Bayesian Inference for CAD-Based Pose Estimation on Depth Images for Robotic Manipulation

Bayessche Inferenz für CAD-basierte Posenschätzung auf Tiefenbildern zur Roboter-Objektmanipulation

Von der Fakultät für Maschinenwesen der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von
Tim Redick geb. Übelhör

Berichter/in: Univ.-Prof. Dr.-Ing. Dirk Abel
Univ.-Prof. Dr.-Ing. Barbara Hammer

Tag der mündlichen Prüfung: 15.04.2024

To my family

Abstract

Camera-based 6D Pose estimation is specifically relevant in the context of robotic manipulation and augmented reality [1]. Robots require the pose estimates to transform local plans, such as grasp points from the object's coordinate frame to the robot's base frame. This thesis originated in medical applications, e.g., estimating the 6D pose of surgical instruments and tracking bones. The challenges are, however, similar to many industrial applications: Textures are missing or unreliable, and parts of the object might be occluded.

Unlike deep learning methods, which recently dominated computer vision, *probabilistic/Bayesian* methods can operate without large datasets and intuitively fuse other sensor measurements or user inputs. Moreover, probabilistic methods include uncertainty estimates, enabling decision-making, e.g., by exploring different viewpoints instead of cutting a bone if the uncertainty is high. Compared to deep learning, insufficient utilization of the computational power of graphics processing units (GPUs) holds back Bayesian inference on images.

This work develops probabilistic models and adapts sampling-based Bayesian inference algorithms for the 6D object pose estimation with known CAD models using depth images from a 3D camera. It only requires a prior for the position of a point, which can be provided, e.g., by attaching tags to the objects or a surgeon pointing onto a bone. Utilizing a GPU improves the method's runtime to enable robotic manipulation applications. Moreover, the experiments ablate the influence of different model components and inference algorithms. Finally, this thesis demonstrates the methods' versatility in different applications: industry-relevant Benchmark for 6D Object Pose Estimation (BOP) datasets, a synthetic dataset of surgical instruments, and tracking the pose of a bone.

The results show that per-object runtimes of $\approx$1 s are possible, while achieving an average recall of 0.475 on BOP for unseen objects. State-of-the-art methods participating in the same category achieve recalls of 0.674, but additionally require color images and per-image runtimes >30 s [2]. The Bayesian pose inference is competitive if the runtime is limited. As the models have no semantic understanding, combining them with learning-based methods might be a promising direction for future research.

Kurzfassung

Kamerabasierte 6D Posenschätzung ist insbesondere im Kontext der Roboter-Objektmanipulation von Bedeutung [1]. Roboter benötigen diese, um lokale Pläne wie Greifpunkte aus dem Koordinatensystem eines Objekts in das des Roboters zu transformieren. Diese Arbeit hat ihren Ursprung in medizinischen Anwendungen wie der Schätzung der 6D-Position von chirurgischen Instrumenten und dem Tracking von Knochen. Jedoch bieten industrielle Anwendungen ähnliche Herausforderungen: Texturen fehlen oder sind unzuverlässig, und Teile des Objekts können verdeckt sein.

Im Gegensatz zu Deep Learning, dem aktuellen Goldstandard in der Bildverarbeitung, benötigen Bayessche Methoden in der Regel keine großen Datensätze und ermöglichen eine intuitive Einbindung weiterer Sensoren. Darüber hinaus können die geschätzten Unsicherheiten eine informierte Entscheidungsfindung ermöglichen. Im Vergleich zu Deep Learning wird in der Bayesschen Inferenz zur Bildverarbeitung die verfügbare Rechenleistung von GPUs unzureichend genutzt, was diese Methoden zurückhält.

In dieser Arbeit werden probabilistische Modelle für die Bayessche Inferenz auf Tiefenbildern mit bekannten CAD-Modellen entwickelt und samplingbasierte Inferenzalgorithmen angepasst. Lediglich ein Vorwissen über die Position eines Objektpunktes ist erforderlich. Durch die Verwendung einer GPU wird die Laufzeit der Methode so weit verkürzt, dass der Einsatz im Rahmen robotischer Objektmanipulation möglich ist. Die Einflüsse verschiedener Modellkomponenten werden systematisch evaluiert. Schließlich wird die Vielseitigkeit der gezeigten Methoden für die Posenschätzung in diversen Anwendungen gezeigt: Industrielle Objekte, chirurgische Instrumente und das Tracking von Knochenposen.

Die Ergebnisse zeigen, dass Laufzeiten von <1 s pro Objekt möglich sind, während ein durchschnittlicher Recall von 0,475 auf BOP Datensätzen für ungesehene Objekte erreicht wird. Der Stand der Technik erreicht einen Recall von 0,674, benötigt jedoch zusätzlich Farbbilder und Laufzeiten pro Bild von >30 s [2]. Die Bayessche Poseninferenz ist bei begrenzter Laufzeit kompetitiv. Da die Modelle kein semantisches Verständnis haben, könnte die Kombination mit lernbasierten Methoden ein vielversprechender Ansatz für künftige Forschungen sein.

Contents

List of my Publications

[1] Tim Redick, Christina Kohler, Kirsten Lena Voß, Christian Blume, Daniel Delev, Hans Rainer Clusmann, Dirk Abel, and Heike Vallery. IMU-based displacement detection of spinal vertebrae during image-based surgeries. In *Abstracts of the 57th Annual Meeting of the German Society of Biomedical Engineering 26 – 28 September 2023, Duisburg, Including: The Artificial Vision Symposium – the International Symposium on Visual Prosthetics*, volume 68 of *Biomedical Engineering*, page 87, Berlin [u.a.], 2023. de Gruyter / 57th Annual Meeting of the German Society of Biomedical Engineering 26 – 28 September 2023, Duisburg,. doi: 10.1515/bmte-2023-2001.

[74] Tim Übelhör, Jonas Gesenhues, Nassim Ayoub, Ali Modabber, and Dirk Abel. 3D camera-based markerless navigation system for robotic osteotomies. *at - Automatisierungstechnik*, 68(10):863–879, October 2020. ISSN 2196-677X, 0178-2312. doi: 10.1515/auto-2020-0032.

[3] Tim Übelhör, Jonas Gesenhues, Nassim Ayoub, Ali Modabber, and Dirk Abel. Augmented Reality Schablonen für intelligente kollaborative Roboter in der Chirurgie;. In *Automation 2020 : 21. Leitkongress Mess- u. Automatisierungstechnik : Shaping Automation for Our Future, 30. Juni u. 01. Juli 2020 / VDI VDE Mess- Und Automatisierungstechnik*, volume 2375 of *VDI-Berichte / Verein Deutscher Ingenieure. - Düsseldorf : VDI-Verl., 1955-*, pages 603–615, Düsseldorf, June 2020. VDI Verlag GmbH.

[4] Tim Übelhör, Jonas Gesenhues, Nassim Ayoub, Ali Modabber, and Dirk Abel. Depth Camera Based Particle Filter for Robotic Osteotomy Navigation. *arXiv:1910.11116 [cs]*, October 2019.

[104] Ali Modabber, Nassim Ayoub, Tim Redick, Jonas Gesenhues, Kristian Kniha, Stephan Christian Möhlhenrich, Stefan Raith, Dirk Abel, Frank Hölzle, and Philipp Winnand. Comparison of augmented reality and cutting guide technology in assisted harvesting of iliac crest grafts – A cadaver study. *Annals of Anatomy - Anatomischer Anzeiger*, 239:151834, January 2022. ISSN 0940-9602. doi: 10.1016/j.aanat.2021.151834.

[6] Philipp Winnand, Nassim Ayoub, Tim Redick, Jonas Gesenhues, Marius Heitzer, Florian Peters, Stefan Raith, Dirk Abel, Frank Hölzle, and Ali Modabber. Navigation of iliac crest graft harvest using markerless augmented reality and cutting guide technology: A pilot study. *The International Journal of Medical Robotics and Computer Assisted Surgery*, August 2021. ISSN 1478-5951, 1478-596X. doi: 10.1002/rcs.2318.

[115] Johannes Kummert, Alexander Schulz, Tim Redick, Nassim Ayoub, Ali Modabber, Dirk Abel, and Barbara Hammer. Efficient Reject Options for Particle Filter Object Tracking in Medical Applications. *Sensors*, 21(6):2114, January 2021. doi: 10.3390/s21062114.

ORCID Tim Redick: https://orcid.org/0000-0001-5719-297X

List of Symbols

Symbol	Description	Unit
Greek Symbols		
α	Acceptance ratio	
β	Scale parameter	
$\boldsymbol{\mu}_{img}$	Expected depth image	m
μ	Mean	
$\boldsymbol{\theta}$	Vector of generic parameters	
$\boldsymbol{\Phi}$	Random rotation	rad
σ	Standard deviation	
τ	Threshold	
Latin Symbols		
Bern	$\mathrm{Bern}(o)$ Bernoulli distribution with the probability o for the *true* event	
$\mathbf{K}$	Kamera intrinsics matrix	
Cat	$\mathrm{Cat}(p_1, \ldots, p_k)$ Categorical distribution with k category probabilities	
c_x	Optical center in x-direction	
c_y	Optical center in y-direction	
c_o	Classification (is object)	
o	Classification probability	
L_0	Image likelihood regularization using classification probabilities	
c_{reg}	Regularization constant	
$\varnothing$	Diameter	m
ELBO	Evidence lower bound	

e	Error metric	
erf	Error function	
$\mathbb{E}$	Expected value	
Exp	$\mathrm{Exp}(\beta)$ Exponential distribution with scale parameter β	
f_x	Focal length in x-direction	
f_y	Focal length in y-direction	
Gumbel	$\mathrm{Gumbel}(\mu, \beta)$ standard Gumbel distribution for $\mu = 0, \beta = 1$	
u	Horizontal image coordinate	
v	Vertical image coordinate	
$\mathbb{1}$	Indicator function	
J	Jacobian Matrix	
KL	Kullback–Leibler divergence	
LSE	LogSumExp	
Mix	$\mathrm{Mix}(a, b)$ Mixture model of the distributions a and b	
N_{eff}	Effective sample size	
$\mathcal{N}$	$\mathcal{N}(\mu, \sigma^2)$ Normal distribution with mean μ and variance σ^2	
N_{px}	Number of pixels in an image	
$\mathbf{z}$	Observation / measurements	
$\mathbf{P}$	Pose as transformation matrix	
pdf	Probability density function	
p	Probability distribution	
$\mathbf{q}$	Rotation as unit quaternion	
$\mathbb{R}$	Real numbers	
$\mathbf{R}$	Rotation as a $\mathbb{R}^{3\times3}$ matrix	
L_{px}	Image likelihood regularization using number of pixels	
$\mathbf{x}$	State vector	
$\mathbf{t}$	Translation as vector	m
$\mathcal{U}$	$\mathcal{U}(a, b)$ Uniform distribution for interval $[a, b]$	
$\mathbf{v}$	Vertex or 3D point	
V_M	Vertices of a 3D model	

w	Weight	
z_{max}	Upper bound of measured depth	m
z_{min}	Lower bound of measured depth	m

Acronyms and Abbreviations

ADD	average distance of model points
ADD-S	average distance of model points with indistinguishable views
API	application programming interface
BOP	Benchmark for 6D Object Pose Estimation
CAD	computer-aided design
CNN	convolutional neural network
CPU	central processing unit
CT	computed tomography
ESS	effective sample size
FN	false negative
FP	false positive
GAN	generative adversarial network
GPU	graphics processing unit
HMC	Hamiltonian Monte Carlo
ICP	iterative closest point
ITODD	MVTec industrial 3D object detection dataset
KNN	k-nearest neighbors
LM	Linemod
LM-O	Linemod-Occluded
MCMC	Markov chain Monte Carlo
MDD-S	maximum distance of model points for symmetric objects
MH	Metropolis-Hastings
MSPD	maximum symmetry-aware projection distance

MSSD	maximum symmetry-aware surface distance
MTM	multiple-try Metropolis
PF	particle filter
PnP	perspective-n-point
PPL	probabilistic programming language
RANSAC	random sample consensus
RFID	radio-frequency identification
ROS	Robot Operating System
SE(3)	3D special Euclidean group
SMC	sequential Monte Carlo
SO(3)	3D rotation group
STERI	synthetic surgical instruments
T-LESS	texture-less rigid objects
TN	true negative
TP	true positive
VI	variational inference
VSD	Visible Surface Discrepancy

1 Introduction

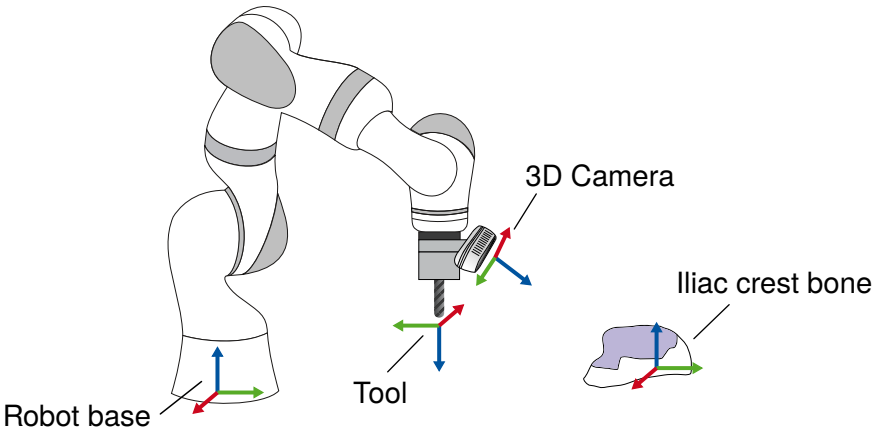


Figure 1.1: Robotic manipulator setup for 3D camera-based surgeries with relevant coordinate systems.

This work originated in medical applications involving dangerous and repetitive tasks predestined for robotic automation, such as sorting contaminated surgical instruments for sterilization. Individualized applications like facial reconstruction surgeries require high precision and flexibility compared to these repetitive applications. Collaborative robots can potentially relieve understaffed sectors, e.g., the sterile supply, and offer the precision required to enable new surgical applications.

Robotic manipulation of rigid objects, such as instruments and bones, requires transforming a plan defined in the object's coordinate frame to the robot's base frame to control the pose of a tool mounted to the robot. The 6D pose of an object describes the required transformation rule. For example, when replacing a defective jaw bone with pieces from the iliac crest (hip) bone, surgeons plan the cutting edges in the bone's frame as depicted in Fig. 1.1. Current robotic surgery systems attach markers as local references to the bone and require an additional registration step to align the plan to the bone in the marker frame. Compared to application-specific marker systems, a 3D camera offers more flexibility and is suitable for a broader range of applications, e.g., bin picking. One challenge in medical and industrial applications is that the objects do not

possess a reliable texture, which many computer vision algorithms require. 3D cameras can overcome this challenge since depth images are purely geometric. Recent 3D cameras[1] enable capturing even difficult scenes with reflective and small objects.

Nevertheless, using a 3D camera introduces uncertainty to the system, as it perceives an unstructured environment and introduces sensor noise. Bayesian methods can explicitly handle this uncertainty by formalizing the iterative inclusion of new data. Moreover, formulating models for Bayesian inference follows an intuitive description of the measurements' generative process. Given a known state, this generative process describes what the measurement would look like. The Bayesian workflow allows models to be adapted to new applications by fusing additional sensor measurements or including learned black-box models.

1.1 Scope

In addition to medical applications, the methods developed in this work also cover industrial applications, which offer similar challenges and chances. Typically, 3D camera-based pose estimation algorithms must handle a wide variety of rigid objects for which 3D models are available, e.g., as computer-aided design (CAD) models from the manufacturer or preoperative computed tomography (CT) scans of bones.

Hodan et al. defined two different 6D pose estimation problems depending on the prior information available: the 6D localization problem and the 6D detection problem [3]. While the former provides the classes and number of object instances in an image, the latter provides no information on how many objects are present. Almost all pose estimation algorithms use one model to detect the objects and another to estimate the pose in a second step [2]. This work focuses on the second step and assumes that the detections are available from another model, previous timesteps, or additional sensors, e.g., radio-frequency identification (RFID) tags attached to the objects. The methodological focus of this work is the application of sampling-based Bayesian inference algorithms to the problem of 6D pose estimation using depth images as the primary sensor signal.

[1] Zivid One+, Oslo, Norway

1.2 Relevance

Deep neural networks have revolutionized the field of computer vision by providing an approach that does not necessitate an extensive understanding of image processing and feature engineering. This simplification has made the field more accessible to various individuals and organizations. However, neural networks typically require large annotated datasets for training. Often, these large datasets are unavailable for specific medical or industrial applications. Simulators like BlenderProc allow the generation of vast amounts of annotated data and reduce the sim-to-real gap using high-fidelity ray tracing [2, 4]. However, medical and industrial applications might require detecting thousands of classes. Scaling deep learning methods to so many classes relies on zero-shot learning of multimodal models, which can, for example, output detections from free text [5]. However, how these methods perform with different variants of the same class remains unclear, e.g., scissors and clamps of different sizes. Additional sensors like RFID tags could provide per-instance classification. Integrating additional sensors into neural networks is not straightforward and requires additional training steps and data.

Generative models for image-based Bayesian inference replace complex feature engineering with a stochastic scene generator, a renderer, and a likelihood model [6]. Moreover, generative modeling is a white-box approach that can intuitively integrate expert knowledge and additional sensors. Besides, downstream decision-making and control tasks can use the uncertainty estimates that Bayesian methods include.

Still, sampling-based Bayesian inference algorithms typically exhibit long computation times of several minutes, especially for high dimensional problems such as image-based pose estimation [7]. Real-world robotic applications require that the pose estimation finishes within the cycle time. Furthermore, the lack of good tooling held back Bayesian inference compared to deep learning methods [8, 9]. While tools for generative modeling and Bayesian inference have improved recently, combining them with the rendering process is not straightforward. Moreover, state-of-the-art Bayesian inference algorithms use gradient information to speed up convergence. Classical rasterization-based rendering does not offer gradients, and differential renderers are in a research stage with prohibitively long runtimes [8, 10].

1.3 Research Hypothesis and Structure

The computational budget hinders the application of sampling-based inference algorithms to image-based pose estimation. As no gradients are available to enable advanced sampling algorithms, the straightforward alternative is to use more computational power to generate more samples with a given time budget. GPUs can enable this increase in computation power by using parallel processing. Deep learning frameworks have simplified the efficient utilization of GPUs, which is a primary reason for the breakthroughs of deep neural networks in a wide range of applications. Thus, this work hypothesizes that Bayesian inference can be a viable alternative to deep learning methods for 6D pose estimation when using a GPU. Subsequently, this work answers the following research questions:

- What does a probabilistic model for depth image-based pose estimation look like?
- Which Bayesian inference algorithms lend themselves for parallelization on a GPU?
- How do the components of the models and inference algorithms influence the performance?
- How competitive is the resulting method to the state of the art in a standardized pose estimation benchmark?
- Are the developed methods applicable to various domains and tasks?

First, this work introduces relevant theoretical foundations on Bayesian statistics, probabilistic models, 3D poses, and cameras in Chapter 2. The following Chapter 3 explores related work focusing on image processing, pose estimation, and image-based Bayesian inference.

Afterward, the modeling Chapter 4 answers the first research question by introducing a generative model that draws depth images from 6D poses using 3D rendering on the GPU. Moreover, this chapter explores suitable distributions to model the uncertainty of the position, orientation, and pixel measurements. Special care is taken to formulate a model of the depth measurement for the execution on the GPU. Moreover, this work proposes approaches to handle occlusions and regularize an image's likelihood. The subsequent Chapter 5 introduces the Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) inference algorithms and explains which variants are suitable for the parallelization on a GPU. Moreover, essential best practices are introduced for a successful pose inference.

Chapter 6 evaluates the algorithms and models developed throughout this work. This chapter starts by selecting suitable datasets and metrics to calculate a performance score. Then, the influence of individual components, for example, the sampler runtime and occlusion models, is investigated. After an automatic parameter tuning, the algorithms are tested on datasets of the Benchmark for 6D Object Pose Estimation (BOP) and compared to the state of the art. Finally, two medical applications demonstrate the applicability of the methods to a wide range of challenging problems, estimating the pose of slim surgical instruments and tracking bone poses in Chapter 7.

Finally, Chapter 8 concludes the results and limitations of the developed methods. Furthermore, possible directions for future research to improve the methods are presented. Possible approaches to the use of probabilistic pose estimation for decision-making and control of robot manipulators are given.

2 Theoretical Foundations and Notations

This chapter introduces foundational concepts of Bayesian statistics used throughout the thesis, based on the explanations from [11, 12]. 6D poses are described as transformations by translational and rotational components. This chapter also describes the pinhole camera model for transforming 3D points to 2D images, which is the foundation for image-based pose estimation. Moreover, this chapter explains the notations from this.

2.1 Foundations of Bayesian Statistics

Probability density functions describe how likely observing a variable x is given some parameters. For example, a normal distribution is parametrized by the mean μ and standard deviation σ and has the following probability density function:

$$p_N(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{2.1}$$

In this case, x is a *random variable*, which states that x is subject to uncertainty. If a problem consists of multiple random variables, the probability of simultaneously observing specific values for all variables is described by the *joint* probability distribution. For example, $p(x, z)$ is the notation of a joint probability distribution of the random variables x and z. If the value of some variable is known or observed, the joint probability distribution can be *conditioned* on it. The notation of a *conditional probability distribution* for a known z is $p(x \mid z)$. It reads as: "the probability of observing x given z has a known value."

The chain rule of probability allows the calculation of the joint probability distribution using a conditional distribution. For example, the joint distribution of two variables x and z can be calculated as:

$$p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x) \tag{2.2}$$

It is important to note that conditioning on a variable is always possible. Independence assumptions do, however, not always apply. If x and z are *independent*, the probability of observing x does not change if the value of z changes - the same holds for z if x is known. In this case, Eq. (2.2) simplifies to:

$$p(x \mid z)p(z) = p(z \mid x)p(x) = p(x)p(z) \tag{2.3}$$

Conditional independence describes that two or more variables are independent given other variables:

$$p(x, y \mid z) = p(x \mid z)p(y \mid z) \tag{2.4}$$

Finally, a variable can be removed from the distribution via *marginalization*, also known as the law of total probability. Marginalization integrates over all possible values of a variable:

$$p(x) = \int p(x, z)dz = \int p(x \mid z)p(z)dz \tag{2.5}$$

If the variable is discrete, the integral becomes a sum. The notations used above are handy for mathematical manipulation. Another common notation describes a model as a *generative* process [13].

$$x \sim \mathrm{Exp}(\beta) \tag{2.6}$$

$$z \sim \mathcal{N}(x, \sigma) \tag{2.7}$$

The first reads: "x is distributed according to an exponential distribution parametrized by β." which implies that x can be generated by drawing samples from the exponential distribution. Eq. (2.7) defines a hierarchical model where x must first be known or sampled to evaluate the normal distribution. Variables that do not serve as parameters of other distributions are usually *observed*, while the other variables are called hidden or *latent variables*. The objective is to infer the posterior distribution $p(x \mid z)$ of the hidden variable x given the observation z.

2.2 Probability Distributions Used in This Thesis

As defined in Eq. (2.1), the *normal* distribution is a continuous probability distribution. It is a symmetric distribution centered around the mean μ with the standard deviation σ. Many noise characteristics can be modeled sufficiently well with a normal distribution, and it is a common choice because of its analytical properties. Being a part of the exponential family of distributions allows for closed-form solutions such as the Kalman filter [12].

Another member of the exponential family is the *exponential* distribution. It is parametrized by the rate parameter λ or by the scale parameter $\beta = \lambda^{-1}$. The probability density function is defined as:

$$p_{\text{Exp}}(x \mid \beta) = \begin{cases} \frac{1}{\beta} e^{-\frac{x}{\beta}} & , x \geq 0 \\ 0 & , x < 0 \end{cases} \tag{2.8}$$

Uniform distributions describe that a variable can take on any value in a given interval $[x_{min}, x_{max}]$. As all values are equally likely, the uniform distribution denotes the absence of knowledge and is referred to as *uninformed*. The probability density function is defined as:

$$p_{\mathcal{U}}(x \mid x_{min}, x_{max}) = \begin{cases} \frac{1}{x_{max} - x_{min}} & , x \in [x_{min}, x_{max}] \\ 0 & , \text{else} \end{cases} \tag{2.9}$$

The *Bernoulli* distribution models the probability of a discrete binary event, which is typically denoted by 0 for *false* and 1 for *true*. This distribution is parametrized by a single variable $o \in [0, 1]$, which is the probability of the *true* event.

$$p_{\text{Bern}}(x \mid o) = \begin{cases} o & , x = 1 \\ 1 - o & , x = 0 \end{cases} \tag{2.10}$$

The *Categorical* distribution is a discrete probability distribution where a random variable can be drawn from k classes. Each class has a probability weight w_i associated with it, where $\sum_{i=1}^{k} w_i = 1$. The probability density function is:

$$p_{\text{Cat}}(x_i \mid w_1, \ldots, w_k) = w_i, i \in [1, \ldots, k] \tag{2.11}$$

Elements from a *mixture* distribution can be sampled using a categorical distribution. A mixture distribution consists of k distributions p_i and the same number of weights w_i. Generating random variables distributed according to a mixture distribution requires two steps. First, draw a class i according to the probability weights w_i from a categorical distribution parametrized by the same weights. Second, draw a random value from the distribution p_i. Drawing from a mixture with two components $x \sim \text{Mix}(p_1, p_2; w_1, w_2)$ is equivalent to the following notation.

$$\begin{aligned} i &\sim \text{Cat}(w_1, w_2) \\ (x \mid i = 1) &\sim p_1 \\ (x \mid i = 2) &\sim p_2 \end{aligned} \tag{2.12}$$

This is similar to an if-else statement in programming: If $i = 1$, draw from p_1, else if $i = 2$, draw from p_2. The probability density function of a mixture distribution is defined as:

$$p_{\text{Mix}}(x \mid p_1, \ldots, p_2; w_1, \ldots, w_k) = \sum_{i=1}^{k} w_i p_i(x) \tag{2.13}$$

Even though drawing from a mixture involves only a single component at a time, the density functions of all components are evaluated and weighted accordingly.

2.3 Probabilistic Models

Historically, probabilistic models have been used to fit a set of parameters $\boldsymbol{\theta}$ on stateless data $\mathbf{z}$ using *Bayesian inference* [11]. In signal processing, the goal is to estimate a time-variate state $\mathbf{x}(t)$, sometimes jointly with the parameters given a stream of measured observations $\mathbf{z}(t)$, called *filtering*. Even though the naming differs, the same methods can infer latent variables using a probabilistic model given noisy data or measurements. This thesis focuses on estimating the state, and the parameters are considered hyperparameters, so they are not part of the inference problem. For static problems, the state $\mathbf{x}$ is time-independent, and for the general case, the time variable t is omitted and used as an index for the algorithm step instead.

Bayes' theorem is the basis of probabilistic models. It allows calculating the *posterior* distribution of the state $\mathbf{x}$ given a set of observations $\mathbf{z}$:

$$p(\mathbf{x} \mid \mathbf{z}) = \frac{p(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} \tag{2.14}$$

The term $p(\mathbf{z} \mid \mathbf{x})p(\mathbf{x}) = p(\mathbf{z}, \mathbf{x})$ is what will be referred to as *probabilistic model* from here on. First, the probabilistic model contains the probability $p(\mathbf{z} \mid \mathbf{x})$ of observing $\mathbf{z}$ for a specific state $\mathbf{x}$, which is called *likelihood*. Second, the *prior* $p(\mathbf{x})$ can encode an expert's knowledge about the distribution of the state and parameters. Another possibility is to use data and the posterior of one model as the prior of another model.

The denominator of Eq. (2.14) is called *marginal likelihood* $p(\mathbf{z})$ which is calculated by marginalizing the latent variables $\mathbf{x}$ of the probabilistic model:

$$p(\mathbf{z}) = \int p(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{2.15}$$

Integrating over all latent variables can be interpreted as the probability that the observations $\mathbf{z}$ have been generated from the specific probabilistic model.

Intuitively, the value resulting from the integral increases if the model better fits the data. Therefore, $p(\mathbf{z})$ is often called the evidence of the model or *model evidence*. Calculating the integral in Eq. (2.15) is intractable for most problems with a continuous state-space. Conjugate priors and likelihoods are a special case where a closed-form solution is generally possible. However, conjugate priors significantly restrict the choice of models. One example is the exponential family of distributions, where the conjugate prior is typically also a member of the exponential family. For discrete state-spaces with a low dimensionality, it can also be tractable to sum over all possible states and calculate a closed-form solution of Eq. (2.15).

In most cases, it is sufficient if the posterior probabilities of states can be compared relative to each other. In these cases, $p(\mathbf{z})$ can be interpreted as a normalization constant since it does not depend on the inferred state $\mathbf{x}$. Thus, the *probabilistic model* used in Bayesian inference is typically formulated proportional to the posterior:

$$p(\mathbf{x} \mid \mathbf{z}) \propto p(\mathbf{z} \mid \mathbf{x}) p(\mathbf{x}) = p(\mathbf{z}, \mathbf{x}) \tag{2.16}$$

The joint probability distribution $p(\mathbf{z}, \mathbf{x})$ is also called *generative model*, which allows the simulation of synthetic data, ideally similar to actual observations [13]. This model can also be used for Bayesian inference, which will be examined in Chapter 5.

2.4 Graphical Model for Inference

In practice, a probabilistic model is commonly formulated using a probabilistic programming language (PPL). Probability distributions are defined for each latent variable, which might depend on other latent variables. After defining the distributions, a PPL engine compiles the model for inference. In the case of models with a fixed number of variables, a Bayesian network can be generated which is a directed acyclic graph [13].

Figure 2.1 represents the Bayesian network for the following exemplary model, which consists of the latent variables a and b, a deterministic node d, and the observed variable c:

$$\begin{aligned} a &\sim \mathcal{N}(0, 1) \\ b &\sim \mathrm{Exp}(1) \\ d &= a + b \\ c &\sim \mathcal{N}(d, 1) \end{aligned} \tag{2.17}$$

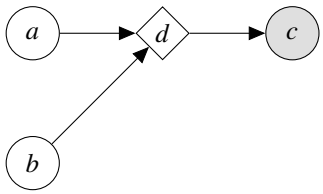


Figure 2.1: Example of a Bayesian network for Eq. (2.17).

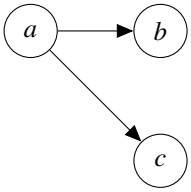


Figure 2.2: Bayesian network with multiple nodes conditioned on a.

In Bayesian networks, the arrows point to variables influenced by the predecessors, so a and b are the parents of d. On the contrary, a program that processes the graph stores the structure in the opposite direction where c is the root node, which knows d as its child. This thesis uses the Bayesian convention. The rhombus-shaped node represents a deterministic function. Moreover, nodes a and b also encode the prior of the model, while the c represents the observation by convention using a gray color. Prior nodes must be fully parametrized, while child nodes are parametrized by their parents.

Sampling from this model is a forward evaluation: Generate random values from the prior/root nodes, then generate random values for their child nodes, the children of the children, and so forth until the leaf/observation nodes are evaluated. Algorithmically, this can be implemented by running a depth-first search every time the graph has to be evaluated; performing a topological sorting once can improve the performance. Care has to be taken that a node is not evaluated twice if multiple nodes are conditioned on the same node as in Fig. 2.2. Consequently, all values have to be stored and associated with the node. A node can only be asked to return a random sample if no value has been stored before. Storing values for certain variables before sampling from the graph also adds a notion of *conditioning* to the graph on these variables. Conditioning on variables is specifically helpful if only blocks of variables are sampled, which is a common practice in Bayesian inference; see Section 5.2.6.

During the **evaluation**, each variable has a value assigned to it. Subsequently, all nodes are d-separated, and the network's likelihood is the product of the individual nodes given their ancestor's values. In case of the network in Fig. 2.1 this results

in:

$$p(c \mid a, b, d) = p(c \mid d)p(a)p(b) \tag{2.18}$$

After inference, the distribution of the latent variables is known, and one can use this model to generate new data from the *posterior predictive distribution*.

2.5 6D Pose and Transformations

Generally, a 6D pose has three degrees of freedom for the position and three degrees of freedom for the orientation. In the literature, a 6D pose is also referred to as an element of the 3D special Euclidean group (SE(3)) [14]. For simplicity, this work does not introduce SE(3) formally but uses it as a synonym for a transformation rule $\mathbf{P}$ of a 3D point $\mathbf{v}$ expressed or measured from frame A to a target frame B:

$${}^{\mathrm{B}}\mathbf{v} = {}^{\mathrm{B}}\mathbf{P}^{\mathrm{A}}\,{}^{\mathrm{A}}\mathbf{v} \tag{2.19}$$

Note that the notation ${}^{\mathrm{B}}\square^{\mathrm{A}}$ will be used for transformations throughout this thesis describing the pose of a frame A measured from frame B. By convention, the pose first rotates the point by a $\mathbb{R}^{3\times3}$ $\mathbf{R}$ matrix and then translates it by a $\mathbb{R}^3$ translation vector $\mathbf{t}$:

$${}^{\mathrm{B}}\mathbf{v} = {}^{\mathrm{B}}\mathbf{R}^{\mathrm{A}}\,{}^{\mathrm{A}}\mathbf{v} + {}^{\mathrm{B}}\mathbf{t}^{\mathrm{A}} \tag{2.20}$$

This convention also enables one to express the same pose as an affine $\mathbb{R}^{4\times4}$ transformation matrix and homogenous coordinates:

$$\begin{pmatrix} {}^{\mathrm{B}}\mathbf{v} \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{\mathrm{B}}\mathbf{R}^{\mathrm{A}} & {}^{\mathrm{B}}\mathbf{t}^{\mathrm{A}} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{pmatrix} \begin{pmatrix} {}^{\mathrm{A}}\mathbf{v} \\ 1 \end{pmatrix} \tag{2.21}$$

Affine transformation matrices and homogenous coordinates are commonly used in 3D computer graphics, which is the focus of Section 4.3.1.

2.6 3D Rotation Representations

In the previous section, rotations have been described by $\mathbb{R}^{3\times3}$ rotation matrices. These matrices allow the transformation of a 3D point to be calculated simply by multiplication. Nevertheless, representing the 3D rotations as matrices, often called the 3D rotation group (SO(3)), requires nine instead of the minimal three parameters.

In theory, three parameters are enough to describe a 3D orientation. For example, Euler angles intuitively represent a rotation. Three consecutive rotations around different axes describe an SO(3) rotation. One problem is that the specification of the order of the axes and whether the axes are static or rotating with the object often leads to confusion. Another problem is that Euler angles have singularities, which lead to a phenomenon known as *gimbal lock* [15]. Moreover, 3D rendering uses transformation matrices to transform vertices, and converting Euler angles to $\mathbb{R}^{3\times3}$ rotation matrices is expensive due to the trigonometric functions involved[1].

It is considered best practice to use an alternative representation that does not suffer from singularities. This work uses unit quaternions in the Hamiltonian convention, and the equations used throughout this thesis are largely based on Sola et al. [15]:

$$\mathbf{q} = [q_w, \mathbf{q}_v]^\mathsf{T} = [q_w, q_x, q_y, q_z]^\mathsf{T} \tag{2.22}$$

The first part q_w is the real part, and $\mathbf{q}_v$ is the imaginary part of the quaternion. An additional advantage of quaternions is that they can easily be converted to $\mathbb{R}^{3\times3}$ rotation matrix using the exponential map, which does not involve trigonometric functions[2]. Compared to Euler angles, one drawback is that four components are an over-parametrization of SO(3) rotations, resulting in non-unique representations known as the *double cover* of SO(3). Specifically, negating a quaternion results in the same rotation.

2.7 Pinhole Camera Model

The pinhole camera model is a simple approach to modeling the projection of 3D points into an image. It neglects nonlinear effects like lens distortion, which can be removed in a preprocessing step. A point ${}^{\mathrm{C}}\mathbf{v} \in \mathbb{R}^3$ measured in the camera frame is transformed to image coordinates ${}^{\mathrm{I}}\mathbf{v} \in \mathbb{R}^2$ using the intrinsic camera matrix $\mathbf{K}$:

$$\underbrace{w \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}}_{{}^{\mathrm{I}}\mathbf{v}} = \underbrace{\begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_{{}^{\mathrm{C}}\mathbf{v}} \tag{2.23}$$

[1] *Rotations.jl*, `RotMat(r::RotXYZ)`: 28.963 ± 6.799 ns
[2] *Rotations.jl*, `RotMat(q::QuatRotation)`: 4.954 ± 1.993 ns

Note that the coordinates u and v represent the pixel coordinates. A third component and the factor $w = z$ are required for the formulation as an affine transformation. The parameters f_x and f_y in Eq. (2.23) are the focal lengths of the camera in x- and y-direction, s is the axis skew, and c_x and c_y describe the location of the principal point in x- and y-direction. In addition, the nonlinear distortion caused by the lens could be considered [16]. Because the datasets used in the evaluation (Chapter 6) do not provide any distortion coefficients, distortions are not considered further.

The extrinsic parameters describe the transformation ${}^{\mathrm{R}}\mathbf{P}^{\mathrm{C}}$ of a point in the optical camera frame to a reference frame, e.g., the mounting point. In practice, 3D cameras are either shipped pre-calibrated or can be calibrated easily using open-source tools, for example, the *Robot Operating System (ROS) camera_calibration* package which is based on *OpenCV* [16, 17, 18].

2.8 Summary

This chapter presented the theoretical foundations of Bayesian statistics and probabilistic models. It showed that the full posterior distribution described resulting from Bayes' law is intractable in most cases. Therefore, the concept of probabilistic models, which can be evaluated up to a constant factor, was introduced. These probabilistic models use probability distributions to describe the generative process of observations or measurements. Commonly used probability distributions from this thesis were additionally introduced. Graphical models have been introduced as a flexible and formal model description. Besides, graphical models enable an efficient sampling and evaluation of the model.

The second half focused on the foundations of 6D poses, which transformation rules can describe. 6D poses consist of a translational and a rotational component, and quaternions were introduced to represent the latter. Quaternions are more concise than rotation matrices and avoid problems of Euler angles like gimbal lock. The pinhole camera model was introduced to bridge the transformation of 3D points to 2D images. It is the simplest model used to render 3D scenes to 2D images.

3 Related Work on Camera-Based 6D Pose Estimation

The BOP challenge introduced standardized datasets and evaluation metrics in 2018, significantly improving the comparability of 6D pose estimation methods [19]. Many datasets of this challenge focus on industrial applications with similar challenges as the medical applications also considered in this thesis. Gorschlüter et al. recently published an extensive survey that compares relevant 6D pose estimation methods for industrial applications [1]. Both serve as a good starting point for the literature review, and the BOP challenge is still open for submissions in 2023.

Pose estimation is often a two-step approach of detecting the object first and estimating its pose using a second algorithm or model [2]. After a short historical overview, Section 3.2 introduces object detection methods. The following Sections 3.3 and 3.4 introduce general algorithms for pose estimation, which can roughly be divided into classical methods focusing on heuristically designed features and (deep) learning-based methods for 6D pose estimation. As many pose estimators use a refinement step similar to pose tracking, Section 3.5 jointly introduces these methods. The final Section 3.6 gives an overview of Bayesian inference for image analysis.

3.1 Short History of Computer Vision

The beginnings of machine vision can be found in the 1950s, when James Gibson introduced optical flow in his work [20]. In the 1960s, it was possible to extract three-dimensional geometric information from 2D images. Subsequently, in the early 1970s, methods were developed for low-level tasks, such as edge detection and segmentation, to solve the problems in a bottom-up approach. In the 1970s and 80s, research focused on understanding and reverse engineering human vision. Researchers have explored the capabilities of neural networks during this time, and most notably, Yann LeCun laid the foundation of using backpropagation for training these networks together with Geoffrey Hinton [21]. Computational

power held these methods back, leading to the AI winter of the 1990s and 2000s. During this period, research focused on hand-crafting color-based and geometric image features. Deep learning methods got their breakthrough due to leveraging the computational power of GPUs with *AlexNet*, known as the *ImageNet moment* in 2012.

3.2 Object Detection and Segmentation

This thesis focuses on the 6D localization task. It assumes that some prior information can be provided, for example, via the localization of RFID tags attached to the object or surgeons pointing to the object of interest. An image-based object detector can be a fallback solution for generating this prior information by predicting the object center. Object (instance) segmentation is about classifying whether a pixel belongs to an object and is an extension of the detection task. Deep learning led to vast growth in the field, and the following methods only represent an overview of practical and easy-to-use models.

In 2012, the convolutional neural network (CNN) *AlexNet* outperformed classical feature-based methods, reducing the error by a large margin [22]. Soon afterward, deep learning methods have also pushed state-of-the-art object detection benchmarks like *PASCAL VOC* [23]. Usually, a base CNN like *ResNet* is trained on a large dataset and used as the *backbone* with modified heads to output detections and segmentations. An often-used example for a CNN-based object segmentation model with a ResNet backbone is *Mask-RCNN* [24]. If inference time is a concern, the *YOLO* family with adaptions to segmentation like *YOLACT++* is a common choice [25].
Large *transformer* networks, known from large language models as *foundation models*, have gained traction in object detection recently. While transformer models dominate the language space, large CNNs still challenge these newer architectures in the image space [26].

3.3 Classical Pose Estimation

Classical pose estimation is based on extracting **sparse hand-crafted features**, which in term consists of two steps [16]: A detector localizes keypoints that can be relocalized robustly in different scenes and lighting conditions: see Fig. 3.1 a). The second step involves calculating a descriptor, which must be unique to match the same keypoint in different views or a CAD model. Commonly used features

Figure 3.1: Principles of pose estimation illustrated on an iliac crest bone: a) matching sparse features, b) template matching, c) dense object coordinates, d) surface fragments, e) direct regression

include SIFT, SURF, and ORB [27]. After calculating and matching similar image features to the model features, estimating the 6D object pose requires solving the perspective-n-point (PnP) problem. As the matchings are noisy in practice, the PnP problem is formulated as an optimization problem and solved using the random sample consensus (RANSAC) algorithm.

Hinterstoisser et al. introduced Linemod as a **template-matching** method for 6D pose estimation in 2013 [28]. They focus on estimating the pose of textureless objects by creating templates of color gradients and surface normals and matching them to the measured image: see Fig. 3.1 b). Afterward, they refine the pose estimates using the iterative closest point (ICP) algorithm; see Section 3.5. This method has gained particular notoriety as it published one of the first standardized benchmark datasets with associated evaluation metrics. The Linemod dataset and the corresponding point distance metrics are described in depth in Section 6.1.3. Hodan et al. focus on improving the growing complexity of using more templates by rigorously filtering detections and using a hash-based voting scheme [29]. Finally, they refine the pose using a particle swarm optimization algorithm, less prone to getting stuck in a local optimum than ICP. Moreover, they achieve run times of <1 s using a GPU implementation. As template-based methods require larger parts of the object to be visible, they are prone to occlusions [1].

Methods that only require depth images or point clouds are invariant to colors, as they purely rely on geometric features. Drost et al. introduced the sparse **point pair feature** in 2010 to recognize objects and estimate their pose in point clouds; compare for Fig. 3.1 a) [30]. This method selects two points on the object and calculates a feature vector based on their distance and corresponding surface normals. These features are invariant to rigid motions, which allows for the calculation of the object pose as a point cloud transformation.
Point pair features have the weakness that they are prone to sensor noise. Hinterstoisser et al. improved on this weakness by spatially spreading the features [31]. This allows them to be matched in other regions during the voting. Additionally, they introduce a subsampling scheme to reduce the runtime significantly.

Only in 2020, deep learning-based methods were able to catch up to point pair features, which was the best-performing method in the BOP challenge until then [32].

3.4 Learning-Based Pose Estimation

Learning-based methods have made the image processing space accessible and led to a significant increase in publications. Again, this thesis can only give an overview of the many recent publications. Classical pose estimation requires knowledge and experience to hand-craft features that work under various conditions. Learning these features from data also requires less tunable parameters in practice, making them more robust under varying conditions [1].
Learning-based methods differ in the input modalities (color, depth, or both modalities), the learning method, and the output representation. Arguably, grouping the methods based on the outputs and corresponding techniques to regress the pose is more purposeful, similar to Hodan's work [33].

One group of approaches aggregates pose predictions via Hough **voting**, initially developed for detecting lines and later other shapes like circles [34]. Tejan et al. adapt the Linemod template matching method into a scale-invariant patch descriptor and train a Hough forest with it [35]. The forest processes image patches, and the leaf nodes store votes for a pose. Accumulating the votes for the position and then the votes for the orientation results in the final pose estimate.
Kehl et al. extended this method to introduce one of the first CNN-based pose estimators [36]. They train a convolutional autoencoder to extract scene descriptors for local image patches. These descriptors are matched to pose votes using an approximate k-nearest neighbors (KNN) algorithm.

Another early tried approach is to predict **object coordinates**, also called *dense 2D-3D correspondences*. Object coordinates assign a unique location on the object's 3D surface to the image pixels, shown via different colors in Fig. 3.1 c). Brachmann et al. introduced this approach to 6D pose estimation using random forests [37]. A random forest assigns votes for object class labels, and the object coordinates to each image pixel. Afterward, they use a RANSAC optimization to regress the pose from the noisy votes.
Krull et al. extended this method using a CNN in 2015, the earliest deep learning approach for pose regression in this literature review [38]. They use Brachmann's random forest to generate predictions for the object class, object coordinates, and pose. Subsequently, they render a depth image and an image of the object coordinates. Finally, a CNN compares the rendered images to the observed depth, the predicted object class, and coordinates.

While Krull et al. used color and depth images, Park et al. proposed a method that regresses the object coordinates using only a color image [39]. They note that training a CNN is difficult if the objects are symmetric, and propose a modified loss function. Moreover, they handle occlusions by training a generative adversarial network (GAN) to reconstruct the occluded parts.

Furthermore, a notable method regressing object coordinates from color images is the *GDR-NET* by Wang et al. [40]. A modified version won almost all categories of the latest BOP challenge in 2022 [2]. Their network outputs a segmentation mask, a dense object coordinate map, and a map of **surface fragments** as proposed by Hodan et al. [41]; see Fig. 3.1 d). They use the segmentation mask only as a supervision signal during training and feed the other two representations into another network, which has the task of solving the PnP problem instead of using RANSAC. Using a network for regressing the pose allows end-to-end training of the pose estimator and the intermediate representations.

Dense object correspondences also lend themselves to the pose estimation of unknown objects, also known as **category-level** pose estimation. The work of Wang et al. introduces a so-called normalized object coordinate space where all category instances are consistently oriented. For example, the tip of a shoe always points in the same direction [42]. Li et al. developed this idea further by including constraints for articulated objects, e.g., scissors or human joints [43]. Another approach has been presented by Park et al., who employ a neural network as a differentiable renderer [44]. This method does not require a CAD model, but only a few annotated images of the object for learning an internal representation. As the neural network is differentiable, they can use a gradient descent approach to compare the renderings to the measurements and optimize the pose.

A similar approach is to predict **sparse keypoints** and match them, like in the classical approaches of Section 3.3 and Fig. 3.1 a). For example, Peng et al. generate a vote for each pixel that points to keypoints and then solve the PnP problem in an uncertainty-driven way [45]. Other approaches rely on manually annotated keypoints, which can be tedious and prohibit a widespread application [46].

The final line of works regresses **direct pose outputs** without the need for solving the PnP problem, as shown in Fig. 3.1 e). Xiang et al. use a CNN architecture for color images, which solves multiple tasks to output pose estimates [47]: It predicts object labels, the object center, the center distance, and a quaternion for the orientation. They use the ICP algorithm to refine the pose if a depth image is available.
Meanwhile, Wang et al. focus on regressing the pose from color images and point clouds [48]. They use a CNN to extract a dense feature vector from a color image, and a *PointNet* [49] to extract another feature vector from the corresponding

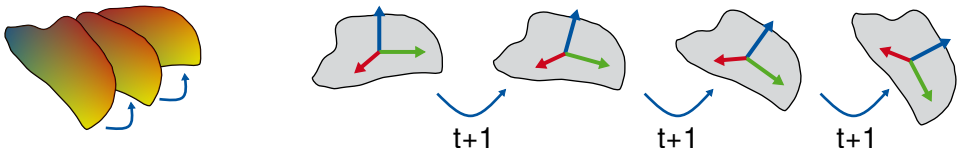


Figure 3.2: Pose refinement iteratively improves an initial guess for a static image (left). Pose tracking updates the pose for a sequence of images (right).

point cloud. After concatenating the feature vectors, a pose predictor network regresses a pose and a confidence value for each pixel in the image. They use the pose with the highest confidence score for the final pose output.
Compared to the previous method and many others, Wu et al. do not require a color image for a detection step. Instead, they create the segmentation mask as an intermediate step from a point cloud [50]. Afterward, they use a keypoint voting scheme to regress the 6D pose of the object.

By **discretizing** the output space, the pose estimation task can be reformulated as a classification problem. Kehl et al. extended the single-shot detection paradigm to 6D pose estimation [51]. A single-shot 2D detector discretizes the color image into overlapping bounding boxes and outputs classification labels for each box. In addition to the classification labels, Kehl et al. output scores for discrete viewpoints and in-plane rotations. For the final output, they select the viewpoint with the highest score and apply an ICP refinement if a depth image is available.
Sundermeyer et al. increase the flexibility of this approach by training an object-independent autoencoder instead of training one network with a given set of classes [52]. Training the autoencoder to reconstruct the object is only an auxiliary task, and the method uses the internal feature embedding for the pose estimation. A codebook can be generated by rendering synthetic views of an object and storing the feature embeddings in a codebook. Similar views result in similar feature embeddings, so they use a KNN algorithm to find the most similar embeddings and use the pose of the corresponding rendering as the estimate.

3.5 Pose Refinement And Tracking

Pose refinement and tracking are similar problems because they assume a prior pose (distribution) close to the measured pose to be available. Conceptually, pose refinement is often formulated as an optimization problem that iteratively predicts the difference between the current pose and the measured one; see Fig. 3.2 left. In contrast, pose tracking is usually formulated as a probabilistic

filter. Filters include a motion model to predict the pose distribution for the next image and a measurement model to update the distribution: see Fig. 3.2 right. Moreover, tracking typically requires that the algorithm runs at the frame rate of the camera. Optimization and filtering approaches can be used to some extent for the refinement and tracking problems.

Many pose estimation methods rely on a depth-image or point cloud-based **refinement** step to improve the performance in benchmarks [32]. Commonly, the ICP algorithm is used for refinement. This algorithm iteratively matches the closest points of the source and target cloud to minimize the point-to-point distances. For large point clouds, the algorithm can become costly, at worst $\mathcal{O}(N^2)$ for N points in each cloud [53]. Tree-based algorithms can improve the performance of ICP but require more memory [54].
Another approach to refinement is to use a differentiable renderer to compare a rendered image of the pose to the measured image and use a gradient-based optimization [55]. The neural network-based differentiable renderer introduced in Section 3.4 refines the Pose estimate similarly [44]. Li et al. train a neural network to compare a rendered image of the object pose and the measured image to predict the difference directly [56]. Their method is also applied iteratively to achieve better results.

Robotic control and augmented reality applications require **tracking** an object at a high frame rate to reduce latency. Even if a high tracking frequency is not required, a higher tracking frequency reduces the distance an object can travel from image to image. Kalman and particle **filters** have been successfully applied to the localization of mobile robots in 2D since the late 1990s [57]. The computational cost of sampling-based algorithms like particle filters grows exponentially with the problem's dimensionality, known as the *curse of dimensionality*.
It took almost fifteen years until Choi et al. introduced the first real-time capable algorithm to track 6D poses [58]. They use *OpenGL-CUDA interop* to efficiently render color, depth, and normal images of pose hypotheses and access them to calculate the likelihood function of a particle filter directly on the GPU.
As Choi's method is not robust against occlusions, Wüthrich et al. added an estimate for each pixel's occlusion probability to the particle filter [59]. They use a relatively complex formulation for the likelihood of a range measurement, involving marginalizing variables via an analytical integration.
In contrast to said approach, Krull et al. focus on improving the proposal model to reduce the number of required particles by placing them in better locations [60]. They use the random forest-based object coordinate predictor from [37] to generate global proposals and mix them with local proposals from the motion model.

Another line of work applies **gradient-based optimization** specifically to tracking applications. Early examples use a region-based approach, which estimates a segmentation mask discriminating between the object's foreground and background and compares it with color images [7]. They implement various steps of their algorithm on the GPU to achieve a tracking rate of ≈20 – 25 Hz. Tjaden et al. improved this method by deriving a Gauß-Newton optimization scheme, which allows them to run the calculations on the central processing unit (CPU) at ≈50 – 60 Hz [61].
One limitation of the region-based trackers is that they either require textured objects or that the object must be distinguishable from the background. Therefore, Kehl et al. have included depth images in their tracking algorithm [62]. They have also improved the performance to ≈370 Hz on the CPU by pre-calculating information in an offline rendering step and using a sparse approximation for the optimization. Stoiber et al. improved the performance even further to ≈790 Hz by combining a Gauß-Newton optimization and a sparse precomputed viewpoint model for the region-based model. Region-based optimizations generally have the disadvantage that the object must be distinguishable from the background, which often is not the case in industrial or medical applications [63].

Most **learning-based** methods for tracking predict the difference/error between the current pose and the next measured image. Tan et al. use random forests to solve this regression problem [64]. The forests learn how the object's pose should change for a given change of the depth images and can be executed on the CPU at 500 Hz. However, this approach can only handle mild occlusions.
Building on the success of deep learning in other applications, Garon et al. train a CNN to predict the pose difference between a render and a measured image [65]. It is similar to the approach for pose refinement in [56], which has also been applied to tracking. Using a CNN makes these methods relatively robust to occlusions and allows them to recover from large errors. This approach can achieve a tracking rate of ≈100 Hz using a GPU for inference.
Deng et al. combine deep learning with a particle filter using *Rao–Blackwellization* [66]. They extend the work of Sundermeyer et al. on using autoencoders to match views to a codebook of feature embeddings for the application in tracking. Compared to Sundermeyer, their method maps the observation's feature embeddings and the codebook to a probability distribution over the discrete orientations instead of selecting the most similar embedding. By marginalizing the orientations, they can also calculate the likelihood of the position.

3.6 Bayesian Inference on Images

Few publications exist on using Bayesian inference for 6D object pose estimation most likely due to the *curse of dimensionality* [67]. Therefore, this section first highlights pose estimation and tracking algorithms from the previous sections, which contain probabilistic elements. Then, it gives an overview of the previous applications of Bayesian inference on general image analysis. Finally, it presents prior work on sampling-based pose estimation.

Some methods introduced in Section 3.4 contain probabilistic/Bayesian elements. For example, the forest-based methods for voting and predicting object coordinates output probabilities [35, 37]. Krull et al. use MCMC for approximate Bayesian inference to train a CNN, which calculates an energy function. This energy function is used in a probability distribution of the predicted object coordinates [60].

Bayesian inference uses a generative model that describes how the data might have been generated instead of building complex bottom-up processing pipelines. This approach has been used since the late 1990s to localize planar robots, and 2013 for tracking 6D object poses using particle filters; see Section 3.5 [57, 59]. Particle filters are sampling-based Bayesian inference algorithms for temporal data.
Early applications of using the Bayesian inference algorithm MCMC on images are based on low-level visual features like color histograms or edges. In 2002, Tu et al. segmented gray-scale images into regions of four texture types: uniform, clutter, textured, and shaded [68]. Vikash et al. have shown that images can be analyzed using compact probabilistic programs, e.g., for recognizing characters and detecting road lanes [6].

Picture is a probabilistic programming language specifically for image analysis introduced by Kulkarni et al. [10]. It has been applied to human pose estimation, face reconstruction, and 3D shape reconstruction. The authors rely on third-party renderers like *Blender* to generate images and compare them using a likelihood or distance function to the measured image. Using the differentiable renderer *OpenDR* allowed them to use advanced gradient-based samplers like Hamiltonian Monte Carlo (HMC) even though they note that it only "somewhat" speeds up inference [69]. Moreover, they use the idea of data-driven proposals from Jampani et al. to improve convergence [70]. Data-driven proposals mix samples from a global discriminative model like a CNN with the local proposals of the sampler.
General probabilistic programming languages like *Gen.jl* and *Turing.jl* have been developed since then, which allow formulating almost any problem as a generative model [8, 9]. They offer an even wider range of general inference algorithms,

although obtaining gradients for image-based inference still requires a third-party differentiable renderer.

Bayesian inference for 6D pose estimation using an iterated likelihood weighting has been proposed by Chen et al. [67]. They initialize the sampler using bounding box detections from a CNN to achieve robustness in adversarial scenarios. The slow runtime of ≈10 s per inference has been addressed in a follow-up work. By implementing the method on a field programmable gate array and a GPU, the inference times have been reduced to ≈1 second [71].
Gothoskar et al. estimate a full scene graph that contains multiple objects, whether the objects are stacked on each other, and the poses [72]. Their method uses the point cloud-based pose estimator "DenseFusion" from [48] to initialize their pose estimator. The estimated poses of their method are slightly better when inferring the scene graph than not. They indicate that their method is ≈20 x slower than their own DenseFusion baseline; the authors of DenseFusion report 16 Hz which results in ≈1.25 Hz.
Finally, Pavlasek et al. introduce belief propagation via message passing for the inference of poses of the articulated object [73]. It is inspired by human pose estimation and uses a CNN, which outputs a score for the object classes to evaluate the likelihood function.

3.7 Summary

Approaches for image-based pose inference vary with different input modalities like color, depth, or both. The variety of methods, their respective preprocessing steps, and outputs are much larger. This chapter first introduced classical approaches that use hand-crafted features and templates. Features engineered for color images are often not robust enough for adversarial conditions like challenging lighting. Purely geometric features for point clouds like the point pairs have proven to be very competitive for 6D pose estimation until now.

Learning-based methods promise to be more robust as they can internally learn feature representations that suit different conditions from the dataset. Deep learning methods have caught up to point pair features in 2020 and surpassed them in the 2022 BOP challenge [2]. While object coordinate predictors are currently the best-performing methods in the benchmark, discretizing methods trade off sampling accuracy for inference time and are typically faster than other methods. Therefore, these methods are also used in the refinement and tracking applications which Section 3.5 presents.

Bayesian inference has been applied to mobile localization since the late 1990s with the use of Kalman and particle filters. Later, these filters were also applied to track an object's 6D pose. Optimization-based tracking algorithms run faster at ≈790 Hz using only the CPU compared to particle filters at ≈90 Hz on the GPU [74]. However, optimization-based trackers are less intuitive to implement as they require a careful derivation of the gradients, Hessian, and are constrained to local minima [63]. Probabilistic methods also provide an intuitive way of incorporating additional sensors. Tracking using (deep) learning is relatively new and does not outperform other methods yet. If the trend in other image processing tasks can be extrapolated, deep learning might soon outperform other methods on the tracking task [2, 22].

However, Bayesian inference has rarely been applied to the pose estimation problem without temporal priors. The few practical implementations are informed by a CNN to guide samplers in the right direction in the light of the curse of dimensionality.

Many ideas for tracking an object's 6D pose with particle filters influenced this thesis. For example, explicitly modeling occlusions efficiently using the GPU; see Chapters 4 and 5. The idea of offloading implementations into a generic rendering and inference engine allows for rapid prototyping and comparing different models [6, 10].

4 Probabilistic Models for 3D Camera-Based 6D Pose Estimation

Formulating the probabilistic model is one essential part of Bayesian inference, besides developing and tuning the inference algorithms described in Chapter 5. It is possible to use existing probabilistic programming languages (PPL) like *Turing.jl* to write the model for 3D camera-based pose estimation. These PPLs are designed for general applicability. Therefore, computational optimizations like using a GPU can only be implemented with difficulty. Accordingly, the runtime of the algorithms is too slow and can, at most, enable a proof of concept.

This chapter explains the probabilistic top-down modeling approach, starting with an overview by intuitively describing it as a generative model for the data in Section 4.1. Afterward, Section 4.2 elaborates on modeling the translational and rotational components of the pose. Furthermore, Section 4.3 deals with modeling depth images generated by a calibrated 3D camera, the possibility of including pixel-object classifications, and how to deal with overconfident image likelihoods using regularization. Finally, Section 4.4 highlights performance considerations to enable inference times shorter than typical robot cycle times.

4.1 Probabilistic Model Overview

Hand-deriving a model for the posterior $p(\mathbf{x} \mid \mathbf{z})$ can be tedious and error-prone [13]. Often, an analytical solution does not even exist. An intuitive alternative that has gained popularity in probabilistic programming is to define a generative model $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})$ and use an inference algorithm from Chapter 5, to approximate the posterior (see Section 2.3).

Using domain knowledge, a prior can be defined for the 6D pose, which consists of the position $\mathbf{t}$ and orientation $\mathbf{R}$. The pose is defined in the camera frame

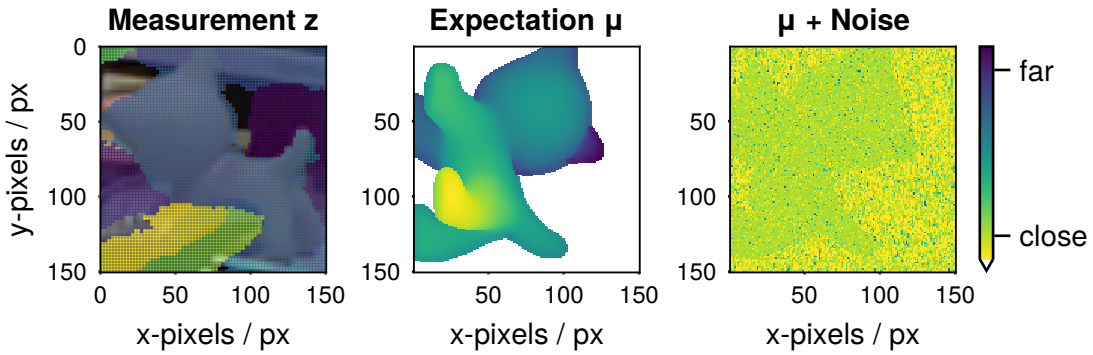


Figure 4.1: Left: measured depth overlaid onto the color image, middle: rendered image for a pose sampled from the prior, right: noise added from the likelihood model.

for the image-based pose estimation. Moreover, $\mathbf{t}$ and $\mathbf{R}$ are assumed to be independent:

$$p(\mathbf{x}) = p(\mathbf{t}, \mathbf{R}) = p(\mathbf{t})p(\mathbf{R}) \tag{4.1}$$

Specifically, a normal distribution parametrized by the prior position $\mathbf{t}_0$ and standard deviation $\boldsymbol{\sigma}_0$ can be used for the position $\mathbf{t}$ and a uniform distribution for Quaternions $\mathcal{U}_q$ for the orientation $\mathbf{R}$. More details follow in Sections 4.2.1 and 4.2.3.

$$\begin{aligned} \mathbf{t} &\sim \mathcal{N}(\mathbf{t}_0, \boldsymbol{\sigma}_{\mathbf{t}0}) \\ \mathbf{R} &\sim \mathcal{U}_q \end{aligned} \tag{4.2}$$

As the name *generative* model implies, it describes how to generate the observations given the latent state. Rather than determining the likelihood of an observation, the approach is reversed to model how an observation is generated from latent variables, such as the state $\mathbf{x}$. Generating a synthetic (depth) image from a scene description is called *rendering*. A scene description minimally includes an object's 3D model, the object's pose, and a parametrized camera with a pose. Rendering using a calibrated camera is described in detail in Sections 2.7 and 4.3.1. The result is a deterministic function that generates the expected depth image $\boldsymbol{\mu}_{img}$ for a given pose state $\mathbf{x} = [\mathbf{t}, \mathbf{R}]$ as visualized in Fig. 4.1. This image would result if the sensor measurements did not exhibit any noise.

Each camera pixel is assumed to be independent to generate images with sensor noise similar to the measured image $\mathbf{z}$ as visualized in Fig. 4.1. A mixture of the

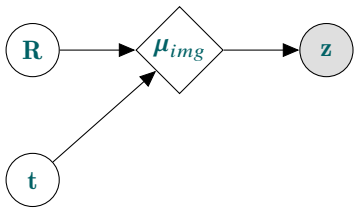


Figure 4.2: Bayesian network for the model of Eqs. (4.2) and (4.3) with the latent variables $\mathbf{R}$ and $\mathbf{t}$. The expected depth $\boldsymbol{\mu}_{img}$ results from deterministic rendering, and $\mathbf{z}$ is observed.

following distributions models three noise sources:

$$\begin{aligned} i &\sim \mathrm{Cat}(w_n, w_e, w_u) \\ (z \mid i = n) &\sim \mathcal{N}(\mu, \sigma_z) \\ (z \mid i = e) &\sim \mathrm{Exp}(\beta) \\ (z \mid i = u) &\sim \mathcal{U}(z_{min}, z_{max}) \end{aligned} \tag{4.3}$$

Sensor noise is modeled by a normal distribution centered at the expected depth μ with the standard deviation σ_z weighted by w_n. Occlusions are modeled using an exponential distribution with the scale parameter β and random outliers by a uniform distribution parametrized by the minimum and maximum ranges z_{min} and z_{max}. Section 4.3.3 discusses the pixel model choices and the results of generating noise from Eq. (4.3). The model from Section 4.1 can be compiled to the graphical representation shown in Fig. 4.2 as described in Section 2.4.

Besides a relatively simple model generating the noise, the independence assumption also simplifies the calculation of the image likelihood, which is the product of the pixel likelihoods over all N_{px} pixels:

$$p(\mathbf{z} \mid \boldsymbol{\mu}_{img}) = \prod_{i}^{N_{px}} p(z_i \mid \mu_i) \tag{4.4}$$

Naturally, this assumption is oversimplified as nearby pixels often correlate and share the same noise characteristics [12]. Examples are occluding objects or over-exposure due to bright lighting or reflective object surfaces. Occlusions are therefore modeled in Section 4.3.5 and Section 4.3.6 introduces regularization strategies to counter the over-confidence of the model.

4.2 Position and Orientation Models

Estimating a pose posterior involves estimating a position $\mathbf{t}$ and an orientation $\mathbf{R}$. This section introduces prior and proposal models for both components. Both priors require centering the object's origin coordinate frame inside the CAD model.

4.2.1 Position Priors

An informative prior is mandatory as the solution space would otherwise be infinitely large. Even if a region of interest can be defined, e.g., a tabletop, sampling the whole region to find an object is prohibitively expensive. This thesis considers two sources for the position prior, depending on the use case:

Mask priors from camera images: Deep learning-based object detectors can output bounding boxes and segmentation masks of different object instances. This scenario is considered in the BOP challenge and can be adapted to a wide range of use cases since only a color image is required. However, this approach might not scale well for thousands of similar object classes. For example, for surgical instruments, only subtle differences like a slightly longer blade might separate these classes.

Typically, object detectors output 2D bounding boxes in the image coordinates with the center point (u, v). Inverting the pinhole camera model from Section 2.7 allows us to reproject the image coordinates to 3D. Reprojecting the 2D coordinates requires that a depth value z is known. This depth can be extracted from the measured depth image. A single pixel in the center of the bounding box might have a high uncertainty, or might even belong to another occluding object. Subsequently, the median depth z is calculated over all measured pixel z_i, which are part of the segmentation mask $o_i = 1$ that is robust against outliers:

$$z = \operatorname{med}\{z_i \mid z_i \in \mathbf{z} \wedge o_i = 1\} \tag{4.5}$$

If no pixels are part of the mask, the depth value at the center of the bounding box can be used as a fallback solution. Only the front of an object is visible, so the z component is always biased towards the camera. Afterward, Eq. (2.23) can be used to calculate the missing x and y coordinates of the prior given the u and v coordinates in the image:

$$y = \frac{z}{f_y}(v - c_y) \tag{4.6}$$

$$x = \frac{z}{f_x}(u - c_x) \tag{4.7}$$

These estimates might also be biased depending on the object geometry or if parts are occluded. Each component of the position $(x, y, z)^\mathsf{T}$ is modeled using a normal distribution, which encodes local uncertainty. All components of the position prior share the same standard deviation σ_t:

$$\mathbf{t} \sim (\mathcal{N}(x, \sigma_t), \mathcal{N}(y, \sigma_t), \mathcal{N}(z, \sigma_t))^\mathsf{T} \tag{4.8}$$

This formulation neglects possible cross-correlations of the x, y, and z components, which might be possible to extract from the images. As segmentation masks do not contain semantic information about the visible object parts, a heuristic for the cross-correlations would be speculative. For this reason, the preferred solution is tuning a single parameter, the position's standard deviation σ_t; refer to Section 6.6.

Point priors from additional sensors: New generations of RFID sensors could allow directly triangulating the tag's position in addition to the object identifier. Moreover, a covariance matrix may be provided alongside it. However, these sensors are currently unavailable and part of ongoing research. Thus, the same prior from Eq. (4.8) is used, centered at the annotated ground truth position. A possible disadvantage of this approach is that it does not provide any prior segmentation masks for object instances.

4.2.2 Position Proposals

Proposal models suggest the next state to be evaluated in sampling algorithms. One possibility is to sample from the prior distribution, which is often regarded as a *global move*, since the prior's support must cover all possible realizations of the variables.

$$\mathbf{t}_{k+1} \sim p(\mathbf{t}_{k+1} \mid \mathbf{t}_k) = p(\mathbf{t}) \tag{4.9}$$

Sampling from the prior makes it hard for the algorithm to explore regions of higher probability. Alternatively, local moves like in the original version of the Metropolis-Hastings algorithm can be used. Usually, a normal distribution centered at the previous estimate $\mathbf{t}_k$ is used:

$$\mathbf{t}_{k+1} \sim \mathcal{N}(\mathbf{t}_k, \boldsymbol{\sigma}_{t+1}) \tag{4.10}$$

Local proposals allow the sampler to explore other regions if the standard deviation $\boldsymbol{\sigma}_{t+1}$ is sufficiently large. If $\boldsymbol{\sigma}_{t+1}$ is too large, the algorithm might fail to converge. In contrast, if $\boldsymbol{\sigma}_{t+1}$ is too small, the algorithm tends to refine a local optimum. Properly tuning $\boldsymbol{\sigma}_{t+1}$ is essential for a good tradeoff between exploration and exploitation. Algorithms like SMC have been developed to support adaptive proposals to avoid the tuning; see Section 5.1.3.

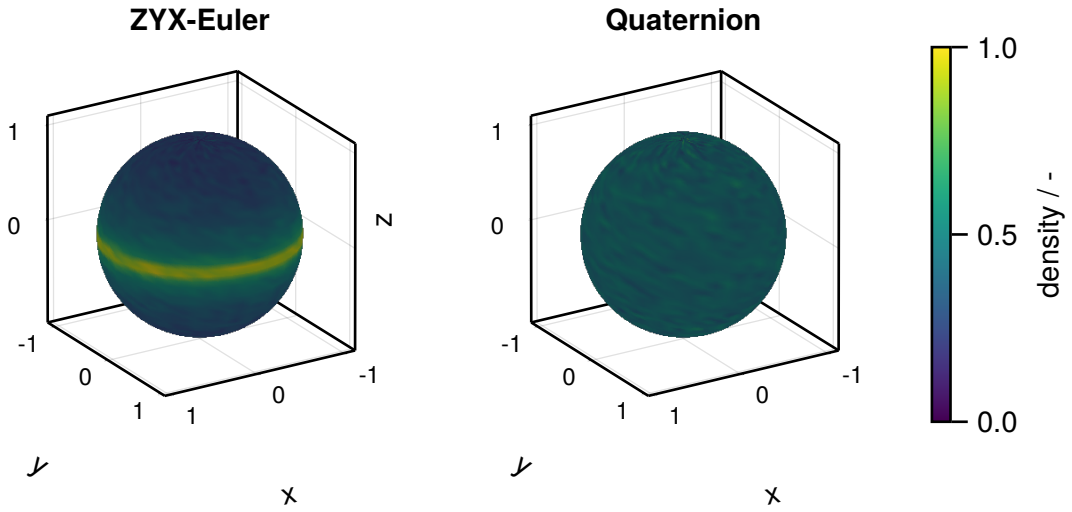


Figure 4.3: The density of a point $[0, 0, 1]^\mathsf{T}$ which is rotated using 500 000 randomly sampled rotations on the unit sphere. Brighter colors designate areas with a higher density.

4.2.3 Orientation Priors

While it is possible to extract approximate position priors from image-based object detectors, extracting an orientation from a bounding box or segmentation mask is not straightforward. Even if additional sensors like RFID tags are placed on the object, at least three sensors are required to determine a 3D orientation. A uniform distribution over SO(3) rotations describes the absence of prior knowledge.

However, it is not straightforward to sample rotations uniformly. For example, uniformly sampling Euler angles in ZYX order would result in a higher rotation density around the poles of the SO(3) manifold, as shown inf Fig. 4.3. In contrast, a uniform distribution $\mathcal{U}_\mathbf{q}$ for quaternions can be defined using standard normal distributions for each component q_* of the quaternion [75]:

$$\mathcal{U}_\mathbf{q} = \frac{(q_w \sim \mathcal{N}, q_x \sim \mathcal{N}, q_y \sim \mathcal{N}, q_z \sim \mathcal{N})^\mathsf{T}}{\left\|(q_w, q_x, q_y, q_z)^\mathsf{T}\right\|_2} \tag{4.11}$$

First, the four components of a quaternion $\mathbf{q} = (q_w, q_x, q_y, q_z)^\mathsf{T}$ are drawn individually from a normal distribution. Second, the randomly drawn quaternion must be normalized as rotations are represented by unit quaternions. Formally, the prior distribution of orientations can be written as follows:

$$p(\mathbf{R}) = \mathcal{U}_\mathbf{q} \tag{4.12}$$

4.2.4 Orientation Proposals - Quaternion Perturbations

Similar to the position proposals in Section 4.2.2, one could either sample global moves from the prior $p(\mathbf{R})$ or choose local moves $p(\mathbf{R}_{k+1} \mid \mathbf{R}_k)$ depending on the current state. Using global moves for the orientation is especially useful to escape local optima as the posterior distribution of the orientation is expected to be multimodal. Objects may look similar from different viewpoints or have the same appearance as symmetric objects. Moreover, fewer features are visible for heavily occluded objects, and multiple orientations might explain the visible features equally well.

In contrast to the position prior, which might focus on a biased location, uniformly sampled rotations are truly global. Still, local moves can be helpful to refine the estimates because it takes many samples to sufficiently cover the support of SO(3), i.e., all possible 3D rotations. Sola et al. describe local moves on SO(3) by defining a plus operator $\oplus$ which composes a reference quaternion $\mathbf{q}_{ref}$ with a typically small random rotation $\mathbf{\Phi} \in \mathbb{R}^3$ [15]:

$$\mathbf{q}_{new} = \mathbf{q}_{ref} \oplus \mathbf{\Phi} = \mathbf{q}_{ref} \otimes \mathrm{Exp}(\mathbf{\Phi}) \tag{4.13}$$

With $\otimes$ denoting the quaternion product and Exp the exponential map from the Lie algebra to the Lie group. The reader can refer to [15] for an extensive explanation of quaternions and Lie algebra. Intuitively, $\mathbf{\Phi}$ is defined in the velocity space using a discrete integral of the angular velocity $\omega \Delta t$. The exponential map can be interpreted as a tool to convert angular perturbations to quaternions and is readily implemented in programming libraries like *Quaternions.jl*.

Now, the local proposal model can be defined using a small local perturbation rotation $\Delta\mathbf{\Phi}$ sampled using a zero-centered normal distribution with a small standard deviation σ_{r+1}:

$$\Delta\mathbf{\Phi} \sim (\mathcal{N}(0, \sigma_{r+1}), \mathcal{N}(0, \sigma_{r+1}), \mathcal{N}(0, \sigma_{r+1})) \tag{4.14}$$

Using the quaternion perturbation from Eq. (4.13), the local proposal becomes:

$$\mathbf{q}_{k+1} = \mathbf{q}_k \oplus \Delta\mathbf{\Phi} \tag{4.15}$$

Sola et al. state that using small angle approximation for Eq. (4.15) should result in a shorter runtime. A significant advantage could however not be reproduced in benchmark experiments[1]. Finally, Eq. (4.15) can be interpreted as drawing a random rotation distributed around the previous rotation. This results in a description of the local proposal model similar to the one for the position, see Eq. (4.10):

$$\mathbf{R}_{k+1} \sim \mathcal{N}(\mu = \mathbf{R}_k, \boldsymbol{\sigma}_{r+1}) \tag{4.16}$$

[1] *Quaternions.jl* exponential map 1.461 ± 0.476 ns, small angle approximation 1.396 ± 0.840 ns

4.3 Depth Image Models

Depth images represent the observations in the probabilistic model and are used in the likelihood function to update the pose prior. This section describes generating depth images and formulating pixel-wise probabilistic models to consider the different sources of noise. Moreover, this section combines the individual pixel models in an image likelihood and introduces strategies for regularizing this image likelihood.

4.3.1 Generating Depth Images via Rendering

Rendering generates 2D depth images from a 3D scene, typically consisting of a calibrated camera (see Section 2.7), 3D mesh models, and lights. GPUs have been developed for rendering and offer general computing capabilities today. One can render images with high-level 3D render engines like Unity and Unreal Engine or low-level application programming interfaces (APIs) like *OpenGL* or *Vulkan*. While render engines offer much functionality out of the box, they usually target game development, with the goal of rendering a single color image to the screen. Parallel evaluations of multiple pose hypotheses for a single object are required to enable short inference times. Moreover, the communication between the CPU and GPU is slow. Ideally, multiple calculations are transferred from the CPU to the GPU at once. Consequently, this work uses an *OpenGL* pipeline, which is interoperable with the *NVIDIA CUDA* compute library for the GPU and allows avoiding memory transfers; see Section 4.4. This section sketches out the rendering pipeline. The detailed implementation is available in the accompanying code of *SciGL.jl*[2].

Since depth images are purely geometric, the rendered scene only contains the mesh of the object and a camera but no lights. Both scene elements have a pose measured from the world frame associated with them, ${}^{\mathrm{W}}\mathbf{P}^{\mathrm{O}}$ for the object and ${}^{\mathrm{W}}\mathbf{P}^{\mathrm{C}}$ for the camera. The first step is to transform each vertex ${}^{\mathrm{O}}\mathbf{v}_i$ of the object's mesh into the camera frame:

$$ {}^{\mathrm{C}}\mathbf{v}_i = {}^{\mathrm{C}}\mathbf{P}^{\mathrm{W}}\, {}^{\mathrm{W}}\mathbf{P}^{\mathrm{O}}\, {}^{\mathrm{O}}\mathbf{v}_i = \left({}^{\mathrm{W}}\mathbf{P}^{\mathrm{C}}\right)^{-1}\, {}^{\mathrm{W}}\mathbf{P}^{\mathrm{O}}\, {}^{\mathrm{O}}\mathbf{v}_i \tag{4.17} $$

In 3D graphics, ${}^{\mathrm{C}}\mathbf{P}^{\mathrm{W}}$ is commonly called the *view matrix* and ${}^{\mathrm{W}}\mathbf{P}^{\mathrm{O}}$ the *model matrix*. A caveat of *OpenGL* is that it uses a different convention than *OpenCV* or ROS, which are commonly used for calibrating and integrating cameras. Thus,

[2] SciGL.jl: https://github.com/rwth-irt/SciGL.jl

the respective rows for the y and z components of the homogenous coordinates of the view matrix have to be negated to convert the *OpenCV* view matrix $\mathbf{P}_{CV}^{\mathrm{W}}$ to the corresponding *OpenGL* matrix ${}^{\mathrm{C}}\mathbf{P}_{GL}^{\mathrm{W}}$:

$$ {}^{\mathrm{C}}\mathbf{P}_{GL}^{\mathrm{W}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} {}^{\mathrm{C}}\mathbf{P}_{CV}^{\mathrm{W}} \tag{4.18} $$

After this transformation, the model vertices are in the camera's view. *OpenGL* uses *normalized device coordinates* to define a clipping space by a cube ranging from -1 to $+1$ in x-, y-, and z-direction. All vertices outside this cube are discarded for rendering. To transform the vertices from the camera's view to the clipping space, a projection matrix can be constructed using a perspective $\mathbf{M}_{persp}$ and an orthographic projection $\mathbf{M}_{ortho}$:

$$ \mathbf{M}_{proj} = \mathbf{M}_{ortho}\mathbf{M}_{persp} \tag{4.19} $$

The matrices are left-multiplied to the vertices, so the perspective projection is applied first. Perspective projection describes the geometrical optics of the camera - in this case as a pinhole camera model from Eq. (2.23):

$$ \mathbf{M}_{persp} = \begin{pmatrix} f_x & -s & -c_x & 0 \\ 0 & -f_y & -c_y & 0 \\ 0 & 0 & z_{min} + z_{max} & z_{min} \cdot z_{max} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{4.20} $$

Compared to the *OpenCV* intrinsic camera matrix from Eq. (2.23), some components are negated to account for the different coordinate system conventions in *OpenGL*. If the intrinsic matrix has been modified, e.g., because of cropping, the respective entries at the location of the original intrinsics matrix can be used. Moreover, z_{min} and z_{max}, as well as a -1 entry in the last, are introduced to map the z-component to the range $[-z_{max}, -z_{min}]$. Choosing z_{min} and z_{max} determines the precision of the GPU's depth buffer, determining which vertices are in front of others. The precision of the depth values in the rendered image is not affected by the choice of z_{min} and z_{max} as they can be extracted in the *OpenGL* program before the projection. The fourth component of the homogenous coordinates is used for normalization by *OpenGL*.

After the perspective projection, the visible vertices are located in a cube, which has to be resized to $[-1, 1]$ by an orthographic projection:

$$\mathbf{M}_{ortho} = \begin{pmatrix} \frac{2}{x_{max}-x_{min}} & 0 & 0 & -\frac{x_{max}+x_{min}}{x_{max}-x_{min}} \\ 0 & -\frac{2}{y_{max}} & 0 & -\frac{y_{min}+y_{max}}{y_{min}-y_{max}} \\ 0 & 0 & -\frac{2}{z_{max}-z_{min}} & -\frac{z_{max}+z_{min}}{z_{max}-z_{min}} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.21}$$

Combining the previous equations leads to the following formulation to transform the model points for rendering in normalized device coordinates:

$${}^{\mathrm{NDC}}\mathbf{v}_i = \mathbf{M}_{proj}\, {}^{\mathrm{C}}\mathbf{P}^{\mathrm{W}}_{GL}\, {}^{\mathrm{W}}\mathbf{P}^{\mathrm{O}}\, {}^{\mathrm{O}}\mathbf{v}_i \tag{4.22}$$

If modeling the lens distortion is required, the reader can refer to [74].

4.3.2 Resizing and Cropping the Pinhole Camera Model

Resizing rasterized images not only reduces the number of computational operations, but also the information on the respective image. Cropping before resizing the image to a region of interest reduces the loss of information and lets the algorithms focus on the scene's relevant parts.

When resizing an image, the intrinsic parameters change as well. Intuitively, the new image pixels must point at the same object vertices as the old ones prior to the operation. Given the scaling factors $\gamma_x = {width_{new}}/{width_{old}}$ and $\gamma_y = {height_{new}}/{height_{old}}$, the rows of the intrinsic matrix $\mathbf{K}$ can be scaled via:

$$\mathrm{resize}(\mathbf{K}) = \begin{pmatrix} \gamma_x & 0 & 0 \\ 0 & \gamma_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{K} \tag{4.23}$$

Only a region of interest is kept when cropping an image. For this operation, only the principal point of the intrinsic matrix has to be adjusted by the new origin $(\hat{u}_0\,, \hat{v}_0)$:

$$\mathrm{crop}(\mathbf{K}) = \mathbf{K} - \begin{pmatrix} 0 & 0 & \hat{u}_0 \\ 0 & 0 & \hat{v}_0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} f_x & s & (c_x - \hat{u}_0) \\ 0 & f_y & (c_y - \hat{v}_0) \\ 0 & 0 & 1 \end{pmatrix} \tag{4.24}$$

The operations can be chained. For example, a crop followed by a resize operation results in the following camera matrix:

$$(\text{resize} \circ \text{crop})(\mathbf{K}) = \begin{pmatrix} \gamma_x f_x & \gamma_x s & \gamma_x (c_x - \hat{u}_0) \\ 0 & \gamma_y f_y & \gamma_y (c_y - \hat{v}_0) \\ 0 & 0 & 1 \end{pmatrix} \tag{4.25}$$

4.3.3 Pixel Likelihood

A beam-based model is commonly employed to model 3D sensors like laser scanners or depth cameras [12, 59]. Under the assumption of conditional independence given an expected depth image $\boldsymbol{\mu}_{img}$ for the state $\mathbf{x}$ as shown in Fig. 4.2, each pixel of a depth camera can be modeled individually as $p(z \mid \mu)$. The following model includes three causes of uncertainty for the individual beams: sensor noise, random outliers, and occlusions. As described by Thrun et al. and in Section 4.1, if each pixel is generated differently for each cause, the resulting model is a mixture of different probability distributions [12]:

$$\begin{aligned} i &\sim \text{Cat}(w_n, w_e, w_u) \\ (z \mid i = n) &\sim \mathcal{N}(\mu, \sigma_z) \\ (z \mid i = e) &\sim \text{Exp}(\beta) \\ (z \mid i = u) &\sim \mathcal{U}_{tail} \end{aligned} \tag{4.26}$$

The **sensor noise** is relevant if a depth pixel captures the object of interest. A normal distribution centered at the rendered expected depth μ models the sensor noise of the depth camera with a standard deviation of σ_z. Strictly speaking, σ_z grows quadratically for stereo and structured light cameras [74, 76]. However, a quadratic model contains more tunable parameters, and the region of interest usually covers only a small range of depth measurements. Therefore, this thesis neglects the effects of the measured distance on the noise and overestimates a constant σ_z as it is common practice in Bayesian filtering [12]. Overestimating the standard deviation also helps with inaccuracies in the 3D model. Moreover, this work omits truncating and normalizing the normal distribution to the measurable range as the standard deviation is much smaller than this range:

$$\sigma_z \ll z_{max} - z_{min}. \tag{4.27}$$

Stereo sensors often produce random **outliers** if the feature matching of the image pairs fails. Textureless objects and overexposed areas, e.g., on specular surfaces, are problematic. Usually, an invalid measurement returns a depth value

of zero, which cannot physically be measured, or random values in the range of the sensor. Differentiating between the causes of outliers would only change the likelihood function at the minimum and maximum measurable depth, so they do not change the likelihood function in the region of μ. Moreover, a non-zero term is crucial, as the image likelihood is the product of all pixel likelihoods; see Eq. (4.4). To cover the long tail of the measurements, an unnormalized **uniform** distribution $\mathcal{U}_{tail}$ assigns the same constant probability, even if outliers exceed the measurable range.

Occlusions are often modeled using an exponential distribution in the literature [12, 59, 74]. The idea is to cast a ray from the pixel to the object of interest and divide the ray into equidistant segments. If no information exists, whether an existing object is in a segment or not, all segments share the same probability that they could contain an occluding object. A segment is only visible if no previous segment contains an occluding object, so the probabilities must be multiplied. For the limit of infinitesimally small segments, this results in an **exponential** distribution. Another limit exists for only one segment: the constant probability of an **outlier** or, in other words, a **uniform** distribution. The latter case leads to the simplest version of the depth pixel model:

$$\begin{aligned} i &\sim \mathrm{Cat}(w_n, w_u) \\ (z \mid i = n) &\sim \mathcal{N}(\mu, \sigma_z) \\ (z \mid i = u) &\sim \mathcal{U}_{tail} \end{aligned} \tag{4.28}$$

Fig. 4.4 displays the resulting probability density as *only uniform* among other more complex occlusion formulations. Mixing the exponential and the normal distribution causes the mode at 1 m to skew slightly towards a shorter distance. However, the exponential distribution is scaled in the figure to emphasize the effect. In practice, values of $\sigma_z = 0.01$ m and equal weights $w_n = w_e = w_u$ are reasonable choices that reduce the effect.

As argued by Thrun et al. [12], occlusions can only occur in front of the object. Truncating the exponential distribution at the expected depth μ also involves recalculating the normalization constant, which is not negligible; compare *unmodified* and *truncated* in Fig. 4.4.

$$p_{\mathrm{Exp}}(z \mid \beta, z \leq \mu) = \frac{1}{\beta(1 - e^{-\frac{\mu}{\beta}})} e^{-\frac{z}{\beta}} \tag{4.29}$$

Here, $z \leq \mu$ denotes the truncation of the distribution. Naturally, the resulting density function has a discontinuity at μ, and after that, the *truncated* occlusion model has the same course as the *only uniform* one. The mode of the truncated exponential is also slightly skewed towards the shorter distance. Pixels with an

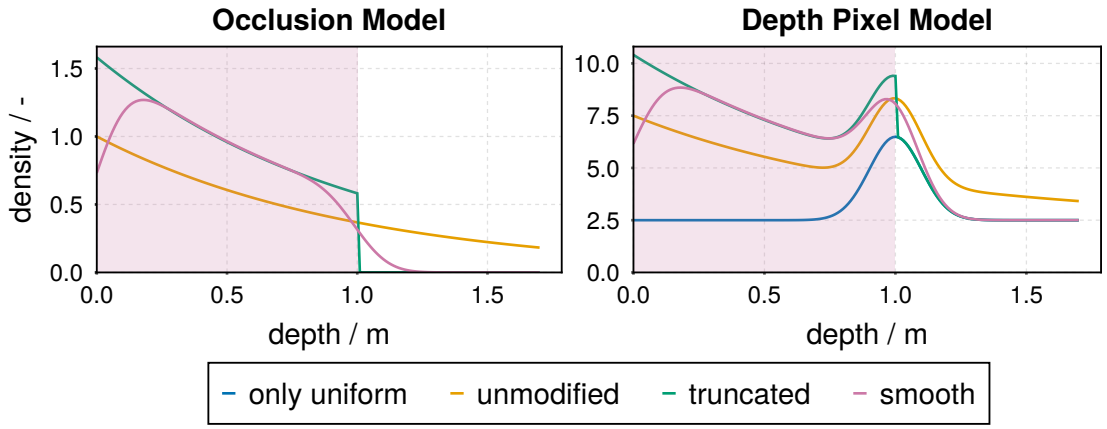


Figure 4.4: Left: plot of different occlusion models. Right: full depth pixel model, including the occlusion models as described in Eq. (4.26). Parameters: $\mu = 1\,\text{m}$, $\sigma_z = 0.1\,\text{m}$, $\beta = 1$, $z_{min} = 0.5\,\text{m}$, $z_{max} = 1.5\,\text{m}$. The exponential and uniform distributions are scaled by a factor of five for improved visualizations.

invalid rendered distance of $\mu = 0$ would cause a division by zero in Eq. (4.29). Section 4.3.4 explains handling these invalid depth values to consider them as outliers.

While the discontinuity poses no problem for the sampling-based inference algorithms used in this thesis per se, it might result in a more complex likelihood landscape with hard-to-escape minima. Instead of truncating the exponential distribution at the expected value, this thesis uses the idea that measurements of occluding objects are also subject to sensor noise. Introducing an auxiliary variable a that represents the true depth of an occluding object yields the following model of a *compound distribution* similar to [59]:

$$\begin{aligned} a &\sim \text{Exp}(\beta \mid a \leq \mu) \\ i &\sim \text{Cat}(w_u, w_n) \\ (z \mid i = u) &\sim \mathcal{U}(0, z_{max}) \\ (z \mid i = n) &\sim \mathcal{N}(a, \sigma_z) \end{aligned} \tag{4.30}$$

In this case, an occlusion might appear between the camera lens and the object of interest, yielding the right truncation of the exponential distribution to μ. The model in Eq. (4.30) is similar to the mixture model in Eq. (4.28) with the only difference that the normal distribution is centered at the depth of the occluding object a. To handle the newly introduced variable a, two choices arise:

1. Add the depth image containing the estimated depth a of every pixel to the state variables. Even for low resolutions of 30 px × 30 px, this results in 900 additional state variables.

2. Marginalize a from the model. An analytical solution exists as a and z are sampled from the exponential family of probability distributions.

Choice number one leads to an intractable inference problem, while choice number two involves solving the following convolution:

$$p_s(z \mid \mu) = \int p_{\mathcal{N}}(z \mid a, \mu) p_{\mathrm{Exp}}(a \mid \beta, a \leq \mu))\, da \tag{4.31}$$

Here, the truncated exponential distribution $p_{\mathrm{Exp}}(a \mid \beta, a \leq \mu)$ models the occlusion, and $p_{\mathcal{N}}(z \mid a, \mu)$ models the noise and random outliers the sensor produces. Inserting the respective probability density functions yields:

$$p_s(z \mid \mu) = \int \Bigg(\underbrace{\frac{w_u}{z_{max}}}_{\mathcal{U}} + \underbrace{w_n \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{-\frac{(z-a)^2}{2\sigma_z^2}}}_{\mathcal{N}} \Bigg) \underbrace{\mathbb{1}_{(0,\mu)}(a) \frac{e^{-\frac{a}{\beta}}}{\beta(1 - e^{-\frac{\mu}{\beta}})}}_{\text{truncated Exp}}\, da \tag{4.32}$$

Where $\mathbb{1}_{(0,\mu)}$ denotes the indicator function, which is one if $a \in (0, \mu)$ or zero otherwise. The reader can find details on this equation and the steps for solving the integral in Appendix A.3. The following equation presents the resulting density function:

$$p_s(z \mid \mu) = \underbrace{\frac{w_u}{z_{max}}}_{\mathcal{U}} + \underbrace{w_n \frac{e^{-\frac{z}{\beta} + \frac{1}{2}\left(\frac{\sigma_z}{\beta}\right)^2}}{2\beta(1 - e^{-\frac{\mu}{\beta}})}}_{\mathrm{Exp}} \left[\mathrm{erf}\left(\frac{\mu + \frac{\sigma_z^2}{\beta} - z}{\sqrt{2\sigma_z^2}} \right) - \mathrm{erf}\left(\frac{\frac{\sigma_z^2}{\beta} - z}{\sqrt{2\sigma_z^2}} \right) \right] \tag{4.33}$$

Conveniently, the uniform distribution for the outliers yields a uniform distribution after the integration, so the mixture distribution from Eq. (4.26) can still be used. However, the second term replaces the unmodified exponential distribution consisting of a shifted and normalized exponential distribution and two error functions, denoted by erf. The error functions replace the truncated exponential distribution's lower and upper bound with smooth limits according to the normal distribution's standard deviation σ_z; displayed as *smooth* in Fig. 4.4. Compared to the *unmodified* and *truncated* occlusion models, the *smooth* version is skewed even further towards closer distances. Section 4.3.5 shows that the skewness depends on the weights associated with the exponential and normal distributions.

4.3.4 Handling Invalid Pixels

Pixels in the rendered image not showing the object have a value of $\mu = 0$ and do supposedly not add any information for the inference. Moreover, $\mu = 0$ can

cause a division by zero in the truncated distributions in Eqs. (4.29) and (4.33), which must be avoided. Naturally, pixels with an invalid depth of zero should be considered outliers, meaning only the uniform distribution should contribute to the mixture.

One possibility is **ignoring** pixels with $\mu = 0$ because they do not contain any information and always assign a likelihood of one. Nevertheless, this introduces more complexity, leading to even more complexity in the regularization in Section 4.3.6.

Introducing a **preprocessing** for the measurement $\mathbf{z}$ is a more straightforward approach that does not require fiddling with the regularization. This step replaces all zero-valued pixels of $\mathbf{z}$ with positive infinity. Subsequently, the mixture Eq. (4.26) behaves as desired because the (truncated) exponential and the normal distribution have a zero density for infinity measurements, avoiding the division by zero. Therefore, invalid measurements are considered outliers; only the uniform distribution contributes to the pixel likelihood.

The *preprocessing* does not alter the model's behavior for pixels with a measurement $z \neq 0$ and a rendered depth of $\mu = 0$. Evaluating the densities of all distributions from the mixtures with such a measurement results in:

- Normal $\mathcal{N}$: As the standard deviation is much smaller than the typically measured distance $\sigma_z \ll z$, the density is almost zero[3]. Consequently, the normal distribution does not contribute to the mixture
- Uniform $\mathcal{U}$: This distribution covers the long tail of measurements and always returns the same value.
- (Smoothly) truncated exponential $\mathrm{Exp}(\beta \mid z \leq 0)$: Truncating to $[0, 0]$ means that any measurement $z \neq 0$ is outside the support of the distribution and results in a zero probability.
- Unmodified Exponential Exp: This distribution still gets evaluated and returns different probabilities for different values of z.

In conclusion, only the uniform distributions and the unmodified exponential contribute to the mixture. As intended, the uniform distribution always contributes the same uninformed probability. Only the unmodified exponential distribution returns varying values depending on the measured depth, which might lead to undesired behaviors. Using a truncated version might be preferable, and Section 6.5.2 evaluates this hypothesis.

[3] Often equal to zero due to floating-point numerics.

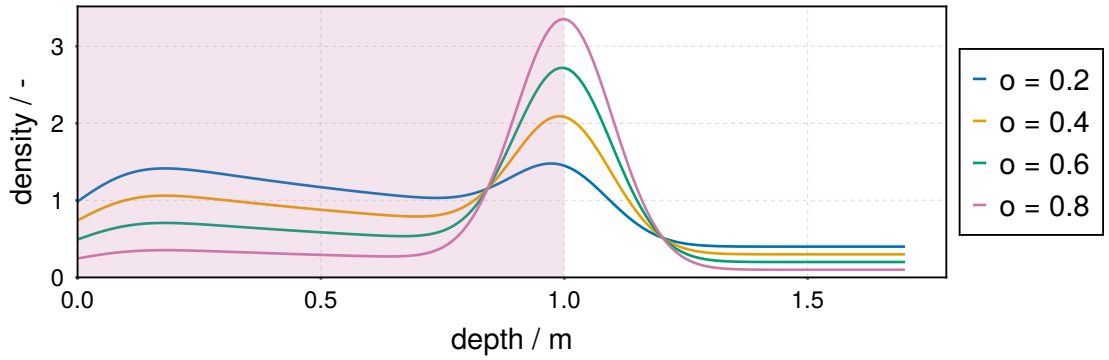


Figure 4.5: Probability density of the likelihood model in Eq. (4.34) for a smooth truncated exponential distribution and varying object classification probabilities o. Parameters: $\mu = 1\,\text{m}$, $\sigma_z = 0.1\,\text{m}$, $\beta = 1$, $z_{min} = 0.5\,\text{m}$, $z_{max} = 1.5\,\text{m}$.

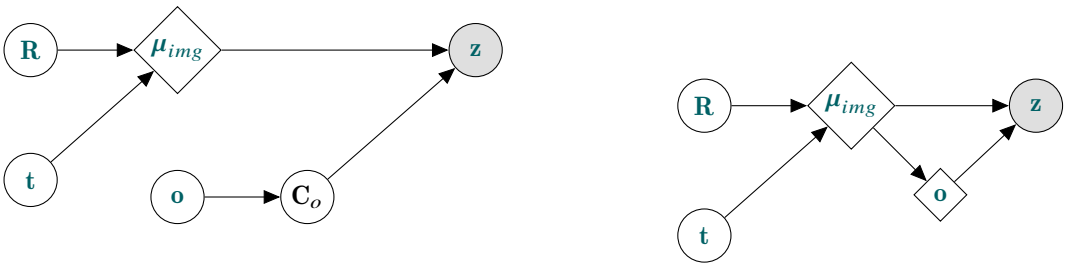
4.3.5 Incorporating Masks: Pixel-Object Classification

Pixel-object classifications are typically the output of machine-learning models called segmentation masks. Other methods use these masks to cut out the objects from the image, reducing distractions with the danger of discarding information if the segmentation masks are inaccurate. Including the classification of pixel-object classifications in the probabilistic model allows updating these masks as required and using them softly as the weights of the mixture model in Eq. (4.26). Moreover, explicitly including the pixel-to-object classification in the model might remove the requirement of prior segmentation masks.

Wüthrich et al. explicitly model whether an object is occluded in their particle filter formulation [59]. Similarly, this work extends pixel model with a categorical variable $c_o \in [0, 1]$, estimating the probability of a pixel showing the object of interest. Using the probability o as an adaptive weight allows shape the pixel likelihood from Eq. (4.3) to the densities shown in Fig. 4.5:

$$\begin{aligned} c_o &\sim \text{Bern}(o) \\ (z \mid c_o = 1) &\sim \mathcal{N}(\mu, \sigma_z) \\ (z \mid c_o = 0) &\sim \text{Mix}(\text{Exp}(\beta), \mathcal{U}(z_{min}, z_{max}); w_e, w_u) \end{aligned} \tag{4.34}$$

Classifying a pixel as the object ($c_o = 1$) or not ($c_o = 0$) has a binary outcome modeled using a Bernoulli distribution parametrized by the classification probability o. A natural choice for the prior o is the beta distribution as the conjugate prior of the Bernoulli distribution. If objects are always placed in a similar location, the beta distribution parameters could be calculated analytically from a labeled dataset. Otherwise, the beta distribution reduces to a uniform distribution without any prior data. For high probabilities of o, the normal distribution dominates the



(a) additional variables **(b)** marginalizing $\mathbf{C}_o$

Figure 4.6: Bayesian networks for including the pixel classification.

density, and the skewness of the mode towards closer distances disappears. In contrast, low probabilities increase the influence of the exponential and uniform distributions. Hence, this approach should make the likelihood model more robust to outliers and occlusions.

Figure 4.6a depicts the Bayesian network for Eq. (4.34) with $\mathbf{o}$ as an additional variable from which the classification labels $\mathbf{C}_o$ are drawn. Again, introducing a new variable per pixel would result in an intractable problem because of the *curse of dimensionality*.

Another alternative is to use deep learning-based methods like *YOLACT++* to predict a segmentation image $\mathbf{C}_o = f(\mathbf{z})$. Segmentation neural networks usually use a softmax or sigmoid function to assign class confidences to the pixels. These confidences are, strictly speaking, no probabilities but could be used as an estimate for $\mathbf{o} = p(\mathbf{C}_o \mid \mathbf{z})$ [77]. Usually, the confidences are compared to a threshold to generate binary classification labels that can be used for $\mathbf{C}_o$. Both approaches exclude $\mathbf{o}$ from the inference problem, reducing the number of variables. On the downside, removing the variables from the inference problem prevents updating them if the neural network is wrong. The training data for both approaches could be generated automatically by a physical simulator using *BlenderProc* [4].

Lastly, it is possible to marginalize the two states of the binary variable c_o to infer the classification probability o for each pixel analytically. This approach is called *collapsed Gibbs sampling*, and the resulting model is presented in Fig. 4.6b. Although nearby pixels likely share the same classification in practice, assuming that the pixels are independent simplifies the derivation. Using Bayes' law results

in the following formulation:

$$\begin{aligned} o &= p(c_o \mid z, \mu) \\ &= \frac{p(z \mid \mu, c_o)p(c_o \mid \mu)\cancel{p(\mu)}}{p(z \mid \mu)\cancel{p(\mu)}} \\ &= \frac{p(z \mid \mu, c_o)p(c_o \mid \mu)}{\int p(z \mid \mu, c_o)p(c_o \mid \mu)\, dc_o} \end{aligned} \tag{4.35}$$

From Fig. 4.6a, it follows that z and c_o are independent causes of the measured depth z and are therefore independent as long as z is not involved. Thus, $p(c_o \mid \mu)$ simplifies to $p(c_o)$. Since $c_o = 1$, $\overline{c_o} = 0$ are discrete, the integral in the denominator turns into a sum:

$$o = \frac{p(z \mid \mu, c_o)p(c_o)}{p(z \mid \mu, c_o)p(c_o) + p(z \mid \mu, \overline{c_o})p(\overline{c_o})} \tag{4.36}$$

The prior for $p(c_o)$ is a single number, and from the Bernoulli distributed c_o follows that $p(\overline{c_o}) = 1 - p(c_o)$. Without prior knowledge, $p(c_o)$ can be set to 0.5, representing that a pixel has the same chance of being the object or anything else. If a learning-based method outputs a binary image, one should replace the ones and zeros with values $p(c_o) \in (0, 1)$. Otherwise, either $p(c_o)$ or $p(\overline{c_o})$ is zero, and the likelihood cannot update the prior as terms get removed from Eq. (4.36). For example, $p(c_o) = 0.7$ is a reasonable choice to express that the pixel probably contains the object.

Two corner cases have to be considered: The first corner case occurs if the rendered object does not occupy a pixel. In this case, no information is available to update the prior. The second corner case occurs if the prior is $p(c_o) = 0$ or $p(c_o) = 1$ which leads to $p(\overline{c_o}) = 0$. In this case, one term of the denominator from Eq. (4.36) disappears. Thus, the observation does not update the prior in the best case, or a division by zero occurs in the worst case. Both corner cases are solved by directly returning the prior instead of calculating the fraction.

4.3.6 Image Likelihood and Regularization

Assuming the pixels of an image to be independent allows using relatively simple models for the individual pixels and factorizing the joint probability in Eq. (4.4). This assumption might not hold since the measurements are often strongly correlated in a local region [12]. For example, the depth value only changes slightly from one pixel to the next if they are on the same smooth surface of an object. This correlation leads to an overconfident likelihood, since a single image might not provide as much information as the independent pixels might suggest. In practice,

the strongly peaked likelihood functions lead to a degeneration of the estimated posterior, and sampling algorithms frequently get stuck in local minima.

Regularization is a common strategy to prevent overconfidence in such ill-posed problems. It aims to modify the target function for a better generalization, typically by shifting the objective from the likelihood to the prior. A common practice to regularize the likelihood of laser scanners, which suffer from the same problems as depth cameras, is to incorporate only a subset of the beams [12]. Subsampling reduces local correlations and overconfidence due to too many highly accurate and correlated measurements. Subsampling can also be interpreted as a form of likelihood tempering [78]. When estimating the pose of small objects like surgical instruments, this approach might lead to missing the pixels showing the object altogether. Moreover, subsampling does not lend itself to GPU accelerations since GPUs work best on dense arrays.

Another relatively simple regularization strategy that works for the dense image representation is multiplying a factor smaller than one by the likelihood. It can be interpreted as reducing the information value of the measurements [12]. This work extends this regularization strategy by introducing N to normalize the image likelihood. Defining N is more straightforward in the logarithmic domain, which (Section 5.2.3) introduces as a best practice for sampling. For readability, the following equation repeats the result of the derived image log-likelihood Eq. (5.24):

$$\ln p(\mathbf{z} \mid \boldsymbol{\mu}_{img}) = \sum_{i}^{N_{px}} \ln p(z_i \mid \mu_i)$$

The number of elements in the sum grows linearly with the number of pixels N_{px} in the image. While the individual pixel log-likelihoods are still non-linear in z and μ, the image log-likelihood should still grow relatively linearly with N_{px} when resizing the image. Therefore, the first version of the regularization uses N_{px} to make the image likelihood independent of the resolution. It is deemed L_{px} *pixel count regularization*:

$$L_{px} \ln p(\mathbf{z} \mid \boldsymbol{\mu}_{img}) = \frac{c_{reg}}{N_{px}} \sum_{i}^{N_{px}} \ln p(z_i \mid \mu_i) \tag{4.37}$$

Due to the division by N_{px}, the regularization constant c_{reg} might be larger than one in contrast to the original proposal in [12]. In analogy to the subsampling regularization, c_{reg} can be interpreted as the number of pixels considered in the likelihood. The constant c_{reg} is a tunable parameter that weighs the prior against the likelihood and changes with the standard deviations of the pose prior and the camera's sensor noise.

Besides differing image resolutions, the object's geometry can lead to different values of the image likelihood. For example, the number of object pixels in an image is smaller for slim surgical instruments than for large box-shaped objects. Moreover, the number of object pixels changes with different views of the same object. Modeling pixels that do not display the object as outliers with a uniform distribution as described in Section 4.3.4 expresses that they do not provide any information for the inference. Therefore, a more advanced regularization should consider the number of visible object pixels. Using the estimates of the object classification-probability o from Section 4.3.5, the normalization constant is defined as:

$$N_0 = \sum_i^{N_{px}} o_i \tag{4.38}$$

Summing the pixel-object classification probabilities can be interpreted as a soft version of summing the number of object pixels. Using a threshold like $o_i > 0.5$ to determine whether a pixel shows the object results in discretization errors. These errors might cause unnecessary particle deprivation, resampling, and higher discretization errors of the pose estimates.

Using this condition is similar to the L_0 class regularization in machine learning, where the number of non-zero parameters, the l^0 norm, is used as a cost to encourage sparseness. The inverse is used here, and the normalization factor emulates sparse measurements. Combining the normalization with the log-likelihood from Eq. (5.24) leads to the L_0 *class regularization*:

$$L_0 \ln p(\mathbf{z} \mid \boldsymbol{\mu}_{img}) = \frac{c_{reg}}{N_o} \sum_i^{N_{px}} \ln p(z_i \mid \mu_i) \tag{4.39}$$

Other approaches for defining the normalization constant N have been tried but failed and are briefly described in Appendix A.2.

4.4 Performant Evaluation using Julia and the Graphics Processing Unit

Efficient parallelization of the graphical model on the GPU is achieved through its implementation[4] in *Julia* [79]. Using a naive implementation of the models on a CPU resulted in inference times of 5 min per pose, which is prohibitively slow.

[4] BayesianPoseEstimation.jl: https://github.com/rwth-irt/BayesianPoseEstimation.jl

With the optimizations proposed in this section, the inference time can be reduced to ≈0.5 s, which enables experiments on a larger scale; compare for Section 6.3. This section discusses implementation details, which are essential for efficient parallelization. A unique feature of *Julia* as a programming language is its ability to run identical code on both the CPU and GPU. This functionality is supported by *CUDA.jl* packages' compatibility with *Julia*'s broadcasting syntax, which allows operations on arrays and scalars of different dimensions.

Two major requirements exist to parallelize array operations on the GPU using *CUDA.jl*:

1. *Scalar indexing* is disallowed: the same operation should be repeated on all elements of an array and not on specific indices. It is possible to operate on so-called *view*s of an array or *sub-arrays*.
2. Functions must be *type-stable*; in other words, the types of all variables can be inferred, so the function can be compiled statically.

Existing packages for probability distributions do not satisfy these requirements, so a slimmed-down version of *Distributions.jl* has been implemented, which supports existing interfaces for generating random values and evaluating the log-density[5]. Additional distribution types have been implemented to support uniform sampling of random rotations, a uniform distribution for the long tail, and a smooth truncated exponential distribution, used throughout Chapter 4. At the core of evaluating the likelihood of multiple rendered images $\boldsymbol{\mu}_{img}$ in parallel is a new implementation of a product distribution, deemed `BroadcastedDistribution`. It resembles an array of probability distributions of the same type with possibly different parameters. Compared to other implementations, the `BroadcastedDistribution` allows supplying the dimensions that are used to accumulate the probability densities. Specifically, this allows additional dimensions to encode different particles in a sampling algorithm and spawn only a single GPU kernel, as spawning a kernel always comes with some communication overhead.

One of the most important performance considerations is avoiding memory transfers from the CPU to the GPU and vice versa. *CUDA*s' graphics-interoperability allows mapping graphics memory to arrays without copying. After rendering depth images using the *OpenGL* pipeline described in Section 4.3.1 to a 3D texture, the same data is directly accessible in *CUDA*. The 3D texture is beneficial in conjunction with the `BroadcastedDistribution` mentioned above, as the third dimension allows rendering multiple poses into one texture and evaluating the

[5] KernelDistributions.jl: https://github.com/rwth-irt/KernelDistributions.jl

likelihood separately for each pose. Details on how to correctly map *OpenGL* to *CUDA* can be found in the source code[6].

The last part is a Bayesian network package which automatically keeps track of the dimensions for each variable and uses the broadcasted distribution internally[7]. Sequentialization is the process of converting a directed acyclic graph into a sequential list. The ordering of this list guarantees that all predecessor nodes have been evaluated before evaluating a node that depends on them. This process is called topological sorting and needs to be executed only once instead of running the depth-first search on every graph evaluation. Another benefit of using sequentialization in the *Julia* implementation is that looping over the list of nodes results in type-stable code, while the recursions do not. Thus, the latter requires an interpreted execution while the former compiles into efficient machine code.

4.5 Summary

This chapter's models lay the foundation for using the Bayesian inference algorithms described in the upcoming chapter. One part of the model is the description of the pose. Two alternatives can be used as sources for the prior information of the position component. Image mask priors from machine learning models enable broad applicability, since only images from the depth camera are required. Conversely, point priors from additional sensors can enable specific applications like the pose estimation of surgical instruments. In the context of this thesis, no prior information is available for the object's orientation. Thus, the prior of the orientations is formulated as a uniform distribution for rotations based on quaternions. Global proposals are sampled from the prior, while normal distributions are used for local moves of the position and orientation. Applying the concepts of a normal distribution to rotations requires special care.

Formulating the likelihood of the measured depth images consists of two steps: First, the generative process of depth images is described as 3D rendering. Second, the noise is added to the depth images using a per-pixel mixture model, which models measuring the object of interest, occlusions, and outliers. A smooth truncated exponential distribution has been formulated to avoid discontinuities while modeling occlusions more accurately than an unmodified exponential distribution. Explicitly modeling the pixel-to-object classification probabilities has been

[6] SciGL.jl: https://github.com/rwth-irt/SciGL.jl
[7] BayesNet.jl: https://github.com/rwth-irt/BayesNet.jl

proposed to let the algorithms focus on the relevant pixels. A closed-form solution enables the use of these probabilities during inference, as introducing hundreds of pixel-wise probabilities would lead to an intractable high dimensionality of the problem. Finally, the pixel-wise models lead to overconfident likelihoods. Thus, regularization strategies have been proposed based on the number of pixels in an image or the pixel classifications. Crucial performance considerations have been explained in this context to evaluate the models on a GPU.

5 Approximate Bayesian Inference Algorithms

Analytical solutions of the posterior are not possible if no closed-form solution exists for the evidence integral in Eq. (2.15). Approximate inference algorithms can broadly be categorized into sampling-based and optimization-based methods. Most inference algorithms are generic and can be used to estimate a posterior distribution with the models described in the previous chapter. However, some algorithms have special requirements, such as requiring a differentiable model for gradients as in HMC. This chapter gives an overview of different inference algorithms and selects appropriate algorithms and their modifications.

5.1 Overview of Approximate Inference Algorithms

This section presents an overview of Monte Carlo sampling and importance sampling methods. Moreover, it presents variational inference as a versatile instance of optimization-based techniques. Finally, this section highlights some practical considerations and requirements for choosing sampling-based algorithms for 6D pose estimation.

Monte Carlo methods approximate the posterior by drawing discrete values from it instead of specifying a continuous function. Two major approaches exist to approximate a distribution via discretization: Monte Carlo sampling and importance sampling. Drawing a *sample* from a distribution includes realizing all state variables $\mathbf{x}$. Since sampling directly from the target $p(\mathbf{x} \mid \mathbf{z})$ is not possible in most cases, the samples are drawn sequentially from another proposal distribution $q(\mathbf{x})$ and evaluated up to a constant on the probabilistic model $p(\mathbf{x}, \mathbf{z})$. After collecting N samples, integrating over all states can be approximated by summing over the samples. For example, the expected value of a function $g(\mathbf{x})$ can be calculated via:

$$\mathbb{E}_{p(\mathbf{x},\mathbf{z})}\left(g(\mathbf{x})\right) = \int p(\mathbf{x}, \mathbf{z})\, g(\mathbf{x})\, d\mathbf{x} \approx \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{x}_i) \tag{5.1}$$

An interpretation of this approximation is that regions with many samples have a higher probability density than regions with fewer samples. Choosing $g(\mathbf{x}) = \mathbf{x}$ results in the expected value of the state. To eliminate the discrepancy between the proposal and target distribution, Monte Carlo methods reject samples that do not represent the target distribution. Many Monte Carlo samplers draw samples sequentially, and Section 5.1.1 introduces the *Metropolis-Hastings* acceptance-ratio, which compares the previous and the current sample to decide which one to keep.

Importance sampling uses a cloud of samples, called particles, to approximate a target distribution [11]. It makes use of the following approximation:

$$\mathbb{E}_{p(\mathbf{x},\mathbf{z})}\left(g(\mathbf{x})\right) = \int p(\mathbf{x},\mathbf{z})\, g(\mathbf{x})\, d\mathbf{x} \approx \sum_{i=1}^{N} \frac{p(\mathbf{x}_i,\mathbf{z})}{q(\mathbf{x}_i)} g(\mathbf{x}_i) = \sum_{i=1}^{N} w^{(i)} g(\mathbf{x}_i) \tag{5.2}$$

Again, as it usually is not easy to sample from $p(\mathbf{x},\mathbf{z})$, the samples are drawn from another easy-to-sample proposal distribution $q(\mathbf{x})$. Instead of accepting or rejecting samples, the discrepancy between p and q is resolved by introducing the ratios $w^{(i)} = {}^{p(\mathbf{x}_i,\mathbf{z})}/_{q(\mathbf{x}_i)}$, which are called the *importance weights*. A technique called *resampling* connects importance sampling to Monte Carlo sampling. Resampling draws N particles with replacement and a probability according to their weights. These samples closely but not fully match the approximation from Eq. (5.1).

Variational inference (VI) turns the inference problem into an optimization problem to minimize the Kullback-Leibler-divergency $\mathrm{KL}(p\|q)$, which is "a measure of dissimilarity of two distributions $p(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\mathbf{x})$" [11]. Specifically, the goal is to find a good approximation $q_{\boldsymbol{\theta}}(\mathbf{x})$ for the posterior distribution $p(\mathbf{x} \mid \mathbf{z})$ given a specific set of observations z. Compared to the posterior, this approximation should be easy to sample from and evaluate. These constraints on the choice of $q_{\boldsymbol{\theta}}(\mathbf{x})$ lead to the approximate nature of VI. It can be shown that this optimization goal is equivalent to maximizing the lower bound of the model evidence $p(\mathbf{z})$, commonly called *evidence lower bound*.

$$\ln p(\mathbf{z}) \geq \mathrm{ELBO}(\boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\mathbf{x})}\left[\ln p(\mathbf{x},\mathbf{z}) - \ln q_{\boldsymbol{\theta}}(\mathbf{x})\right] \tag{5.3}$$

In this equation, $\ln p(\mathbf{x},\mathbf{z})$ is the expectation of the generative probabilistic model, and $\ln q_{\boldsymbol{\theta}}(\mathbf{x})$ is the entropy of the approximating distribution. Historically, variational inference required hand-deriving approximate distributions and the optimization scheme. *Automatic differentiation variational inference* simplifies this process by providing gradients $\nabla_{\boldsymbol{\theta}}\mathrm{ELBO}$ for efficient optimization algorithms [80]. Still, calculating the model's expectation Eq. (5.3) typically lacks a closed-form solution. Therefore, *automatic differentiation variational inference* approximates the integral

via Monte Carlo sampling and requires only a single sample in practice. Hence, $p(\mathbf{x}, \mathbf{z})$ and $q_\theta(\mathbf{x})$ must be differentiable and easy to sample from. Black box VI is an alternative that does not require analytical gradients of the probabilistic model. Instead, it uses stochastic optimization with gradients estimated by falling back to Monte Carlo sampling [81].

Variational inference is classically used for problems where much data is available since it is usually faster than Monte Carlo methods. However, it has the disadvantage of generally underestimating the variance of the posterior. Monte Carlo methods are often preferred for smaller and more expensive to collect datasets [82]. The variance is unimportant if only a single pose is used for benchmarking, as described in Section 6.1.4. Still, a single image can be considered a tiny dataset, and the results of deriving wrong decisions from it can be costly in robotic applications; see Section 7.3.

Multimodal posteriors are likely to occur in pose estimation problems and pose a challenge for MCMC methods and VI algorithms as they tend to get stuck in a single mode [83, 84]. In practice, the algorithms can run multiple times with different initializations to increase the chances of covering multiple modes. Importance sampling-based algorithms handle multiple modes well, but run into problems with high-dimensional problems [11].

Another perspective is the practicality of implementing the inference algorithms. A major bottleneck is that classical rendering using rasterization is not differentiable. Differentiable rendering is ongoing research. In preliminary experiments, differentiable rendering has been slow and unable to produce usable gradients; see Appendix A.2.3. The rendering step currently prevents using modern tools such as automatic differentiation VI and HMC, which use gradients for improved performance. Vanilla versions of sampling-based inference algorithms do not require gradients, are relatively easy to implement, and work with almost any model. The applicability of the algorithms is the primary reason this work focuses on sample- and particle-based algorithms.

5.1.1 Metropolis-Hastings

The Metropolis-Hastings (MH) algorithm generates a Markov chain of samples to approximate the target distribution via Eq. (5.1). The Markov property of a chain is that every new sample $\mathbf{x}_{t+1}$ only depends on the previous sample $\mathbf{x}_t$ via a proposal model $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$, as shown in Fig. 5.1. At its core is the *acceptance ratio* for two consecutive samples, which is the basis of many advanced sampling

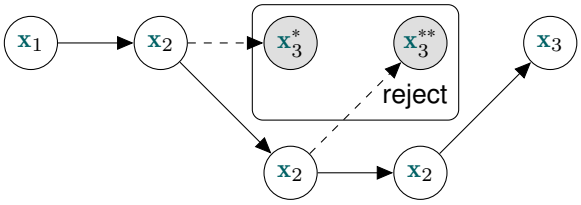


Figure 5.1: Drawing samples as a Markov chain using an accept/reject step. The gray samples are rejected, and $\mathbf{x}_2$ is reused in the chain.

algorithms and is defined as:

$$\alpha(\mathbf{x}_t, \mathbf{x}_{t+1}) = \min\left[1, \frac{p(\mathbf{x}_{t+1}, \mathbf{z})q(\mathbf{x}_t \mid \mathbf{x}_{t+1})}{p(\mathbf{x}_t, \mathbf{z})q(\mathbf{x}_{t+1} \mid \mathbf{x}_t)}\right] \tag{5.4}$$

For symmetric proposals, i.e., normal distributions centered at the sample, this equation simplifies to:

$$\alpha(\mathbf{x}_t, \mathbf{x}_{t+1}) = \min\left[1, \frac{p(\mathbf{x}_{t+1}, \mathbf{z})}{p(\mathbf{x}_t, \mathbf{z})}\right] \tag{5.5}$$

The acceptance ratio represents the probability of accepting the proposed sample. If the new sample is not accepted, the previous sample is reused, i.e., $\mathbf{x}_2$ is reused in Fig. 5.1. In the symmetric case of Eq. (5.5), if the proposed sample has a higher probability according to the probabilistic model $p(\mathbf{x}, \mathbf{z})$ than the previous one, it is always accepted with a probability of one. Otherwise, a less likely proposed sample yields a proportionally smaller probability of acceptance. If the proposal distribution is not symmetric as in Eq. (5.4), the probability q of proposing the sample is also considered. As the probability of transitioning to the new state is in the denominator, it favors states that are less likely to transition to. From a theoretical background, the acceptance ratio has been designed to draw samples from the stationary target distribution [85].

5.1.2 Multiple-Try Metropolis

Evaluating multiple poses in parallel can improve the performance of the GPU-based parallelization by reducing the number of computation kernels, as written in Section 4.4. The previously described MH algorithm evaluates only one pose per step. Multiple-try Metropolis (MTM) computes a chain of samples to approximate the target distribution, similar to MCMC. However, MTM allows evaluating multiple hypotheses in a single step by borrowing from the idea of importance sampling [86, 87].

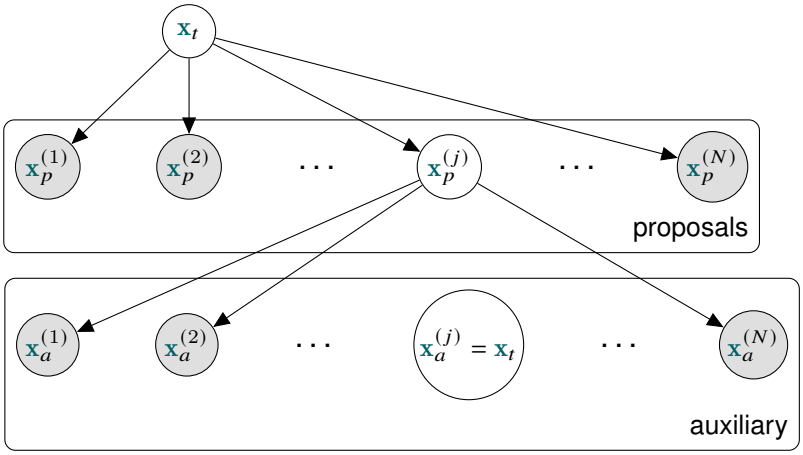


Figure 5.2: A general multiple-try Metropolis (MTM) step with proposal and auxiliary variables. The proposal $\mathbf{x}_p^{(j)}$ is the candidate to accept. If the candidate is rejected, the previous sample $\mathbf{x}_t$ is kept.

The general case of the MTM sampler is depicted in Fig. 5.2. First, MTM generates N proposal samples and calculates the importance weights via:

$$w(\mathbf{x}_p^{(i)} \mid \mathbf{x}_t) = \frac{p(\mathbf{x}_p^{(i)}, \mathbf{z})}{q(\mathbf{x}_p^{(i)} \mid \mathbf{x}_t)} \tag{5.6}$$

Afterward, the algorithm selects one of the proposals $\mathbf{x}_p^{(j)}$ according to the importance weights as the candidate for x_{t+1}:

$$j \sim \mathrm{Cat}\left(w(\mathbf{x}_p^{(1)} \mid \mathbf{x}_t), w(\mathbf{x}_p^{(2)} \mid \mathbf{x}_t), \ldots, w(\mathbf{x}_p^{(N)} \mid \mathbf{x}_t)\right) \tag{5.7}$$

In the general case, the next step is to propose $N-1$ auxiliary samples $\mathbf{x}_a$ and insert the previous sample $\mathbf{x}_t$ at position j. The acceptance ratio for $\mathbf{x}_p^{(j)}$ is defined as:

$$\alpha(\mathbf{x}_t, \mathbf{x}_p^{(j)}) = \min\left[1, \frac{\sum_{i=1}^{N} w(\mathbf{x}_p^{(i)} \mid \mathbf{x}_t)}{\sum_{i=1}^{N} w(\mathbf{x}_a^{(i)} \mid \mathbf{x}_p^{(j)})}\right] \tag{5.8}$$

The weights of the auxiliary variables in the denominator can be calculated according to Eq. (5.6). A significant drawback of the general MTM sampler is that it requires the evaluation of $2N-1$ samples per step due to the auxiliary variables. If the proposal distribution is independent, $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t) = q(\mathbf{x}_{t+1})$, no auxiliary variables are required, and the acceptance ratio is defined as [87]:

$$\alpha(\mathbf{x}_t, \mathbf{x}_p^{(j)}) = \min\left[1, \frac{\sum_{i=1}^{N} w(\mathbf{x}_p^{(i)} \mid \mathbf{x}_t)}{w(\mathbf{x}_t \mid \mathbf{x}_t) + \sum_{i=1, i\neq j}^{N} w(\mathbf{x}_p^{(i)} \mid \mathbf{x}_t)}\right] \tag{5.9}$$

The denominator uses the same sum as the nominator, except for replacing $\mathbf{x}_p^{(j)}$ with $\mathbf{x}_t$.

Even though MTM uses multiple samples within a step, the resulting chain consists of single sequential samples. Consequently, the sampler discards many hypotheses and the algorithm might struggle with multiple modes and local minima similar to MH. In unfortunate probability landscapes, MTM might get stuck longer than MH. To eliminate this behavior, Martino et al. propose to select a random number $N_r \in \{1, \ldots, N\}$ of proposals and auxiliaries [88].

5.1.3 Sequential Monte Carlo Samplers

Compared to MCMC methods, sequential Monte Carlo (SMC) is a particle-based method which evolves a cloud of N samples and corresponding importance weights $\{w_t^{(i)}, \mathbf{x}_t^{(i)}\}$ with $i = \{1, \ldots, N\}$, called particles. SMC approximates the posterior distribution $p_t(\mathbf{x} \mid \mathbf{z})$ at each step t using the importance weights from Eq. (5.2).

Historically, particle filters (PFs), also called sequential Monte Carlo *method*, have been used as an alternative to Kalman filters in multimodal, nonlinear state-space problems. PFs work on sequential observations $\mathbf{z}_t$ and are used to estimate the posterior distribution of the state $p(\mathbf{x}_t \mid \mathbf{z}_t)$ at each timestep t. Chopin modified PFs to estimate posterior distributions $p(\mathbf{x} \mid \mathbf{z})$ that does not change over time, e.g., by iteratively estimating posteriors for subsets of the data $p_t(\mathbf{x} \mid z_1, \ldots, z_t)$ with $t \leq \dim(\mathbf{z})$ [89]. Note that the evaluated densities p_t change with the steps t as different subsets of the data are considered. Afterward, Moral et al. generalized the PF further, which enables a parallel usage of MCMC kernels [78]. They term these algorithms sequential Monte Carlo *samplers*. To avoid confusion, SMC will be used to denote sequential Monte Carlo *samplers* and PF to denote particle filters from here on.

As shown in Fig. 5.3, each SMC step starts by generating a proposal for each of the N particles. Instead of directly using a proposal distribution $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$, SMC uses a forward kernel $K_t(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$. In this context, a kernel has a density associated with it but might include additional steps, such as an acceptance step. After proposing the new states, SMC calculates incremental weights using:

$$\tilde{w}_{t+1}^{(i)} = \frac{p_{t+1}(\mathbf{x}_{t+1}, \mathbf{z}) L_t(\mathbf{x}_t \mid \mathbf{x}_{t+1})}{p_t(\mathbf{x}_t, \mathbf{z}) K_{t+1}(\mathbf{x}_{t+1} \mid \mathbf{x}_t)} \tag{5.10}$$

The incremental weights update each particle weight via:

$$w_{t+1}^{(i)} = w_t^{(i)} \, \tilde{w}_{t+1}^{(i)} \tag{5.11}$$

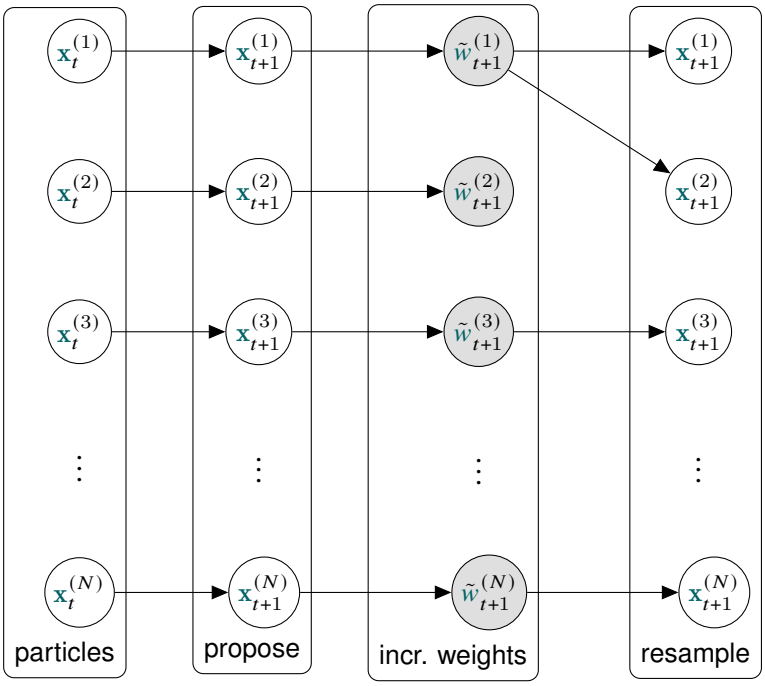


Figure 5.3: A SMC step for N particles with an optional resampling step [90].

In addition to the forward kernel K_t and the target densities p_t and p_{t+1}, Eq. (5.10) makes use of an artificial backward kernel $L_t(\mathbf{x}_t \mid \mathbf{x}_{t+1})$. The choice of L is arbitrary, but Del Moral et al. propose some (sub)optimal choices for L based on the choice of K [78]. One generic choice is to use an MCMC forward kernel, which uses the MH acceptance ratio from Eq. (5.4) in this work. The backward kernel for an MCMC forward kernel is [78]:

$$L_t(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \frac{p_{t+1}(\mathbf{x}_t, \mathbf{z}) K_{t+1}(\mathbf{x}_{t+1} \mid \mathbf{x}_t)}{p_{t+1}(\mathbf{x}_{t+1}, \mathbf{z})}, \tag{5.12}$$

and the resulting weight increment:

$$\tilde{w}_{t+1}^{(i)} = \frac{p_{t+1}(\mathbf{x}_t, \mathbf{z})}{p_t(\mathbf{x}_t, \mathbf{z})}. \tag{5.13}$$

Note that the nominator and denominator evaluate the previous state $\mathbf{x}_t$, and the incremental weight is equal to one if the previous p_t and the new target distribution p_{t+1} are the same. If the weights do not change, the algorithm runs N independent Markov chains which do not interact. Otherwise, if the weights change, resampling can be used as a natural mechanism to let the chains interact and avoid particle deprivation. Resampling is triggered after the weight updates and if the effective sample size (ESS) is below a threshold N_{eff} with more details in Section 5.2.3. Evolving the target $p_t(\mathbf{x} \mid z_1, \ldots, z_t)$ using Chopin's method of including one

measurement at a time is unsuitable for the depth image model from Chapter 4 [89]. Many pixels do not show the object of interest, and the corresponding computational cycles would be wasted. Instead, this work evolves the densities using a tempered likelihood, which is explained in detail in Section 5.2.1 and has been proposed in [89, 91].

As a particle-based method, SMC should generally be favorable for multimodal posteriors like the ones expected for ambiguous object poses. Moreover, drawing and evaluating samples is "trivially" parallelizable, as these steps repeat the same operation for all particles. Parallelizing the resampling step is not trivial but requires much fewer operations than evaluating the likelihood. A sequential resampling step is preferred, since the pose states reside on the CPU. Another benefit of SMC algorithms is that they allow estimating the model evidence using the unnormalized weights, which can be used in model comparison [78, 91].

5.1.4 Particle Filtering as Special Case of Sequential Monte Carlo Samplers

This section shows that the SMC algorithm is a generalization of the particle filter (PF) similar to the derivations in [90]. A bootstrap particle filter uses the transition probability as the forward kernel, which is described by the noisy system dynamics:

$$K_{t+1}(\mathbf{x}_{t+1} \mid \mathbf{x}_t) = p(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \tag{5.14}$$

Choosing a backward kernel in the form of

$$L_t(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \frac{p_t(\mathbf{x}_t, \mathbf{z}) K_{t+1}(\mathbf{x}_{t+1} \mid \mathbf{x}_t)}{p(\mathbf{x}_{t+1})}, \tag{5.15}$$

results in the bootstrap PF. In this case, the weight increment is the likelihood function:

$$\tilde{w}_{t+1}^{(i)} = p(\mathbf{z} \mid \mathbf{x}_{t+1}^{(i)}) \tag{5.16}$$

The bootstrap PF kernel has the drawback that only local proposal moves can be used. An independent global move would replace all particles and discard the progress of the sampling iterations. Still, the kernel involves simpler computations compared to the MCMC kernel and might profit from running more SMC iterations using the same compute budget. More importantly, it allows the same implementation to be reused for particle filtering applications on sequential data.

5.2 Best Practices and Sampler Modifications

Using the vanilla version of a sampling algorithm often does not result in satisfactory results if the target distribution is too complex [92]. This section presents likelihood tempering and adaptive proposals to improve the exploration and exploitation behavior of the algorithms. Moreover, this section introduces logarithmic sampling to avoid numerical issues and unconstrained sampling to enable a flexible combination of prior and proposal models.

5.2.1 Likelihood Tempering

Likelihood tempering reduces the influence of the likelihood function in the early sampling phase to avoid focusing on a single high-likelihood region too early. Instead, the modified probabilistic model favors the exploration of the prior. It is a generally applicable concept, so this work also uses it in the MCMC algorithms. Mathematically, the tempering schedule ϕ_t modifies the likelihood [78, 91, 92]:

$$p_t(\mathbf{x}, \mathbf{z}) = p(\mathbf{z} \mid \mathbf{x})^{\phi_t} p(\mathbf{x}) \tag{5.17}$$

In this work, the tempering parameter increases linearly from zero to one with the number of steps T:

$$\phi_t = \frac{t}{T}, \; t \in \{1, \ldots, T\}, \tag{5.18}$$

As the number of inference steps T must be known before running the sampler, tempering does not apply to particle filters. Plugging Eq. (5.17) into Eq. (5.13) results in the following incremental weights for an MCMC forward SMC kernel:

$$\tilde{w}_{t+1}^{(i)} = p(\mathbf{z} \mid \mathbf{x}_t^{(i)})^{\phi_{t+1} - \phi_t} \tag{5.19}$$

5.2.2 Adaptive Proposals for Sequential Monte Carlo Samplers

A good proposal model must offer a tradeoff between exploration and exploitation. The algorithm gets stuck in local optima if the step sizes are too small. If the step sizes are too large, the algorithm might not converge. Therefore, the proposals introduce tunable parameters, which can be avoided using an adaptive scheme to parametrize the proposals [93].

In particle-based sampling algorithms such as SMC, the current particle distribution $\{w_t^{(i)}, \mathbf{x}_t^{(i)}\}$ can be used to derive an adaptive proposal. The adaptive proposal

distribution is a multivariate normal distribution with a covariance according to the particle cloud [91]:

$$\mathbf{x}_{t+1}^{(i)} \sim \mathcal{N}\left(\mathbf{x}_t^{(i)}, \Sigma_t\right), \tag{5.20}$$

with the covariance matrix:

$$\Sigma_t = \sum_i^N w_t^{(i)} \left(\mathbf{x}_t^{(i)} - \mu_t\right) \left(\mathbf{x}_t^{(i)} - \mu_t\right)^\mathsf{T}, \tag{5.21}$$

and the mean used to calculate the covariance matrix:

$$\mu_t = \sum_i^N w_t^{(i)} \mathbf{x}_t^{(i)}. \tag{5.22}$$

The idea is that the previous target density $p_t(\mathbf{x}, \mathbf{z})$ should, by design, be similar to the next target density $p_{t+1}(\mathbf{x}, \mathbf{z})$. In this case, the covariance at time t is a good scaling at $t + 1$. Intuitively, if the variance of the current particle cloud is high, the algorithm has not converged and should explore the space using larger proposal steps. If the variance is low, the algorithm has converged to the target distribution and should sample from it.

Calculating the covariance of the particle cloud implies that it must be distributed approximately according to a multivariate normal distribution. However, only the position but not the orientation is expected to be distributed sufficiently normally. Thus, adaptive proposals are only applied to the position but not the orientation components.

5.2.3 Logarithmic Sampling

Since densities can differ in the order of many magnitudes, a best practice in statistics is to evaluate **log densities** to avoid numerical issues. In particular, the image likelihood from Eq. (4.4) involves multiplying the individual likelihoods of hundreds of pixels. Multiplying a large number of values can result in numerical over- or underflows:

$$p(\mathbf{z} \mid \boldsymbol{\mu}_{img}) = \lim_{N\to\infty} \prod_{i=1}^{N} p(z_i \mid \boldsymbol{\mu}_{img}) = \begin{cases} 0 & , \mathbb{E}\left[p(z_i \mid \boldsymbol{\mu}_{img})\right] < 1 \\ \infty & , \mathbb{E}\left[p(z_i \mid \boldsymbol{\mu}_{img})\right] > 1 \end{cases} \tag{5.23}$$

In the logarithmic domain, the image likelihood becomes numerically more stable. The image likelihood turns into a sum with i iterating over all N_{px} pixels in the images:

$$\ln p(\mathbf{z} \mid \boldsymbol{\mu}_{img}) = \ln\left(\prod_i^{N_{px}} p(z_i \mid \mu_i)\right) = \sum_i^{N_{px}} \ln p(z_i \mid \mu_i) \tag{5.24}$$

Reformulating a product of densities as the sum of the respective log densities is a concept that is also applicable in other places, like evaluating the acceptance ratio from Eq. (5.4).

Sometimes, evaluating the sum of logarithmic elements in the non-log domain is necessary. For example, updating the weights in the SMC sampler involves simpler calculations and is numerically more stable in the log domain. However, normalizing the weights involves summing over the non-log weights:

$$\ln \hat{w}_t^{(i)} = \ln \left(\frac{w_t^{(i)}}{\sum_{i=1}^{N} w_t^{(i)}} \right) = \ln w_t^{(i)} - \ln \sum_{i=1}^{N} \exp \left(\ln w_t^{(i)} \right) \tag{5.25}$$

Exponentiating the logarithmic weights might lead to over- or underflows, which can be avoided using the *log-sum-exp* trick:

$$\mathrm{LSE}\left(\ln x_1, \ldots, \ln x_N\right) = \ln \sum_{i=1}^{N} \exp\left(\ln x_i\right) = c + \ln \sum_{i=1}^{N} \exp\left(\ln x_i - c\right), \tag{5.26}$$

where $c = max\{\ln x_1, \ldots, \ln x_N\}$. This function can also be used to calculate the **ESS**, which determines whether to resample the particles:

$$\ln N_{eff} = \ln \left(\frac{1}{\sum_{i=1}^{N} \left(w_t^{(i)} \right)^2} \right) = - \left(\sum_{i=1}^{N} e^{2 \ln w_t^{(i)}} \right) \tag{5.27}$$

$$= -\mathrm{LSE} \left(2 \ln w_t^{(1)}, \ldots, 2 \ln w_t^{(N)} \right) \tag{5.28}$$

Another example of summing weights in this thesis is calculating the MTM acceptance ratio in Eq. (5.8).

Similarly, systematic **resampling** requires iteratively adding logarithmic weights. A specific version of the LSE trick for two variables is referred to as *logaddexp*. Algorithm 1 expresses the systematic resampling in the log domain used in this work. The systematic resampling algorithm has a lower computational cost, and results in a lower sampling variance than randomly drawing all samples from a categorical distribution [12].

Selecting a single value from a categorical distribution is required in the MTM sampler for choosing the acceptance candidate; see Eq. (5.7). To avoid summing and iterating over the weights as in Algorithm 1, the *Gumbel-max trick* can be used to draw according to unnormalized logarithmic weights [1]. The idea is to draw N values $G^{(i)}$ independently from a Gumbel distribution and add them to each of the log weights:

$$I = \underset{i \in (1,\ldots,N)}{\mathrm{argmax}} (\ln w^{(i)} + G^{(i)}), \; G^{(i)} \sim \mathrm{Gumbel} \tag{5.29}$$

Algorithm 1: Systematic resampling in log-domain.

Input: N particle states $\mathbf{x}$ and logarithmic weights $\mathbf{w}$
Result: resampled particle states $\hat{\mathbf{x}}$ and logarithmic weights $\hat{\mathbf{w}}$

$c \leftarrow \mathbf{w}[0]$ // cumulative log-weight
$r \leftarrow \text{random}(0, 1/N)$ // random starting point
$i \leftarrow 1$ // current sample
for $n \leftarrow 0$ **to** N **do**
  /* systematic steps */
  $U \leftarrow \ln(r + n/N)$
  while $U > c$ **do**
    $i \leftarrow i + 1$
    $c \leftarrow \text{logaddexp}(c, \mathbf{w}[i])$
  end
  $\hat{\mathbf{x}}[n] \leftarrow \mathbf{x}[i]$
  $\hat{\mathbf{w}}[n] \leftarrow -\ln(N)$
end

After the addition, the index I with the largest corresponding sum is chosen as the random sample. Repeating this process for many draws from the Gumbel distribution results in indices distributed according to the categorical distribution for the respective weights.

Another benefit is that the densities of the frequently used exponential family of probability distributions can be calculated more efficiently in the log domain. For example, the pdf of the normal distribution becomes:

$$\ln(p_{\mathcal{N}}(x \mid \mu, \sigma)) = -\ln(\sqrt{2\pi}) - \ln(\sigma) - \frac{1}{2\sigma^2}(x - \mu)^2 \tag{5.30}$$

Every term in this equation except $(x - \mu)^2$ can be precalculated at compile time if σ does not change, improving the online performance.

5.2.4 Sampling in Constrained Domains

Many sampling algorithms require that the variables are defined in the unconstrained domain $\mathbb{R}$. However, commonly used prior distributions like the uniform distribution are limited to an interval $x \in [x_{min}, x_{max}]$. One possibility to satisfy the constraints is to repeat the random draws until all variables are in their respective valid bounds. However, this approach results in an undetermined long runtime. A change of variables can be used to transform the samples from the constrained

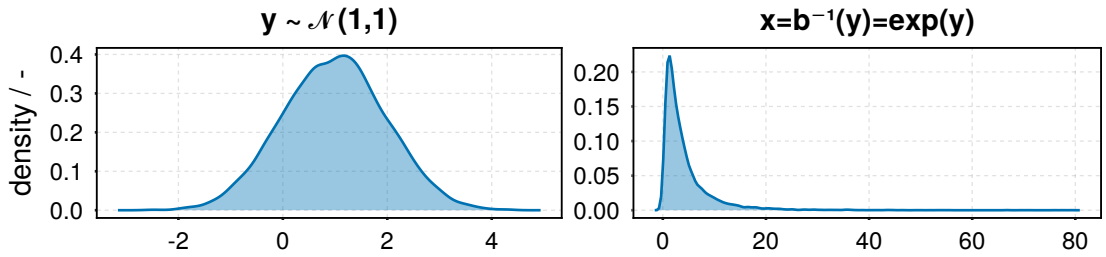


Figure 5.4: Distortion as a result of transforming a random variable.

model domain $\mathcal{D} \subset \mathbb{R}^n$ to $\mathbb{R}^n$ which avoids drawing samples for an undefined time. By convention, this mapping is defined as $b(x) \mapsto y, b : \mathcal{D} \mapsto \mathbb{R}^n$ and is referred to as *bijector*. A bijector must be differentiable and invertible [94].

For example, if a variable x has an exponential prior $p_x(x) : x \sim \mathrm{Exp}(\beta = 1)$, the corresponding bijector for mapping $x \in \mathbb{R}^+ \mapsto y \in \mathbb{R}$ is $b(x) = \ln x$ and the inverse mapping $b^{-1}(y) = e^y$. Let the proposal distribution in the unconstrained domain be a normal distribution centered at the previous sample $q(y) : y_{t+1} \sim \mathcal{N}(\mu = y_t =, \sigma = 1)$. As visible in Fig. 5.4, the proposed samples for $y_t = 1$ are not distributed according to the proposal after transforming them back to the prior distribution's domain. Instead, they are distorted according to the transformed distributions $b^{-1}\left(p_y(y)\right) \mapsto p_x(x)$. Therefore, when evaluating the prior $p_y(y)$ for the transformed variable, an adjustment of the density using a *Jacobian factor* is required [11]:

$$p_{\mathbf{y}}(\mathbf{y}) = p_{\mathbf{x}}\left(b^{-1}(\mathbf{y})\right) \left|\det J_{b^{-1}}(\mathbf{y})\right| = p_{\mathbf{x}}\left(b^{-1}(\mathbf{y})\right) \left|\det J_b(\mathbf{x})\right|^{-1} \tag{5.31}$$

In the example above, Eq. (5.31) results in the corrected density

$$p_y(y) = p_x(e^y) \left|\frac{d}{dy} e^y\right| = p_x(e^y)\, e^y = p_x(e^y) \left(\frac{1}{x}\right)^{-1}.$$

5.2.5 Metropolis Hastings in Unconstrained Domains

As stated in Section 5.2.4, a correction by the Jacobian factor is required when evaluating a transformed probability variable. Applying Eq. (5.31) to the MH acceptance ratio from Eq. (5.4) results in the following term for acceptance ratio:

$$\alpha = \frac{p_{\mathbf{y}}(\mathbf{y}_{t+1})\; q_{\mathbf{y}}(\mathbf{y}_t \mid \mathbf{y}_{t+1})}{p_{\mathbf{y}}(\mathbf{y}_t)\; q_{\mathbf{y}}(\mathbf{y}_{t+1} \mid \mathbf{y}_t)} \tag{5.32}$$

$$= \frac{p_{\mathbf{x}}(b^{-1}(\mathbf{y}_{t+1}))\, \left|\det J_{b^{-1}}(\mathbf{y}_{t+1})\right| q_{\mathbf{y}}(\mathbf{y}_t \mid \mathbf{y}_{t+1})}{p_{\mathbf{x}}(b^{-1}(\mathbf{y}_t))\, \left|\det J_{b^{-1}}(\mathbf{y}_t)\right| q_{\mathbf{y}}(\mathbf{y}_{t+1} \mid \mathbf{y}_t)} \tag{5.33}$$

Algorithm 2: Sampling in the unconstrained domain

Data: N inference steps, prior $p_x(x)$, proposal $p_y(y_{t+1} \mid y_t)$, model $p_x(x, z)$
Result: N samples $\mathbf{x} \sim p(x \mid z)$

1 $\mathbf{y}[1] \sim b(p_x(x))$ // store unconstrained
2 **for** $i \leftarrow 1$ **to** $N - 1$ **do**
3 $\quad y_t = \mathbf{y}[i]$; // current sample
4 $\quad y_{t+1} \sim q_y(y_{t+1} \mid y_t)$ // propose unconstrained
$\quad$ /* probabilistic model in constrained domain */
5 $\quad \alpha \leftarrow \frac{p_{\mathbf{x}}(b^{-1}(y_{t+1}),z)\left|\det J_{b^{-1}}(y_{t+1})\right| q_y(y_t|y_{t+1})}{p_{\mathbf{x}}(b^{-1}(y_t))\left|\det J_{b^{-1}}(y_t)\right| q_y(y_{t+1}|y_t)}$
$\quad$ /* $u > \alpha \Leftrightarrow \neg(u < min(\alpha, 1))$ */
6 $\quad$ u = rand($\mathcal{U}(0, 1)$) **if** $u > \alpha$ **then**
7 $\quad\quad \mathbf{y}[i + 1] \leftarrow y_t$
8 $\quad$ **else**
9 $\quad\quad \mathbf{y}[i + 1] \leftarrow y_{t+1}$
10 $\quad$ **end**
11 **end**
12 $\mathbf{x} = b^{-1}(\mathbf{y})$// return in model domain

In case of a symmetric proposal $q(\mathbf{y}_{t+1} \mid \mathbf{y}_t) = q(\mathbf{y}_t \mid \mathbf{y}_{t+1})$ the equation can be simplified further, which is similar to the Metropolis step without the extension by Hastings:

$$\alpha = \frac{p_{\mathbf{x}}(b^{-1}(\mathbf{y}_{t+1}))\left|\det J_{b^{-1}}(\mathbf{y}_{t+1})\right|}{p_{\mathbf{x}}(b^{-1}(\mathbf{y}_t))\left|\det J_{b^{-1}}(\mathbf{y}_t)\right|} \tag{5.34}$$

Thus, the scheme from Algorithm 2 improves the chance of accepting new samples and is a more robust implementation. If the samples are stored in the model domain, they need to be transformed once for the proposal and again for storing them. These frequent forward and backward transformations can lead to numerical inaccuracies. Consequently, the samples are stored in the unconstrained domain immediately after sampling the initial sample from the prior in line 1 of the algorithm. The probabilistic model is formulated in the constrained domain, so the samples must be transformed back in line 5 to calculate the acceptance rate. For efficiency, the result of evaluating the previous sample $p_{\mathbf{x}}(b^{-1}(y_t))\left|\det J_{b^{-1}}(y_t)\right|$ can be stored and reused in the next inference step. Finally, the samples are transformed back to the model domain in line 12.

5.2.6 Sampling Blocks of Variables

If a set of variables is highly correlated, sampling them in blocks can speed up the convergence [95]. Instead of proposing the whole state vector $\mathbf{x}$ at each inference step, *blocked sampling* groups variables $\mathbf{x} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(i)}, \ldots, \mathbf{x}^{(N)}\}$ to propose and evaluate only a block (i) at each step.

$$q^{(i)}(\mathbf{x}_{t+1} \mid \mathbf{x}_t) = q(\mathbf{x}_t^{(1)}, \ldots, \mathbf{x}_{t+1}^{(i)}, \ldots, \mathbf{x}_t^{(N)} \mid \mathbf{x}_t) \tag{5.35}$$

All variable blocks except (i) keep the value from the previous time step in this proposal. In practice, this allows using different samplers for different blocks of variables, as in *Turing.jl* [9].

Choosing which variables to group into one block is not always obvious, especially for complex models. Approaches for automatic blocking have been developed, but the physical intuition behind the 6D pose model lends itself to a rigorous choice [96]. This choice consists of grouping the translational components $\mathbf{t} = [t_x, t_y, t_z]$ and the rotational components $\mathbf{R} = [q_w, q_x, q_y, q_z]$ of the pose. Specifically, the rotational components are expected to be highly correlated, as they can significantly change the object's appearance. Moreover, rotations are represented as quaternions, and sampling the whole rotation allows using the proposal model from Section 4.2.4.

5.3 Summary

Closed-form solutions usually do not exist for inferring the posterior distribution $p(\mathbf{x} \mid \mathbf{z})$ of a problem. Therefore, approximate inference algorithms use the probabilistic model $p(\mathbf{x}, \mathbf{z})$, which can be evaluated pointwise, to evaluate the posterior up to a constant.

Variational inference restricts the family of distributions and fits these to the data using optimization algorithms. A significant drawback when applying variational inference to the problem of depth camera-based pose estimation is that efficient optimizers require gradients of the model. The rasterization used when rendering images is not differentiable and would require other approximations, for example, via sampling the environment of the current state.

Sampling-based algorithms approximate the posterior using discrete samples drawn using auxiliary proposal distributions. Due to the mismatch of the proposal and the target distribution, the samples have to be adjusted. Either the samples are accepted or rejected in a Markov chain Monte Carlo (MCMC) algorithm, or a

cloud of particles is adjusted using importance weights in sequential Monte Carlo (SMC). Unlike variational inference, sampling-based algorithms are typically used in scenarios with little data. Moreover, these algorithms do not rely on gradients and are, therefore, used for the pose estimation in this thesis.

The pose estimation problem is multimodal due to symmetries and occlusions. Baseline versions of the sampling algorithms struggle with exploring multiple modes. Tempering the likelihood is a crucial adaption that allows exploring multimodal distributions in the early sampling phase [92]. Moreover, this work relies on adaptive proposals in the context of SMC, eliminating additional tunable parameters and automatically switching from exploration to exploitation. Logarithmic sampling improves the numerical stability, and unconstrained sampling enables a flexible combination of prior and proposal models.

Many other adaptions for improving sampling algorithms, such as adaptive cooling strategies, exist. However, as Chapter 6 will show, the adaptions above are sufficient. While the GPU optimizations lead to a low execution time, reusing older samples could be a worthwhile adaption to avoid the expensive model evaluations and further improve the inference time [91].

6 Experimental Comparison of Models and Samplers for 6D Pose Estimation

This chapter evaluates several model-sampler configurations introduced in Chapters 4 and 5 on challenging industry-relevant datasets. Table 6.1 gives an overview of these model-sampler configurations. The experiments evaluate the following representatives of sampler families: MH algorithm as a sequential MCMC sampler, MTM as a parallelizable MCMC sampler, and SMC with an MH kernel as a parallelizable particle-based sampler. First, the experiments ablate the sampler configurations to enable successful inferences with a target time of 0.5 s per pose in Section 6.3. Second, to maximize the recall of pose estimates, the experiments ablate the

- choice of priors for position and classification in Section 6.5.1.
- modeling of occlusions: exponential distributions, pixel classification, and image regularization in Section 6.5.2.
- parameter tuning of probabilistic models for different datasets in Section 6.6.

The following sections first present the datasets, pose error metrics, and performance scores used to evaluate the pose estimates quantitatively. Then, baseline model-sampler configurations are introduced to evaluate each component from Table 6.1. Due to the long evaluation times, non-competitive configurations are

Table 6.1: Overview of experiments to compare different model-sampler configurations. * The resolution is only evaluated for the SMC sampler.

sampling algorithm	resolution* / runtime	pixel $p(z_i \mid \mu_i)$	class & image reg. $p(\mathbf{z} \mid \mathbf{x}, \mathbf{z})$	parameter tuning
MH	X			X
MTM	X			
SMC	X*	X	X	X

not part of the ablation. Still, one sampling-based (MH) and one particle-based algorithm (SMC) participate in the automatic parameter tuning in Section 6.6 for a fair comparison. Finally, the best-performing configuration is tested on real data from the BOP challenge and compared to the state-of-the-art in Section 6.7.

6.1 Experiment Design

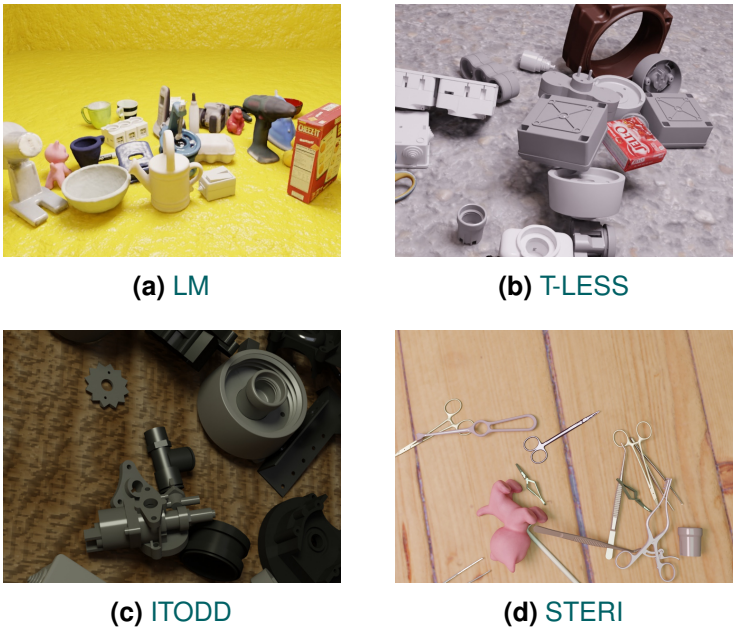
Each model-sampler configurations' accuracy is evaluated based on the Benchmark for 6D Object Pose Estimation (BOP) challenge, making 6D pose estimation methods comparable [19]. First, this section presents the unified BOP datasets for scenarios such as household items, industrial items, or shiny objects. Second, this section introduces metrics for pose errors that handle ambiguous views for symmetric or occluded objects.

6.1.1 Datasets

The performance of the model-sampler configurations is evaluated on a selection of datasets from the BOP challenge, referred to as BOP [19]. One of the earliest datasets is Linemod (LM), named after the template-matching-based pose estimation method for which the dataset was generated [28]. Because of its widespread use, the LM dataset is also used in this thesis to improve the comparability of results. Brachmann et al. added annotations to one LM scene to include more challenging occlusions, resulting in the Linemod-Occluded (LM-O) dataset [37]. Two of the BOP datasets, texture-less rigid objects (T-LESS) and MVTec industrial 3D object detection dataset (ITODD), focus on industrial settings where CAD models are available, objects tend to be symmetric, texture-less or shiny, and similar objects are visible in each scene [97, 98]. Moreover, the ITODD dataset captures shiny metallic parts with a high-quality depth sensor, similar to the surgical instruments in Section 7.1.

In the later iterations of BOP, reducing the sim-to-real gap is a particular focus [32]. The methods presented in this thesis specifically target problems where only CAD models are available beforehand. Similar to BOP, synthetic datasets have been generated using BlenderProc[1], preventing tuning the algorithms on the test sets [4]. The synthetic LM, T-LESS, and ITODD datasets use similar scene configurations as the original BOP datasets. Each dataset contains scenes

[1] BlenderProc Pipeline: https://github.com/rwth-irt/BlenderProc.DissTimRedick



(a) LM **(b)** T-LESS

(c) ITODD **(d)** STERI

Figure 6.1: Example scenes from the synthetic datasets.

rendered from five different views, as shown in Fig. 6.1, resulting in 25 images per dataset.

- LM: 5 scenes, 15 objects textureless household items, nine distractors
- T-LESS: 5 scenes, 20 objects sampled from 30 textureless industrial relevant objects, six distractors
- ITODD: 5 scenes, 25 objects sampled from 28 textureless industrial relevant objects, five distractors
- STERI: 10 scenes, 20 objects sampled from 40 surgical instruments, 2-4 distractors

The synthetic surgical instruments (STERI) dataset contains more scenes, since no real annotated datasets are available for surgical instruments. Therefore, a more diverse dataset is required to test the application in Section 7.1. Each scene of the STERI dataset consists of a pile of instruments generated by dropping CAD models from an orthopedic standard sieve using a physics simulation. According to the BOP convention, the origin of all mesh models is centered in the object.

6.1.2 Computer Hardware

The experiments ran on a workstation equipped with an AMD Ryzen Threadripper PRO 5975WX 32-Cores CPU, 256 GB RAM, and an NVIDIA GeForce RTX 4090 GPU with 24 GB VRAM. CPU parallelization was not used, only GPU parallelization. A Laptop equipped with an Intel(R) Core(TM) i7-7700HQ CPU, 16 GB RAM, and an NVIDIA GeForce 940MX mobile GPU with 3 GB VRAM is also capable of running the inference with longer runtimes. The algorithms can also run on a CPU only with integrated graphics and OpenGL support, but ≈60 x slower.

6.1.3 Pose Error Metrics

An overview and rationale behind 6D pose error metrics can be found in the evaluation methodology of BOP 2020 [32]. The goal is to select metrics commonly used in the literature to enable comparability while focusing on the relevance of robotic manipulation tasks. In particular, metrics designed for other applications like augmented reality are not considered here. This thesis' application setting is similar to the industrial robotics setting presented by Gorschlüter et al. in the following ways [1]:

- CAD models are available for surgical instruments from the manufacturer. The systems are indoors, and lighting can be controlled to some extent. However, end users may alter the lighting depending on their needs.
- High-end depth cameras are necessary to capture thin and shiny surgical instruments, as no usable images could be captured with low-cost solutions.
- In contrast to industrial settings, cost is an important factor in clinics. Still, research shows promising results in enabling lower-cost stereo-vision for reflective materials [99].

Together with the LM dataset, the Linemod paper introduced two of the earliest pose error metrics named **average distance of model points (ADD)** and **average distance of model points with indistinguishable views (ADD-S)**, also known as ADI [28]. Because of their early introduction, they are the most widely used metrics and thus lend themselves to enable comparability. **ADD** can be used for non-symmetric objects:

$$e_{ADD}(\hat{\mathbf{P}}, \mathbf{P}, V_M) = \underset{\mathbf{v} \in V_M}{\text{avg}} \|\hat{\mathbf{P}}\mathbf{v} - \mathbf{P}\mathbf{v}\|_2 \tag{6.1}$$

Each of the 3D model's vertices $\mathbf{v} \in V_M$ is transformed using the estimated pose $\hat{\mathbf{P}}$ and the ground truth pose $\mathbf{P}$ in Eq. (6.1). Then, the Euclidean distance between

the transformed vertices is calculated. Finally, all distances are averaged, which results in a single value for the ADD metric.

Pose estimates of symmetric objects are ambiguous. Consequently, the distance between the same vertex can be large for two poses even though the models are well aligned. Thus, Hinterstoisser et al. also proposed the **ADD-S** metric. It is similar to ADD from Eq. (6.1), but avoids matching the vertices by indices and searches the closest vertices via the $\min$ operator instead. Afterward, the distances of the closest correspondences are averaged.

$$e_{ADDS}(\hat{\mathbf{P}}, \mathbf{P}, V_M) = \underset{\mathbf{v}_1 \in V_M}{\operatorname{avg}} \min_{\mathbf{v}_2 \in V_M} \|\hat{\mathbf{P}}\mathbf{v}_1 - \mathbf{P}\mathbf{v}_2\|_2 \tag{6.2}$$

As pointed out in [32], the average point distance metrics have three significant shortcomings: First, these metrics heavily depend on the object geometry and the vertex density. Averaging the distances results in parts of the model with a high vertex density dominating the metric. Second, matching the nearest neighbor vertices can result in unintuitively low pose errors of ADD-S because of the many-to-one matching [19]. Finally, despite its original name, ADD-S is not invariant to ambiguities caused by self-occlusions since all model points are included in the calculation [3]. It is only invariant to object symmetries, so the namings "distance of model points for symmetric objects" and ADD-S are more fitting for the metric. Although this metric is not ideal, it is still considered in this thesis because it is one of the most widely used metrics and enables comparability to other research.

To mitigate the issues of ADD-S, the **maximum symmetry-aware surface distance (MSSD)** has been introduced in a later iteration of BOP:

$$e_{MSSD}(\hat{\mathbf{P}}, \mathbf{P}, S_m, V_M) = \min_{\mathbf{S} \in S_m} \max_{\mathbf{v} \in V_M} \|\hat{\mathbf{P}}\mathbf{v} - \mathbf{P}\mathbf{S}\mathbf{v}\|_2 \tag{6.3}$$

Compared to ADD, MSSD calculates the maximum vertex distance for each symmetry instead of averaging all vertex distances in Eq. (6.1). Moreover, each transformation $\mathbf{S}$ from the set of global symmetries S_m is applied to the ground truth pose $\mathbf{S}$, and the one with the lowest maximum distance is the MSSD error. The MSSD metric is less dependent on the geometry and sampling density of the model since it utilizes the maximum instead of the average used by the ADD-S metric [32]. Moreover, the authors of BOP argue that the maximum vertex distance is a strong indicator for a successful grasp of a robot.

One issue of the MSSD metric is that the symmetries require manual annotations or an algorithmic search. Annotating the symmetries is tedious and heavily depends on the choice of coordinate frames, making manual labeling prohibitive for large datasets. The annotation effort is intractable, especially for the many surgical instruments considered in this thesis. Algorithmic solutions must cover

a vast search space and might miss obvious symmetries due to computational constraints. The set of symmetries S_m requires a discretization of continuous symmetries, introducing discretization errors in Eq. (6.3).

A possible solution that has not been introduced in the literature yet combines the benefits of ADD-S and MSSD. The idea is to replace the average with the maximum operation in Eq. (6.2), resulting in the **maximum distance of model points for symmetric objects (MDD-S)**. This modification eliminates the geometry and sampling density issues of ADD-S, as well as the time-consuming labeling that MSSD requires.

$$e_{MDDS}(\hat{\mathbf{P}}, \mathbf{P}, V_M) = \max_{\mathbf{v}_1 \in V_M} \min_{\mathbf{v}_2 \in V_M} \|\hat{\mathbf{P}}\mathbf{v}_1 - \mathbf{P}\mathbf{v}_2\|_2 \tag{6.4}$$

Even so, the many-to-one matching of ADD-S as well as the self-occlusion ambiguities of ADD-S and MSSD remain. Furthermore, many error metrics have already been proposed and used throughout the literature, thereby hindering the comparability of yet another metric. Thus, this thesis does not use the MDD-S metric.

Instead of calculating point distances, the **Visible Surface Discrepancy (VSD)** compares 3D rendered distance maps $\hat{D}$ and D of the visible object surface for the estimated pose $\hat{\mathbf{P}}$ and ground truth pose $\mathbf{P}$. Initially presented by Hodan et al. in [3], VSD has been used and modified over the years in the BOP challenge resulting in the following definition [19, 32]:

$$e_{VSD}(\hat{D}, D, \hat{V}, V, \tau) = \operatorname*{avg}_{z_i \in \hat{V} \cup V} \begin{cases} 0 & , z_i \in \hat{V} \cap V \wedge \left|\hat{D}(z_i) - D(z_i)\right| < \tau \\ 1 & , otherwise \end{cases} \tag{6.5}$$

The visibility masks $\hat{V}$ and V express whether the rendered distance maps $\hat{D}$ and D are in front of the measured distance map or occluded. Refer to [3] and [32] for the details of generating the visibility masks. If parts of the object surface that should not be visible are visible $z_i \in (\hat{V} \cap V)$, the pixel z_i is penalized. Additionally, if the absolute difference of the visible surfaces $\left|\hat{D}(z_i) - D(z_i)\right|$ exceeds the tolerance τ, the corresponding pixel is also penalized. Averaging the pixels over the union of visible pixels $\hat{V} \cup V$ results in a kind of *Complement over Union*.

3D cameras typically record depth images containing the distance from the camera *plane* to the object points. In contrast, distance maps contain the Euclidean distance from the camera center *point* to the object points. To convert a depth image to a distance map, the pinhole camera model from Eq. (2.23) is inverted to

reproject the depth pixels to 3D.

$$D(z_i) = \left\| \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} (u - c_x)\, z \div f_x \\ (v - c_y)\, z \div f_y \\ z \end{pmatrix} \right\|_2 \tag{6.6}$$

$D(z_i)$ is the resulting distance of pixel i, $(u, v)^\mathsf{T}$ are the image coordinates, $(x, y, z)^\mathsf{T}$ the coordinates of the 3D point, and f_x, f_y, c_x, c_y the pinhole camera parameters from Eq. (2.23).

VSD is the only metric that only considers the visible parts of the object. A grasp can only be successful if the robot reaches for a visible part of the object. Therefore, metrics like MSSD, which consider the whole object surface, might not correctly represent a grasp success in scenarios featuring occlusions. By considering only the visible parts of the object, the VSD metric inherently handles ambiguities caused by (self) occlusions or symmetries.

The conclusion is similar to Hodan et al. [3]: Even if the metrics are similar, subtle differences in the definition can result in substantially different error values. Different pose error metrics should be used according to the use case's goal, for example, the success probability of grasping in robotics or visual consistency in augmented reality applications. Similar to the review of Gorschlüter et al. the ADD-S and VSD error metrics are used in this work as they lend themselves to industrial robotic applications which are similar to the setting in the sterile supply [1]: They can be used with symmetric objects (eliminates ADD), do not require labeling of symmetries (eliminates MSSD), and are comparable to the literature (eliminates MDD-S).

6.1.4 Performance Score for Pose Estimation

A performance score summarizes the quality of a method on a whole dataset of images that might contain multiple object poses. This score builds on the previously introduced error metrics specific to a single object pose. A simplistic approach would be to use the individual pose errors and average them directly. However, an image-based pose estimation consists of two tasks: object detection and object localization, which is the regression of the pose. An unsuccessful detection leads to significant pose errors, distorting the overall performance score. Thus, pose estimation performance scores are typically formulated as a binary classification problem [3, 19, 28]. The machine learning nomenclature yields the following definitions: A correctly estimated pose is considered a true positive (TP), and a wrongly estimated one is a false positive (FP). If no pose is estimated for

a ground truth pose, it is a false negative (FN). Since no annotations exist for locations without any object, the true negative (TN) class does not exist in this context.

Early research on 6D pose estimation using the Linemod dataset uses the accuracy as the overall score, defined as the division of the number of TP by the number of predictions [28]. The localization problem involves detecting a known number of object instances; see Section 1.1. If the pose of the same object instance is estimated repeatedly, it could lead to an undesired high accuracy score while all other instances have been ignored. Furthermore, Hodan et al. and the BOP challenge use the recall as a performance score, which is the number of correct poses divided by the number of annotated poses:

$$recall = \frac{TP}{TP + FN} \tag{6.7}$$

The difference in the evaluation is that matching the estimated and ground truth poses is required. At most, one estimate must be matched to each ground truth [3, 19]. Therefore, if the matchings are known, the number of predictions is the same as the number of ground truth annotations. The number of annotated poses is the denominator of the recall, and both scores are consequently the same for known matchings.

As stated in [1], $FN \leq FP$ in the localization task since the number of predictions is limited to the number of annotated instances, and each FP results in a FN. Therefore, using a precision- or F1-based score only makes sense if the goal is to evaluate the detection quality and reduce the number of FP.

The considered use case is similar to the industrial one presented in [1], so identical average-recall-based scores are used. For each score, a different criterion of correctness is used:

- **ADD-S**: A pose is considered correct if the ADD-S error is below a threshold of $e_{ADDS} < 0.1 \varnothing(V_M)$ which is less than 10 % of the object's diameter [28].
- **VSD**: A pose is considered correct for a VSD tolerance of $\tau = 20$ mm and thresholding the error score by $e_{VSD} < 0.3$ as defined in BOP 2018 [19]. The authors deem this parametrization suitable for robotic applications. Conversely, Hodan et al. stated that a threshold scaled by the object's diameter is less suitable for robotic manipulation tasks, as a successful grasp depends on the absolute positioning error [3].
- **VSDBOP**: BOP 2019 and later evaluate a range of the object's diameter for the tolerance $\tau \in [0.05, 0.1, \ldots, 0.5] \varnothing(V_M)$. Moreover, the method evaluates a range of thresholds $e_{VSD}(\tau) < [0.05, 0.1, \ldots, 0.5]$ for each τ resulting in

100 evaluations per estimated pose and a factor of 100 for the number of annotated poses [32]. The ten evaluations for different τ are expensive since they involve operations on all image pixels. While the evaluation is more expensive, averaging recalls for different thresholds summarizes the recall/threshold curve similarly to the average precision in the *PASCAL VOC* challenge [100]. Moreover, BOP 2019 and later target a broader range of applications and do not focus on robotic tasks as much as BOP 2018.

For all criteria, only objects with >10 % of the projected surface visible are considered similar to the BOP challenge [19]. Similar to BOP, the *average recall* combines the ADD-S, BOP, and VSDBOP recalls by averaging the three.

6.1.5 Evaluation of Pose Distributions

Probabilistic algorithms estimate a distribution of possible poses, but the BOP has been designed for frequentist methods and expects a single pose to evaluate the performance score. Among others, these are possible choices to select a single sample from the distribution:

- **Maximum posterior sample**: The sample which has the highest unnormalized posterior $p(x \mid z) \propto p(z \mid x)p(x)$ is selected. This choice is beneficial if good prior information is available but not much data.
- **Maximum likelihood sample**: The sample which has the highest likelihood $p(z \mid x)$ is chosen. This choice is beneficial if the prior is unreliable. Here, tolerances of attached measurement devices or the unreliability of object detectors such as CNNs might cause problems.
- **Maximum of the distribution**: Since the samples approximate the posterior distribution, most samples should be located around the maximum of the true posterior distribution. A histogram could be generated to find the maximum, leading to discretization errors. Another possibility is to approximate the posterior via a kernel density. In this case, the maximum must be interpolated, which is not trivial for quaternions. Both approximations have a computational overhead, while the previous values are calculated during inference. The maximum posterior sample is likely close to the maximum of the sample distribution.

Using the maximum posterior or likelihood sample implies that commonly used MCMC techniques, such as burn-in and thinning, would be disadvantageous. Both techniques aim to better represent the posterior distribution by discarding samples to reduce correlation, which might include discarding the best-performing sample. If the distribution's maximum is used, these techniques could be helpful,

Table 6.2: Baseline configuration consisting of parametrized prior and posterior distributions.

model	$p(\mathbf{t})$	$p(\mathbf{o})$	$p(\mathbf{z} \mid \mathbf{x}, \mathbf{o})$			
distribution	$\mathcal{N}$	Bern	$\mathrm{Mix}(\mathcal{N}, \mathrm{Exp}, \mathcal{U})$			L_{px}
parameter	σ_t	$p(c_o)$	σ_z	β_z	$z_{min} - z_{max}$	c_{reg}
value	30 mm	0.5	10 mm	1 m	0.5 – 1.5 m	50

as the histogram might be biased towards highly correlated samples. This work uses the maximum likelihood sample because the algorithms presented mainly rely on unreliable prior information for the initialization.

6.2 Baseline Model-Sampler Configurations

Baseline configurations should be as simple as possible to achieve usable results. Here, simplicity regards the complexity of the equations and the ease of implementation. All parameter choices in this section have been hand-tuned in preliminary experiments to enable convergence for the qualitative analysis of the samplers in Section 6.2.3. Only after determining a sufficient image crop resolution and a reasonable particle count in Section 6.3 the baseline scores can be determined. Therefore, the quantitative evaluation follows in Section 6.4.

6.2.1 Probabilistic Model

Table 6.2 gives an overview of the probability distribution choices for the prior and likelihood. Block-wise sampling is used for the position $\mathbf{t}$ and the orientation $\mathbf{R}$ components; see Section 5.2.6. The position prior is modeled as a normal distribution, which uses the center of the object's bounding box as the mean. Choosing a parametrization for the standard deviation σ_t of the position prior represents the expected uncertainty of extracting a position from segmentation masks or RFID tags. In the simplest case, no prior information is available for the object masks, and only a position prior is known. Thus, an uninformed prior of $p(c_o) = p(\overline{c_o})$ is assigned to the object classification, where all pixels have a 50:50 chance of being classified as the object or background. No prior knowledge exists about the object's orientation. Hence, the orientation prior is modeled by a uniform distribution for SO(3) rotations.

Table 6.3: Baseline configurations of the sampler parameters.

sampler	N_{steps}	$N_{particles}$	$\frac{N_{eff}}{N_{particles}}$	$p(\mathbf{t}_{k+1} \mid \mathbf{t}_k)$	$p(\mathbf{R}_{k+1} \mid \mathbf{R}_k)$
MH	1250	n.a.	n.a.	$\mathcal{N}(\mathbf{t}_k, 1\,\text{cm})$	$\mathcal{N}(\mathbf{R}_k, 0.1\,\text{rad}), \mathcal{U}_\mathbf{q}$
MTM	500	10	n.a.	$\mathcal{N}(\mathbf{t}_k, 1\,\text{cm})$	$\mathcal{N}(\mathbf{R}_k, 0.1\,\text{rad}), \mathcal{U}_\mathbf{q}$
SMC	200	100	0.5	n.a.	$\mathcal{N}(\mathbf{R}_k, 0.1\,\text{rad}), \mathcal{U}_\mathbf{q}$

The standard advice for probabilistic perception is to overestimate the noise of range sensors, which avoids overconfidence in the measurements and increases robustness [12]. However, this work needs to balance precision for robotic manipulation and robustness. Therefore, a standard deviation of $\sigma_z = 10\,\text{mm}$ overestimates a typical depth sensor noise[2]. Wüthrich et al. explain that the model is not sensitive to the parametrization of the occlusion model, and the exponential distribution is parametrized by the scale $\beta_z = 1\,\text{m}$ [59]. The model uses the L_{px} pixel count regularization from Eq. (4.37). Preliminary tests suggest that $c_{reg} \in [50, 100]$ is a reasonable choice, which likely depends on the choice of the prior parametrization.

6.2.2 Samplers and their Proposals

Table 6.3 shows the parameters of the baseline sampler, including their proposals. An appropriate number of particles $N_{particles}$ is determined systematically in Section 6.3. Resampling applies only to the SMC sampler, and the resampling threshold is set to half of the particle count. Because the position priors are expected to be sufficiently good, the position proposals are local moves parametrized by a relatively small σ_{t+1} to focus more on pose refinement than exploration. Adaptive sampling is implemented as best practice for the SMC samplers, so no tunable parameter exists for the position proposal of this sampler. For the orientation, all samplers use local moves from a normal distribution parametrized by σ_{r+1} and global moves from a uniform distribution. Global moves should increase the chance of samplers escaping from local optima.

As outlined in Section 5.2, the vanilla versions of the sampling algorithms typically fail to yield satisfactory outcomes. Thus, all samplers use the previously described best practices throughout the experiments. All algorithms use a GPU parallelized version of the image likelihood. Moreover, all inherently parallelizable sampling

[2] Intel Realsense D400 series specifies an accuracy of $\approx 2.5 - 5\,\text{mm}$, Zvid One+ Medium an accuracy of 0.11 mm at 1 m distance.

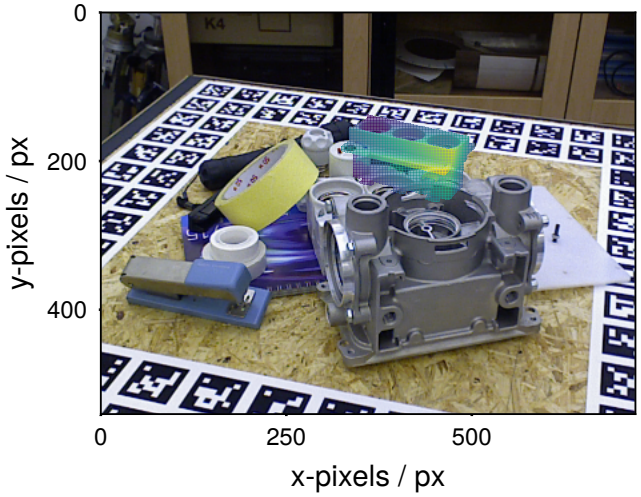


Figure 6.2: Full scene from T-LESS dataset, with the object of interest highlighted.

algorithms also parallelize the evaluation of the particles on the GPU, namely MTM and SMC. Furthermore, all samplers target an inference time of ≈0.5 s and use a resolution of 30 px × 30 px if not stated otherwise; these values are determined systematically in Section 6.3.2.

6.2.3 Qualitative Analysis of Samplers

This section aims to give an intuition of the sampler's behaviors for estimating 6D poses. Representatively, the MH and SMC samplers have been executed on a challenging example scene containing a symmetric object in clutter, shown in Fig. 6.2. A smooth truncated exponential distribution and pixel classifications have been used to qualitatively evaluate the classification probabilities. Moreover, this section employs a higher resolution of 100 px × 100 px for improved visualizations.

Results

Figure 6.3 shows the chain and the resulting density of the MH sampler. The sampler is stuck in one optimum in the first third. Afterward, the orientation jumps to another optimum. Thus, the corresponding position density plot shows sharp modes for the position components with small modes nearby. In contrast, the orientation's x- and y-components show some mixing behavior after escaping the optimum. The orientation density consists of two modes for all components.

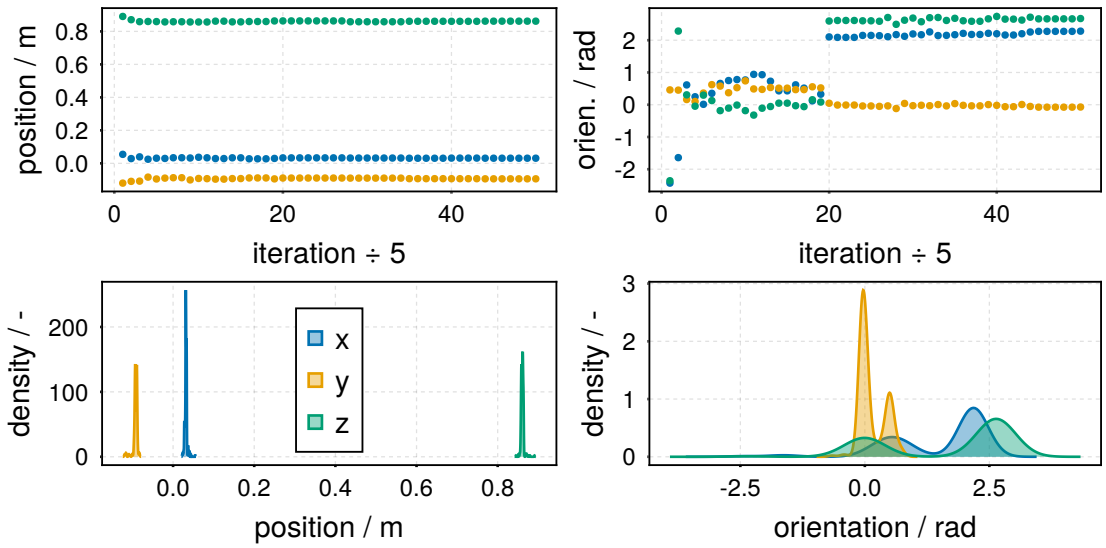


Figure 6.3: Qualitative result of the MH sampler: chain and densities for the scene in Fig. 6.2.

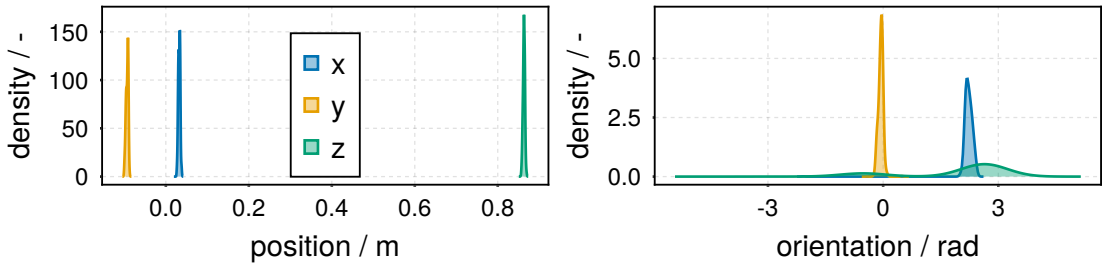
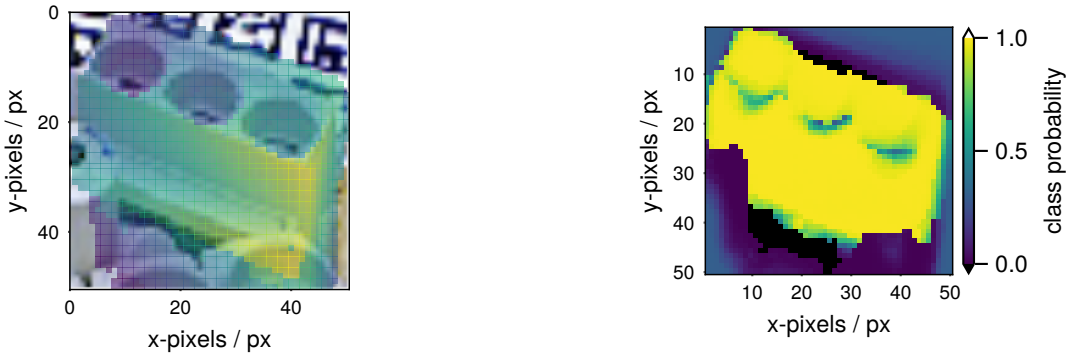


Figure 6.4: Qualitative result of the SMC sampler: densities of the particle cloud for the scene in Fig. 6.2.



(a) SMC max. likelihood estimate

(b) Weighted mean of class probability

Figure 6.5: Best pose of the SMC inference in Fig. 6.4, resolution of 100 px × 100 px for the scene in Fig. 6.2.

SMC samplers evolve a particle cloud instead of a chain. Therefore, Fig. 6.4 only shows the density of the final particle cloud instead of a chain. The position density is sharply peaked compared to the orientation density, which exhibits two pronounced modes for the x- and y-components and is flat for the z-component. Moreover, Fig. 6.5a shows a rendering of the maximum likelihood estimate from the SMC sampler overlaid on the color image. It is only slightly different from the ground truth pose from Fig. 6.2. Moreover, Fig. 6.5b shows the estimated pixel classification probabilities as the weighted mean image of the per-particle classifications. The occluding objects in the front are clearly separated and not classified as the object of interest. Around the edges, the classification probabilities are less confident. No poses cover the other regions, which retain the prior probability of 0.5.

Discussion

Generally, the behavior of the samplers is as expected: The MH sampler tends to get stuck in local minima and requires a long burn-in phase until it samples from the posterior. Consequently, the distribution displayed in the density plots does not represent the posterior distribution. As the burn-in period varies, advanced chain diagnostics would be required to discard samples from this period. Meanwhile, the SMC algorithm converges to a distribution that contains a maximum likelihood pose that can be visually confirmed as the actual pose. These observations are a first indicator that chain-based algorithms perform worse than the SMC algorithm.

The narrow distribution of the position compared to the flat multimodal landscape of the orientation posterior indicates that the orientation is more challenging to estimate. This hypothesis aligns with the experience from the literature, which often focuses on estimating viewpoints [66]. However, interpreting the density plots requires some care since Gaussian kernels approximate the curves from discrete samples. As a result, the density might look more normally distributed and smooth than it is. Even more, care is required when interpreting the orientation xyz-Euler angles. The nonlinear transformations distort the distribution as described in Section 4.2.3. Thus, the flat peak of the orientation's z-component from Fig. 6.4 indicates the expected multimodality caused by object's symmetries.

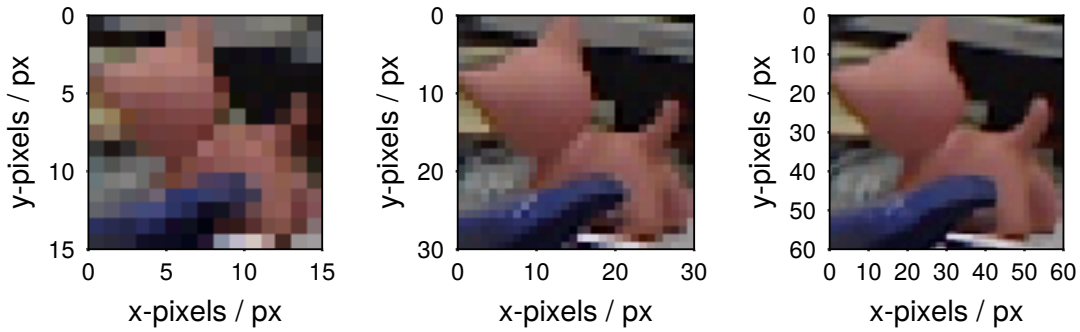


Figure 6.6: Image crops with resolutions ranging 15-60 px of a cat from the LM dataset.

6.3 Image Resolution, Number of Particles, and Runtime

Existing vision-based 3D bin-picking solutions promise cycle times of approximately 10-30 s for a pick and place task and Gorschlüter et al. deem an inference time of 9 s as not acceptable [1, 101, 102, 103]. Moreover, for planning, it is beneficial to estimate the pose of multiple objects in a scene to determine the best candidate; see Section 7.3. Besides, faster inference allows rapid iterating on the models during research. With this consideration, the goal is to limit inference time to 0.5 s. The main factor influencing the runtime is the number of calculations per inference. Preliminary experiments have indicated that the complexity of the models only marginally influences the runtime. This section hypothesizes that an optimal configuration can be found for the image resolution and number of particles given a constrained time budget.

6.3.1 Influence of Image Resolution

The quality of pose estimates should be expected to improve with higher resolutions, since more object details can be captured. Figure 6.6 shows what is recognizable for different resolutions. However, the number of calculations grows quadratically with the edge lengths of an image. Subsequently, fewer inference steps can be performed given a constrained time budget. Moreover, launching compute kernels on the GPU has an overhead that might dominate the inference time for low resolutions. Thus, the task of this section is to determine a resolution that offers enough details without degrading the performance with too few iterations.

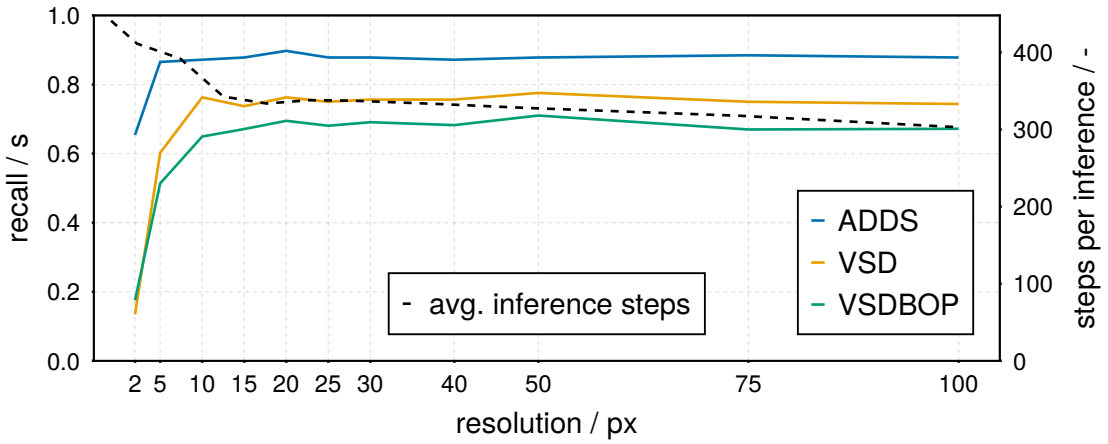


Figure 6.7: Recall and mean number of inference steps of the SMC algorithm for different quadratic image sizes a time budget of 0.5 s.

The SMC has been executed using the baseline configuration from Sections 6.2.1 and 6.2.2 with different quadratic image resolutions of 2-100 px for 0.5 s per pose inference. One scene from each synthetic dataset has been used to calculate the recalls.

Results

Figure 6.7 shows the recall over the side length of the quadratic image crop. While the ADD-S recall starts and stays at a high value, the VSD-based recalls start with a low recall of <0.2. These recalls saturate for the range of 15-75 px and then drop slowly.

Moreover, Fig. 6.7 displays the mean number of inference steps for the 0.5 s inference budget. After starting with 441 iterations for a resolution of 2 px × 2 px, the number quickly drops to 333 iterations for 20 px × 20 px and steadily drops afterward to 303 iterations for a resolution of 100 px × 100 px.

Discussion

The results suggest that a wide range of resolutions is possible without any noticeable impact on the overall performance. Compared to color images, depth images also have a resolution for the third dimension, which is not reduced when cropping. Therefore, surprisingly low resolutions of 15 px × 15 px result in a good performance, even though the cat from Fig. 6.6 becomes unrecognizable for a

human. As a rule of thumb, if a human can recognize the object pose in an image, an algorithm should also be able to do so. Depending on the object size in the image, cropping might upscale the image if the target resolution is too high. In the best case, unnecessary computations are the consequence; in the worst case, the extrapolation might introduce additional errors.

6.3.2 Optimal Number of Particles and Inference Time

Samplers might benefit from more particles in multimodal problems to avoid local minima. Moreover, sampling algorithms generally profit from more inference steps. This section aims to find the optimal number of particles which maximizes the recall. Based on this choice, a tradeoff can be determined between the inference time and average recall of the pose estimates.

Consequently, the samplers have been executed by combining different numbers of particles and inference time budgets. Again, only a single scene from the LM, T-LESS, and ITODD datasets has been used to keep the experiment time tractable.

Results

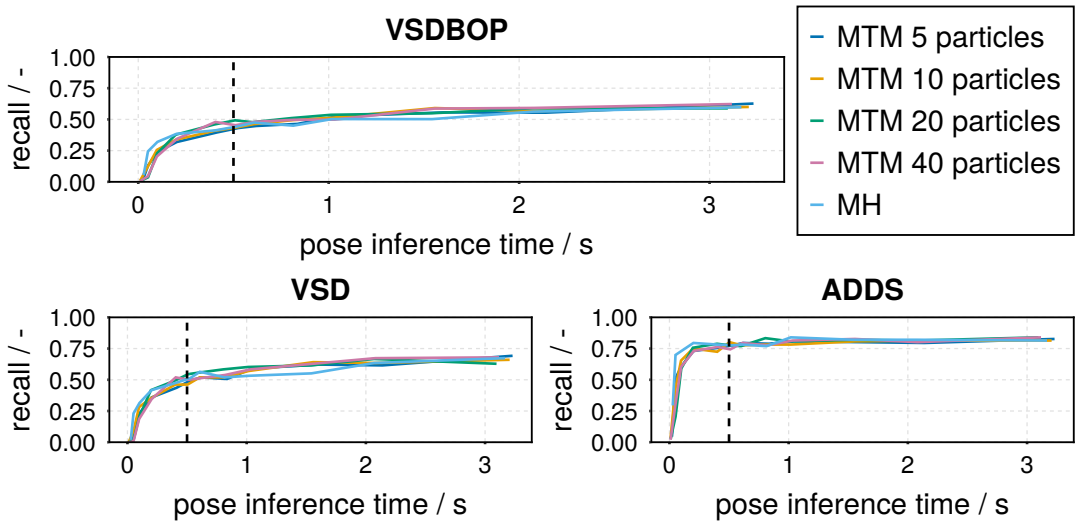


Figure 6.8: Recalls of chain-based samplers on synthetic data for varying particle numbers and inference steps. MTM uses particles, MH uses only a single sample per inference step.

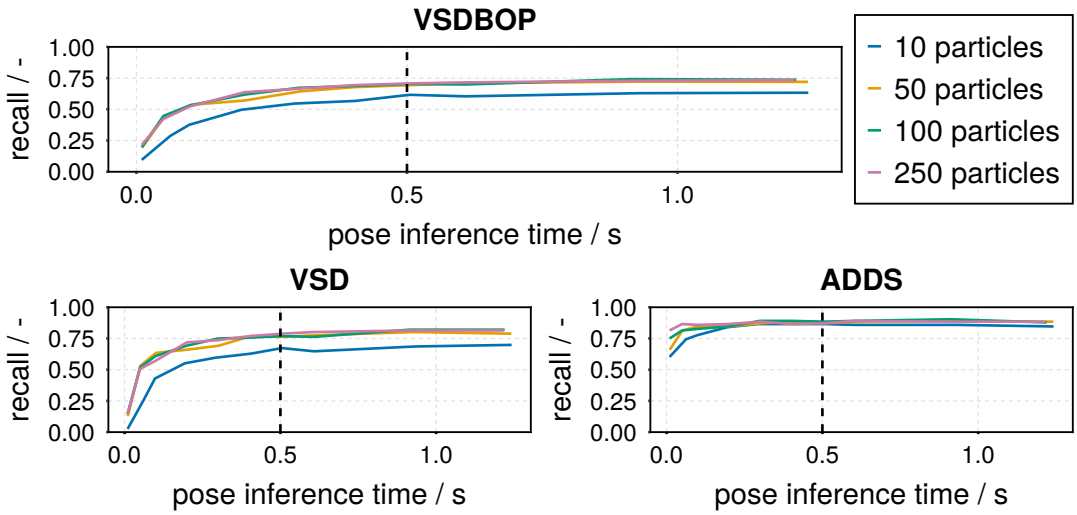


Figure 6.9: Recalls of the SMC sampler for varying particle numbers and inference steps.

Figure 6.8 presents the results for the chain-based MH and MTM algorithms. A vertical dashed line at 0.5 s highlights the target inference time. Once more, all curves show a saturating behavior except the ADD-S curve, which is almost a step function. No significant difference is observable for a varying number of particles in the MTM sampler. Besides, the recall curve rises slightly faster for the MH algorithm than the MTM variants. The VSD-based recalls do not saturate even if the algorithms run for 3 s.

Figure 6.9 shows the results for the SMC algorithm. The curves represent the combined recalls for all three datasets plotted over the measured inference time. All curves show a saturating behavior; only the ADD-S recalls almost represent a step function. The VSD recalls saturate after ≈1 s. Notably, all particle counts lead to a similar performance except a low particle count of ten.

Discussion

Most noticeably, the SMC algorithm outperforms the chain-based algorithms for lower inference times. Even when running the MCMC and MTM algorithms for more than five times longer, the recalls are still lower than the SMC. The MH and MTM recalls have not saturated yet, indicating that the chain-based algorithms eventually sample from the posterior. However, it takes very long for these samplers to escape local minima by generating a good sample from the prior.

The performance of the MTM sampler shows no significant difference for different particle numbers. It also performs similarly to the MH sampler, even though it should have a higher acceptance rate [87]. Much of the literature on approximate Bayesian inference focuses on the acceptance rate to evaluate the algorithm performance, dismissing that more evaluations result in longer iterations. Additionally, the general MTM algorithm has the drawback of requiring two model evaluations per proposal. In summary, no improvements are expected in practical applications with limited runtime. Finally, the MTM algorithm is more complicated than the MH algorithm, so the latter is the preferred one.

A difference is only noticeable between the SMC samplers for the low number of ten particles. All other particle numbers lead to the same performance, which enables the targeted time of 0.5 s per pose inference. Using at least 50 particles might help to spread the initial particle wide enough, so some particles are close enough to the true pose and avoid local optima. For weaker hardware, the runtime might jump suddenly if the memory requirements become too high due to the number of particles and image pixels. These jumps are likely due to the large memory allocations required on the GPU. Using 100 particles for the SMC sampler is a sensible choice.

6.4 Quantitative Evaluation of Baseline Samplers on Synthetic Data

This section quantitatively evaluates the baseline configurations of the MH and SMC samplers, as described in Sections 6.2.1 and 6.2.2. The baseline configuration has been executed on the full synthetic LINEMOD, T-LESS, and ITODD datasets to calculate the recalls and error densities.

Results

Figure 6.10 shows the recall curves for the three baseline samplers over the error thresholds. Because the errors are normalized, all error thresholds range from zero to one. The highlighted areas with the vertical lines represent the thresholds from the BOP challenge to consider a pose estimate correct. ADD-S and VSD use a single threshold, while VSDBOP uses a range of thresholds to average the recalls. One observation is that the ADD-S recall curves are similar for all three samplers. Besides, the ADD-S curves have a much steeper slope than the other two error metrics. Characteristically, the VSD and VSDBOP curves feature

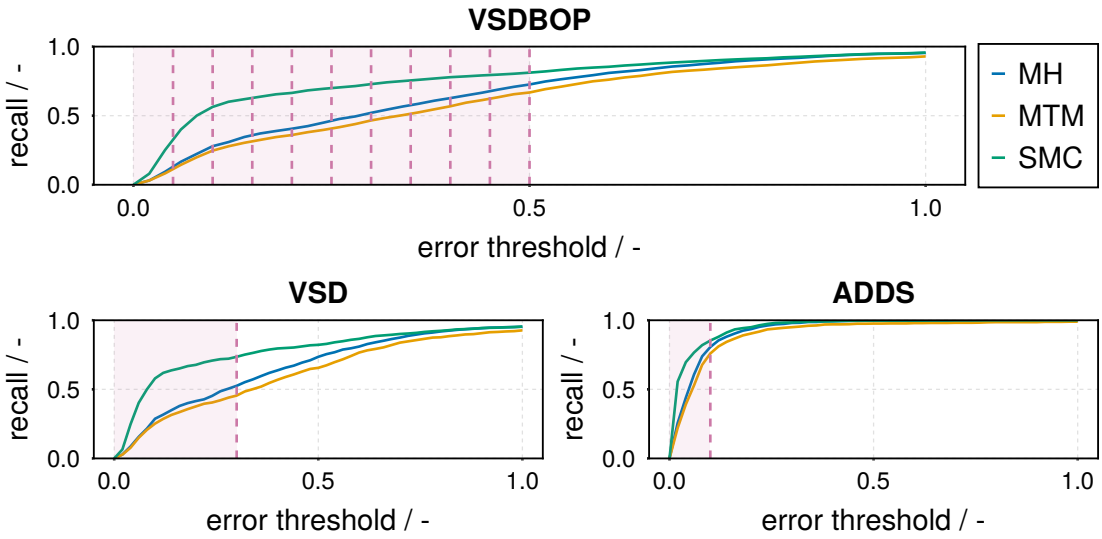


Figure 6.10: Recalls of the baseline configuration on synthetic data.

Table 6.4: Recalls of the baseline samplers on synthetic data.

sampler	ADD-S	VSD	VSDBOP	average
MH	0.806	0.526	0.477	0.603
MTM	0.759	0.455	0.428	0.547
SMC	**0.853**	**0.735**	**0.675**	**0.754**

the same characteristics, with the only difference that VSD raises slightly faster than VSDBOP. Moreover, the chain-based algorithms' curves are almost linear for these metrics, while the particle-based SMC features a bump and outperforms the other two samplers. The MTM sampler performs slightly better than the MH sampler.

Figure 6.11 provides insights into the distribution of the pose errors. These density plots are directly related to the recall curves, as the recall is the sum of the errors distributed below a threshold. Most errors are located at the extremes: The errors are either very low or very high. While hte VSD-based metrics have some errors distributed between the limits, the errors of ADD-S are located almost exclusively at the limits.

Finally, Table 6.4 shows the recalls based on the remarks from Section 6.1.4. By a large margin, the SMC algorithm has the highest score for all metrics. MH and MTM perform similarly.

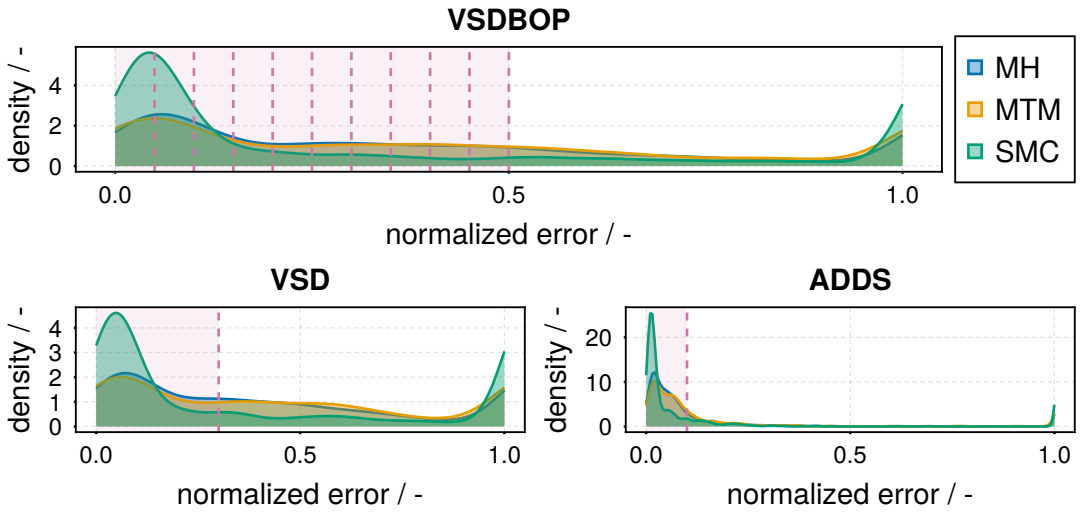


Figure 6.11: Density of the pose errors for the baseline configuration on synthetic data.

Discussion

One reason for the sub-par performance of the chain-based samplers could be that these samplers require a different parametrization to perform well. However, a more likely explanation is that the chains get stuck in local optima, and random global moves rarely escape from these optima, as clarified in the qualitative analysis of Section 6.2.3. This section validates what has been outlined in Section 6.3.2: The performance characteristics of the MH and MTM algorithms are comparable, with the latter performing slightly worse. Even though MTM might theoretically have a higher acceptance rate, approximately the same number of pose hypotheses can be evaluated given a constrained time budget. Therefore, the suggestion is to prefer the MH over the MTM algorithm for its more straightforward implementation.

Looking at the distribution of errors in Fig. 6.11 signals that the SMC sampler either converges to the true pose or diverges to a wrong estimate. The errors of the chain-based samplers are spread out over the whole interval. As the initial sample is drawn from the global prior, these spread values could be local optima close to this global initialization. These density plots also highlight the similarity of the average recall scores for the VSDBOP and VSD error metrics. Most of the VSDBOP thresholds contain the same error distribution as the single VSD threshold. One threshold for the VSD-recall is cheaper to evaluate than multiple thresholds for the VSDBOP recall. Hence, the VSD recall should be used if many evaluations are required, e.g., for automatic parameter tuning. On the contrary,

the VSDBOP is a good benchmark choice since it provides a better summary of the recall-threshold curve and saturates later.

6.5 Ablating the Probabilistic Model

Until now, the focus has been on ablating the inference algorithms. The SMC sampler outperformed the other samplers in the previous experiments. Thus, it is the only algorithm used during the ablation of the models.

For the prior model, three choices are available: using a segmentation mask from a learned model, a point prior from a localization sensor, or both. Additionally, the likelihood can use different probability distributions to model occlusions, pixel classification can be modeled, and different regularization strategies can be used. The following sections will evaluate the average recalls for these combinations on the full synthetic LM, T-LESS, and ITODD datasets.

6.5.1 Choice of Priors for Position and Classification

Section 4.2.1 proposed using segmentation masks from a machine learning model or additional sensors for localizing the position of a point, e.g. from RFID tags. Until now, all experiments only used a point prior. This section examines using masks to calculate the point prior and as a prior for the pixel-object classifications; see Section 4.3.5. Since recent deep learning-based segmentation models perform well on a wide variety of tasks, the probabilities for the mask represent a high confidence of $p(c_o) = 0.7$ and $p(\overline{c_o}) = 0.3$. Using only the point prior, e.g., from an RFID sensor, involves a flat prior $p(c_o) = p(\overline{c_o}) = 0.5$ for the pixel classification. When both prior sources are used, the point prior is used for the position, and the mask prior is used only for the observation model.

Results

Figure 6.12 displays the recalls for each prior and each synthetic dataset separately. In almost all cases, the prior models achieve the best results on the LM dataset, followed by the T-LESS and ITODD datasets. Moreover, using both prior sources results in the highest recalls. Using only the point prior achieves better results on the LM dataset, while the mask prior achieves better results on T-LESS and ITODD.

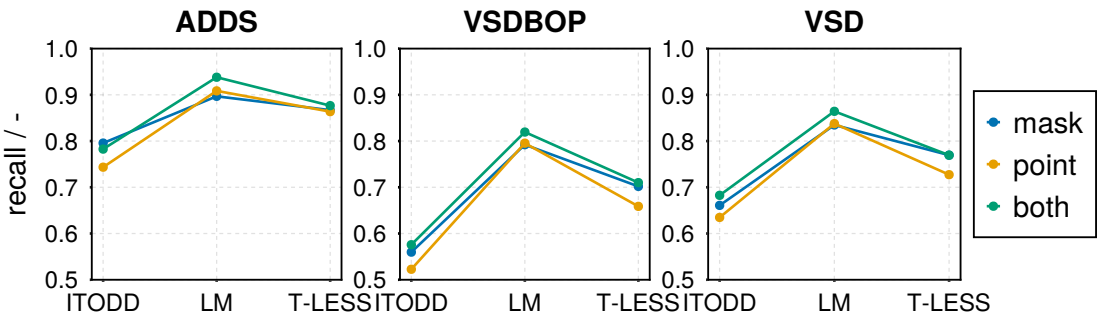


Figure 6.12: Recalls on synthetic data for different prior choices.

prior	ADD-S	VSD	VSDBOP	average
point	0.849	0.746	0.676	0.757
mask	0.860	0.766	0.700	0.775
both	**0.876**	**0.783**	**0.717**	**0.792**

Table 6.5: Recalls on synthetic data for different prior choices. The best results are bold.

Discussion

The point prior model is the same as the one from the baseline in Section 6.4 and achieves similar sores. Compared to this baseline, the mask prior scores ≈1.5 % points better, and the combination of mask and point prior scores ≈3 % percent points better. The improvement from the point to the mask prior is likely due to an improvement in the likelihood, which can focus more on the visible object parts. The reason for the improvement from the mask prior compared to using both is most likely due to the bias of the calculation of the position prior; see Section 4.2.1. In conclusion, more prior information should be used if it is available. However, the similar scores underline that the SMC sampler and probabilistic models are generalizable enough to perform similarly well regardless of the prior source.

To classify the difference in percent points: A difference of one to three percent points between the mask and the point prior can be enough to score about five places better on the BOP 2023 leaderboard. The BOP leaderboard also shows that methods typically achieve much lower ITODD scores than on the other datasets. A possible explanation is that ITODD features smaller objects, such as screws, which provide less information on the pose.

Most large-scale scenarios like logistics do not offer point priors since attaching RFID tags to all objects is a significant cost factor. Therefore, using a trained object detector and segmentation masks is often cheaper and easier to implement.

On the other hand, regulations in medical applications like the sterile supply require precise instrument tracking. Thus, RFID tags might already be available. Moreover, surgeons can point to the object, so point priors are more likely to be available.

To classify the present capabilities of mask priors: Based on the most recent BOP leaderboards[3], zero-shot segmentation models perform much worse than models trained on known objects. On the other hand, this thesis assumes that CAD models are available for generating synthetic training data. If the number of classes is not too large, models trained on synthetic training data perform similarly[4] to models using real data.

6.5.2 Modeling Occlusions: Exponential Distribution, Classification, Regularization

Occlusions reduce the information on the object pose, cause more ambiguities, and are thus challenging to model. Ergo, different exponential distributions for modeling the measurement of occlusions and pixel-object classification have been presented as possible solutions in Sections 4.3.3 and 4.3.5. The experiments in this section compare the recalls for different combinations of the:

- exponential distribution: Using **no** exponential to determine whether modeling occlusions is required or if the uniform distribution for modeling outliers suffices. Using an unmodified **exp**onential or a **smooth** truncated exponential to test if more precise modeling improves the estimates.
- classification: How does modeling the pixel-object classification or not (**yes** or **no**) change the performance?
- regularization: If the pixel-object classification is modeled, these estimates can be used for the L_0 class regularization. This regularization is compared to the L_{px} pixel count regularization.

Because the choice of the classification prior directly influences the mixture model of the likelihood function, it is expected to lead to different results. Thereby, the same prior choices from the previous Section 6.5.1 are compared: a flat prior of $p(c_o) = p(\overline{c_o}) = 0.5$, and a mask prior with $p(c_o) = 0.7$ and $p(\overline{c_o}) = 0.3$.

[3] October 2023 average precision: 0.412 (zero-shot), 0.619 (known objects)
[4] Average precision: 0.579 (synthetic only), 0.619 (synthetic and real)

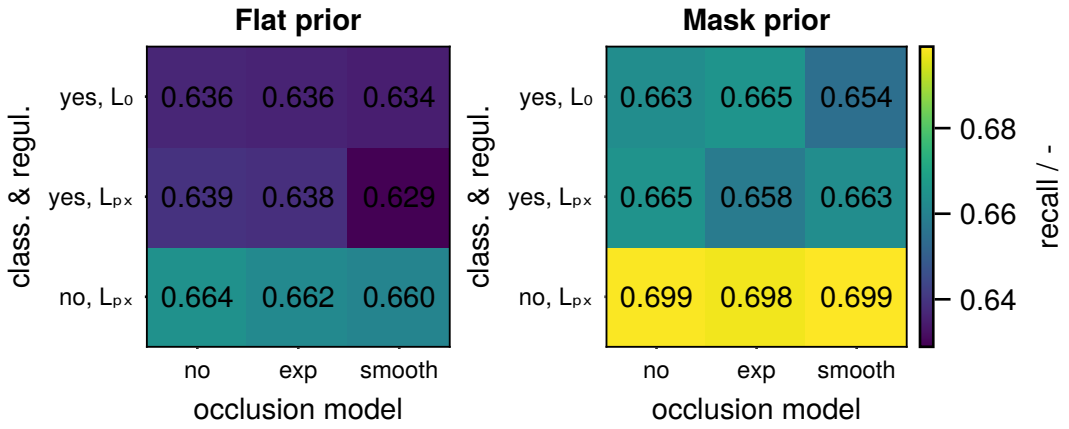


Figure 6.13: VSDBOP recalls for combinations of different exponential distributions with classifications and regularizations.

Results

Figure 6.13 presents the average VSDBOP recalls in two matrices. The left matrix presents the recalls for the flat prior, which is indifferent and does not include the masks. In contrast, the masks were used to assign a classification prior to pixels in the right matrix. All scores increase from the flat to the mask prior by $3 - 4\,\%$ points.

The rows contain the results for using or not using pixel classifications and the different regularization types. Using object classifications results in similar scores for the upper two rows, regardless of the regularization strategy. For both prior choices, the results of not using the pixel classifications score $3 - 4\,\%$ points higher. Combining mask priors with pixel classifications results in scores almost as good as the flat prior without classifications.

Different choices for the occlusion models are displayed in the columns. There is no significant difference for different occlusion models.

Discussion

Intuitively, more complex models should be able to represent complex relationships more accurately. Still, similar to other optimization domains like model predictive control, a simple occlusion model produces results similar to more complex ones. This result suggests that occlusions can be interpreted as outliers, so modeling them using a uniform distribution is sufficient.

Table 6.6: Bounds of the tunable parameters. * is only required in MH sampler.

parameter	$p(c_o)$	σ_z	σ_{r+1}	$\sigma_{t+1}{}^*$
value	0.5-1	0.1-100 mm	0.01 – π rad	1-100 mm

Similar to Section 6.5.1, using the mask prior improves the result of the model without pixel-object classifications. One has to remember that the synthetic dataset contains perfect mask annotations, so results on real data with imprecise mask estimates might result in worse results.

Like complex occlusion models, pixel classification has been developed to improve the model's performance under heavy occlusions by focusing on the relevant parts. Nevertheless, the result is the opposite and generally degrades the performance. One possible explanation is that the classification model utilizes the same models used for the likelihood of the depth measurements. In contrast, machine learning models for segmentation can use additional information from the training data. Therefore, this model can infer no additional information, and it lends itself more to produce an additional refined mask output for downstream tasks.

6.6 Automatic Parameter Tuning

Tuning the probabilistic model and sampler aims to find a good tradeoff between exploration and exploitation. An automatic tuning algorithm can determine this tradeoff systematically and improve the performance of the samplers. Moreover, a fair comparison is only possible for properly tuned algorithms. Specifically, this section re-evaluates the tuned MH and SMC algorithms using optimized parameters.

Since the parameters are optimized for the BOP benchmark, segmentation mask priors are available, but no point priors from additional sensors. Following the best performance from the previous section, a model without pixel-object classifications is tuned. The choice of the occlusion models and regularization strategy is arbitrary and uses the unmodified exponential due to its simplicity.

Table 6.6 gives an overview of the tunable parameters and their ranges. These parameters are only a subset, since tuning all parameters is intractable due to the high dimensionality and evaluation times. The ranges exceed reasonable values by one magnitude to allow exploration of unintuitive values. Only one parameter is tuned for highly correlated parameters to avoid singularities in the optimization

landscape. Moreover, multivariate variables are simplified by parametrizing them using only a single standard deviation for all components.

The weight of the prior and likelihood is primarily determined by the standard deviation σ_t of the position prior, the regularization constant c_{reg}, and the standard deviation σ_z of the depth sensor noise. A reasonable parametrization for the position prior based is possible based on typical object sizes, which are in the scale of centimeters. The parametrization of the depth sensor noise also influences the weight of measuring the object versus measuring occlusions or outliers. Thus, σ_z is tuned to balance multiple objectives, and c_{reg} stays fixed.

Tuning the proposal models calibrates the tradeoff between exploration and exploitation. The MH sampler requires tuning the position and orientation proposals via their standard deviations σ_{t+1} and σ_{r+1} respectively. Because of the adaptive proposal for the position, only the orientation proposal has to be tuned for the SMC algorithm.

A readily available *bilevel centers algorithm* for parameter tuning[5] is used. Grid- and random-sampling-based approaches did not result in good scores after a reasonable number of iterations. Since different datasets contain different types of objects and scene configurations, the optimal parameters might vary per dataset. Therefore, each model-dataset combination is optimized separately. The optimizer is executed for 200 iterations, resulting in ≈2 h of tuning per combination.

Results

Table 6.7 shows the optimized parameter values and the corresponding VSD recalls using the SMC sampler. All pixel classification probabilities $p(c_o)$ increased from 0.7 to values of 0.86-0.9. The standard deviation σ_z of depth sensor noise decreased from 10 mm to 2.6-8.7 mm. While the optimum of the previous parameters is similar to the hand-tuned ones, the standard deviation of the rotation proposals is magnitudes larger and increased from 0.1 rad to 2.64 – π rad. Moreover, the recall increased significantly to 0.9 for the best per-dataset parameter sets from 0.74 for the baseline parameters.

The trends and magnitudes of the optimized model parameters for the MH are similar to the ones of the SMC sampler: First, the values for the pixel classification prior $p(c_o)$ are even higher at 0.9-0.99. Second, the standard deviation σ_z of the depth sensor noise is again similar to the hand-tuned one, albeit higher at 7.22-16.8 mm. Finally, the standard deviation σ_{t+1} of the position proposals is

[5] HyperTuning.jl: https://github.com/jmejia8/HyperTuning.jl

Table 6.7: Best parameter set and corresponding VSD recall using the SMC sampler.

dataset	$p(c_o)$	σ_z	σ_{r+1}	VSD recall
ITODD	0.863	2.679 mm	2.641 rad	0.878
LM	0.865	8.705 mm	π rad	0.938
T-LESS	0.901	4.887 mm	π rad	0.883

Table 6.8: Best parameter set and corresponding VSD recall using the MH sampler.

dataset	$p(c_o)$	σ_z	σ_{r+1}	σ_{t+1}	VSD recall
ITODD	0.987	7.219 mm	2.476 rad	3.414 mm	0.634
LM	0.907	16.823 mm	π rad	10.524 mm	0.672
T-LESS	0.984	13.702 mm	2.973 rad	4.567 mm	0.706

Table 6.9: Selection from optimized parameters similar to SMC for ITODD. * is only required in MH sampler.

parameter	$p(c_o)$	σ_z	σ_{r+1}	$\sigma_{t+1}{}^*$
value	0.9	5 mm	π rad	10 mm

Table 6.10: Recalls of the tuned samplers on the synthetic validation data.

sampler	ADD-S	VSD	VSDBOP	average
MH selection	0.812	0.529	0.477	0.606
MH individual	0.783	0.513	0.452	0.583
SMC selection	**0.877**	0.776	0.724	0.792
SMC individual	0.875	**0.785**	**0.735**	**0.798**

similar to the hand-tuned one with a range of 3.4-10.5 mm compared to 10 mm. The proposal distribution for the orientation is again parametrized by very high values of σ_{r+1} close to π.

Table 6.10 contains the recall values with optimized parameters on the synthetic validation scenes, which have not been used during the optimization. Since the optimized parameters show a similar trend, the recalls were evaluated on a *selection* of the parameters. Table 6.9 contains this selection, which is based on the optimization results for the SMC algorithm on the ITODD dataset from Table 6.7. Additionally, the *individual* best per-dataset parameters were evaluated.

For the SMC sampler, the individual parametrization outperforms the selection on the VSD recalls by $\approx$1 % point, but the ADD-S recall is almost the same. In

contrast, the selection outperforms individual parameters for the MH sampler. Compared to the baseline configuration with mask priors from Sections 6.5.1 and 6.5.2, the improvements for the VSD errors range between 0 – 3 % points. Overall, it is essential to note that the recalls on the holdout data are much lower than the ones of the best runs per dataset.

Discussion

In most cases, the optimized parameters are in the same order of magnitude as the hand-tuned baseline parameters, indicating that the hand-tuned values were reasonable choices. Only the standard deviation of the local rotation proposals σ_r is many magnitudes larger than the baseline. A possible conclusion could be that exploration is more important than exploitation, since no prior information exists on the orientation. If the accuracy of the estimates was the optimization goal instead of the recalls, the results would likely be different because the recalls tolerate some error by using a threshold to discriminate between correct and wrong estimates.

The similar recall scores of the baseline, selected, and individual parameters demonstrate that the algorithm is insensitive to the parametrization. As long as the parameters are in the same order of magnitude as the presented ones, they should work similarly well. At least for the three BOP datasets used during the optimization, an individual parameter optimization shows no significant benefit. However, most objects' sizes and shapes are relatively similar in these three datasets. For example, only a few long, thin, and tiny objects are present.

Finally, some skepticism is necessary towards the parameter tuning results, since the VSD recalls during the optimization are much higher than those for the holdout data. This behavior typically indicates an overfitting, which might be the case here, since a single scene does not contain many annotated poses. However, rerunning the sampler using these parameters on the same scenes resulted in scores similar to the ones on the holdout data. Consequently, the high scores of the best runs are most likely due to a lucky random seed leading to an excellent initial particle distribution. One possibility to reduce the effect of the randomness is to increase the amount of data used during the parameter optimization, which quickly leads to multiple hours or days of runtime.

In conclusion, the performance of the inference is insensitive to the selected parameters, and the evaluation results during the optimization steps are very noisy. Therefore, the parameter optimization from this section can only improve performance to a limited extent.

Table 6.11: Recalls of the best sampler on synthetic and validation data.

dataset	ADD-S	VSD	VSDBOP	average
synthetic	0.875	0.785	**0.735**	**0.798**
validation	**0.899**	**0.790**	0.654	0.781

6.7 Benchmark for 6D Object Pose Estimation: Results on Real Data

The BOP validation datasets for ITODD, occluded LM, and T-LESS are used to evaluate the synth-to-real transferability of the results from the previous sections. This section only evaluates the best-performing model-sampler configuration: an SMC sampler using an unmodified exponential for occlusions, which does not model the pixel-object classifications and uses the tuned selection of parameters from Section 6.6. The ground truth masks are used instead of detections from a machine learning model to improve comparability with the previous sections.

Later, the SMC-based pose estimation is executed on the much larger BOP test sets to compare it to the state of the art. For the test sets, no ground truth masks are available, but the BOP challenge 2023 provides default segmentation masks[6] from a machine learning model [40]. Moreover, the ground truth poses are unavailable to the participants. Instead, the challenge provides an official submission form[7], guaranteeing that all submissions are comparable. Compared to the previous sections, the BOP system averages the VSDBOP, MSSD, and maximum symmetry-aware projection distance (MSPD) recalls, to include augmented reality applications.

Results

Table 6.11 compares the results on the real validation data to the results on the synthetic data from section Section 6.6. Moreover, Fig. 6.14 the corresponding densities of the pose error metrics. The recalls for the ADD-S and VSD metrics are almost identical for the synthetic and real validation data. Only the VSDBOP recalls dropped significantly by $\approx 8\,\%$ points. Compared to the error densities from the baseline evaluation in Fig. 6.11, the peak of the error values is shifted to higher values and compressed.

[6] GDR-Net masks for task 1: https://bop.felk.cvut.cz/challenges/bop-challenge-2023/
[7] http://bop.felk.cvut.cz/sub_upload

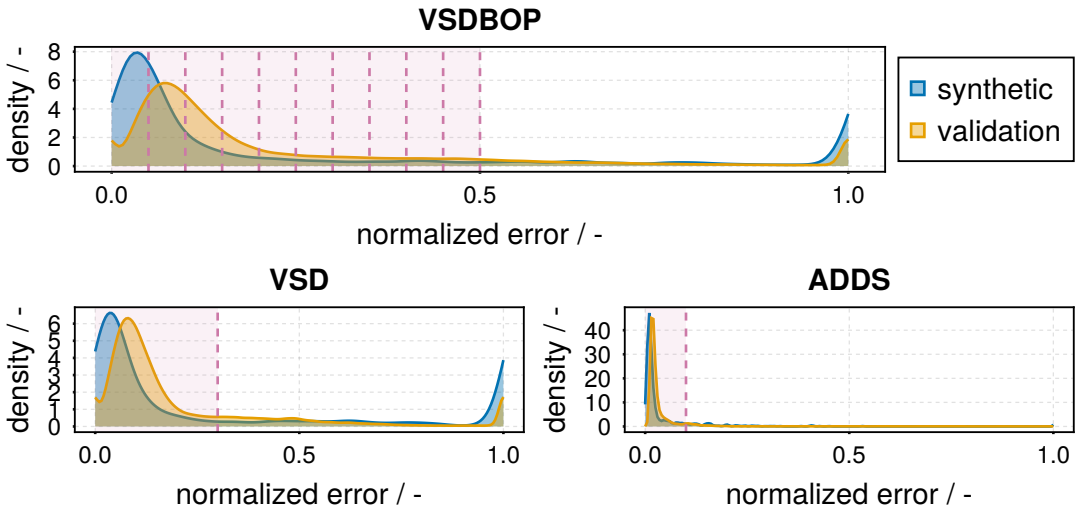


Figure 6.14: Density of the pose errors for the tuned models on the synthetic and the validation datasets.

Table 6.12 shows the results on the test data using the official BOP leaderboard[8]. Selected methods from the leaderboard November 2023 are included and sorted with the best average recalls on top. GPose2023 is the best-performing method for seen objects and gives an impression of possible scores. Compared to the localization of seen objects, localizing unseen objects does not allow training on the objects from the dataset. This work has been evaluated on the later task with default detections. GenFlow is the current leader in localizing unseen objects using default detections. SAM6D is the only method that outperforms the SMC sampler while requiring significantly less than 10 s per image for the unseen object localization required for robotic manipulation tasks (November 2023). Most methods require more than 30 s per image on this task. The SMC sampler from this work has been executed with different compute budgets and is called SMC-<time>s-CNOS. All methods except the SMC sampler use a neural network with a fixed compute budget. In contrast to this work, which uses only depth images, all other methods also use the corresponding color images.

Compared to the overall best method, the average recalls are much lower, with a difference of almost 40 % points. This difference shrinks to ≈20 % points if the methods use the default detections and do not see the objects during training. This difference shrinks even further to ≈14 % points compared to SAM6D, the only method with a comparable runtime for localizing unseen objects. Notably,

[8] https://bop.felk.cvut.cz/leaderboards/pose-estimation-unseen-bop23

Table 6.12: Average recalls from BOP submission for the task: 6D localization of unseen objects. GPOSE2023 is grayed out since it was submitted to the 6D localization of seen objects task.

method	LM-O	T-LESS	TUD-L	IC-BIN	ITODD	HB	YCB-V	avg.	$seconds/image$
GPose2023	**0.794**	**0.914**	**0.964**	**0.737**	**0.704**	**0.950**	**0.928**	**0.856**	**2.670**
GenFlow-MultiHypo16	0.635	**0.521**	**0.862**	**0.534**	**0.554**	**0.779**	**0.833**	0.674	34.578
SAM6D-CNOSmask	**0.648**	0.483	0.794	0.504	0.351	0.727	0.804	0.616	3.872
SMC-1.0 s-CNOS	0.558	0.423	0.599	0.316	0.389	0.585	0.454	0.475	6.051
SMC-0.5 s-CNOS	0.512	0.415	0.511	0.290	0.358	0.538	0.403	0.433	**2.992**

the most considerable differences occur on datasets with texture-rich objects, such as TUD-L and YCB-V. The difference shrinks for datasets focusing on industrial objects, such as T-LESS and ITODD. On ITODD, the SMC sampler even outperforms SAM6D. Finally, the scores of the SMC inference improve significantly by increasing the per-object inference time from 0.5 s to 1 s.

Discussion

If ground truth masks are known, the Bayesian pose estimator performs similarly on synthetic and real data. While the distribution of the VSD and VSDBOP errors both shifted to higher values, only the VSDBOP recall declined. The VSD threshold from the BOP 2018 challenge for considering a pose correct is high enough that the recall does not get affected. In contrast, averaging several thresholds in VSDBOP successfully discriminates this density shift. Still, the VSD recall is more suitable for the targeted robotic applications than VSDBOP as it does not include scaling; see Section 6.1.4. Therefore, the sim-to-real performance is considered good. This performance also implies that synthetic datasets can evaluate the algorithms quantitatively in applications without annotated data.

Compared to the state-of-the-art methods in the BOP leaderboard, the Bayesian inference pose estimator has much improvement potential. However, the quality of the default segmentations provided by BOP is also much lower than the best ones for seen objects, with an average precision of only 0.412[9] compared to 0.619[10]. The Bayesian pose estimation results exhibit a reasonable performance compared to other methods using only default detections. GenFlow indicates that higher scores are possible, but they use much more computation time with a more capable NVIDIA A100 GPU and additionally color images. SAM6D also uses

[9] CNOS (FastSAM) https://bop.felk.cvut.cz/leaderboards/segmentation-unseen-bop23
[10] ZebraPoseSAT-EffnetB4 https://bop.felk.cvut.cz/leaderboards/segmentation-bop22

depth and color images and performs better than the SMC sampler. Using color images might be the differentiating advantage, as the most considerable differences occur on texture-rich datasets. This conclusion is similar to the ablation of the prior information in Section 6.5.1: Using more information generally improves performance; in this case, more information is available to the observation model instead of the prior model.

A major advantage of the sampling-based methods from this work over the neural networks is that the computing budget can be limited as required. A reasonable performance can be expected even for a per-object time budget as low as 0.5 s. However, the performance difference from 0.5 s to 1 s compute time suggests that more runtime optimizations are required to allow more inference steps per pose estimate. Moreover, the scores on the validation dataset indicate that the Bayesian pose inference can produce results similar to the state-of-the-art given good segmentation masks.

6.8 Summary

This section establishes the evaluation methodology, including datasets and relevant pose error metrics[11]. The ADD-S is one of the oldest metrics, invariant to object symmetries, and enables comparability to the literature. However, the experiments demonstrate that this metric fails to differentiate the performance, as ADD-S saturates very early. In contrast, the selected VSD and VSDBOP metrics successfully differentiate the performance while being invariant to object symmetries and occlusions.

Starting by constituting a hand-tuned baseline configuration, the presented models and samplers have been ablated on synthetic datasets similar to real datasets from the BOP challenge. Even the first qualitative results demonstrate that the chain-based MH and MTM samplers perform much worse than the particle-based SMC sampler. Using multiple parallel hypotheses seems to help to explore the multimodal posterior distribution. The chain-based samplers might be able to achieve similar results but require much longer inference times. Benchmarking the samplers for different image resolutions and particle counts shows that the performance is almost constant for both parameters if they are not excessively low. Resolutions as low as 30 p × 30 pixels can be sufficient to distinguish poses. Even the baseline model configuration with the SMC sampler can achieve an average

[11] Implementations: https://github.com/rwth-irt/PoseErrors.jl

VSD recall of 0.735, which should enable a successful robotic manipulation of most objects.

Different prior sources have been compared in this chapter for the quantitative ablation. Naturally, more sources of prior information and more precise priors help to achieve better results. Using precise point priors, which could be provided by RFID tags attached to objects and segmentation masks, helps to increase the average VSD recall to 0.783. Using more complex models for occlusions proves unnecessary, since even simple models that interpret occlusions as outliers perform similarly. Moreover, explicitly estimating the pixel-object classification probabilities decreases the performance and should only be used as an additional output, not as part of the probabilistic model. Finally, the automatic parameter tuning does not significantly improve the performance of the models, with an average VSD recall of 0.785. A much larger tuning dataset would be required to overcome overfitting the random noise, resulting in hours or even days required for tuning. It can be suggested that simple models suffice, and the algorithms are insensitive to the exact parameter choices, so a simple tuning might suffice.

Finally, the best-performing model sampler configuration[12] has been evaluated on BOP validation, and test sets. During the validation, ground truth masks have been available. The validation results show, that the performance of the Bayesian pose estimation on real datasets is comparable to that of synthetic datasets. State-of-the-art methods perform much better than the SMC sampler on the test set of the BOP challenge. One reason for the performance difference is that the BOP default segmentation masks used to evaluate the SMC sampler are much worse than the masks produced by the state-of-the-art methods. Still, methods that use the same default segmentations outperform the SMC sampler but with a lower difference. This difference might be explained by the other methods using color images as an additional input modality. These methods, in particular, score better on datasets with textured objects. Comparing state-of-the-art methods to the performance of the SMC sampler using ground truth masks on the validation data indicates that the SMC sampler could be competitive if better segmentation masks were available.

[12] Occlusion model: unmodified exponential, no pixel-object classifications, and an SMC sampler using a MH kernel.

7 Applications in Medical Robotics

Previously, the focus was on ablating and evaluating the methods and algorithms presented in Chapters 4 and 5 on standardized datasets and metrics. This chapter aims to determine how flexible these methods are by applying them to two medical applications.

The first application is the pose estimation of surgical instruments in the context of robotic bin-picking applications in Section 7.1. Compared to the previous datasets, surgical instruments are long, thin, and shiny, making them particularly challenging for camera-based pose estimation. The second application targets pose tracking of bones in the context of facial reconstruction surgeries in Section 7.2. For this application, the SMC algorithm from the previous chapters is reformulated as particle filter (PF) to track the 6D pose based on a stream of images. Moreover, Section 7.3 provides insight into the practical steps required to use the pose estimates in robotic manipulation tasks.

7.1 Pose Estimation of Surgical Instruments

Compared to most objects from the previous datasets, surgical instruments are typically long and thin. Sieves returned from the operating room expose much clutter of stacked instruments and might contain additional garbage. Additionally, many instrument classes look similar, for example, clamps and scissors. Furthermore, many models in slightly different sizes exist for each class. This variety makes it impossible to train a model that can reliably detect the exact model identification number. Due to regulations, instrument manufacturers attach RFID tags or print small QR codes on the instruments to document their flow efficiently. Therefore, additional sensors for point priors are considered to be available for the pose estimation.

As no sufficiently large dataset of real instrument poses exists, the algorithm is evaluated on the synthetic STERI dataset generated with BlenderProc. The sim-to-real gap has proven to be sufficiently small for the Bayesian inference pose estimation to generate interpretable results; see Section 6.7. Each scene of the

Table 7.1: Results for the STERI specific model parameter tuning using the SMC sampler.

dataset	$p(c_o)$	σ_z	σ_{r+1}	VSD recall
STERI	0.995	6.587 mm	0.105 rad	0.662

dataset consists of 20 objects sampled from 40 standard surgical instruments for orthopedic surgeries and two to four distractors. The instruments have been dropped onto a flat surface using a physics simulation to generate ten scenes and five views per scene. Moreover, the Zivid camera used to capture the instruments for robotic manipulation has very low noise and can filter specular artifacts. Therefore, the simulated and real data results are expected to be comparable.

This section only evaluates the SMC sampler and automatically tunes the model parameters as presented in Section 6.6. The sampler is executed on the remaining nine scenes to evaluate the performance with different time budgets, similar to Section 6.3.2. Since many surgical instruments are thin, the crop resolution is increased to 60 px × 60 px. Otherwise, only one or even zero pixels cover the thinnest parts. Running the SMC sampler using only a point prior without a mask fails in most cases; see Appendix A.4. Thus, the model uses the ground truth segmentation masks to generate interpretable results.

7.1.1 Results for Simulated Instruments

Except for the measurement noise σ_z, the tuned parameters from Table 7.1 are different from the ones of the BOP datasets in Section 6.6. The prior probability $p(c_o)$ for the classification prior is much higher and almost one. In contrast, the local orientation proposal's standard deviation σ_{r+1} is much smaller and close to the hand-tuned baseline configuration. Finally, even the best-performing parameter tuning run scored a VSD recall of only 0.662, which is the worst SMC performance in all experiments.

Visualizing the recalls in Fig. 7.1 is similar to particle-runtime-recall plots in Section 6.3.2. At the target runtime of 0.5 s, highlighted by the vertical dashed line, the VSDBOP and VSD have a value of ≈0.5. After ≈3 s, the VSD-based recalls start to saturate, which takes six times longer compared to the BOP datasets. However, the limits of these recalls are similar to the ones on the BOP datasets at scores of 0.75 – 0.8.

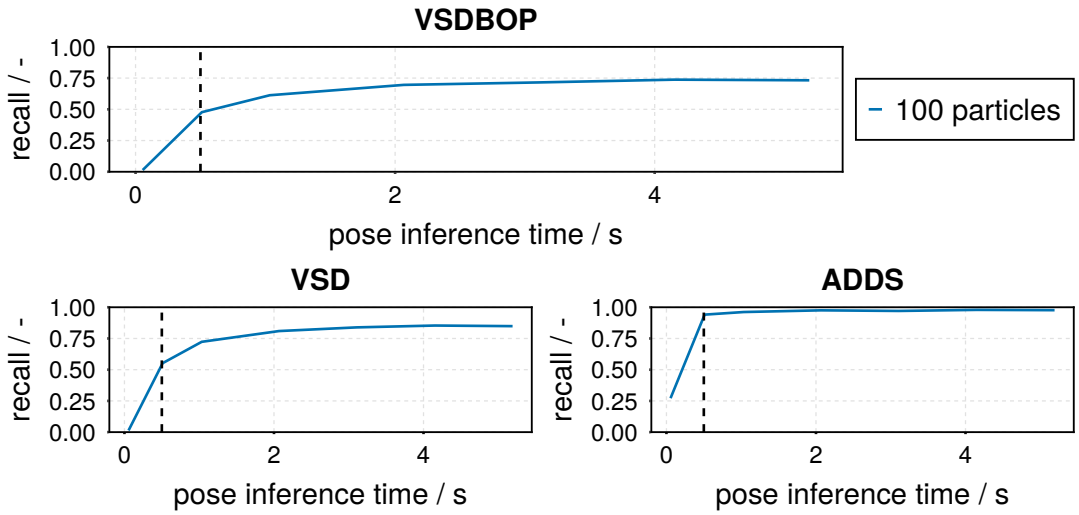


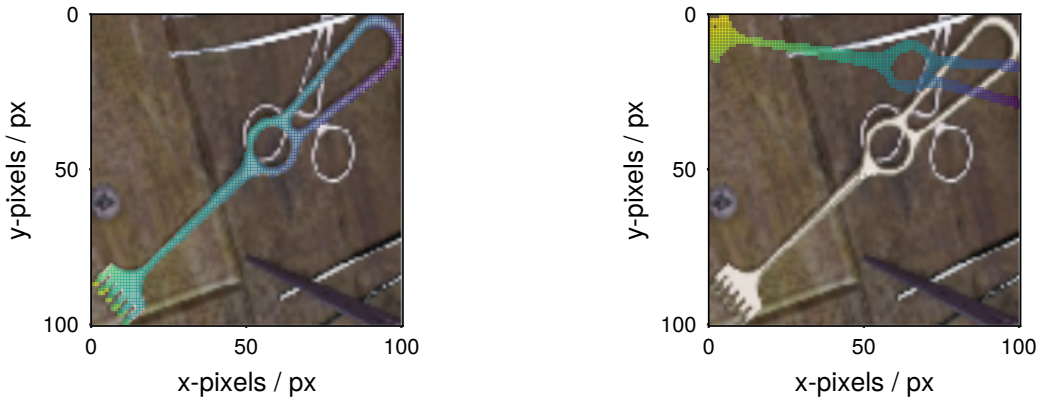
Figure 7.1: Recall over inference time for the STERI dataset.

7.1.2 Discussion on Pose Estimation of Surgical Instruments

The results suggest that estimating the pose of long and thin instruments is much harder for presented pose estimation algorithms than estimating the pose of objects from the BOP datasets. A first indicator of why it is so hard can be found in the tuned value of the classification prior $p(c_o)$ parameters. This value is almost one, meaning the algorithm requires confident masks to estimate a pose. On the flip side, this implies that if no masks are available, regions around the instrument have a high likelihood.

Furthermore, the inference time required to achieve similar recalls on the STERI dataset as on the BOP datasets is much larger. Longer times until convergence indicate that the probability of randomly generating a good pose estimate is much lower for these thin instruments. This lower chance points out a limitation of the pixel-wise comparison for calculating the image likelihood. Figure 7.2 shows that almost no rendered pixels overlap with the instrument pixels in the measured image if the orientation of the instruments deviates from the true orientation. In the case of only a few overlapping pixels, the pixel-wise likelihood fails to distinguish poses. Methods that extract and match features would be able to overcome this issue, e.g., by calculating a 3D distance instead.

Moreover, the frequent failures without masks can also be explained by the thin nature of the instruments. One explanation is that the likelihood of an instrument located just below the surface is similar to the likelihood of matching the instrument. Figure 7.3 validates this behavior, as the distribution of the x and z orientation



(a) Ground truth pose.

(b) Pose diverged into surface.

Figure 7.2: Overlay render of a surgical instrument for the described poses.

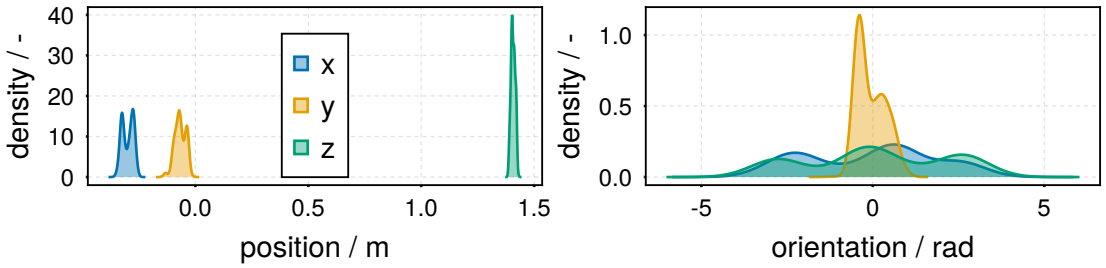


Figure 7.3: Density plots for the estimated poses in Fig. 7.2b

components is almost uniform. The constrained orientation matches exactly the surface as shown in Fig. 7.2b, and the instrument is free to rotate in the axis normal to the surface. Even simple segmentation algorithms for flat surfaces could counter this behavior. A more advanced approach would be to train a classifier that only has to know two classes: instrument or background. Learning two classes should simplify the training instead of learning to classify thousands of different instrument variants.

To summarize, Bayesian inference can be used for pose estimation of surgical instruments, but is significantly more challenging than previous datasets. Segmentation masks are a mandatory requirement to prevent the algorithm from diverging into flat surfaces, and the inference time budget should be at least 3 s instead of 0.5 s. One possible improvement could be a likelihood function, which contains semantic information such as object coordinates to compare non-overlapping parts; see Section 8.1.

7.2 Particle Filtering for 6D Pose Tracking of Iliac Crest Bones

In facial reconstruction surgeries, bone grafts, such as the iliac crest, can replace defective jaw bones [104]. A long-term goal is to control a robot to cut bones precisely so defective bones can be replaced seamlessly. In contrast to bin-picking applications, the scene is not static, as the surgeons or the robot might move the bone. One of our previous publications has shown that a particle filter can be used to track the 6D pose of an iliac crest bone in the context of facial reconstruction [74]. This section shows that the general SMC framework for the pose estimation on static images can also be applied to the state estimation of dynamic systems. In addition to the previous work, explicitly modeling pixel classification probabilities (Section 4.3.5), the choice of exponential distribution for modeling occlusions, and block-wise sampling (Section 5.2.6) of the position and orientation components are evaluated.

7.2.1 Motivation, Boundary Conditions, and Goal

Currently, 3D-printed templates are the gold standard for facial reconstruction [104]. Using computed tomography scans, surgeons digitally plan the templates, so a replacement bone fragment can exactly replace a defective bone. This approach is precise but unflexible, with long manufacturing and delivery times. The aim is to use robotic surgery for similarly precise but more flexible interventions [74].

Fully automated robotic surgeries will not be feasible soon, and surgeons will still handle most of the work collaboratively with the robot. In control theory, the surgeons' actions must be considered disturbances or a source of uncertainty for the filter algorithm. They might move a bone to a different position for better accessibility, so the system has to handle uncertain dynamics. Moreover, the surgeons might cause heavy occlusions when handling the bone. Even during the cutting procedure, mild to medium occlusions must be taken into account since at least the tools are in view.

Robust tracking with high precision and low noise is required to control a robot even under disturbances. A reasonable tradeoff is to allow larger deviations during occlusions and movements as long as the tracking does not diverge. While cutting, the surgeons keep the bone still, which requires less robustness but high precision with low noise.

Table 7.2: Particle filter parameters. The standard deviations of the motion model are time-normalized and without time units.

motion model			observation model				
$\ddot{\sigma}_t$	$\ddot{\sigma}_r$	λ_v	resolution	σ_z	β_z	$z_{min} - z_{max}$	$p(c_o)$
1 mm	0.001 rad	0.9	80 px × 60 px	1 mm	1 m	0.15-10 m	0.5

7.2.2 Motion and Observation Models

The motion model for the system dynamics is described using the same constant decaying velocity model as in [74].

$$\mathbf{x}_{t+1} = \mathbf{x}_t \oplus (\dot{\mathbf{x}}_t + 0.5\,\mathbf{a}_t) \tag{7.1}$$

$$\dot{\mathbf{x}}_{t+1} = \lambda_v\,\dot{\mathbf{x}}_t + \mathbf{a}_t \tag{7.2}$$

For simplicity, the discrete Euler integration uses a normalized $\Delta T = 1$, so the tracking rate is directly encoded in the noise vector $\mathbf{a}_t \sim \mathcal{N}(0, [\ddot{\sigma}_t, \ddot{\sigma}_r]^\mathsf{T})$ resembling a random acceleration. In Eq. (7.1), the $\oplus$ operator highlights that the state's orientation is a quaternion perturbed by the integrated angular velocity; see Eq. (4.13). The $\oplus$ operator becomes a regular addition for the position vector. Energy dissipation motivates the decaying velocity factor $\lambda_v < 1$ in Eq. (7.2) and prevents the filter from diverging due to integrating positive velocities over a long time.

The observation model is the same as the one used to infer poses in static scenes, but with the following changes Section 4.3. Cropping images introduces additional discretization errors, leads to jitter during tracking, and is not used: see Appendix A.2.6. Instead, the resolution of the images is increased to 80 px × 60 px. The experiments evaluate the robustness against occlusions and compare

- a *simple* model, similar to the previously published one. It uses an unmodified exponential to model occlusions and does not model pixel classifications.
- a *complex* model. This model uses the smooth truncated exponential to model occlusions, calculates the pixel class probabilities, and uses the L_0 class regularization.

Table 7.2 contains parameter values for the particle filters' model. These values are deliberately similar to the ones from the previous publication to aid comparability. Due to the high frame rate of 90 Hz of the depth camera, the motion model is characterized by low noises $\ddot{\sigma}_t$ and $\ddot{\sigma}_r$. A high decay factor λ_v allows higher velocities and enables tracking dynamic motions. Compared to the previous

sections' models, the depth camera's measurement noise is characterized by a very low standard deviation $\sigma_z = 1\,\text{mm}$. Moreover, σ_z is constant instead of a function of the previous publication's depth $\sigma_z = f(z)$.

Moreover, the previous publication's baseline filter implementation used a truncated exponential distribution to model occlusions. No pixel classifications have been modeled, nor has a regularization been used in the image likelihood. The baseline filter used only 500 particles to achieve 90 Hz on weaker hardware.

7.2.3 Filter Design

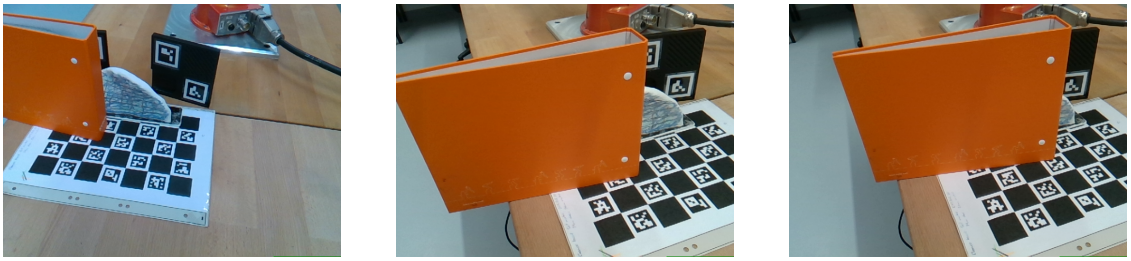
This work compares *jointly* proposing and evaluating a new position and orientation to a *block-wise* sampling as described in Section 5.2.6. When sampling block-wise, the sampler proposes one component, evaluates the likelihood, reweights the particles, and resamples them if required. The particle filter runs these steps twice per image to incorporate all observations into the position and orientation. Consequently, the block-wise sampler only uses 600 particles compared to the 1250 particles of the joint sampler to achieve the camera's frame rate of 90 Hz.

A similar strategy has been proposed in [105], deemed *coordinate particle filter*, but they sample each subcomponent, instead of blocks of variables. For example, they would sample the x component of the position instead of the 3D vector. Sampling two blocks is a compromise, reducing the number of particles by about two instead of seven times.

Instead of using the maximum likelihood or maximum-a-posterior estimate, the particle filter outputs the weighted mean of the particles. Compared to the other two alternatives, the mean can smooth the discretization error of discrete particles. Using the mean is only reasonable for unimodal distributions, which should be the case, as the motion model leads to very local proposals.

7.2.4 Evaluation

For the evaluation, the *smc pf* particle filter configurations are evaluated on the dataset from our previous publication [74]. A static bone phantom has been recorded at 90 Hz using a moving robot-mounted camera, which enabled annotating the camera's ground truth pose using the precise robot position measurements. The dataset features varying degrees of occlusion (see Fig. 7.4): the robot starts at a view with low occlusions, moves to a view with heavier occlusions, and finally back to the start position. All filter configurations performed similarly well for low



(a) mild occlusions **(b)** medium occlusions **(c)** heavy occlusions

Figure 7.4: Images from the dataset with different levels of occlusions [74].

to medium occlusions (≈20 – 50 %), so this work focuses on medium to heavy occlusions (≈50 – 95 %) to differentiate the configurations.

Two issues are present in this dataset: First, the timestamps of the robot and camera are not precisely synchronized. Second, the dataset is missing camera frames for ≈300 ms at 5 s and 10 s due to bottlenecks while recording the dataset.

Moreover, the particle filters from the previous publication are used as a baseline: The *at robot* particle filter had access to the robot kinematics and did not miss frames as opposed to the offline dataset. When not using the robot kinematics, the bone appears to move in the image and mimics a dynamic scenario without using additional markers to annotate the pose. This scenario was also tested previously by the *at only* tracker, which only used the camera images for tracking.

7.2.5 Results

For clarity, the results of the simple model with block-wise sampling and the complex model with joint sampling are located in Appendix A.7. For all trackers, the error increases towards the middle of the period and decreases towards the end. The baseline *at only* particle filter, which does not use the forward kinematics of the robot, is the one with the highest noise and jitter. Using the forward kinematics in *at robot* results in lower errors and noise. In all cases, the *smc pf* shows lower jitter than the *at* particle filters. As shown in Appendix A.7, using the maximum likelihood instead of the mean pose results in high-frequency noise.

Figure 7.5 shows the position and orientation errors of the filter using the simple model and joint sampling of the position and orientation. Besides, the figure depicts the relative ESS. This filter diverges after ≈5 s, where frames are missing and the movement direction changes. In contrast, the block-wise sampler using

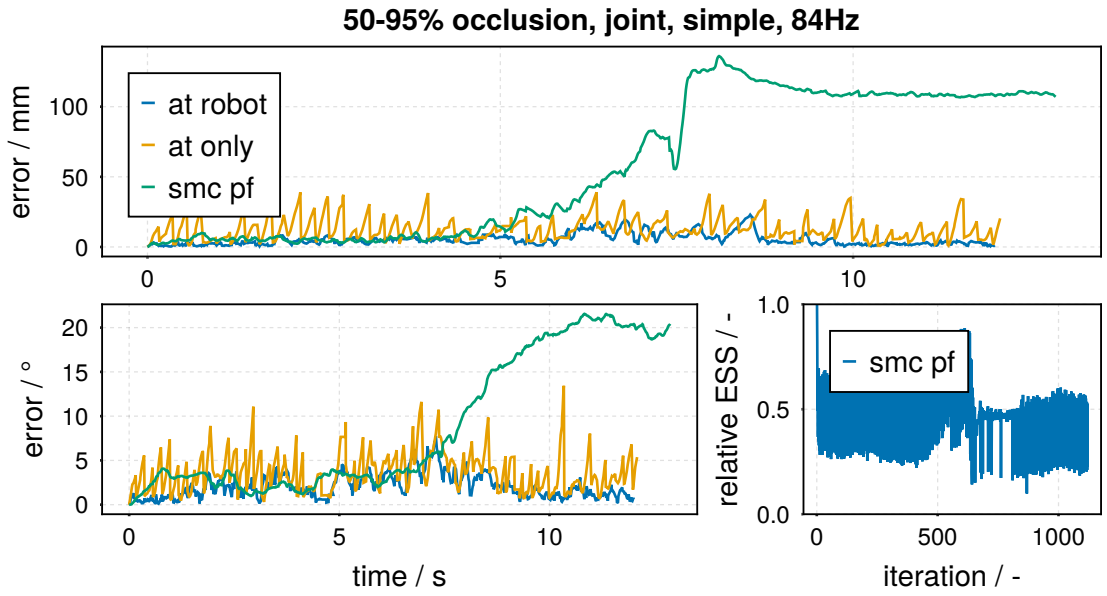


Figure 7.5: Particle filter results using joint sampling and an unmodified truncated exponential without pixel classifications.

the same model remains stable, albeit with significant position and orientation errors; see Appendix A.7. After diverging, the ESS drops below the resampling threshold of 0.5 for most iterations. Moreover, the ESS of the block-wise sampler in Appendix A.7 is generally higher than the ESS of the joint sampler.

Figure 7.6 shows one of the best-performing models for heavy occlusions featuring the complex model, which calculates the pixel-object classifications. Block-wise and joint sampling perform similarly with this model. The magnitude of the position error is similar to the baseline, which had access to the robot kinematics. For the phase of heavy occlusions in the period of 5-8 s, the error is lower than the baseline most of the time. The jitter is also less pronounced during this period than the baseline's. Moreover, using the complex model for both sampling strategies, the ESS is characterized by a lower noise level. Again, the ESS is higher for the block-wise than the joint sampling strategy.

7.2.6 Discussion

Before running the experiments, the expectation was that the baseline particle filter with access to forward kinematics would perform best, since this filter only has to track a static object. With a constant velocity near zero, the noise should stay low. Block-wise sampling was expected to target the course of dimensionality

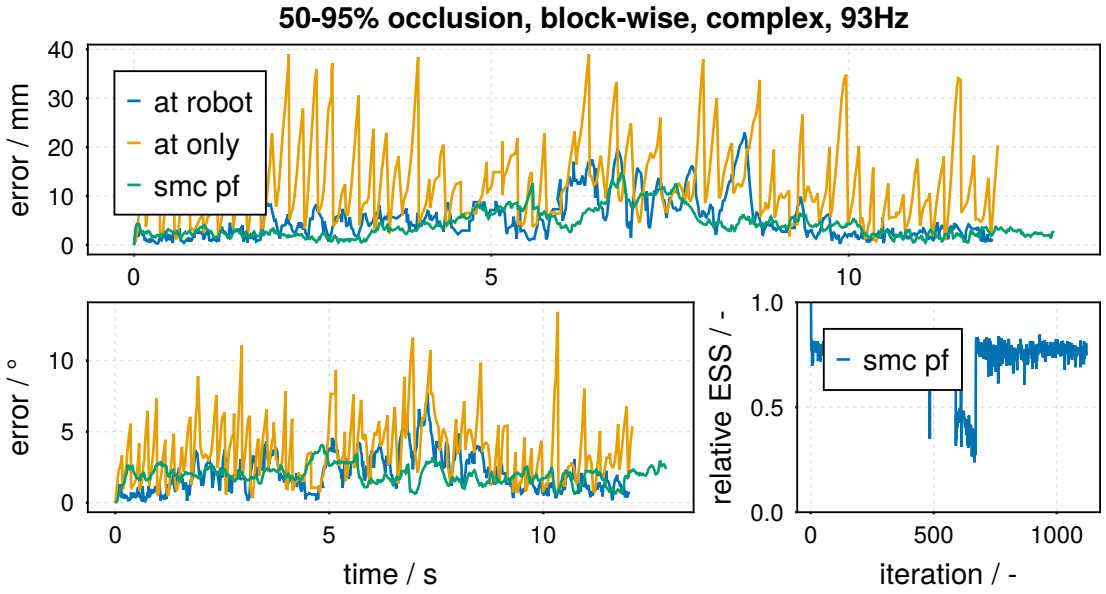


Figure 7.6: Particle filter results using block-wise sampling and a smooth truncated exponential with pixel classifications.

and reduce particle deprivation. Moreover, the expectation was that the smooth truncated exponential and the pixel classification model occlusions would be more accurate and improve the accuracy during heavy occlusions.

Remarkably, the *smc pf* using the complex model outperforms the baseline filter *at robot* with access to the robot kinematics. Since the *at robot* filter used significantly fewer particles, discretization errors most likely cause the jitter. Using a higher particle count due to the improved hardware increases the granularity of sampling the pose space and reduces the jitter. Another possible reason is that the mean of the particle cloud is used as the output instead of the maximum likelihood particle. However, using the mean only filters high-frequency noise and is most likely not the reason for the reduced jitter. Still, the baseline filter uses a model comparable to this work's simple model but does not suffer from high errors during heavy occlusions. Thus, having additional information for the motion model makes the filter more robust but does not improve the accuracy. The camera sensor, amount of occlusion, and number of particles are likely the main factors influencing the achievable accuracy.

As expected, sampling the position and orientation block-wise improves the stability of the filter. One indicator that supports this behavior is the increased ESS compared to jointly sampling the whole pose. A higher ESS indicates a more diverse particle cloud, which keeps track of more hypotheses. More hypotheses also improve the chance of tracking the object if the motion changes directions.

Moreover, a lower ESS results in more frequent resampling. Each resampling step introduces sampling errors, which can also degrade performance.

Higher errors occur for the position during heavy occlusions when using the simple model in Fig. A.7. However, the orientation errors are usually similar to the baseline. A similar dodging behavior could be observed during the practical evaluation of the baseline samplers during surgeries. If the model is not tuned well, the estimated position shifts away from the occluded object towards the free space. In contrast, the complex model with pixel classifications maintains low errors. Note that this contrasts the results on pose estimation from a single image in Section 6.5.2 where the results got worse when using the classifications. Therefore, modeling classifications might help under heavy occlusions but require a good initialization for the orientation. Using the pixel classifications might allow the algorithm to focus on the relevant visible parts and increase the robustness.

In conclusion, all sensors that can improve the proposals should be used to prevent the degeneration of the particle cloud and increase robustness. If the camera is not attached to a robot, many modern 3D cameras have an integrated inertial measurement unit, which can be used in the motion model. Block-wise sampling improves accuracy and increases the ESS, which leads to less resampling. Using a more complex and arguably more accurate model with pixel classifications and a smooth truncated exponential drastically improves the performance. However, the parameters of the motion and observation model require some tuning. Automatic tuning for specific applications to eliminate this *special care* is advisable, but only possible if sufficiently diverse annotated data is available [106].

7.3 Practical Use of Pose Estimation in Robotic Manipulation Tasks

This section aims to give a practical overview of using the pose estimates to control a robot in manipulation tasks. Controlling a robot based on probabilistic perception is often referred to as *decision-making*, and an outlook can be found in Section 8.1. This section focuses on the practical steps to use 6D pose estimates to control a robot in manipulation tasks, such as bin picking or bone cutting. Practical experiences were collected for bin-picking surgical instruments. The required steps include calibrating the robot camera setup, calculating a goal pose, planning and executing a collision-free trajectory, and compliant control of the robot when making contact with the object and environment. For practicality, this section introduces tools from the Robot Operating System (ROS) if available and focuses less on the theoretical foundations. An open and up-to-date resource

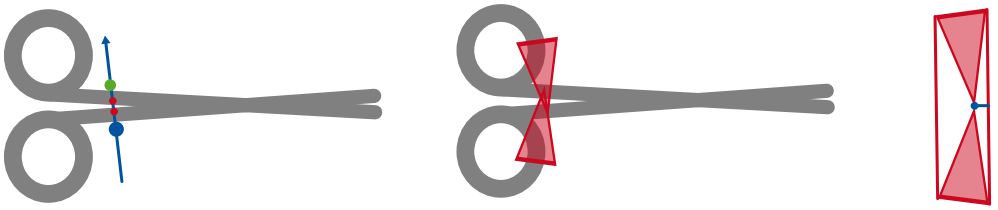


Figure 7.7: Grasp planning on a surgical clamp. Left: sampling and ray tracing candidate points, middle: friction cones, right: grasp quality via convex hull.

that provides theoretical foundations is the MIT course on robotic manipulation[1] [107].

7.3.1 Local Plan: Model-Based Grasp Candidates

Experts could annotate grasp poses similar to surgery planning, where surgeons manually specify the cutting edges. However, thousands of different objects make this approach impractical, so grasp poses should be planned automatically. While magnetic and suction grippers simplify grasp planning by requiring only a single contact point, the former might grab multiple instruments, and the latter struggles with thin objects. This section focuses on parallel grippers, which have been used to grasp surgical instruments successfully.

For a parallel gripper, a grasp candidate consists of two points on the object's surface, indicating where the gripper should make contact. The first step consists of randomly sampling a point on the object's surface, calculating the corresponding surface normal vector, and tracing a ray along this vector. If the ray crosses multiple surfaces, the point with the largest ray distance avoids grasping candidates between two surfaces. For example, Fig. 7.7 shows the sampled point in blue, two points on the ray between surfaces, and the selected second grasp point. The candidate can be discarded if the distance exceeds the gripper's width.

Classically, the quality of a grasp candidate is determined by calculating the friction cones, which are normal to the surface. The grasp quality is the distance of the closest facet of the convex hull of the wrenches, highlighted as the blue line in the right image of Fig. 7.7 [108]. If the origin is outside the convex hull, the force-closure condition is not met, and the grasp will fail.

[1] Robotic Manipulation; Perception, Planning, and Control: https://manipulation.csail.mit.edu/index.html

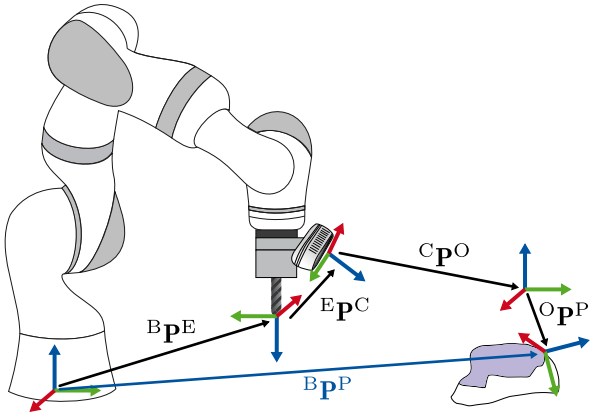


Figure 7.8: Transform the plan from the object frame to the robot base frame for the eye-in-hand scenario.

After determining a grasp candidate, the two contact points must be lifted to a 6D pose, allowing the end effector to be positioned. The grasp center point is located in the middle of the two points. Moreover, the connection line between the two points defines the axis of the end effector frame, which points from one finger to the other. However, the end effector can rotate around this axis, resulting in infinite solutions. In practice, this is solved by considering grasps only from the top and checking for reachability using the robot kinematics. If a model of the world exists, grasps that would cause collisions can be rejected using a simulation as described in Section 7.3.3. More recent approaches use deep learning to generate the grasp candidates by training in simulations [109]. These learning-based methods operate directly on the measured point cloud and can learn to avoid collisions in most cases.

7.3.2 Transforming Local Plans using Camera-Based Pose Estimates

Plans are typically defined in the local object coordinate system. However, the goal poses of the end effector and trajectories are typically defined in the robot base coordinate frame. Calculating the planned goal pose in the robot base frame $^{B}\mathbf{P}^{P}$ requires chaining transformations as depicted in Fig. 7.8:

$$^{B}\mathbf{P}^{P} = {}^{B}\mathbf{P}^{E}\,{}^{E}\mathbf{P}^{C}\,{}^{C}\mathbf{P}^{O}\,{}^{O}\mathbf{P}^{P} \tag{7.3}$$

where $^{O}\mathbf{P}^{P}$ is the **P**lanned end effector pose in the **O**bject frame, $^{C}\mathbf{P}^{O}$ the pose of the object in the **C**amera frame, $^{E}\mathbf{P}^{C}$ the pose of the camera in the **E**nd effector frame, and $^{B}\mathbf{P}^{E}$ the pose of the end effector in the robot **B**ase frame. By definition,

the planned pose ${}^{\mathrm{O}}\mathbf{P}^{\mathrm{P}}$ of the end effector is known in the object frame. The object's pose in the camera frame ${}^{\mathrm{C}}\mathbf{P}^{\mathrm{O}}$ results from the pose estimation and tracking algorithms presented in this thesis.

Assuming that the robot is calibrated and the tool can be attached precisely, only the camera mounting needs to be calibrated. Figure 7.8 displays the eye-in-hand scenario, where the camera is rigidly attached to the end effector. Because of the rigid attachment, the camera's pose in the end effector frame ${}^{\mathrm{E}}\mathbf{P}^{\mathrm{C}}$ is constant and needs to be calibrated only once. For the calibration procedure, a known object is placed statically in the camera's view, so the object's pose in the robot's base frame ${}^{\mathrm{B}}\mathbf{P}^{\mathrm{O}}$ is constant, too. Then, the robot moves the end effector to different poses ${}^{\mathrm{B}}\mathbf{P}_i^{\mathrm{E}}$ so the camera can capture corresponding object poses ${}^{\mathrm{C}}\mathbf{P}_i^{\mathrm{O}}$. These measurements result in the following equation system:

$$ {}^{\mathrm{B}}\mathbf{P}_i^{\mathrm{O}} = {}^{\mathrm{B}}\mathbf{P}^{\mathrm{E}}\,{}^{\mathrm{E}}\mathbf{P}^{\mathrm{C}}\,{}^{\mathrm{C}}\mathbf{P}_i^{\mathrm{O}} \tag{7.4} $$

This equation system can be solved using optimization algorithms, typically by rewriting it in the form $\mathbf{AX} = \mathbf{YB}$ [110]. Theoretically, this calibration could be executed online using the algorithms from this work to estimate the poses ${}^{\mathrm{C}}\mathbf{P}_i^{\mathrm{O}}$ of a known object. If a color or gray-scale camera is part of the 3D camera, calibration packages, such as MoveIt Calibration[2], use printable calibration patterns for which the pose can be estimated robustly and precisely. Alternatively, the camera can be mounted rigidly to the robot base as in Fig. 7.9, in which case the calibration pattern is attached rigidly to the end-effector instead of the camera. In this case, the robot hand moves a rigidly attached calibration target instead of the camera.

7.3.3 Collsion Free Motion and Task Planning

Motion planning calculates a trajectory from the current robot pose to the goal pose. It can be interpreted as an optimization problem to minimize the execution time of the trajectory to maximize the throughput. The problem's constraints are the robot's axis limits and (self) collisions.

MoveIt[3] is a tool that combines the required components for motion planning [111]: It keeps track of a planning scene, which includes a predefined workspace setup, an occupancy map, and custom objects as visualized in Fig. 7.9. The predefined workspace includes a model of the robot station and fixed objects such as tables or walls. In addition, MoveIt can use the 3D camera to map occupied

[2] https://github.com/ros-planning/moveit_calibration
[3] https://moveit.ros.org/

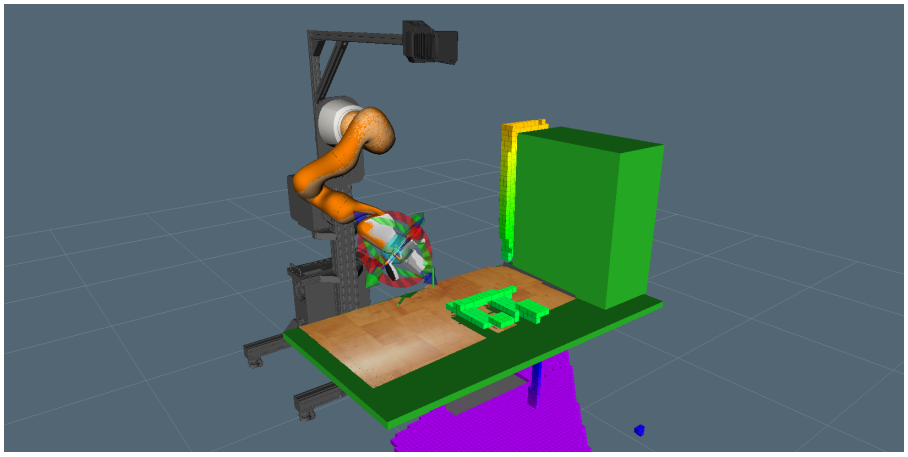


Figure 7.9: MoveIt digital twin with workspace model and occupancy mapping visualized in RViz.

spaces with cubes that have not been predefined, e.g., boxes or humans. Custom objects can be used with this thesis' pose estimation to enable higher fidelity planning. If, for example, the robot needs to grasp an object, the cube-based occupancy map would be too coarse and always trigger collisions. MoveIt includes several planning libraries and algorithms, for example, stochastic samplers or gradient-based approaches. Since motion planning occurs in joint coordinates, but manipulation goals are defined in Cartesian coordinates, inverse kinematics libraries are also included. The IKFast library generates source code with fast analytical solutions for inverse kinematics to speed up motion planning.

Task planning takes this approach one step further and allows the definition of subgoals and the joint optimization of multiple trajectories between them. During task planning, it is also possible to introduce varying degrees of freedom, e.g., by proposing several grasp poses. Jointly optimizing all subgoals avoids cases where an optimal trajectory for one goal results in much longer execution times for subsequent goals. In the worst case, grasping an object in the wrong position might lead to infeasible trajectories in one of the next steps. The MoveIt Task Constructor integrates with MoveIt, allowing it to access collision avoidance and motion planning capabilities [112]. Therefore, it can also handle grasp poses, which would lead to collisions and reject them early.

Historically, trajectory planning and execution were separated and stopped if the planning scene changed. With MoveIt2, hybrid motion planning allows the robot to react to a changing environment and avoid dynamic obstacles. This is especially useful in collaborative scenarios where humans might enter and exit the planning scene at any time. It is theoretically possible to model human limbs as cylinders and track their pose with this work's PF. However, human pose

estimation and tracking is an extensive research field with specific advanced methods. For example, OpenPose[4] and ROS4HRI[5] can easily be integrated into ROS and MoveIt [113].

7.3.4 Compliant Control for Contacts

Most surgical instruments are thin and tend to lay flat in the sieve or on top of other instruments. Therefore, the margin between reaching the instrument and colliding with an object below is tiny. If a position-controlled cobot collides with an object, it throws an error that an operator must acknowledge. If the robot does not reach far enough, it misses the instrument. Instead of avoiding collisions, embracing them with compliant behavior can help overcome the abovementioned issues.

The concept is as follows: First, the robot moves to a pre-grasp pose above the object with a position controller. Then, the controller switches to a cartesian impedance control mode and approaches the object. Instead of targeting the grasp point, the goal is set slightly below the object to ensure that the robot reaches far enough. With the impedance control still on, the gripper can be closed to avoid damaging it. After moving up again, the robot can continue the planned task trajectory using a position controller. This approach benefits from a proper impedance behavior, where the robot acts as a spring, which is possible with a KUKA LBR iiwa. If a force-torque sensor is attached to the robot flange, the robot is usually controlled via admittance control, which requires a compliant environment. In this case, the force suddenly increases when hitting a stiff surface, and the robot bounces.

7.3.5 Using the Pose Estimator in Practice

Pick and place tasks can be fully automated, and operators should only need to interact with the robot in case of exceptions. For example, if the robot repeatedly fails to pick an instrument, an operator could help to sort this instrument. The main effort for using the pose estimation algorithms from this work is in curating the CAD models. First, older parts might not even have CAD models available but only technical drawings, requiring modeling them from the ground up. Second, CAD models might contain annotation artifacts and must be exported to mesh

[4] https://github.com/ravijo/ros_openpose
[5] http://wiki.ros.org/hri

models for rendering. An automated pipeline using FreeCAD[6] and Open3D[7] has been developed for this thesis to process surgical instruments. However, there is not a single parametrization that fits all instruments, as the size of details varies. If the mesh model contains too many details, the runtime of the inference algorithms increases. A low degree of detail might lead to missing parts, e.g., the tip of a clamp. Therefore, every model must be checked for defects and corrected if required.

In contrast, one-off applications like surgeries already include processing CT scans to mesh models. Only downsampling might be required to reduce rendering times in the CAD-based workflow. Moreover, the surgeons could provide a point prior to the algorithms during surgeries via a pointing device. During previous cadaver studies, one person used a computer and provided the prior by aligning the mesh model to the point cloud in 3D.

7.4 Summary

In a sense, medical applications are similar to industrial applications: CAD models of instruments of pre-operative or CT scans enable the application of this thesis' pose estimation algorithms. Still, collaboratively working with surgeons and the large variety of geometries presents specific challenges.

The first section has shown that the CAD-based pose estimation of surgical instruments is possible via Bayesian inference. However, the results highlight the importance of evaluating a method in diverse applications. Compared to the BOP datasets, segmentation masks are necessary to estimate the pose of long and thin instruments. Otherwise, the pose diverges, so the rendered instruments align with the flat surface they lie on. This observation points out a significant limitation of the presented pose inference method: If the pixels of the rendered and measured object do not overlap, the likelihood function fails to discriminate good from bad estimates. Moreover, the inference time required to achieve reasonable recalls is six times higher than the time required for the BOP datasets. Semantic information could enable a likelihood function that overcomes this limitation. The outlook presents one possibility using object coordinates in Section 8.1.

Most sections of this thesis focused on static scenarios, but Section 7.2 shows that reformulating the SMC sampler into a bootstrap particle filter allows reusing

[6] www.freecad.org
[7] www.open3d.org

the framework for dynamic object tracking. The application consists of tracking the pose of an iliac crest bone in the context of facial reconstruction surgeries. This thesis' models and filter formulation were compared to a baseline particle filter tracker from a previous publication. A block-wise sampling strategy and new observation models from Section 4.3 have been evaluated. Block-wise sampling improves robustness and errors by increasing the effective sample size. Pixel-object classifications improve tracking performance during heavy occlusions, in contrast to the decreased performance during static pose inference. Using the more complex smooth truncation can improve the accuracy.

Finally, Section 7.3 introduced the practical steps required to use the pose estimators from this thesis in a robotic manipulation task. Usually, planning is executed in local frames, for example, by calculating grasp points in the object frame. The calibration steps required to transform local plans into the robot base coordinate system for control were explained. Moreover, the section introduces the MoveIt ecosystem to plan and execute collision-free trajectories. Compliant control has been introduced to overcome imprecise calibrations and successfully grasp even thin objects like surgical instruments. Finally, the steps for preparing and using CAD models by a practitioner for the pose estimation were described.

8 Conclusion

Bayesian inference is a relatively unexplored research direction in computer vision, with almost no prior published research in 6D pose estimation. Bayesian methods enable an intuitive way to formulate the pose estimation problem as a top-down generative process, in contrast to bottom-up feature engineering. However, the practical use of these methods has been limited because of long inference times. In particular, robotic manipulation tasks require both high accuracy and a computation time lower than their cycle time, typically <10 s. Similar to the tooling that kick-started the success of deep learning by efficiently utilizing the GPU, this thesis aims to present models and inference algorithms for Bayesian depth image-based pose inference on the GPU.

This thesis introduces probabilistic models for image-based pose estimation. The model consists of a prior, encoding an estimate of the object's position and an observation model describing the generative process of the camera measurement. For the considered cases, a rough prior position estimate is available based on additional sensors, machine learning models, or humans working collaboratively with the robot. Modeling this position estimate is straightforward by using a normal distribution. In contrast, no information on the object's orientation is available, and formulating probability distributions for rotations requires special care due to the nonlinearities. Quaternions allow a formal and efficient description of the probabilistic orientation model.

3D rendering is the generative process that transforms a scene description into depth images measured by the camera using a GPU. This work enables the use of the GPU for calculating the likelihood of the measured image given these renderings by formulating the measurement noise independently for each pixel. One critical insight is that the independence assumption for the pixels results in an overconfident likelihood, which can be countered using regularization strategies. Occlusions present a particular challenge, and this thesis addresses it using different probability distributions and formulating a per-pixel classification problem. However, these formulations introduce additional complexity.

Rendering is a non-differentiable operation that prohibits the use of advanced gradient-based inference algorithms. With this consideration, this work employs

chain-based and particle-based sampling algorithms which do not require gradients. Chain-based samplers sequentially evolve a single hypothesis, while the particle-based SMC sampler promises to explore the expected multimodal distributions more effectively. Several best-practice modifications improve the chances of a successful pose inference, e.g., likelihood tempering and logarithmic sampling in unconstrained domains.

The experiments permit the conclusion of the effectiveness of the samplers and model components. Chain-based samplers fail to explore the multimodal posterior distribution efficiently, even with a time budget of more than three seconds. However, the particle-based SMC sampler performs much better and achieves acceptable results with a time budget of 0.5 s. Runtime is the only parameter that effectively influences the performance of the samplers. The number of particles or the image size does not significantly influence the performance. Moreover, the model performance increases if more and higher quality priors are available for the position and pixel-object classification. However, using more complex distributions to model occlusions makes no difference. Inferring the pixel-object classifications as part of the model decreases average recalls for inference on static images but increases the performance of a particle filter-based pose tracker under heavy occlusions. Automatic parameter tuning did not improve the results significantly because the algorithms can be considered insensitive to the parameters. In conclusion, more prior information should always be used, but the probabilistic model can be as simple as possible. Simpler model formulations might even allow for more inference steps given a fixed compute budget, which could improve the pose estimates.

With an average recall of 0.475 on the BOP core datasets, the SMC sampling-based approach offers room for improvements compared to the state-of-the-art methods. The average recall of the best-performing method in the same category of the BOP challenge is 0.674. However, most methods in the same category require much longer inference times, exceeding 30 s for the same task. Most notably, the scores of other methods are significantly higher on datasets with texture-rich objects. One possible implication is that using more sensor modalities can improve performance. Including color images in the observation model could be a promising future research direction for Bayesian pose estimation. Another observation from the experiments is that the evaluation of the synthetic images and the real BOP validation images yields similar results, indicating that the synth-to-real gap is small for depth images. Therefore, synthetic images are a cost-effective alternative to evaluate depth image-based pose estimators.

Moreover, applying the Bayesian pose estimation to different applications in the context of medical robotic manipulation demonstrates the adaptability of the approach. With only minor modifications, the SMC sampler could be reformulated

as a particle filter to enable tracking a bone pose at the camera frame rate of 90 Hz. Nevertheless, applying the pose estimator to surgical instruments also revealed a limitation of the approach: Renderings of thin and long objects barely overlap, so the pixel-wise likelihood cannot differentiate between poses. Thus, it takes much longer to sample a pose close enough to the true pose for the algorithm to converge. Moreover, the algorithm fails to estimate instrument poses without good segmentation masks. This failure indicates that the likelihood landscape contains global optima, which do not correspond to the true pose. All these observations lead to the conclusion that semantic information is required in addition to the purely geometric information from the depth images to avoid the current failure cases.

The Bayesian method's main advantage lies in its intuitiveness and flexibility in applying it to new applications with different information sources and sensors. With the performance considerations from this work, the runtime is another advantage over competing methods. Moreover, only a CAD model, a depth image, and a coarse position estimate are required, which also aids broad applicability. While this work is not a leader on the BOP benchmark, it shows that alternative research directions exist besides classical feature engineering and neural network-based computer vision. This work is the first to successfully use Bayesian inference for the BOP challenge. Future work can build on this foundation, improve it, and adapt it to the specific needs.

8.1 Outlook

This outlook focuses on possibilities for improving this work's methods and combining Bayesian inference with control for active perception. The most potential for improving the pose estimates lies in employing learning-based methods to include more semantic information in the observation model. In addition, active perception could jointly improve the outcomes of the subordinate manipulation task and the pose estimation.

8.1.1 Future Directions for Bayesian Pose Estimation

One prominent advantage of the method from this thesis is that no annotation of application-specific training data is required. Nevertheless, the semantic capabilities of neural networks could significantly improve the algorithms, as shown by the results of using segmentation masks as prior information.

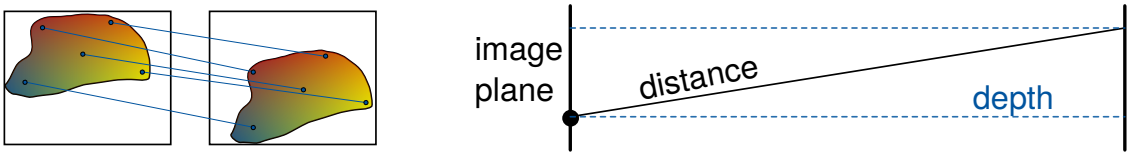


Figure 8.1: Dense correspondences and difference between depth and distance measurements.

A weakness of the current approach is that the pixel-wise likelihood requires that the rendered and measured object pixels overlap to discriminate between poses. One possibility to overcome this limitation could be to match the rendered object surface pixels to the corresponding object surface pixels in the measured images. For example, one could train a model to predict the object coordinates proposed by Brachmann et al. in the rendered images and the measured image; see Fig. 8.1 [37]. Using these correspondences to calculate matrices of matching indices would allow using the pixel-wise likelihood from this work on the GPU. However, the pixels of depth images measure a value normal to the sensor plane, which would result in a likelihood invariant to translations and rotations in the image plane; see Fig. 8.1. One possible solution could be measuring and rendering distance images instead of depth images, reducing the ambiguities to rotations around the central image axis; compare for Fig. 8.1. Another solution could be an additional term that includes the distances images between the correspondences in the image coordinates.

Most methods from the literature that use algorithms similar to Bayesian inference algorithms employ machine learning to generate better proposals [60]. This approach is straightforward, as it does not require modifying the likelihood function or the GPU code. Any probabilistic generator could be used as an independent proposal distribution in the MH or SMC algorithms. For example, the dense correspondences from Fig. 8.1 could also be used to formulate a PnP problem, allowing for the direct regression of a transformation between the rendered and measured images. Another approach to guide the algorithms to better poses could be differentiable rendering to enable advanced gradient-based inference algorithms such as HMC or variational approximations. Differentiable rendering has been tried in the early phase of this research but dismissed because of slow rendering times of multiple seconds; see Appendix A.2.3. If the performance of differentiable renderers improves, they might be worth revisiting.

Besides algorithmic improvements, more insights can be gained by evaluating the Bayesian pose estimation on new applications. For example, by trying to estimate the pose of surgical instruments, the limitation of the pixel-wise likelihood became apparent. An application not considered in this thesis is the pose estimation and tracking of articulated objects. Simple objects like clamps only introduce one

additional degree of freedom to the state of the inference problem. However, the curse of dimensionality causes the required inference time to grow exponentially with each additional degree of freedom. Future work could evaluate whether intelligent formulations of the constraints can be used to propose more likely samples and avoid this increase in required computations. Soft materials that cover objects introduce an infinite number of additional degrees of freedom. For example, a bone transplant is typically covered in soft tissue, and most of the bone is not visible. Modeling the deformable tissue might be a way to enable the pose estimation even in this challenging scenario.

8.1.2 Active Perception: Combining Vision and Control

Benchmarks only evaluate a single pose, meaning that most hypotheses must be discarded in the context of Bayesian inference. However, planning and control of a robot could benefit from a probability distribution instead of a point estimate. Robotic decision-making under uncertainty aims to balance exploration to reduce this uncertainty and exploitation to fulfill the task at hand [114].

For example, tracking the uncertainty could be used to supervise a manipulation task in critical environments. A robot must be very sure about the pose if it cuts a bone. Otherwise, it might cause serious injuries. The robot must continuously decide whether it is safe to continue. In practice, a particle filter could track the pose of the bone, and a change of the particle cloud can indicate tracking failures [115]. In this case, the robot would stop operating until the particle filter is reinitialized.

Conversely, a robot could help to improve the pose estimation results by moving the camera to different viewpoints. Depending on the scene, multiple viewpoints could resolve ambiguities caused by (self) occlusions of the object. Inspired by the work of Patten et al., this viewpoint selection can be automated using a Bayesian approach [116]. The goal is to visit as few viewpoints as possible until the robot is certain enough to start a manipulation task. Patten et al.'s method consists of two key elements: an offline computation to determine the expected quality of different viewpoints and an online mapping of occlusions. They quantify the expected quality of a viewpoint by calculating the information entropy of a probability distribution of object classes. The object must be rendered from different viewpoints to adapt the entropy to pose estimation. A Bayesian pose inference algorithm from this thesis can approximate the posterior pose distribution for each viewpoint, and the entropy can be calculated using these distributions. Moreover, the pixel-object classification model from this thesis can be used to calculate an occlusion probability during the online step. A utility

function can be formulated and then evaluated for each viewpoint candidate using the offline entropy and the online occlusion map. The viewpoint with the highest utility is the next set point of the controller.

A Appendix

A.1 Source Code

My source code is publicly available:

- Main repository, with *scripts* to reproduce results:
  https://github.com/rwth-irt/BayesianPoseEstimation.jl
- OpenGL rendering and CUDA interop:
  https://github.com/rwth-irt/SciGL.jl
- Bayesian network and graph compilation:
  https://github.com/rwth-irt/BayesNet.jl
- CUDA compatible probability distributions:
  https://github.com/rwth-irt/KernelDistributions.jl
- Pose error metrics and scores:
  https://github.com/rwth-irt/PoseErrors.jl
- Generating synthetic datasets:
  https://github.com/rwth-irt/BlenderProc.DissTimRedick

Datasets, raw results, and processed results are stored on the Coscine research data management platform.

http://hdl.handle.net/21.11102/bc759cab-cade-4ab3-8f70-86dbee80b330

A.2 Failed Approaches

I believe that research should report more on the failures and not only the successes. These failure cases can provide valuable lessons or, at the very least, save someone's time.

Nothing really matters . . .

A.2.1 Image Regularization

Using the number of **non-zero rendered pixels** in μ_{img} for Eq. (4.37).

$$N_L = \sum_{\mathcal{S}} 1, \ \mathcal{S} = \{i : \mu_i > 0\} \tag{A.1}$$

The idea is that only rendered pixels make a difference in the likelihood function. Dividing by the number of pixels should make the likelihood less view-dependent if the views have varying silhouettes. However, it encourages poses with fewer visible pixels, especially if no distinct features are available. For example, when estimating the pose of a box-shaped object, the side with the smallest area is usually in front. Moreover, the algorithm tends to diverge to regions where only a handful of pixels are rendered, e.g., the edges of the image. These have a high likelihood by fitting the surface precisely and dividing by a small number of valid pixels.

If masks are available, taking the **union over the mask prior and the non-zero pixels** was considered a possible solution to the abovementioned phenomenon.

$$N_L = \sum_{\mathcal{S}} 1, \ \mathcal{S} = \{i : p(c_{o,i}) = 1 \vee \mu_i > 0\} \tag{A.2}$$

The union enforces the regularization to consider a certain number of pixels, which should avoid the edges of the image. An unintended side effect is that the inference algorithm now tries to fit most of the object surface into the mask's silhouette. This side effect is especially troublesome under heavy occlusions, as the mask's silhouette does not match the object for any view.

A.2.2 Handling Invalid Pixels

Note that the development of the regularization started when trying to exclude non-zero rendered pixels from the likelihood by setting it to one. A value of one does not change the factorized joint probability distribution of the image. However, as it turns out, it should change the value of the joint probability. The regularization is still used, as including the classification probability has a similar effect.

A.2.3 Gradient Based Samplers with Differentiable Renderers

It might be worth trying again, as we experimented with the differentiable renderer from *Pytorch3D* back in 2020. Research has made progress on the performance

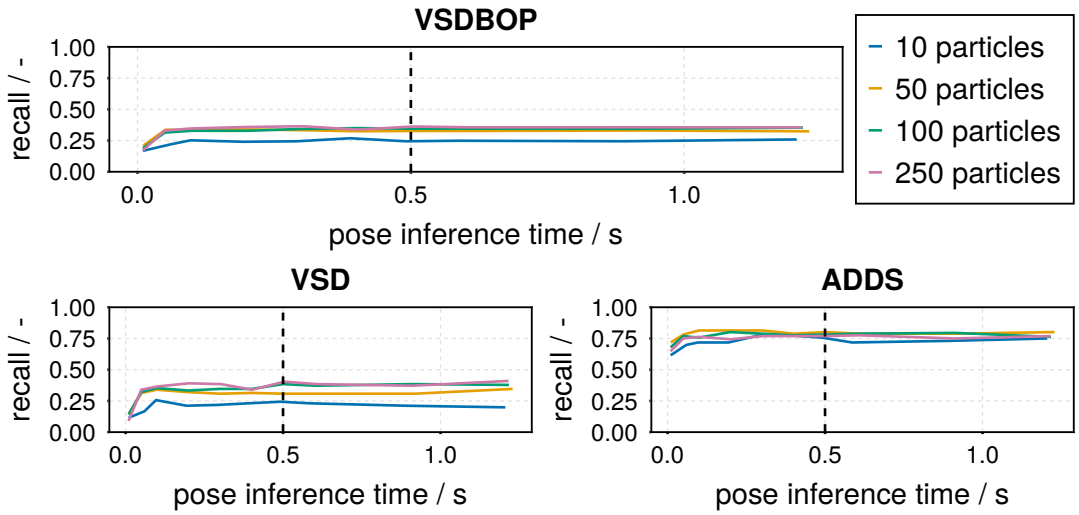


Figure A.1: Recalls of a SMC bootstrap sampler on synthetic data for various inference steps and particles.

and the quality of the estimated gradients in differentiable rendering, and using the latest advancements might yield significantly better results [117].

The discontinuity at the expected depth for the truncated distribution in Fig. 4.4 μ prevents the application of gradient-based methods. Another hurdle for using gradient-based methods for inference on images is the rasterization process of rendering, which is non-differentiable. Differentiable renderers such as *Pytorch3D* are much slower than rasterization-based renderers. Preliminary experiments have indicated that the gradients are not good enough, and the inference time using a Hamiltonian Monte Carlo sampler is prohibitively slow.

A.2.4 Bootstrap SMC Kernel for Static problems

While the bootstrap kernel is the de facto standard for particle filtering, it is typically not used for inference on static data. Chopin et al. published an early work that modified the particle filter for static problems, which did not use the bootstrap kernel from Section 5.1.4. This kernel is not used for a good reason: it performs poorly, as shown in Fig. A.1. The recalls are saturated after a short runtime, indicating that the particle cloud collapses early. Even worse, the bootstrap kernel only allows local moves and is stuck in the local optimum. Therefore, the chance of succeeding depends on an initial particle close to the true pose. This hypothesis

is supported by the fact that the recall in Fig. A.1 is almost only influenced by the number of particles.

A.2.5 Forward Proposal Kernel for SMC

An alternative to using an MCMC kernel is to use the forward kernel as the backward kernel, which is coined the *forward proposal kernel* [118]:

$$L_t(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = K_{t+1}(\mathbf{x}_t \mid \mathbf{x}_{t+1}) \tag{A.3}$$

The resulting weight increment is similar to the MH acceptance ratio in Eq. (5.4) [118]:

$$\tilde{w}_{t+1}^{(i)} = \frac{p_{t+1}(\mathbf{x}_{t+1}, \mathbf{z}) K_{t+1}(\mathbf{x}_t \mid \mathbf{x}_{t+1})}{p_t(\mathbf{x}_t, \mathbf{z}) K_{t+1}(\mathbf{x}_{t+1} \mid \mathbf{x}_t)} \tag{A.4}$$

This kernel has shown similar results to the bootstrap particle filter kernel in Appendix A.2.4. Most likely, the limiting factor is that only local proposals can be used, so the algorithm is more likely to get stuck in a local optimum.

A.2.6 Cropping in Particle Filters

The motivation behind cropping in Monte Carlo sampling is that less information is lost due to the down scaling of the images. However, cropping also introduces an additional sampling bias for each iteration of the particle filter. Thus, cropping leads to very jittery tracking and is therefore not recommended. This behavior might be improved if a different interpolation method is used for resizing depth images. Currently, a nearest neighbor interpolation method is used as it does not result in wrong values at the edges of objects. A linear interpolation could produce smoother results if the views offer large visible surfaces.

A.3 Gaussian Modified Truncated Exponential Distribution (Smooth Truncated Exponential)

We seek a solution for the following convolution from Eq. (4.31):

$$p_s(z \mid \mu) = \int p_{\mathcal{N}}(z \mid a, \mu) p_{\mathrm{Exp}}(a \mid \beta, a \leq \mu)) \, da \tag{A.5}$$

This equation uses the probability densities from Eq. (4.30); a normal distribution and a truncated exponential distribution:

$$p_{\mathcal{N}}(z \mid a, \mu) = \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{-\frac{(z-a)^2}{2\sigma_z^2}} \tag{A.6a}$$

$$p_{\text{Exp}}(a \mid \beta, a \leq \mu) = \mathbb{1}_{[l_b,u_b]}(a) \frac{\frac{1}{\beta} e^{-\frac{a}{\beta}}}{(1 - e^{-\frac{u_b}{\beta}}) - (1 - e^{-\frac{l_b}{\beta}})} \tag{A.6b}$$

$$= \mathbb{1}_{[l_b,u_b]}(a) \frac{\frac{1}{\beta} e^{-\frac{a}{\beta}}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} \tag{A.6c}$$

where $\mathbb{1}_{[l_b,u_b]}$ denotes the indicator function which represents the truncation and is 1 in the interval $[l_b, u_b]$ and 0 otherwise. Moreover, the exponential distribution has to be normalized using the cumulative density function to account for the truncation. Using a generic lower l_b and upper bound up allows us to derive a general formulation instead of only considering the upper bound used in Eq. (4.32). Inserting the respective distributions into the convolution yields:

$$p_s(z \mid \mu) = \int \Bigg(\underbrace{\frac{w_u}{z_{max}}}_{\mathcal{U}} + \underbrace{w_n \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{-\frac{(z-a)^2}{2\sigma_z^2}}}_{\mathcal{N}} \Bigg) \underbrace{\mathbb{1}_{[l_b,u_b]}(a) \frac{\frac{1}{\beta} e^{-\frac{a}{\beta}}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}}}_{\text{truncated Exp}} da \tag{A.7}$$

Splitting the sum of the $\mathcal{U}$ and $\mathcal{N}$ terms allows us to solve two separate integrals:

$$\int \frac{w_u}{z_{max}} \mathbb{1}_{[l_b,u_b]}(a) \frac{\frac{1}{\beta} e^{-\frac{a}{\beta}}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} da + \tag{A.8a}$$

$$\int w_n \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{-\frac{(z-a)^2}{2\sigma_z^2}} \mathbb{1}_{[l_b,u_b]}(a) \frac{\frac{1}{\beta} e^{-\frac{a}{\beta}}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} da \tag{A.8b}$$

Solving the first integral from Eq. (A.8a) is straightforward as the indicator function determines the boundaries of the integral:

$$\frac{w_u}{z_{max}} \frac{\frac{1}{\beta}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} \int_{l_b}^{u_b} e^{-\frac{a}{\beta}} da = \frac{w_u}{z_{max}} \frac{\frac{1}{\beta}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} \left[\beta e^{-\frac{l_b}{\beta}} - \beta e^{-\frac{u_b}{\beta}} \right] = \frac{w_u}{z_{max}} \tag{A.9}$$

Which is again the uniform distribution $\mathcal{U}$. The second integral Eq. (A.8b) is more involved, as it contains the product of a normal and an exponential distribution.

Pulling out the constants and adjusting the boundaries of the integral according to the indicator function yields:

$$w_n \frac{\frac{1}{\beta}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} \frac{1}{\sqrt{2\pi\sigma_z^2}} \int_{l_b}^{u_b} e^{-\left(\frac{(z-a)^2}{2\sigma_z^2} + \frac{a}{\beta}\right)} da \tag{A.10}$$

This integral is similar to the definition of the exponentially modified Gaussian distribution, which is the convolution of a normal and an exponential distribution. Therefore, a similar approach to solving the integral can be used by completing the square [119].

$$w_n \frac{\frac{1}{\beta}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} e^{-\frac{z}{\beta} + \frac{1}{2}\left(\frac{\sigma_z}{\beta}\right)^2} \frac{1}{\sqrt{2\pi\sigma_z^2}} \int_{l_b}^{u_b} e^{-\frac{\left(a + \frac{\sigma_z^2}{\beta} - z\right)^2}{2\sigma_z^2}} da \tag{A.11}$$

However, in [119], a term cancels out because the upper integration bound is chosen as z while it is μ in Eq. (A.11). Thus, a change of variables with $b = (a + \frac{\sigma_z^2}{\beta} - z)^2$ yields an expression that resembles the definition of the error function erf for the integral in Eq. (A.11):

$$\frac{1}{\sqrt{2\sigma_z^2\pi}} \int_{l_b + \frac{\sigma_z^2}{\beta} - z}^{u_b + \frac{\sigma_z^2}{\beta} - z} e^{-\frac{b^2}{2\sigma_z^2}} dt = \frac{1}{2}\left[\mathrm{erf}\left(\frac{u_b + \frac{\sigma_z^2}{\beta} - z}{\sqrt{2\sigma_z^2}}\right) - \mathrm{erf}\left(\frac{l_b + \frac{\sigma_z^2}{\beta} - z}{\sqrt{2\sigma_z^2}}\right)\right] \tag{A.12}$$

The two terms in equation Eq. (A.12) resemble the complementary cumulative density of normal distributions parametrized by $\mathcal{N}(z \mid u_b + \frac{\sigma_z^2}{\beta}, \sigma_z)$ and $\mathcal{N}(z \mid l_b + \frac{\sigma_z^2}{\beta}, \sigma_z)$ respectively. Substituting the results of Eq. (A.12) two integrals back into Eq. (A.7) results in the final formulation of a smooth density function for the occlusion:

$$\begin{aligned} p_s(z \mid \mu) = {} & w_u \frac{1}{z_{max}} + \\ & w_n \frac{\frac{1}{\beta}}{e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}} e^{-\frac{z}{\beta} + \frac{1}{2}\left(\frac{\sigma_z}{\beta}\right)^2} \frac{1}{2}\left[\mathrm{erf}\left(\frac{u_b + \frac{\sigma_z^2}{\beta} - z}{\sqrt{2\sigma_z^2}}\right) - \mathrm{erf}\left(\frac{l_b + \frac{\sigma_z^2}{\beta} - z}{\sqrt{2\sigma_z^2}}\right)\right] \end{aligned} \tag{A.13}$$

If we only consider the Gaussian noise and omit the uniform term, we can define a Gaussian-modified truncated exponential distribution similar to the exponentially modified Gaussian:

$$\begin{aligned} & p_{\mathrm{Exp}^*}(x \mid \beta, \sigma, l_b, u_b) = \\ & \frac{e^{-\frac{x}{\beta} + \frac{1}{2}\left(\frac{\sigma}{\beta}\right)^2}}{2\beta\left(e^{-\frac{l_b}{\beta}} - e^{-\frac{u_b}{\beta}}\right)} \left[\mathrm{erf}\left(\frac{u_b + \frac{\sigma^2}{\beta} - x}{\sqrt{2\sigma^2}}\right) - \mathrm{erf}\left(\frac{l_b + \frac{\sigma^2}{\beta} - x}{\sqrt{2\sigma^2}}\right)\right] \end{aligned} \tag{A.14}$$

A.4 Pose Estimation of Surgical Instruments

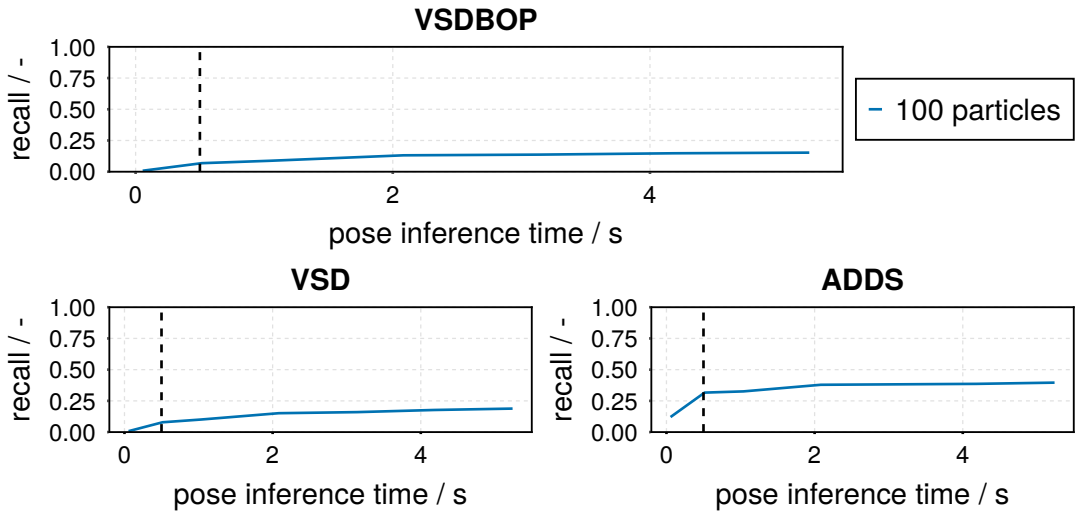


Figure A.2: Recall over inference time for the STERI dataset without using masks.

A.5 Incorporating Masks: Pixel-Object Classification

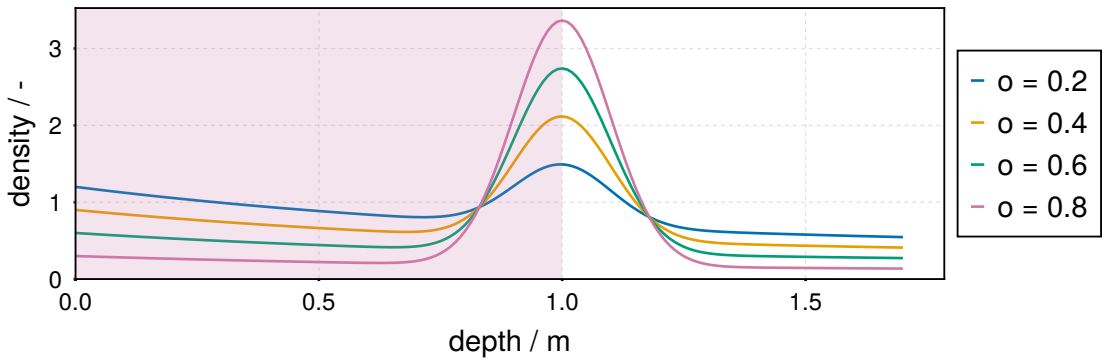


Figure A.3: Probability density of the likelihood model in Eq. (4.34) for an unmodified exponential distribution and varying object classification probabilities $\mathbf{o}$. Parameters: $\mu = 1\,\text{m}$, $\sigma_z = 0.1\,\text{m}$, $\beta = 1$, $z_{min} = 0.5\,\text{m}$, $z_{max} = 1.5\,\text{m}$.

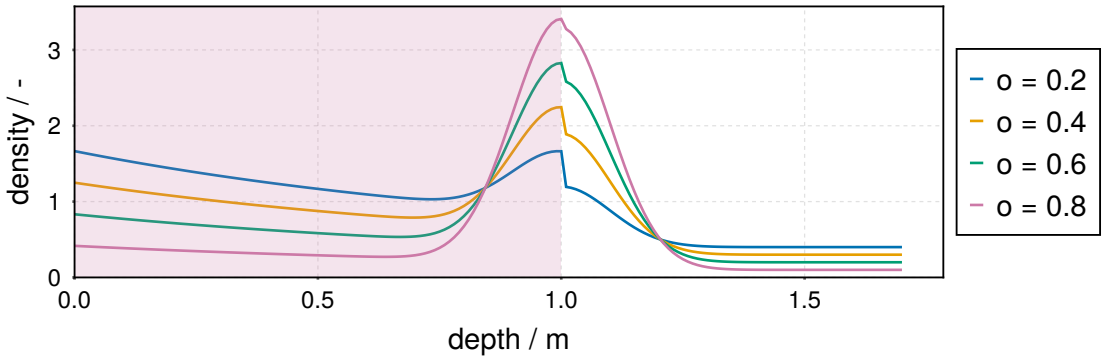


Figure A.4: Probability density of the likelihood model in Eq. (4.34) for a truncated exponential distribution and varying object classification probabilities $\mathbf{o}$. Parameters: $\mu = 1\,\text{m}$, $\sigma_z = 0.1\,\text{m}$, $\beta = 1$, $z_{min} = 0.5\,\text{m}$, $z_{max} = 1.5\,\text{m}$.

A.6 Runtime: Number of Particles and Inference Steps

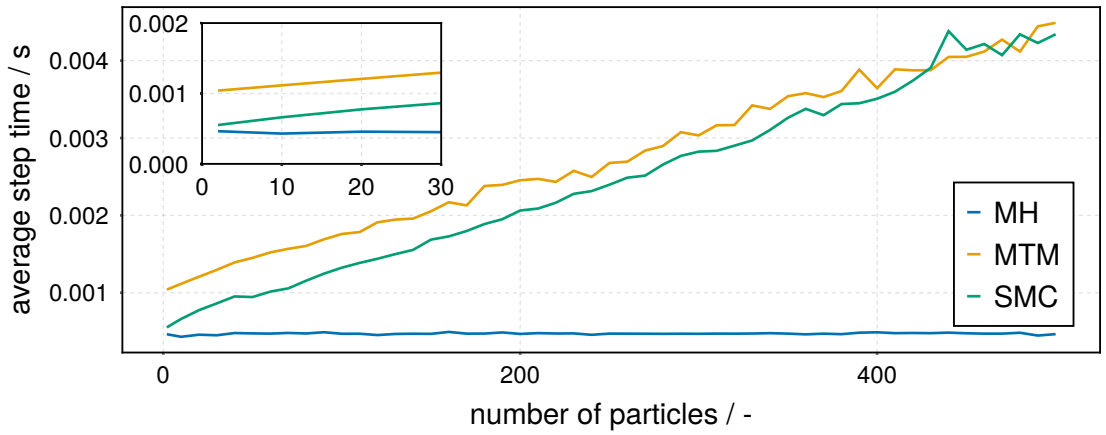


Figure A.5: Benchmarking the time per inference step for a different number of particles and samplers. MCMC-MH uses only one particle resulting in the constant time.

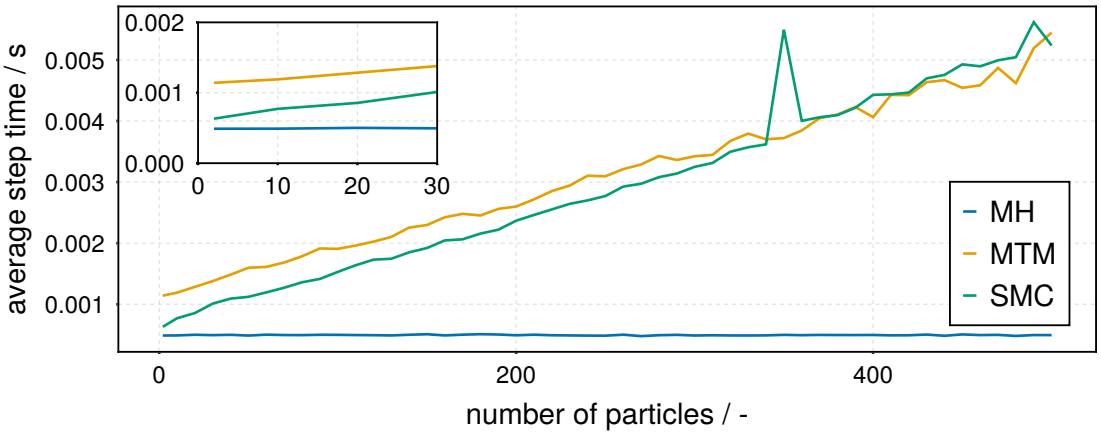


Figure A.6: Benchmarking the time per inference step for a different number of particles and samplers. The size of the image crops is 100 px × 100 px. MCMC-MH uses only one particle resulting in the constant time.

A.7 Particle Filtering for 6D Pose Tracking of Iliac Crest Bones

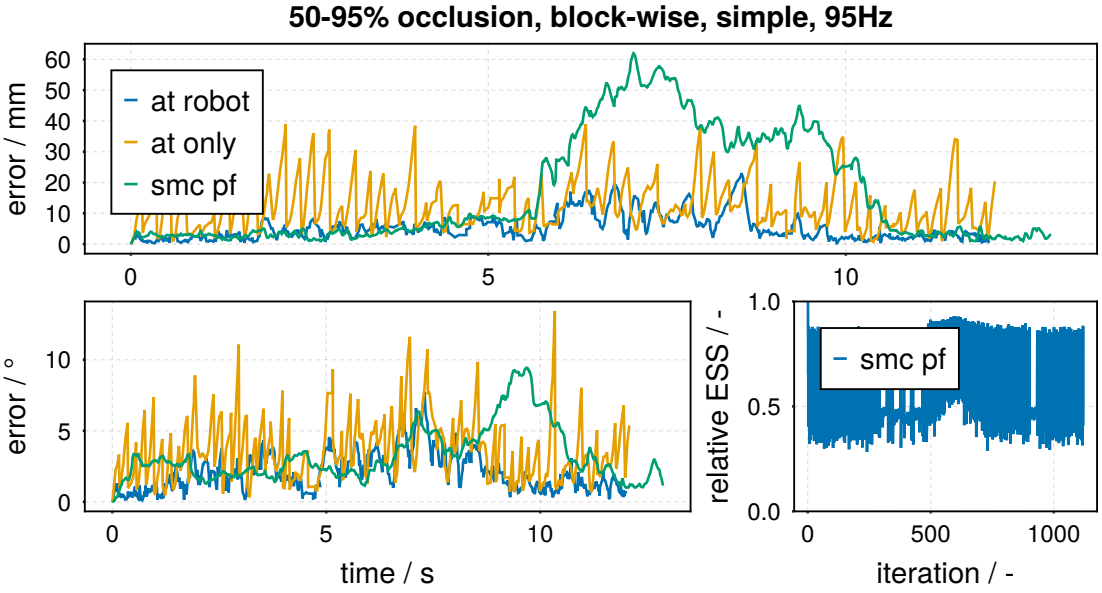


Figure A.7: Particle filter results using block-wise sampling and an unmodified truncated exponential without pixel classifications.

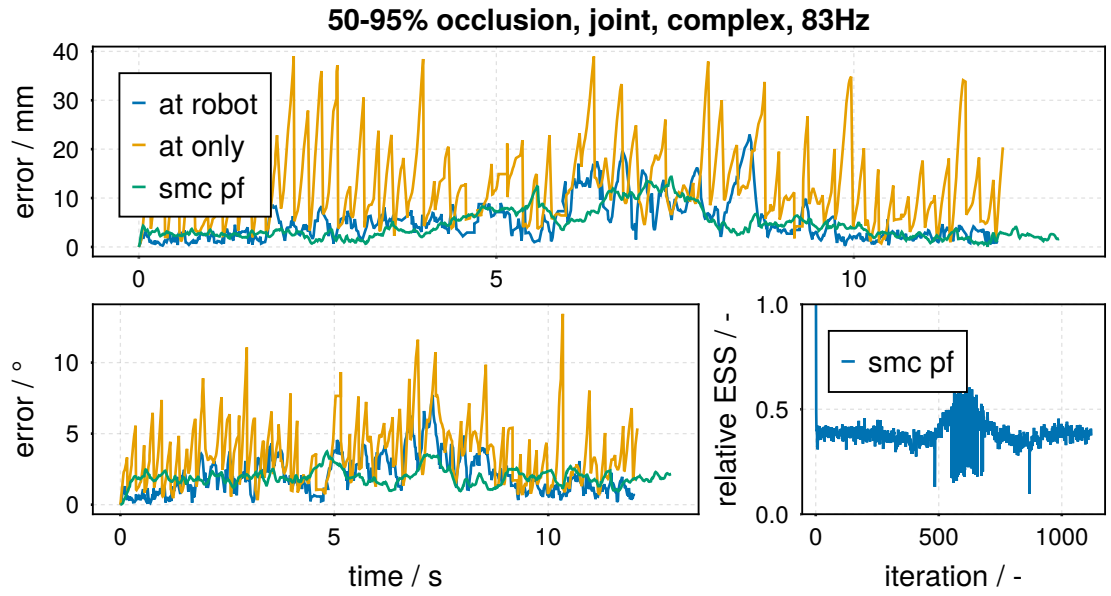


Figure A.8: Particle filter results using joint sampling and a smooth truncated exponential with pixel classifications.

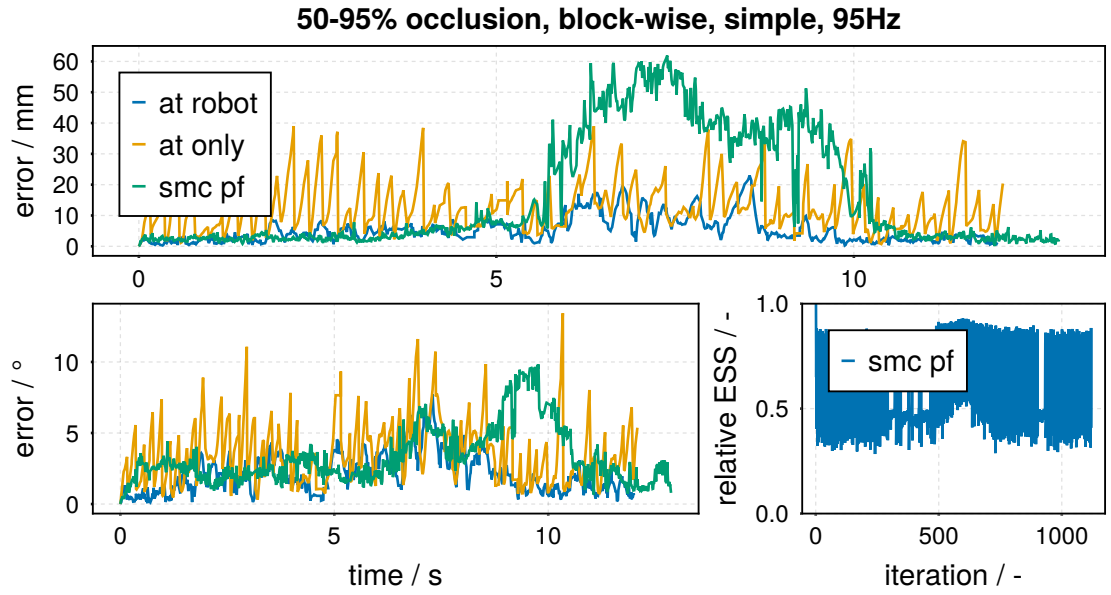


Figure A.9: Maximum likelihood particle filter results using block-wise sampling and an unmodified truncated exponential without pixel classifications.

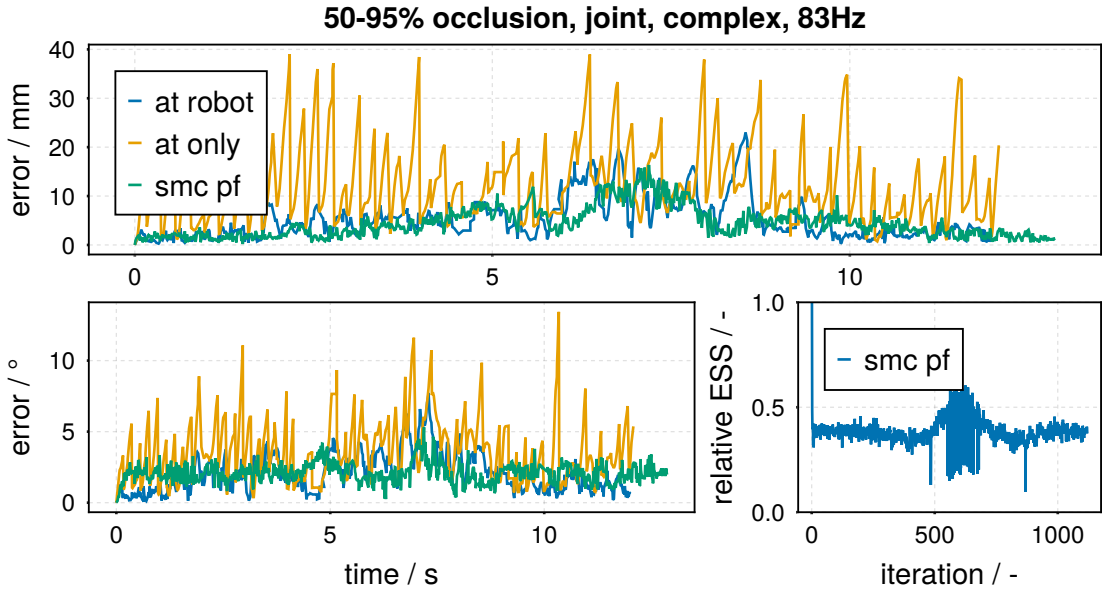


Figure A.10: Maximum likelihood particle filter results using joint sampling and a smooth truncated exponential with pixel classifications.

Bibliography

[1] Felix Gorschlüter, Pavel Rojtberg, and Thomas Pöllabauer. A Survey of 6D Object Detection Based on 3D Models for Industrial Applications. *Journal of Imaging*, 8(3):53, March 2022. ISSN 2313-433X. doi: 10.3390/jimaging8030053.

[2] Martin Sundermeyer, Tomas Hodan, Yann Labbe, Gu Wang, Eric Brachmann, Bertram Drost, Carsten Rother, and Jiri Matas. BOP Challenge 2022 on Detection, Segmentation and Pose Estimation of Specific Rigid Objects, 2023.

[3] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. On Evaluation of 6D Object Pose Estimation. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, volume 9915, pages 606–619. Springer International Publishing, Cham, 2016. ISBN 978-3-319-49408-1 978-3-319-49409-8. doi: 10.1007/978-3-319-49409-8_52.

[4] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. BlenderProc2: A Procedural Pipeline for PhotorealisticRendering. *Journal of Open Source Software*, 8(82):4901, February 2023. ISSN 2475-9066. doi: 10.21105/joss.04901.

[5] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting Twenty-Thousand Classes Using Image-Level Supervision. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, volume 13669, pages 350–368. Springer Nature Switzerland, Cham, 2022. ISBN 978-3-031-20076-2 978-3-031-20077-9. doi: 10.1007/978-3-031-20077-9_21.

[6] Vikash K Mansinghka, Tejas D Kulkarni, Yura N Perov, and Josh Tenenbaum. Approximate Bayesian image interpretation using generative probabilistic graphics programs. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[7] Victor A. Prisacariu and Ian D. Reid. PWP3D: Real-Time Segmentation and Tracking of 3D Objects. *International Journal of Computer Vision*, 98(3):335–354, July 2012. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-011-0514-3.

[8] Marco F. Cusumano-Towner, Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. Gen: A General-purpose Probabilistic Programming System with Programmable Inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pages 221–236, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6712-7. doi: 10.1145/3314221.3314642.

[9] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A language for flexible probabilistic inference. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 1682–1690, 2018.

[10] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4390–4399, Boston, MA, USA, June 2015. IEEE. ISBN 978-1-4673-6964-0. doi: 10.1109/CVPR.2015.7299068.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.

[12] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge, Mass, 2005. ISBN 978-0-262-20162-9.

[13] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An Introduction to Probabilistic Programming, October 2021.

[14] Arunkumar Byravan and Dieter Fox. SE3-Nets: Learning Rigid Body Motion using Deep Neural Networks. *arXiv:1606.02378 [cs]*, June 2016.

[15] Joan Sola. Quaternion kinematics for the error-state KF. *Laboratoire dAnalyse et dArchitecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep*, 2012.

[16] Gary Bradski. The OpenCV library. *Dr Dobb's J. Software Tools*, 25: 120–125, 2000.

[17] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000. ISSN 1939-3539. doi: 10.1109/34.888718.

[18] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob C. Wheeler, and Andrew Y. Ng. ROS: An open-source robot operating system. In *ICRA 2009*, number 3.2. Kobe, Japan, 2009.

[19] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. BOP: Benchmark for 6D Object Pose Estimation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11214, pages 19–35. Springer International Publishing, Cham, 2018. ISBN 978-3-030-01248-9 978-3-030-01249-6. doi: 10.1007/978-3-030-01249-6_2.

[20] James J Gibson. The perception of the visual world. *Houghton Mifflin*, 1950.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[23] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1): 98–136, January 2015. ISSN 1573-1405. doi: 10.1007/s11263-014-0733-5.

[24] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[25] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT++ Better Real-Time Instance Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):1108–1121, February 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.3014297.

[26] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, and Yu Qiao. InternImage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14408–14419, June 2023.

[27] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.

[28] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, *Computer Vision – ACCV 2012*, Lecture Notes in Computer Science, pages 548–562. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-37331-2.

[29] Tomas Hodan, Xenophon Zabulis, Manolis Lourakis, Stepan Obdrzalek, and Jiri Matas. Detection and fine 3D pose estimation of texture-less objects in RGB-D images. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4421–4428, Hamburg, Germany, September 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7354005.

[30] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 998–1005, San Francisco, CA, USA, June 2010. IEEE. ISBN 978-1-4244-6984-0. doi: 10.1109/CVPR.2010.5540108.

[31] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going Further with Point Pair Features. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, volume 9907, pages 834–848. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46486-2 978-3-319-46487-9. doi: 10.1007/978-3-319-46487-9_51.

[32] Tomas Hodan, Martin Sundermeyer, Bertram Drost, Yann Labbe, Eric Brachmann, Frank Michel, Carsten Rother, and Jiri Matas. BOP Challenge 2020 on 6D Object Localization, October 2020.

[33] Tomas Hodan. Pose Estimation of Specific Rigid Objects. *arXiv:2112.15075 [cs]*, December 2021.

[34] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of The Acm*, 15(1): 11–15, January 1972. ISSN 0001-0782. doi: 10.1145/361237.361242.

[35] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-Class Hough Forests for 3D Object Detection and Pose Estimation. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 462–477, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10599-4. doi: 10.1007/978-3-319-10599-4_30.

[36] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 205–220, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9.

[37] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8690, pages 536–551. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10604-5 978-3-319-10605-2. doi: 10.1007/978-3-319-10605-2_35.

[38] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 954–962, Santiago, Chile, December 2015. IEEE. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.115.

[39] Kiru Park, Timothy Patten, and Markus Vincze. Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. page 17, 2019.

[40] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16606–16616, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.01634.

[41] Tomas Hodan, Daniel Barath, and Jiri Matas. EPOS: Estimating 6D Pose of Objects With Symmetries. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11700–11709, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.01172.

[42] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *2019 IEEE/CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 2637–2646, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00275.

[43] Xiaolong Li, He Wang, Li Yi, Leonidas Guibas, A. Lynn Abbott, and Shuran Song. Category-Level Articulated Object Pose Estimation. *arXiv:1912.11913 [cs]*, April 2020.

[44] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. LatentFusion: End-to-End Differentiable Reconstruction and Rendering for Unseen Object Pose Estimation. *arXiv:1912.00416 [cs]*, June 2020.

[45] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4556–4565, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00469.

[46] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G. Derpanis, and Kostas Daniilidis. 6-DoF object pose from semantic keypoints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2011–2018, Singapore, Singapore, May 2017. IEEE. ISBN 978-1-5090-4633-1. doi: 10.1109/ICRA.2017.7989233.

[47] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv:1711.00199 [cs]*, November 2017.

[48] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martin-Martin, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[49] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv:1612.00593 [cs]*, December 2016.

[50] Yangzheng Wu, Alireza Javaheri, Mohsen Zand, and Michael Greenspan. Keypoint Cascade Voting for Point Cloud Based 6DoF Pose Estimation. In *2022 International Conference on 3D Vision (3DV)*, pages 176–186, September 2022. doi: 10.1109/3DV57658.2022.00030.

[51] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *2017 IEEE International Conference on Computer Vision*

(ICCV), pages 1530–1538, Venice, October 2017. IEEE. ISBN 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.169.

[52] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, and Rudolph Triebel. Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection. *International Journal of Computer Vision*, 128(3):714–729, March 2020. ISSN 1573-1405. doi: 10.1007/s11263-019-01243-8.

[53] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992. ISSN 1939-3539. doi: 10.1109/34.121791.

[54] M. Greenspan and M. Yurick. Approximate k-d tree search for efficient ICP. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 442–448, October 2003. doi: 10.1109/IM.2003.1240280.

[55] Pol Moreno, Christopher K. I. Williams, Charlie Nash, and Pushmeet Kohli. Overcoming Occlusion with Inverse Graphics. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, volume 9915, pages 170–185. Springer International Publishing, Cham, 2016. ISBN 978-3-319-49408-1 978-3-319-49409-8. doi: 10.1007/978-3-319-49409-8_16.

[56] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. *International Journal of Computer Vision*, 128(3):657–678, March 2020. ISSN 1573-1405. doi: 10.1007/s11263-019-01250-9.

[57] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1322–1328, Detroit, MI, USA, 1999. IEEE. ISBN 978-0-7803-5180-6. doi: 10.1109/ROBOT.1999.772544.

[58] Changhyun Choi and Henrik I. Christensen. RGB-D object tracking: A particle filter approach on GPU. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1084–1091, Tokyo, November 2013. IEEE. ISBN 978-1-4673-6358-7 978-1-4673-6357-0. doi: 10.1109/IROS.2013.6696485.

[59] Manuel Wuthrich, Peter Pastor, Mrinal Kalakrishnan, Jeannette Bohg, and Stefan Schaal. Probabilistic object tracking using a range camera. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3202, Tokyo, November 2013. IEEE. ISBN 978-1-4673-6358-7 978-1-4673-6357-0. doi: 10.1109/IROS.2013.6696810.

[60] Alexander Krull, Frank Michel, Eric Brachmann, Stefan Gumhold, Stephan Ihrke, and Carsten Rother. 6-DOF Model Based Tracking via Object Coordinate Regression. In Daniel Cremers, Ian Reid, Hideo Saito, and Ming-Hsuan Yang, editors, *Computer Vision – ACCV 2014*, volume 9006, pages 384–399. Springer International Publishing, Cham, 2015. ISBN 978-3-319-16816-6 978-3-319-16817-3. doi: 10.1007/978-3-319-16817-3_25.

[61] Henning Tjaden, Ulrich Schwanecke, Elmar Schomer, and Daniel Cremers. A Region-Based Gauss-Newton Approach to Real-Time Monocular Multiple Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1797–1812, August 2019. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2018.2884990.

[62] Wadim Kehl, Federico Tombari, Slobodan Ilic, and Nassir Navab. Real-Time 3D Model Tracking in Color and Depth on a Single CPU Core. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 465–473, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.57.

[63] Manuel Stoiber, Martin Sundermeyer, and Rudolph Triebel. Iterative Corresponding Geometry: Fusing Region and Depth for Highly Efficient 3D Tracking of Textureless Objects. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6845–6855, New Orleans, LA, USA, June 2022. IEEE. ISBN 978-1-66546-946-3. doi: 10.1109/CVPR52688.2022.00673.

[64] David Joseph Tan and Slobodan Ilic. Multi-forest Tracker: A Chameleon in Tracking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1202–1209, Columbus, OH, USA, June 2014. IEEE. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.157.

[65] Mathieu Garon and Jean-François Lalonde. Deep 6-DOF Tracking. *IEEE Transactions on Visualization and Computer Graphics*, 23(11):2410–2418, November 2017. ISSN 1077-2626. doi: 10.1109/TVCG.2017.2734599.

[66] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking. *arXiv:1905.09304 [cs]*, May 2019.

[67] Xiaotong Chen, Rui Chen, Zhiqiang Sui, Zhefan Ye, Yanqi Liu, R. Iris Bahar, and Odest Chadwicke Jenkins. GRIP: Generative Robust Inference and Perception for Semantic Robot Manipulation in Adversarial Environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3988–3995, Macau, China, November 2019. IEEE. ISBN 978-1-72814-004-9. doi: 10.1109/IROS40897.2019.8967983.

[68] Zhuowen Tu and Song-Chun Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 24(5), 2002.

[69] Matthew M. Loper and Michael J. Black. OpenDR: An Approximate Differentiable Renderer. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8695, pages 154–169. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10583-3 978-3-319-10584-0. doi: 10.1007/978-3-319-10584-0_11.

[70] Varun Jampani, Sebastian Nowozin, Matthew Loper, and Peter V. Gehler. The informed sampler: A discriminative approach to Bayesian inference in generative computer vision models. *Computer Vision and Image Understanding*, 136:32–44, July 2015. ISSN 10773142. doi: 10.1016/j.cviu.2015.03.002.

[71] Yanqi Liu, Giuseppe Calderoni, and Ruth Iris Bahar. Hardware Acceleration of Monte-Carlo Sampling for Energy Efficient Robust Robot Manipulation. In *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pages 284–290, August 2020. doi: 10.1109/FPL50879.2020.00054.

[72] Nishad Gothoskar, Marco Cusumano-Towner, Ben Zinberg, Matin Ghavamizadeh, Falk Pollok, Austin Garrett, Josh Tenenbaum, Dan Gutfreund, and Vikash Mansinghka. 3DP3: 3D scene perception via probabilistic programming. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9600–9612. Curran Associates, Inc., 2021.

[73] Jana Pavlasek, Stanley Lewis, Karthik Desingh, and Odest Chadwicke Jenkins. Parts-Based Articulated Object Localization in Clutter Using Belief Propagation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10595–10602, Las Vegas, NV, USA, October 2020. IEEE. ISBN 978-1-72816-212-6. doi: 10.1109/IROS45743.2020.9340908.

[74] Tim Übelhör, Jonas Gesenhues, Nassim Ayoub, Ali Modabber, and Dirk Abel. 3D camera-based markerless navigation system for robotic osteotomies. *at - Automatisierungstechnik*, 68(10):863–879, October 2020. ISSN 2196-677X, 0178-2312. doi: 10.1515/auto-2020-0032.

[75] Ken Shoemake. UNIFORM RANDOM ROTATIONS. In *Graphics Gems III*, pages 124–132. Elsevier, 1992. ISBN 978-0-12-409673-8. doi: 10.1016/B978-0-08-050755-2.50036-1.

[76] Avishek Chatterjee and Venu Madhav Govindu. Noise in Structured-Light Stereo Depth Cameras: Modeling and its Applications. *arXiv:1505.01936 [cs]*, May 2015.

[77] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. ISBN 2640-3498.

[78] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, June 2006. ISSN 1369-7412, 1467-9868. doi: 10.1111/j.1467-9868.2006.00553.x.

[79] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, January 2017. ISSN 0036-1445. doi: 10.1137/141000671.

[80] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *Journal of machine learning research*, 2017. ISSN 1532-4435.

[81] Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 814–822. PMLR, April 2014.

[82] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.2017.1285773.

[83] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods: Particle Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, June 2010. ISSN 13697412, 14679868. doi: 10.1111/j.1467-9868.2009.00736.x.

[84] Warren Morningstar, Sharad Vikram, Cusuh Ham, Andrew Gallagher, and Joshua Dillon. Automatic differentiation variational inference with mixtures. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3250–3258. PMLR, 2021-04-13/2021-04-15.

[85] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970. ISSN 0006-3444. doi: 10.1093/biomet/57.1.97.

[86] Jun S. Liu, Faming Liang, and Wing Hung Wong. The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association*, 95(449):121–134, March 2000. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.2000.10473908.

[87] Luca Martino. A Review of Multiple Try MCMC algorithms for Signal Processing. *Digital Signal Processing*, 75:134–152, April 2018. ISSN 10512004. doi: 10.1016/j.dsp.2018.01.004.

[88] L. Martino and F. Louzada. Issues in the Multiple Try Metropolis mixing. *Computational Statistics*, 32(1):239–252, March 2017. ISSN 0943-4062, 1613-9658. doi: 10.1007/s00180-016-0643-9.

[89] N. Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, August 2002. ISSN 0006-3444, 1464-3510. doi: 10.1093/biomet/89.3.539.

[90] Chenguang Dai, Jeremy Heng, Pierre E. Jacob, and Nick Whiteley. An invitation to sequential Monte Carlo samplers, June 2022.

[91] Thi Le Thu Nguyen, François Septier, Gareth W. Peters, and Yves Delignon. Efficient Sequential Monte-Carlo Samplers for Bayesian Inference. *IEEE Transactions on Signal Processing*, 64(5):1305–1319, March 2016. ISSN 1941-0476. doi: 10.1109/TSP.2015.2504342.

[92] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, April 2001. ISSN 1573-1375. doi: 10.1023/A:1008923215028.

[93] Ajay Jasra, David A. Stephens, Arnaud Doucet, and Theodoros Tsagaris. Inference for Lévy-Driven Stochastic Volatility Models via Adaptive Sequential Monte Carlo: Lévy-driven stochastic volatility. *Scandinavian Journal of Statistics*, 38(1):1–22, March 2011. ISSN 03036898. doi: 10.1111/j.1467-9469.2010.00723.x.

[94] Tor Erlend Fjelde, Kai Xu, Mohamed Tarek, Sharan Yalburgi, and Hong Ge. Bijectors.jl: Flexible transformations for probability distributions. In Cheng Zhang, Francisco Ruiz, Thang Bui, Adji Bousso Dieng, and Dawen Liang, editors, *Proceedings of the 2nd Symposium on Advances in Approximate Bayesian Inference*, volume 118 of *Proceedings of Machine Learning Research*, pages 1–17. PMLR, December 2020.

[95] G. O. Roberts and S. K. Sahu. Updating Schemes, Correlation Structure, Blocking and Parameterization for the Gibbs Sampler. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 59(2):291–317, July 1997. ISSN 1369-7412, 1467-9868. doi: 10.1111/1467-9868.00070.

[96] Daniel Turek, Perry de Valpine, Christopher J. Paciorek, and Clifford Anderson-Bergman. Automated Parameter Blocking for Efficient Markov Chain Monte Carlo Sampling. *Bayesian Analysis*, 12(2):465–490, June 2017. ISSN 1936-0975, 1931-6690. doi: 10.1214/16-BA1008.

[97] Tomas Hodan, Pavel Haluza, Stepan Obdrzalek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888, Santa Rosa, CA, USA, March 2017. IEEE. ISBN 978-1-5090-4822-9. doi: 10.1109/WACV.2017.103.

[98] Bertram Drost, Markus Ulrich, Paul Bergmann, Philipp Hartinger, and Carsten Steger. Introducing MVTec ITODD — A Dataset for 3D Object Recognition in Industry. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2200–2208, Venice, Italy, October 2017. IEEE. ISBN 978-1-5386-1034-3. doi: 10.1109/ICCVW.2017.257.

[99] Krishna Shankar, Mark Tjersland, Jeremy Ma, Kevin Stone, and Max Bajracharya. A Learned Stereo Depth System for Robotic Manipulation in Homes. *IEEE Robotics and Automation Letters*, 7(2):2305–2312, April 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3143895.

[100] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-009-0275-4.

[101] M. Fujita, Y. Domae, A. Noda, G. A. Garcia Ricardez, T. Nagatani, A. Zeng, S. Song, A. Rodriguez, A. Causo, I. M. Chen, and T. Ogasawara. What are the important technologies for bin picking? Technology analysis of robots in competitions based on a set of performance metrics. *Advanced Robotics*, 34(7-8):560–574, April 2020. ISSN 0169-1864. doi: 10.1080/01691864.2019.1698463.

[102] Le Duc Hanh and Khuong Thanh Gia Hieu. 3D matching by combining CAD model and computer vision for autonomous bin picking. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 15(2):239–247, September 2021. ISSN 1955-2505. doi: 10.1007/s12008-021-00762-4.

[103] Carlos Martinez, Remus Boca, Biao Zhang, Heping Chen, and Srinivas Nidamarthi. Automated bin picking system for randomly located industrial parts. In *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, May 2015. doi: 10.1109/TePRA.2015.7219656.

[104] Ali Modabber, Nassim Ayoub, Tim Redick, Jonas Gesenhues, Kristian Kniha, Stephan Christian Möhlhenrich, Stefan Raith, Dirk Abel, Frank Hölzle, and Philipp Winnand. Comparison of augmented reality and cutting guide technology in assisted harvesting of iliac crest grafts – A cadaver study. *Annals of Anatomy - Anatomischer Anzeiger*, 239:151834, January 2022. ISSN 0940-9602. doi: 10.1016/j.aanat.2021.151834.

[105] Manuel Wüthrich, Jeannette Bohg, Daniel Kappler, Claudia Pfreundt, and Stefan Schaal. The Coordinate Particle Filter - a novel Particle Filter for high dimensional systems. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2454–2461, May 2015. doi: 10.1109/ICRA.2015.7139527.

[106] David Stenger, Maximilian Nitsch, and Dirk Abel. Joint Constrained Bayesian Optimization of Planning, Guidance, Control, and State Estimation of an Autonomous Underwater Vehicle. In *2022 European Control Conference (ECC)*, pages 1982–1987, July 2022. doi: 10.23919/ECC55457.2022.9838053.

[107] Russ Tedrake. *Robotic Manipulation. Perception, Planning, and Control*. 2023.

[108] C. Ferrari and J. Canny. Planning optimal grasps. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295 vol.3, May 1992. doi: 10.1109/ROBOT.1992.219918.

[109] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DOF GraspNet: Variational Grasp Generation for Object Manipulation. *arXiv:1905.10520 [cs]*, May 2019.

[110] F.C. Park and B.J. Martin. Robot sensor calibration: Solving AX=XB on the Euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5): 717–721, 1994. doi: 10.1109/70.326576.

[111] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the Barrier to Entry of Complex Robotic Software: A MoveIt! Case Study, April 2014.

[112] Michael Görner, Robert Haschke, Helge Ritter, and Jianwei Zhang. MoveIt! Task constructor for task-level motion planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 190–196, 2019. doi: 10.1109/ICRA.2019.8793898.

[113] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.143.

[114] Mykel J. Kochenderfer, Tim A. Wheeler, and Kyle H. Wray. *Algorithms for Decision Making*. The MIT Press, Cambridge, Massachusetts, 2022. ISBN 978-0-262-04701-2.

[115] Johannes Kummert, Alexander Schulz, Tim Redick, Nassim Ayoub, Ali Modabber, Dirk Abel, and Barbara Hammer. Efficient Reject Options for Particle Filter Object Tracking in Medical Applications. *Sensors*, 21(6):2114, January 2021. doi: 10.3390/s21062114.

[116] Timothy Patten, Michael Zillich, Robert Fitch, Markus Vincze, and Salah Sukkarieh. Viewpoint Evaluation for Online 3-D Active Object Classification. *IEEE Robotics and Automation Letters*, 1(1):73–81, January 2016. ISSN 2377-3766. doi: 10.1109/LRA.2015.2506901.

[117] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6):1–14, December 2020. ISSN 0730-0301, 1557-7368. doi: 10.1145/3414685.3417861.

[118] P. L. Green, L. J. Devlin, R. E. Moore, R. J. Jackson, J. Li, and S. Maskell. Increasing the efficiency of Sequential Monte Carlo samplers through the use of approximately optimal L-kernels. *Mechanical Systems and Signal Processing*, 162:108028, January 2022. ISSN 0888-3270. doi: 10.1016/j.ymssp.2021.108028.

[119] Douglas. Hanggi and Peter W. Carr. Errors in exponentially modified Gaussian equations in the literature. *Analytical Chemistry*, 57(12):2394–2395, October 1985. ISSN 0003-2700, 1520-6882. doi: 10.1021/ac00289a051.