DOCTORAL THESIS

Objects Pose Tracking on RGB Images

Author: mgr inż. Mateusz Majcher

Supervisor: Prof. Bogdan Kwolek, Ph.D., D.Sc.

AGH University of Science and Technology in Krakow

Faculty of Computer Science

Krakow, 2025

DZIEDZINA: Nauki Inżynieryjno-Techniczne
DYSCYPLINA: Informatyka Techniczna i Telekomunikacja

ROZPRAWA DOKTORSKA

Śledzenie Pozy Obiektów na Obrazach RGB

Autor: mgr inż. Mateusz Majcher

Promotor rozprawy: prof. dr hab. inż. Bogdan Kwolek

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Informatyki

Kraków, 2025

Firstly, I would like to express my sincere gratitude to my thesis supervisor, Professor Bogdan Kwolek, for his support and assistance in carrying out the research described in this dissertation. A special gratitude goes to my wife, Justyna, for her unwavering patience, continuous support, and for always offering invaluable help whenever it was needed. I also acknowledge that this work was supported by Polish National Science Center under research grant 2017/27/B/ST6/01743.

Streszczenie

Śledzenie sześciowymiarowej pozy obiektu jest klasycznym problemem w widzeniu komputerowym, który jest wykorzystywany w wielu dziedzinach takich jak robotyka, autonomiczne samochody, czy medycyna. Celem jest określenie trzywymiarowej rotacji oraz trzywymiarowej translacji względem obserwującej kamery. Ze względu na zmiany oświetlenia, przesłonięcia oraz niejednoznaczność widzianej strony obiektu, wyznaczanie sześciowymiarowej pozy stanowi złożone i wymagające wyzwanie. Niniejsza rozprawa doktorska odnosi się do wybranych wyzwań, proponując metody wykorzystujące obrazy RGB, które poprawiają śledzenie pozy obiektu.

W niniejszej rozprawie zaproponowano kilka rozwiązań, które poprawiają wykrywalność punktów charakterystycznych na obrazach oraz poprawiają dokładność śledzenia pozy obiektu. Jednym z głównych wkładów jest wykorzystanie dodatkowych informacji z wcześniejszych klatek poprzez dostarczenie ich na wejście sieci neuronowej w celu poprawy precyzji określania punktów charakterystycznych w niejednoznacznych sytuacjach. W tym celu zaproponowane zostały nowe architektury sieci neuronowych, które dodatkowo posiadają osobno wytrenowaną część z wejściem do poprawy estymaty pozy obiektu na podstawie informacji o rotacji z poprzedniej klatki. Ponadto opracowano metody wykorzystujące wielomodalne dane w celu poprawy wyznaczania pozy w razie przesłonięć oraz niejednoznaczności. Rozprawa bada metody do analizy zasłoniętych lub niewidocznych punktów charakterystycznych przy pomocy opisanych segmentów kształtu obiektu oraz odległości od rzutowanych krawędzi obiektu. Punkty charakterystyczne, nieznajdujące się na widocznych częściach obiektów, w znaczący sposób pogarszają estymację rotacji i translacji, dlatego istotne jest ich wykrywanie oraz usuwanie. Dodatkowo, zaproponowano przestrzeń rotacji opartą na kwaternionach dla filtrów cząsteczkowych oraz rojów cząsteczek by wspomagać proces wyszukiwania optymalnej pozy. Wyniki prac eksperymentalnych pokazują, że zaproponowane metody pozwalają w znaczym stopniu zniwelować negatywne wpływy przesłonięć oraz optymalnie wykorzystać istniejące dane w trakcie śledzenia pozy obiektów. Zaproponowane metody osiągnęły wyniki state of the art na ogólnodostępnych zbiorach danych, charkteryzujących się zarówno zróżnicowanym ruchem kamery jak i przesłonięciami obiektów.

Metody i ich eksperymentalna ewaluacja przedstawione w niniejszej rozprawie potwierdzają tezy badawcze: Po pierwsze, architektury sieci neuronowych wykorzystujące dodatkową informacje z poprzedniej klatki mogą poprawić precyzję wykrywania punktów charakterystycznych. Po drugie, wykorzystanie wielomodalnych danych w celu poprawy wyznaczania pozy obiektu pozwala zniwelować negatywne wpływy przesłonięć. Filtry cząsteczkowe oraz roje cząsteczek wspomagają proces wyszukiwania optymalnej pozy wraz z metodami optymalizacyjnymi.

Abstract

Tracking the six-dimensional pose of an object is a classic problem in computer vision, with applications in various fields such as robotics, autonomous vehicles, and medicine. The goal is to estimate the three-dimensional rotation and the three-dimensional translation relative to the observing camera. However, due to changes in lighting, occlusion, and ambiguities on the visible side of the object, determining the six-dimensional pose is a complex and challenging task.

This dissertation addresses these challenges by proposing methods utilizing RGB images that improve object pose tracking. The research introduces several solutions that enhance the detectability of characteristic points on images and improve the accuracy of the tracked object pose. One of the main contributions is the use of additional information from previous frames to provide input to a neural network, improving the precision of point detection in ambiguous situations. New architectures are proposed that have an extra branch trained separately to improve estimation based on information from the previous frame. Furthermore, methods utilizing multimodal data are developed to improve pose estimation in occlusion and ambiguity-prone scenarios. The dissertation also investigates techniques for analyzing hidden or invisible characteristic points using described object shape segments and distances from projected object edges. These points significantly degrade rotation and translation estimation, making their detection and removal essential. In addition, quaternion-based rotation space is used in particle filters and swarms of particles to support the search for the optimal pose. Experimental results show that the proposed methods can significantly mitigate the negative effects of occlusion and optimally utilize existing data for object pose tracking. The methods have achieved state-of-the-art results on publicly available datasets characterized by diverse camera motion and occluded objects.

The methods and experimental evaluation presented in this dissertation confirm the research hypotheses: first, neural network architectures utilizing additional information from previous frames can improve point detection precision. Second, multimodal data application for object pose estimation allows for mitigating the negative effects of occlusions. Particle filters and swarms of particles, combined with optimization methods, support the search for the optimal pose.

Contents

List of Symbols and Abbreviations

List of Abbreviations

6 DoF	6 Degrees of Freedom
ADD	Average Distance of Model Points
ADD-S	Average Distance of Model Points for symmetric objects
AR	Augumented Reality
AUC	Area Under Curve
BCE	Binary Cross-Entropy
CNN	Convolutional Neural Network
EPnP	Effective Perspective-n-Point
HOO	Hierarchical Optimistic Optimization
ICP	Iterative Closest Point
KDE	Kernel Density Estimation
LiDAR	Light Detection and Ranging
LM	Levenberg-Marquardt
MAE	Mean Absolute Error
MCTS	Monte Carlo Tree Search
MLP	Multilayer Perceptron
MSE	Mean Square Error
OPT	Object Pose Tracking
PF	Particle Filter
PnP	Perspective-n-Point
PSO	Particle Swarm Optimization
PW	Progressive Widening
RANSAC	Random Sample Consensus
RGB	red, green, and blue
RGBD	RGB with depth channel
RMIS	Robot-Assisted Minimally Invasive Surgery
RMSE	Root Mean Square Error
RMSprop	Root Mean Squared Propagation
SDT	Signed Distance Transform
SLERP	Spherical Linear Interpolation
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
VDC-fuzzyTOPSIS	Voting-based Decision Committee-fuzzyTOPSIS
VR	Virtual Reality
YCB	Yale-CMU-Berkeley

List of Symbols

$\mathbb{R}^4$	four-dimensional space
$\mathbb{S}^3$	three-dimensional unit sphere
$\mathbf{K}$	matrix of intrinsic parameters of the camera
$\mathbf{p}_i$	best position of particle number i
$\mathbf{q}$	quaternion
$\mathbf{R}$	rotation
$\mathbf{x}$	pose $[\mathbf{q}\ \mathbf{z}]$
$\mathbf{x}_i$	particle number i
$\mathbf{z}$	translation
$\tilde{e}_{ij}$	triangle fuzzy number $(e^a_{ij}, e^b_{ij}, e^c_{ij})$
A^+	positive ideal solution
A^-	negative ideal solution
a_i	alternatives
BC	Benefit Criteria
BN	Batch Normalization
$C(k;n;s)$	convolutional layer with a kernel of size $k \times k$, with n filters and stride of s
C_j	criteria number j
c_x, c_y	optical centers
CC	Cost Criteria
$CT(k;n;s)$	transposed convolution with a kernel of size $k \times k$, with n filters and stride of s
F_M	set of triplets of vertices defining model faces
f_x, f_y	focal lengths
FC	Fully Connected
lr	learning rate
M	set of 3D model points
P	3D point
r_{11} to r_{33}	a rotation of the object in rotation matrix
S^3	set of unit-length quaternions forms
t	time
t_x, t_y, t_z	translation of the object
Tr	matrix trace
u, v	position on image plain
V_M	finite set of model vertices
w_j	weight of criterion C_j
X, Y, Z	coordinates of 3D point

Chapter 1

Introduction

This chapter presents the context of the research conducted in this thesis. First, the background and motivation for the research are presented. Then, the research goals and contributions are discussed. Finally, the content of this thesis is presented.

1.1. Motivation

The advancement towards automation in various fields such as robotics, autonomous vehicles, medicine, and augmented reality (AR) relies heavily on the accuracy and speed at which systems can process information. These systems utilize data from RGB cameras, GPS receivers, scanners such as LiDAR (Light Detection and Ranging) which provide information about distances, and other sensors. In autonomous vehicles [54, 114], cameras and other sensors play a crucial role not only in determining and planning the driving route but also in detecting signs, obstacles, and other road users. Accurate positioning of vehicles and other objects on the road is a vital element influencing road safety and quality of traffic congestion. Pose estimation allows the autonomous vehicle to anticipate the movements of nearby cars and people, which is required to make maneuvers or maintain a safe distance. Both cameras, which are cost-effective but struggle to provide distance information at night, and LiDARs, which, due to their significantly higher cost compared to standard RGB cameras, are less commonly used but can be more precise in complete darkness, are employed for this purpose. Despite significant advancements, no single technology has yet gained widespread acceptance as a definitive solution for self-driving vehicles on public roads [63].

It is expected that intelligent robots will be able to observe and interact with their environment, going beyond simple navigation [37]. One of the most important skills of a robot is its ability to grasp and manipulate objects, which is not only applied in industry, where it reduces the need for human labor [188] but also in home settings, where robots can assist the elderly or individuals with disabilities [2]. To grasp an object, a robot must first locate the object, determine its position, and then plan its gripper movement trajectory. All these steps should be executed in real time using an embedded system of the robot.

Over the past decade, Robot-Assisted Minimally Invasive Surgery (RMIS) has undergone significant advancements, fueled by breakthroughs in artificial intelligence (AI) and surgical robotics. The introduction of platforms like the da Vinci system has transformed surgical procedures, offering enhanced control over instruments and intraoperative visualization, thereby improving surgical assistance. Accurate estimation of surgical instrument pose is now a vital component of RMIS, as it enables applications such as autonomous task execution [174], surgical skill assessment [50], and surgical workflow analysis [83].

In recent years, virtual reality and augmented reality have attracted the interest of investors and the general public [23]. Notably, VR tools are no longer just entertainment platforms but also serve as valuable research instruments for scientists from disciplines such as neuroscience, psychology, biology, and beyond, offering new avenues for exploration and discovery [31, 19]. This innovative approach enables users to engage with richly enhanced surroundings that combine the best of both worlds - the authenticity of reality and the versatility of digital content. The process of augmenting reality necessitates two primary steps: first, the region or object within the image that is to be augmented must be accurately detected through object detection; secondly, the orientation or pose of the augmented content must be precisely estimated.

In conclusion, all of the tasks mentioned earlier are partially or completely based on proper estimation and tracking of human or object pose. Monocular camera-based pose estimation poses several unique challenges. Most existing solutions rely on estimating pose from a single image, with few approaches utilizing tracking methods that involve additional complexities. Leveraging information from previous frames, maintaining hypotheses about changing poses, or detecting and removing negative influences on the tracking process are issues that have not received sufficient attention.

1.2. Object pose estimation and tracking

The estimation of a pose in monocular vision, a crucial task in computer vision, involves identifying and determining the orientation and position of objects or individuals within an image [44]. It can be divided into many categories, such as human pose estimation [75], head pose estimation, face pose estimation, and object pose estimation. This PhD thesis aims to contribute to object pose tracking.

Tracking the object's pose on images differs from object pose estimation, as it incorporates additional information about the object's pose from previous frames. Depending on the input data, methods can be categorized into those relying solely on RGB images and those that utilize RGB with depth (RGBD). The use of RGB or RGBD images depends on available cameras and sensors. RGB images lack depth information, making the 6 degrees of freedom (6 DoF) object pose detection task an ill-posed problem. Depth cameras increase the overall cost of systems and cannot be used in every situation, e.g., heavy rain or transparent objects. Existing object pose estimation methods can be broadly categorized [44] into several main groups: direct methods, keypoints methods, refinement methods, and self-supervised methods.

Direct methods [183, 73], which treat pose estimation as a regression [183] or classification problem [73], are often more lightweight and easy to train, but their performance is usually poorer than that of other methods or they are highly dependent on time-consuming pose refinement. By building the 2D-3D correspondences [133, 160], keypoint-based methods are more accurate and robust, but pose detection is highly affected by the keypoints detection results. In the case of failure, analyzing the reason is difficult. Refinement-based methods [89, 186], which estimate the object pose by aligning object renderings with real observed images, have shown great promise as a postprocess step for other object pose estimation methods. However, their computational cost is heavily dependent on the number of iterations, and they can become a bottleneck for the entire system. To reduce annotation costs, self-supervised methods [37] have advanced, but their performance remains far from satisfactory [44]. Additionally, these methods can be grouped into retrieval-based methods [73], in which a dataset of synthetic images covering all possible object poses is first generated. Then, the most similar image of the target object is retrieved to determine its pose.

This group of methods can achieve robust performance, but they require discretizing the rotation space to define the codebook, which can be computationally costly when the discretization gap is small.

In the case of object pose tracking, there are two main groups of methods: tracking by refinement and tracking by optimization [44]. Similarly to refinement-based methods, tracking by refinement is accurate and can achieve real-time performance, but the quality of 3D models and render highly affects the tracking results. Tracking by optimization [36], which uses optimized tracking schemes e.g., particle filtering, is highly interpretable, but it cannot achieve end-to-end training and testing. Also, the optimization process is often slow. The state-of-the-art methods are analyzed in Chapter 2.

Taking into account all of the latest developments in this field, there are a few challenges that must be addressed and possible future research directions to follow. Situations in which an occlusion, truncation, or cluttered background affects the object pose estimation should be analyzed and treated accordingly. Another challenge is the utilization of information from more than one image frame, which leads to object pose tracking, but it can also result in the accumulation of errors and drifting of the bounding box. On top of all that, the execution time of methods should be decreased to operate in real-time. While monocular cameras offer a cost-effective option, they are limited by their inability to provide depth information. This lack of depth perception can lead to significant challenges in accurately determining the distance of objects from the camera. Motivated by the greater versatility of RGB cameras compared to RGBD cameras, this lack of depth information in RGB images is considered one of the key obstacles to overcome. Considering the current state-of-the-art methods, it is evident that there is an immense number of possible applications for new and competitive systems that focus on tracking objects in RGB images.

1.3. Thesis statement

Based on the above considerations, the goal of this thesis is to develop methods that improve tracking the pose of the observed object using only RGB images. In the proposed approaches, a new type of neural network is introduced, in which additional information from the previous frame is fed to a neural network to determine more precise poses in ambiguous situations. The attained pose is then refined by a multi-stage method. The thesis of this research is formulated as follows:

Two-input neural networks with multi-stage pose refinement improve tracking of object's pose on RGB images.

The dissertation is devoted to several aspects, related to improving the tracking ability of systems. First of all, the usage of information from previous frames is addressed. Secondly, new methods based on geometric analysis to detect and counter occlusions are introduced. The proposed solutions, with regard to all of the discussed issues, can operate in real-time, which is a crucial aspect for systems that work on embedded devices.

Main thesis contributions:

- novel two-input architecture for a neural network that utilizes information about the rotation of an object in quaternion representation to enhance the detection of characteristic points of the object
- multi-stage pose refinement that uses keypoints and geometric reasoning

Thesis contributions in detail:

- rotation guided Y-Net, which uses an additional input to encode data from the previous and current frames, and encodes it using heatmaps
- rotation guided X-Net, which encodes the current pose using heatmaps and object masks
- pose optimization and refinement strategy that, apart from keypoints, uses also signed distance maps
- method for detecting invisible or occluded keypoints using labeled segments of the object shape
- method for estimating potential occlusion of an object using values from the distance transform on projected boundary segments
- a unit quaternion representation of the rotational state space for a particle filter and particle swarm optimization
- Siamese neural network, that guides the particle representing the object pose hypothesis toward the predicted pose of the object
- SOTA results on publicly available YCB-Video and OPT datasets

1.4. Dissertation structure

This dissertation is organized as follows. Chapter 2 presents the background for the research. A thorough review of pose estimation and tracking methods is provided. Then, publicly available datasets are discussed. Chapter 3 focuses on representations, and methods of evaluation, and presents the theoretical background. In Chapter 4, the architecture of a neural network for pose tracking with the usage of quaternion information is proposed. Issues regarding object occlusions, geometric reasoning, and object shape are addressed. Chapter 5 focuses on extending the neural network proposed in Chapter 4. The Hierarchical Optimistic Optimization is proposed to choose the correct keypoints. In Chapter 6, a combination of particle filtering and particle swarm optimization, which uses a quaternion representation of rotation, is proposed. An extensive evaluation of the proposed methods is provided. Chapter 7 describes the proposed TOPSIS Object Pose Refinement using the Hourglass neural network. Chapter 8 concludes the dissertation and presents directions for further work.

1.5. Projects and publications

The research in this thesis was supported by the Polish National Science Center under a research grant 2017/27/B/ST6/01743. The findings included in this dissertation have been previously published through publications in the following journal articles:

- M. Majcher and B. Kwolek, Object pose tracking using multimodal knowledge from RGB images and quaternion-based rotation contexts, Applied Soft Computing, vol. 170, p. 112699, Feb. 2025, IF = 7.2, (200 pts.)*
- M. Majcher and B. Kwolek, Multi-Modal Pose Representations for 6-DOF Object Tracking, Journal of Intelligent & Robotic Systems, vol. 110, no. 4, p. 149, Oct. 2024, IF = 3.1, (100 pts.)

* Ministerial list of indexed journals (Wykaz czasopism naukowych), https://www.gov.pl/web/nauka/, accessed 12.02.2025

- M. Majcher and B. Kwolek, TOPSIS Aided Object Pose Tracking on RGB Images, IEEE Access, vol. 11, pp. 139498-139508, 2023, IF = 3.4, (100 pts.)

The results have also been presented at the following conferences:

- M. Majcher and B. Kwolek. Shape Enhanced Keypoints Learning With Geometric Prior for 6D Object Pose Tracking. In Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops, pages 2986-2992, IEEE, June 2022, (200 pts., core Rank $A^{\star}$)[†]
- M. Majcher and B. Kwolek. Pose Guided Feature Learning for 3D Object Tracking on RGB Videos. In Proc. of the 17th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2022, Volume 5: VISAPP, 2022, pages 574-581. SciTePress, 2022, (70 pts., core Rank B)
- M. Majcher and B. Kwolek. Multiple-criteria-Based Object Pose Tracking in RGB Videos. Int. Conf. on Computational Collective Intelligence, ICCCI 2022, LNAI 13501, pages 1-14. Springer, 2022, (70 pts., core Rank B)
- M. Majcher and B. Kwolek. Quaternion-driven CNN for object pose tracking. In 2021 Int. Conf. on Visual Communications and Image Processing (VCIP), pages 1-4, IEEE, 2021, (70 pts., core Rank B)
- M. Majcher and B. Kwolek. Fiducial Points-supported Object Pose Tracking on RGB Images via Particle Filtering with Heuristic Optimization. In Proc. of the 16th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, pages 919-926. INSTICC, SciTePress, 2021, (70 pts., core Rank B)
- M. Majcher and B. Kwolek. Deep Quaternion Pose Proposals for 6D Object Pose Tracking. In 2021 IEEE/CVF Int. Conf. on Computer Vision Workshops (ICCVW), pages 243-251, IEEE, 2021, (200 pts., core Rank $A^{\star}$)
- M. Majcher. and B. Kwolek. 3D Model-based 6D Object Pose Tracking on RGB Images using Particle Filtering and Heuristic Optimization. In Proc. of the 15th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, pages 690-697. INSTICC, SciTePress, 2020, (70 pts., core Rank B)
- M. Majcher and B. Kwolek. 3D Model-Based 6D Object Pose Tracking on RGB Images. In Intelligent Information and Database Systems, ACIIDS, pages 271-282, Lecture Notes in Computer Science, vol. 12033, Springer, 2020, (20 pts., core Rank B)

[†] Core page, https://www.core.edu.au/conference-portal

Chapter 2

Background and related work

This chapter discusses the context of the research and explains the motivation behind addressing specific problems related to object pose. A thorough review of methods concerning various aspects of pose estimation and tracking is conducted. Based on the literature analyzed [44, 63], the following approaches to object pose estimation or tracking are distinguished and discussed in this thesis:

– 6D object pose estimation on single images

– 6D object pose tracking

– 6D object pose refinement

The key methods in each category are described in the following sections. This chapter is organized as follows. The problem of pose estimation on single images is described in Section 2.1. Then, in Section 2.2, 6D object pose tracking is discussed, followed by a description of pose refinement in Section 2.3. This chapter concludes with a presentation and discussion of the available datasets in Section 2.4.

2.1. Pose estimation from single images

While significant progress has been made in recognizing 2D objects within images or videos, the challenge remains to accurately identify a 3D object and determine its three-dimensional description solely from a single image [111]. Two primary Convolutional Neural Network (CNN)-based methods have emerged as approaches for estimating the six degrees of freedom (6 DoF) pose of an object [44]. The first method involves directly predicting the 6 DoF pose from an RGB image, without relying on intermediate processing steps, while the second method entails predicting the 2D key-point locations within the image and subsequently utilizing the Perspective-n-Point (PnP) algorithm to determine the object's orientation; more details are given in Section 3.4. Furthermore, another approach can be mentioned: refinement-based methods, which are described in detail in Section 2.3.

2.1.1. State of the art

One of the most intuitive ways to estimate the object pose is to treat the problem as a regression or classification task and directly predict object pose-related parameters from input images [44]. PoseCNN [183] was the first convolutional neural network (CNN) for direct regression of 6 DoF object poses. This network estimates the object pose in two stages. The initial stage of the network comprises 13 convolutional layers and 4 maxpooling layers, which produce feature maps with varying resolutions from the input image. The core component of this architecture generates features that are shared across all subsequent branches. In the second stage, an embedding step is executed to transform the high-dimensional feature maps produced by the first stage into low-dimensional, task-specific features tailored for specific tasks. Three distinct tasks within this stage contribute to 6D pose estimation:

semantic labeling, 3D translation estimation, and 3D rotation regression. Initially, semantic labeling occurs, wherein each image pixel is categorized according to its corresponding object class. The embedding step of the semantic labeling branch takes two feature maps with channel dimension 512 generated by the feature extraction stage as inputs. The output of this branch has n channels with n denoting the number of the semantic classes. Concurrently, the network determines the 2D center of the object in the image and calculates its distance from the camera. Using the object bounding boxes predicted from the semantic labeling layer, two regions of interest pooling layers are utilized to crop the visual features generated by the first stage of the network for the 3D rotation regression. The pooled feature maps are then concatenated and fed into a sequence of Fully Connected (FC) layers. For each class, the last FC layer outputs a 3D rotation represented by a quaternion. Additionally, a large-scale video dataset for 6D object pose estimation named the YCB-Video dataset was introduced, more details are in Section 2.4.

In SSD-6D [73], the authors treat pose estimation as a classification problem and train neural networks to classify the image features into a discretized pose space. Their network takes a processed RGB image to output localized 2D detections using bounding boxes with a pool of the most likely 6D poses for that instance. To represent a 6D pose, the scores are parsed for viewpoint and in-plane rotation, which have been inferred from the network and then projective properties are used to instantiate 6D hypotheses. It extends the 2D SSD [93] architecture for 6 DoF pose estimation.

Some approaches modify indirect methods into direct ones by utilizing neural networks to establish 2D-3D correspondences directly and by simulating the PnP [177] algorithm using deep networks. For example, GDR-Net [175] is a geometry-guided direct regression network that uses Patch-PnP solver [175] as part of the whole network. The entire GDR-Net framework can be trained end-to-end. A key advantage of direct methods is that they are usually lightweight.

Direct methods offer several benefits, including being lightweight and straightforward to implement. Nevertheless, their performance often lags behind other approaches due to their reliance on deep networks to recover the complete 6 DoF parameters [44].

Directly estimating the 6 DoF pose from a single RGB image is an ill-posed problem and faces many challenges. Instead of focusing on directly regressing the 6D object pose, a more popular approach is to establish 2D-3D correspondences and employ them to predict pose-related parameters. BB8 [133] first uses a deep network to segment the object and identify the 2D centers. Then, another network, based on the VGG network [150], is applied to an image window centered on the 2D object to predict the 2D projections of fixed points, e.g., the 3D corners of the encapsulating bounding box.

Similar to BB8, Tekin et al. [160] directly predict the 2D image locations of the projected vertices of the object's 3D bounding box. The object's 6D pose is then estimated using a PnP algorithm [84]. The method is implemented based on YOLO [135] and was extended for 6 DoF object pose detection, namely YOLO-6D.

Oberweger et al. [121] proposed a heatmap-based method that predicts keypoints from multiple small patches independently and accumulates the results to obtain accurate and robust predictions, even in cases of partial occlusions. To combine the contributions of different patches, a simple ensemble approach was used to average the predicted heatmaps for each 2D projection. Then, the locations of the global maxima, after averaging, were taken as the final predictions for the 2D projections. To compute the pose, PnP estimation with RANSAC [55] on the correspondences between the corners of the object's 3D bounding box and the heatmap locations was performed.

In PVNet [128], a Pixel-wise Voting Network was proposed to regress pixel-wise unit vectors pointing to the keypoints and then use these vectors to vote for keypoint locations via RANSAC. This method also presents a new representation for 2D object keypoints and a modified PnP algorithm for pose estimation. PVNet outputs pixel-wise object labels and unit vectors that indicate the direction from each pixel to every keypoint, allowing all pixels within an object to contribute to estimating the location of a certain keypoint by generating hypotheses and corresponding confidence scores through RANSAC-based voting [46]. Then, using these hypotheses, the mean and covariance of the spatial probability distribution for each keypoint can be estimated. This method enables a versatile approach to locate occluded or truncated keypoints, allowing for more accurate tracking of objects and robustness in challenging scenarios. Additionally, this representation provides uncertainty estimates for keypoint positions, which can be exploited by the PnP solver to refine its solutions and improve overall performance. The above-mentioned methods represent mainstream works, and the review [111] provides an overview of current approaches.

2.2. Pose tracking

While much work has been done in the area of 6D pose estimation, comparatively few studies have focused on 3D object pose tracking [44], as the field is still in a relatively early stage of development.

2.2.1. State of the art

One of the first works to leverage deep learning for object pose tracking is Deep 6-DOF Tracking (D6DT) [51], in which on the basis of the current RGBD image and a synthetic rendered RGBD frame the relative 6 DoF object pose between these two input frames is predicted. A rendering-based method to generate the data necessary to train a deep network is employed. To enhance the network to be robust to various situations, both frames are synthesized by rendering a 3D model of the object and simulating realistic capture conditions, including object positions, backgrounds, noise, and lighting. The trained network estimates the transformation between both frames as concatenated translations and Euler angles in the camera reference frame.

Unlike D6DT, which requires RGBD data, DMB6D [108] takes only RGB images as input and aligns object contours in image sequences. Given the input image and pose hypothesis, the object is rendered, the center of the bounding box of the hypothesis is computed and then a scene patch is cut out, and finally, a patch is rendered. The authors feed them separately into pre-trained InceptionV4 [156] layers to extract low-level features. Thereafter, high-level features are concatenated and computed before diverging into separate branches. Then, the pose update is retrieved as a 3D translation and a normalized quaternion. The final predicted pose is coarse because refinement through a single forward pass is performed.

In contrast to D6DT, where a single forward pass is used, DeepIM [89] iteratively matches the rendered images of the object model against the input image for pose refinement. A 6D pose estimate of an object, obtained from methods like PoseCNN or previous iteration refinements, is fed into the method along with the object's 3D model. DeepIM generates a rendered image that simulates how the target object would appear under this initial pose estimate. This rendered image, paired with its corresponding observed image, serves as input to the network to predict a relative transformation that refines the initial pose. The refined pose can then be used as input in subsequent iterations until convergence or until a predetermined number of iterations is reached, enabling iterative refinement of the object's pose estimate. The network uses optical flow [39]; therefore, it requires a large number of real-world images for training and has considerable computational requirements, which depend heavily on the number of iterations and the renderer used.

The above-mentioned methods are refinement-based approaches, as the pose estimated in the last frame is refined by a neural network. In contrast to the above tracking-by-refinement scheme, PoseRBPF [36] leverages a Rao-Blackwellized particle filter [119] and an auto-encoder network to estimate the 3D translation and the full distribution of the 3D rotation of an object undergoing tracking in an RGB image sequence. This is achieved by discretizing the rotation space in a fine-grained manner and training an auto-encoder network to construct a codebook of feature embeddings for the discretized rotations. For each particle, PoseRBPF first uses the 3D translation to determine the center and size of the object bounding box in the image, then determines the embedding for that bounding box, and finally updates the rotation distribution by comparing the embedding value with the pre-computed entries in the codebook using cosine distance. The weight of each particle is determined by the normalization constant associated with its rotational probability distribution. Efficient motion updating is achieved through the use of a sampling technique that involves drawing from a pose distribution and performing a convolution over the rotational space. In contrast, this distribution captures the uncertainty in the object's pose, and owing to tracking the algorithm can better disambiguate the pose of the object. This means that after temporal occlusion, the algorithm can recover the pose by tracking it from previous frames.

2.3. Pose refinement

Due to occlusions, lighting, backgrounds, and noise, it can be challenging to estimate an accurate pose in a single-shot setting. This has motivated several methods to use iterative refinement techniques to obtain more accurate pose estimates. As mentioned previously, several methods [89, 183] require a refinement step to improve their accuracy. While some of these methods were mentioned earlier, it is essential to revisit and elaborate on them as they often serve as the primary means of enhancing predictive accuracy.

2.3.1. State of the art

Some experimental evaluations in [183, 73] used depth information to refine the object pose using the Iterative Closest Point (ICP) algorithm [6]. This is done by rendering a predicted point cloud from the 3D model and the estimated pose and assuming that each observed depth value corresponds to the predicted depth value at the same pixel location.

There are situations where one algorithm uses another as a starting point for visual pose estimation. For example, PoseCNN [183] is combined with DeepIM [89], where during each iteration, it uses the current estimate of object pose to render the 3D model, then uses both, the rendered object and the image, to regress a pose update to better align the image. Notably, DeepIM can perform object tracking by using the refined pose estimate from the previous frame as the initial pose of the current frame [89].

In [173], the DenseFusion framework was proposed as a generic approach for estimating the 6D pose of known objects from RGBD images. This heterogeneous architecture separates the processing of the two data sources and employs a novel dense fusion network to extract pixel-wise dense feature embeddings, which are then used to estimate the object pose. To further refine the object pose, an end-to-end iterative procedure is employed, resulting in improved tracking accuracy.

CosyPose [81] builds on the refinement idea by using improved network architectures and rotation parameterizations. Multi-view optimization is used, in which the 6D poses of all objects and cameras are refined to minimize a global reprojection error. This approach consists of three main stages. In the first stage, initial object candidates are obtained in each view separately. In the next stage, these object candidates are matched across views to recover a single consistent scene. In the third stage, all objects and camera poses are refined globally to minimize the multi-view reprojection error.

2.4. Datasets

Publicly available datasets are essential for comparing different algorithms. To evaluate existing pose estimation algorithms, a number of benchmark datasets have been proposed [183, 182, 61, 16, 62, 159, 136], see Table 2.1. Each dataset is described by the number of objects, the total number of images, the device, used to record the dataset, how the movement was provided, and the topic on which the dataset is focused. In [61], a dataset of 18,000 images with 15 textureless 3D objects was constructed, which was later extended for multi-instance 3D object detection and pose estimation [159]. The Occlusion LINEMOD dataset, proposed in [16] shares the same images as the LINEMOD dataset [61], but annotates 8 objects in a single video that are heavily occluded by other objects. For the above-mentioned datasets, both color and depth images were recorded using handheld Kinect v1 cameras. The target objects are attached to a planar board surrounded by fiducial markers [52], which provide the corresponding poses of the observed objects. Since markers cannot be accurately localized in blurry images, the recorded targets need to remain static in front of the camera, and thus these datasets do not contain distortions that are crucial for evaluating pose tracking performance in real-world scenarios.

Different from using fiducial markers, the ground truth object poses in the dataset [136] are manually labeled and less accurate. Even though the poses are further refined using the Iterative Closest Point (ICP) method, the estimates remain inaccurate due to noisy depth measurements. Methods proposed in this thesis were evaluated on sequences of real images from two challenging, freely available benchmark datasets for 3D object pose tracking, OPT [182] and YCB-Video [183].

2.4.1. OPT dataset

The OPT dataset [182] consists of RGB-D videos of tracked objects, their true poses, and 3D models. This benchmark dataset contains 690 color and depth videos of objects with varying degrees of texture and geometric complexity totaling over 100,000 frames. The videos were recorded using a Kinect v2 sensor at 1920 x 1080

Table 2.1: Datasets for object pose estimation.

Dataset	no. of objects	no. of frames	camera device	mounting device	focus
YCB-Video [183]	21	133 827	Asus Xtion Pro Live	Handheld	household objects
OPT [182]	6	100 956	Kinect v2	Programmable Robotic Arm	motion patterns
LINEMOD [61]	15	18 000	Kinect v1	Handheld	household objects
Occlusion LINEMOD [16]	8	1 214	Kinect v1	Handheld	occluded objects
T-Less [62]	30	48 000	Kinect v2, Primesense Carmine, Canon IXUS	Turntable	industry-relevant objects
Tejani et al. [159]	6	5 229	Kinect v1	Handheld	multi-instance object detection
Rutgers APC [136]	24	10 000	Kinect v1	Programmable Robotic Arm	warehouse pick-and-place

resolution mounted on a programmable KUKA KR 16-2 CR robot arm under various lighting conditions, different motion patterns, and speeds. The ground truth poses are computed using a designed checkerboard and checkerbox for 2D and 3D objects. The OPT dataset is composed of six 3D objects of various geometric complexity: Soda, Chest, Ironman, House, Bike, and Jet, see Figure 2.1. Each 3D object is generated by a 3D printer with a resolution 300×450 dpi and 0.1 mm layer thickness. The length, width, and height of 3D objects are in the ranges of (57.0, 103.6), (57.0, 103.6), and (23.6, 109.5), respectively in mm.

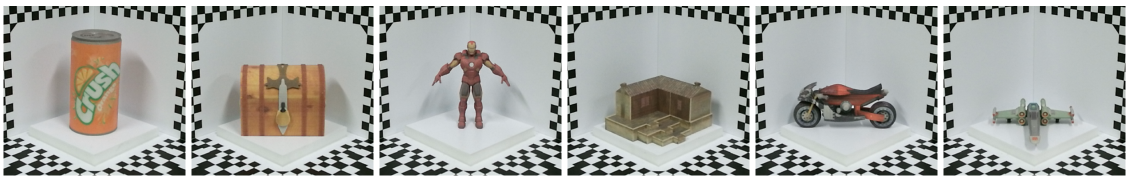


Figure 2.1: 3D objects with simple (Soda, Chest), normal (Ironman, House), and complex (Bike, Jet) geometry.

For each object, there are 92 test videos of varying durations. The test scenarios include translation, zoom, in-plane rotation, out-of-plane rotation, flashing light, moving light, and FreeMotion. The objects move at five different speeds in translation (forward and backward) as well as in in-plane and out-of-plane rotations making the videos more representative of real-world scenarios with different image distortions (e.g., motion blurs). The most natural and demanding evaluation is the FreeMotion scenario, where the object exhibits unconstrained motion in various directions. This test comprises four video sequences for each object, with a total of 323 frames each, displaying the object's movement from different points, including front, back, left, and right views.

2.4.2. YCB-Video dataset

The YCB Video dataset [183] is one of the largest and most challenging datasets for object pose estimation. It provides accurate 6D poses of 21 objects, see Figure 2.2, from the YCB dataset captured in 92 RGBD videos with 133,827 frames at a resolution of 640 x 480. The videos are annotated with 6D poses and segmentation masks. This dataset is challenging as its objects display different symmetries and are arranged in various poses and spatial configurations, leading to severe occlusions between them. The objects vary in size with diameters from 10 cm to 40 cm. For each object, a 3D model is provided. Additionally, 80000 synthetic images were generated for training by randomly placing objects in various scenes. The dataset occupies approximately 265 GB of storage.

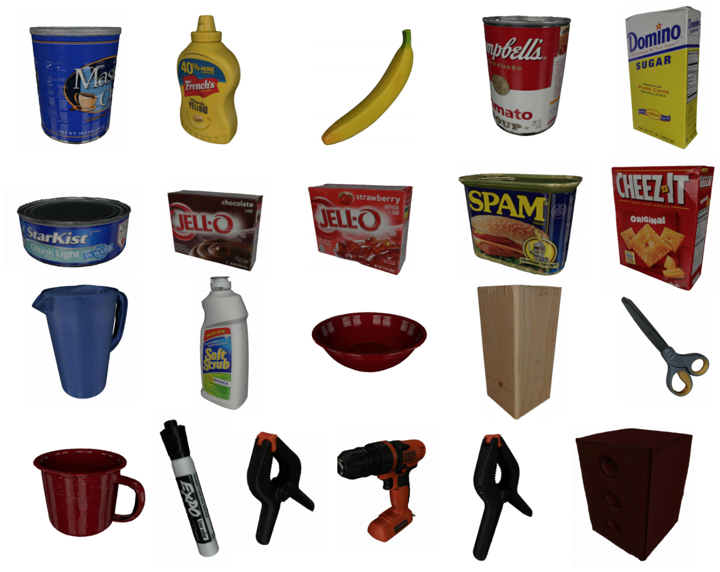


Figure 2.2: Objects of the YCB-Video dataset. The 21 reconstructed object models of the YCB-Video dataset.

2.4.3. Custom dataset

This dataset was introduced in [94] to extend existing benchmarks by incorporating scenarios not included in freely available datasets. The accessible benchmarks often focus on one particular aspect of pose estimation or pose tracking. This dataset is unique, due to the degree of movement in the proposed scenarios, in which objects make full rotation during their movement from side to side. The most popular datasets lack this type of transformation.

The dataset comprises six objects: a drill, multimeter, electrical extension cord, duck, frog, and piggy. Notably, four of these objects (extension cord, duck, frog, and piggy) exhibit minimal or no texture. Blender (software version 2.81), a widely used computer-aided design software (Blender Online Community, 2020), was used to generate 3D models for each object. The diameters of the objects vary significantly, spanning from 103 mm to 228 mm in size. Furthermore, the number of vertices in each model differs substantially, ranging from approximately 119,000 to over 417,000. Specifically, the 3D models have the following specifications: drill (diameter: 228 mm; vertices: 417,777), multimeter (diameter: 138 mm; vertices: 119,273), electrical extension cord (diameter: 103 mm; vertices: 216,233), duck (diameter: 117 mm; vertices: 140,148), frog (diameter: 108 mm; vertices: 135,072), and piggy (diameter: 116 mm; vertices: 132,620). The rendered images depicted in Figure 2.3 were generated based on these 3D models, utilizing their own Python scripts [94] that leveraged Blender's rendering engine capabilities. As demonstrated by the figures, the proposed 3D models enable photorealistic renderings of the objects in various poses.

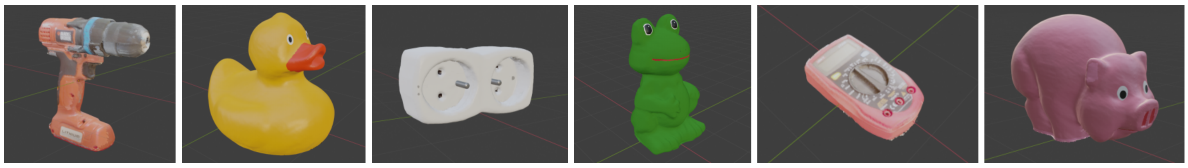


Figure 2.3: Synthetic objects from the dataset.

The synthesized images are employed in neural network training for both object segmentation and detection tasks as well as tracking applications. For object detection and segmentation purposes, binary masks are stored in two formats: PNG images with alpha channels, which contain binary representations of objects, and JSON files adhering to the COCO data format [91]. The PNG images contain binary masks, while the JSON files provide 3D coordinates for eight fiducial points that facilitate accurate tracking. Even when a fiducial point is occluded, it is still generated on the image, ensuring a constant number of points regardless of occlusions or self-occlusions. This enables neural networks like Mask R-CNN [56] to be trained not only for object detection and segmentation but also with minor code modifications for estimating fiducial point positions on RGB images. Additionally, information about 6D object pose is stored in the JSON files. Notably, many popular datasets for six-dimensional object pose estimation often omit information on keypoint locations (fiducial points) and instead provide only 3D coordinates describing the rectangular corners of the object. These corner coordinates are typically derived from camera position or ArUco markers, rather than being explicitly annotated. The tracking dataset comprises images featuring three types of motion: (i) object moves from left to right and then from right to left; (ii) simultaneous movement and rotation between 0° to 180° with subsequent full rotation; and (iii) movement combined with rotation and changes in distance from the camera.

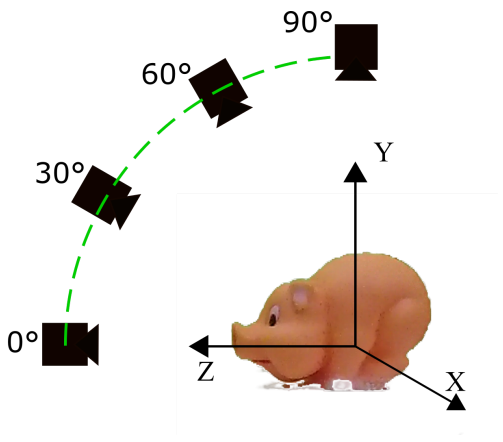


Figure 2.4: Camera angles at which each object has been observed from the custom dataset.

The dataset includes not only synthetically generated data, but also real-world data acquired using an OpenCV-calibrated camera. Ground-truths for the object's poses were determined by measuring its positions with a turntable.

Each object was captured from four different camera angles, as depicted in Figure 2.4. The objects were rotated between 0° and 360°, with images taken every 10 degrees, resulting in 144 images per object for each camera angle.

To train object segmentation models, the objects were recorded from various views by rotating them. The dataset consists of 150 pixel-level, manually annotated images [94] for each object, accompanied by their respective 2D fiducial point coordinates. The number of the projected fiducial points on every image is equal to eight. This subset of real RGB images with ground truth data can be used for training and evaluating neural networks for 6D object pose estimation and tracking. The data is formatted identically to that used in the synthetic subset.

Chapter 3

Pose representation and estimation

This chapter is devoted to object pose representation, parameters, pose estimation, and metrics used in evaluation. It is organized as follows. In Section 3.1 there is a description of the camera model and its parameters. In Section 3.2, the representation of rotation and translation is discussed, followed by a description of quaternions used in object tracking in Section 3.3. Then, in Section 3.4 an overall description of pose estimation is presented. Following this section, an overview of performance metrics used for pose tracking is presented in Section 3.5.

3.1. Camera parameters

The camera is one of the most essential sensors in computer vision [153]. The camera can be understood as a sensor performing a transformation between three-dimensional objects and two-dimensional images. There are various camera models [55] with specific properties that accurately capture the mapping process from 3D space to 2D images.

In modern cameras, light is focused onto the camera image sensor using lenses. In contrast, the human eye employs a curved surface called the retina to capture images, comprising densely packed photoreceptor cells with light-sensitive molecules. The lens model depicts light rays emanating from a point as they pass through the lens, converging at a single point behind it. A crucial factor governing this phenomenon is the focal length of the lens [153], which is defined as the distance behind the lens where rays from an infinitely distant source converge to form a sharp image. Each lens has a specific range within which objects appear "in focus". This property is also linked to the concept of depth [153] of field in photography and computer graphics, which refers to the effective range over which cameras can capture sharp images.

In contrast, a pinhole camera model [55] serves as an idealized representation of the lens where the aperture approaches zero. In a pinhole camera model, light from a point travels along a single straight path onto the image plane, illustrating the fundamental principles of image formation in optical systems. The pinhole camera model describes the mathematical relationship of the projection of points in 3D space onto an image plane.

Figure 3.1 shows a camera with the center of projection O and the principal axis parallel to the Z axis. A 3D point $P = (X, Y, Z)$ is projected on the camera's image plane at coordinate $P_c = (u, v)$. To precisely know the transformation from the real, 3D world into digital images prior knowledge of many of the camera's intrinsic parameters is required. This problem of estimating the intrinsic camera parameters is known as camera calibration [153]. Intrinsic parameters are specific to a camera, so to obtain these parameters it is necessary to take several images of a particular pattern, typically a chessboard, from different positions and extract corners from the images. Then, given the position of the corners of the chessboard in 3D and its position on the image plane, the camera parameters can be calculated:

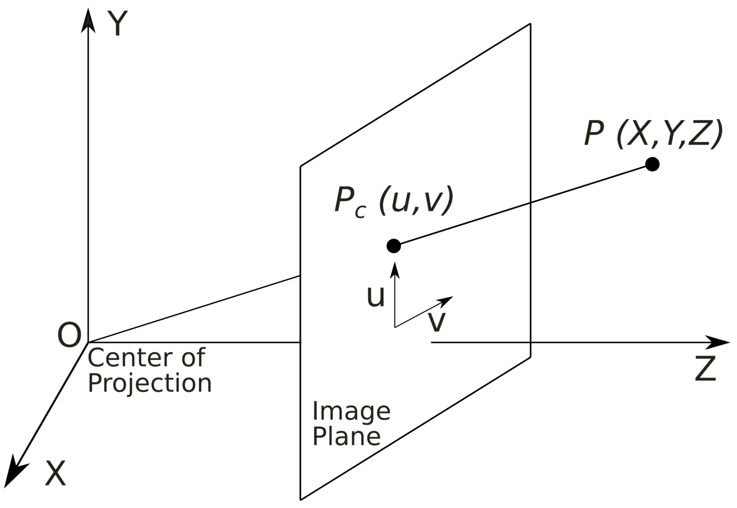


Figure 3.1: Pinhole camera model.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where f_x, f_y are focal lengths and c_x, c_y are optical centers. The position on image plane of the 3D point (X, Y, Z) is given by:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

where u, v are positions on the image plane, r_{11} to r_{33} represent a rotation of the object, and t_x, t_y, t_z represent a translation of the object with respect to the center of projection.

The lens-based model's approximation can lead to various aberrations [151]. One of the most common issues is radial distortion, which affects the image magnification as a function of distance from the optical axis. This phenomenon manifests as pincushion distortion when the magnification increases and barrel distortion when it decreases. Radial distortion causes straight lines to appear curved due to the varying focal lengths of different parts of the lens. Additionally, tangential distortion arises when the image-capturing lens is not precisely aligned with the imaging plane, resulting in certain areas appearing closer than expected. Beyond the standard camera models, there exist a simpler variant known as the weak perspective model [55]. This approach commences by projecting points onto the reference plane through orthogonal projection and subsequently maps them to the image plane using a projective transformation. By further simplifying this process, the orthographic (or affine) projection model is obtained [55]. In this case, the optical center is situated at infinity, leading to projection rays that are perpendicular to the retinal plane. Consequently, this model disregards depth information altogether. The specific cameras used in each of the datasets are listed in Table 2.1.

3.2. Representation of pose

To unambiguously describe the pose of the object, it is required to define a state vector that will represent three-dimensional translation and three-dimensional rotation. The use of Euclidean space is a natural choice for translation coordinates. However, in the case of rotation, the variety of representations makes it necessary to look into the possible options. The rotation can be represented as: Euler angles, rotation vector, rotation matrix, or quaternion [76]. Euler angles are defined as three successive elemental rotations around three independent axes, therefore possible orders of applying rotations lead to different results. This representation can cause a behavior known as gimbal lock, in which two axes become parallel and result in an infinite number of solutions for the same rotation [26].

The axis-angle representation of a rotation parameterizes an object rotation in a three-dimensional Euclidean space by two quantities: a unit vector $\mathbf{n}$ indicating the direction of an axis of rotation, and an angle θ describing the magnitude of the rotation about the axis. Taking the two values $\mathbf{n}$ and θ as is, there can be described an angular displacement in the axis-angle form. Since $\mathbf{n}$ has unit length, it can be multiplied by θ without loss of information, yielding the rotation vector $\mathbf{e} = \theta\mathbf{n}$. The rotation vector is not only more compact than the axis-angle (three numbers instead of four), it elegantly avoids certain singularities and has better interpolation and differentiation properties [11].

A rotation matrix is a transformation matrix that is used to perform a rotation in Euclidean space. It is useful because it allows us to rotate vectors between coordinate spaces and matrix multiplication enables collapsing matrices for nested coordinate spaces into a single matrix [57]. This form of representation takes two to three times as much memory as other techniques.

In various scientific disciplines, quaternions prove to be an ideal tool for combining rotational transformations in an efficient manner, minimizing computational complexity [47, 164, 113]. A significant popularization of quaternions in pure sciences was done by Joachim Lambek [82], who suggested that quaternions can serve as a gateway for pure mathematicians looking to gain insight into specific areas of theoretical physics. In general, quaternions allow reducing the number of operations and parameters compared to vector algebra [184]. Quaternions can be viewed as numbers with one real part and three distinct imaginary parts: $\mathbf{q} = q_w + q_x i + q_y j + q_z k$, where q_w, q_x, q_y, and q_z are real numbers, and i, j, k satisfy $i^2 = j^2 = k^2 = ijk = -1$, and $ij = -ji = k, jk = -kj = i, ki = -ik = j$. This implies that quaternion multiplication is generally not commutative. The quaternion can also be viewed as $\mathbf{q} = w + \mathbf{v}$, where $\mathbf{v} = q_x i + q_y j + q_z k$. Some of the popular systems described in Subsection 2.1.1 use a direct method, which outputs one of the above-presented representations of rotations. The selected representation has an impact on the loss function used to train such a neural network.

3.3. Quaternions for motion tracking

When viewed as a 3D vector, quaternion multiplication can be represented using vector dot product and cross products between vectors [41]. Quaternions with unit magnitude, which restricts the number of degrees of freedom (DoF) to three, correspond to rotations about an arbitrary axis by an angle θ. If the axis passes through the origin of the coordinate system and has a direction given by the vector $\mathbf{n}$ with $|\mathbf{n}| = 1$, this rotation can be parameterized as follows:

$$\mathbf{q} = [q_w\ q_x\ q_y\ q_z] = [cos(\frac{1}{2}\theta)\ \mathbf{n}sin(\frac{1}{2}\theta)] = [w\ \mathbf{v}] \tag{3.3}$$

The set of unit-length quaternions forms a subgroup, denoted by S^3, whose underlying set is identified with the three-dimensional unit sphere $\mathbb{S}^3$ in four-dimensional space $\mathbb{R}^4$. Due to the fact that quaternions $\mathbf{q}$ and **-q** describe the same rotational transformation, only one hemisphere of $\mathbb{S}^3$ is sufficient for consideration. Therefore, we focus on the Northern hemisphere $\mathbb{S}^3_+$, which comprises quaternions satisfying the condition $\mathbf{q} \geq 0$, equivalent to $\theta \in [0, \pi]$.

The formula for quaternion multiplication can be expressed as follows:

$$\mathbf{q}_0 \star \mathbf{q}_1 = [w_0\ \mathbf{v}_0][w_1\ \mathbf{v}_1] = [w_0 w_1 - \mathbf{v}_0 \cdot \mathbf{v}_1\ w_0\mathbf{v}_1 + w_1\mathbf{v}_0 + \mathbf{v}_0 \times \mathbf{v}_1] \tag{3.4}$$

where $\times$ is vector cross product, $\cdot$ denotes vector dot product and $\star$ represents quaternion multiplication.

The logarithm of $\mathbf{q}$ is defined as follows:

$$logMap(\mathbf{q}) = logMap([cos(\alpha)\ \mathbf{n}sin(\alpha)]) \equiv [0\ \alpha\mathbf{n}] \tag{3.5}$$

where $\alpha = \frac{1}{2}\theta$. It should be kept in mind that the $logMap(\mathbf{q})$ is not a unit quaternion. The exponential function can be expressed as:

$$expMap(\mathbf{q}_{exp}) = expMap([0\ \alpha\mathbf{n}]) \equiv [cos(\alpha)\ \mathbf{n}sin(\alpha)] \tag{3.6}$$

where quaternion $\mathbf{q}_{exp}$ is described as $\mathbf{q}_{exp} = [0\ \alpha\mathbf{n}] = [0\ (\alpha x\ \alpha y\ \alpha z)]$ with $\mathbf{n}$ as a unit vector ($\|\mathbf{n}\| = 1$). By definition $expMap(\mathbf{q}_{exp})$ always returns a unit quaternion.

Spherical linear interpolation (SLERP) [147] is used to obtain the object's angular velocity. The SLERP operation is useful because it allows us to smoothly interpolate between two orientations. SLERP avoids all the problems that plagued the interpolation of Euler angles in which the angular velocity cannot be constant during interpolation due to gimbal lock. The angular displacement from $\mathbf{q}_0$ to $\mathbf{q}_1$ is defined in the following manner: $\Delta\mathbf{q} = \mathbf{q}_0\mathbf{q}_1^{-1}$. To take a fraction of this difference quaternion exponentiation was used, which is defined as $\mathbf{q}^t = expMap(t\ logMap\ (\mathbf{q}))$. The fraction of the difference is given by $(\Delta\mathbf{q})^t$. Then, to take the original value and adjust it by this fraction of the difference is the equation for SLERP:

$$SLERP(\mathbf{q}_0, \mathbf{q}_1, t) = (\mathbf{q}_1\mathbf{q}_0^{-1})^t\mathbf{q}_0 \tag{3.7}$$

where t stands for the interpolation parameter and varies from 0 to 1.

3.4. Estimation of object pose

The most common approach to 6D object pose estimation from an RGB is to detect and match local object features [44]. The pose of a calibrated camera can be estimated on the basis of a set of 3D points in the world and their corresponding 2D projections in the image by the Perspective-n-Point (PnP) algorithm [46]. The PnP is often used with a RANSAC [46] framework to support outlier rejection. Over the years, many different versions of the PnP algorithm have been proposed, which demonstrate how relevant this type of solution is. In Gao et al. [48] Ritt-Wu's zero-decomposition method is used to solve the resulting system of equations. AP3P [72] directly determines the camera's attitude by employing the corresponding geometric constraints to formulate a system of trigonometric equations. N 3D points are expressed as a weighted sum of four virtual control points in EPnP [84]. The problem then reduces to estimating the coordinates of these control points in the camera referential, which can be achieved by representing them as a weighted sum of the eigenvectors of a 12×12 matrix. The correct weights are then obtained by solving a limited number of quadratic equations. The DLS [58] computes all pose solutions, as the minima of a nonlinear least-squares cost function, in the general case of $n \geq 3$ points. The UPNP [143] approach can be expressed as the solution of a fixed-size linear set of equations independent of the number of points, similar to the EPnP algorithm for the fully calibrated case. The IPPE [32] is based on locating a point where the transform is best estimated, and using only the local transformation at that point to constrain pose. The SQPNP [161] casts the PnP problem as a quadratically constrained quadratic programming and solves it by conducting local searches in the vicinity of special feasible points from which the global minima are located in a few steps. While PnP algorithms are usually robust when the object is well textured, they can fail when it is featureless or when in the scene there are multiple objects occluding each other. The PnP algorithms are highly dependent on the quality of detected 2D keypoints on RGB images. In most common approaches, detecting corresponding 2D points is carried out by keypoint extractors and descriptors like SIFT [95], which detects, describes, and matches features. SIFT features are invariant to both scale and orientation, and enable efficient matching between different views of objects or persons. As in many other areas, the 6D object pose estimation from an RGB image involves deep neural networks. They are used to directly determine pose or detect characteristic features for keypoint-based methods.

3.5. Metrics

3.5.1. ADD

The Average Distance of Model Points (ADD) score [61] is a commonly employed metric for assessing the quality of 6 DoF object pose estimation. The ADD score is calculated by computing the average Euclidean distance between model vertices transformed using the estimated pose and their corresponding ground-truth pose coordinates. Given $\mathbf{z}$, $\mathbf{R}$ as translation and rotation of ground-truth transformation, and $\hat{\mathbf{z}}$, $\hat{\mathbf{R}}$ corresponding to those of the estimated transformation, it is defined as follows:

$$ADD = \frac{1}{|M|} \sum_{\mathbf{x} \in M} ||(\mathbf{Rx}+\mathbf{z}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{z}})||_2 \tag{3.8}$$

where M denotes a set of 3D model points. The metric evaluates the average disparity between 3D points transformed using an estimated pose and their corresponding ground-truth counterparts. The pose is deemed accurate if the average distance e falls below $k_e d$, where d represents the diameter (i.e., the maximum distance between vertices) of the object M, and k_e is a pre-defined threshold typically set to 10%.

For symmetric objects, the matching between points is ambiguous for some views. The ADD-S metric [61], in which the distance to the closest ground truth point is measured:

$$ADD-S = \frac{1}{|M|} \sum_{\mathbf{x_1} \in M} \min_{\mathbf{x_2} \in M} ||(\mathbf{R}\mathbf{x_1} + \mathbf{z}) - (\hat{\mathbf{R}}\mathbf{x_2} + \hat{\mathbf{z}})||_2 \tag{3.9}$$

can handle symmetric objects. The AUC metric [183] is defined by the area under the accuracy-threshold curve when using the ADD/ADD-S metric. The curve is built by varying the threshold, from 0 to a maximum threshold.

3.5.2. 2D-projection error

The 2D-projection error (reprojection error) measures the average distance between the 2D projections in the image space. The error is calculated on the basis of the following equation:

$$2D-proj = \frac{1}{|M|} \sum_{\mathbf{x} \in M} ||\mathbf{K}(\mathbf{R}\mathbf{x} + \mathbf{z}) - \mathbf{K}(\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{z}})||_2 \tag{3.10}$$

where $\mathbf{K}$ stands for the matrix of intrinsic parameters of the camera, whereas $\mathbf{K}(\mathbf{R}\mathbf{x} + \mathbf{z})$ transforms a 3D point according to the SE(3) transformation and then projects such transformed 3D point onto the image plane. The object pose is considered correct if the average 2D-proj distance is less than 5 pixels [17].

3.5.3. Rotation error

The rotation error can be defined on the basis of the following formula:

$$err_{rot} = arccos((Tr(\mathbf{R}\hat{\mathbf{R}}^{-1}) - 1)/2) \tag{3.11}$$

where Tr stands for the matrix trace, $\mathbf{R}$ and $\hat{\mathbf{R}}$ denote rotation matrices corresponding to ground-truth and estimated pose, respectively.

Chapter 4

Proposed Y-Net

Deep learning-based methods demonstrated high potential in detecting object keypoints [63]. However, most of these methods have been designed for detecting keypoints in single images, and little work has been done toward detecting keypoints in symmetrical objects [44]. Due to the similarity of object elements or views, it can be observed that without information about object pose from the previous frame, the keypoints can be mistakenly estimated, particularly for symmetrical objects. This motivates the introduction of a new attention mechanism to make object keypoints estimation more robust. The first contribution is a quaternion branch, which can be added to any encoder-decoder neural network. This means that a modified neural network will have two inputs, where alongside the standard RGB input, there is a second input for quaternion, which represents the object rotation, for instance, initial object rotation or rotation determined in the former frame. The proposed neural network with quaternion branch was introduced in [100, 101, 103].

Despite recent advancements in computer vision, there is still a significant gap in the utilization of geometric reasoning to leverage object shapes and keypoints for accurate object pose estimation tasks. Object occlusions can lead to a drop in the accuracy of keypoint-based methods for pose estimation [44]. The second contribution in this chapter is to add weights of boundary and keypoints components to the objective function in the pose refinement phase in order to better cope with occluded objects. During each iteration, the boundaries and shapes of the 3D objects are determined through geometric reasoning based on projections of the 3D model onto the image plane and object segmentations. The initial object shapes are extracted using a pre-trained neural network, while the keypoints are derived from the proposed network. Unlike previous methods, the proposed voting scheme is object boundary-based. In [103], the geometric reasoning on object shape was proposed.

This chapter introduces an innovative approach to object pose tracking, which utilizes a multi-branch neural network (Figure 4.1) that uses both a current RGB image (t Image on Figure 4.1) and a quaternion representing the previous frame's pose ($t - 1$ Quaternion on Figure 4.1) as inputs. As can be seen in the aforementioned figure, in the pose tracking phase, the network generates blobs containing fiducial keypoints for the object (Keypoints on Figure 4.1).

In the pose refinement phase, given an initial pose guess, a 3D object boundary is then segmented and projected onto the 2D image plane (Boundary seg. on Figure 4.1). Subsequently, a pre-trained shape network determines the object's shape, which is then compared with the segmented shape to produce confidence weights for the estimated 3D keypoints (Voting on Figure 4.1). The final pose is refined through an optimization-based process, where the initial estimate is computed using the object keypoints and a PnP algorithm. The objective function value is calculated by matching the coarse 3D geometric model with the keypoints predicted by the proposed neural network and projecting the 3D shape onto the estimated object shape.

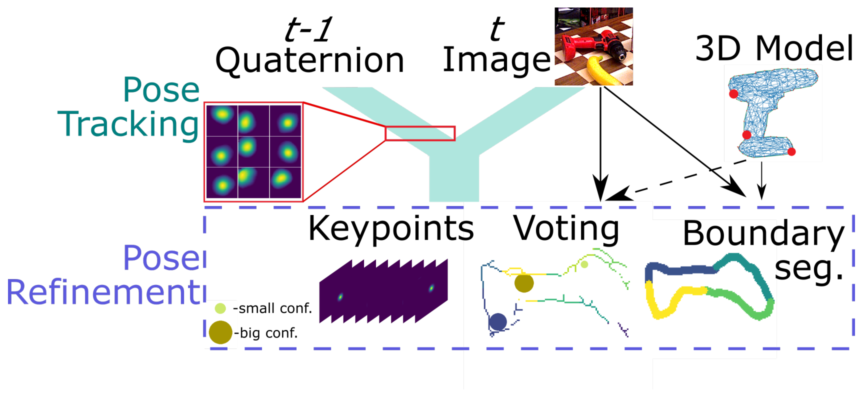


Figure 4.1: Shape enhanced keypoints learning with geometric prior.

This chapter is organized as follows. First, the 2D locations of the keypoints is determined by Y-Net, which is described in Section 4.1. Then, in Section 4.2, the details of pose refinement are presented. Experimental results are presented and discussed in Section 4.4. Section 4.5 provides a short summary of the method.

In this chapter, the following contributions to the object pose tracking are presented:

- rotation guided Y-Net, which uses a two-input architecture to encode data from previous and current frame, and encodes it by heatmaps
- a method for detection of invisible or occluded keypoints using labeled segments of the object shape
- a method permitting estimation of potential occlusion of an object by values of the distance transform on projected boundary segments

4.1. Y-Net architecture

The purpose of the neural network is to estimate 2D locations of fiducial points, which correspond to known locations on the object allowing the estimation of the object pose by PnP or optimization-based algorithms. A new Y-Net neural network architecture for 6D object pose tracking in RGB image sequences is proposed. It is called Y-Net since it has two inputs as shown in Figure 4.2:

- RGB image as the first input
- a quaternion representing the object's rotation at time $t-1$ as the second input

and delivers nine channel output map. Each channel is a gray mask with active pixels forming heatmaps, whose centers represent keypoint locations. Quaternion-based information for the second input results in attention blobs, which are learned in the network branch. Such attention blobs are then forwarded to a bottleneck layer, which is common for all network branches. The goal of the quaternion branch is to provide the neural network with information from the previous frame to focus on a more probable position of keypoints.

The proposed neural network is based on an encoder-decoder architecture [139] with a skip connection between a layer in the encoder path and its corresponding layer in the decoder path. Such skip connections have been introduced in U-Net architecture [139] which is widely used in medical image segmentation, in order to transfer

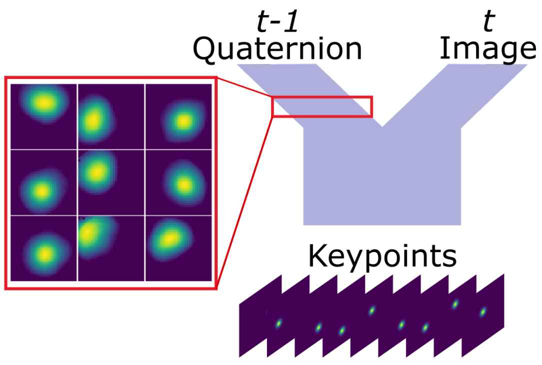


Figure 4.2: Neural model for quaternion driven object pose tracking.

high-level information from layers containing more detailed visual information. Figure 4.3 depicts the architecture of the proposed neural network. It is a Y-like neural network with two inputs and a single output. Let $C(k; n; s)$ denote a convolutional layer with a kernel of size $k \times k$, with n filters and a stride of s, whereas $CT(k; n; s)$ denotes transposed convolution (also known as deconvolution). Let BN stands for batch normalization and FC for fully connected layer. The input image path has the following architecture: the input image is fed to a pyramid pooling module [190] to handle multiple scales, which is followed by $C(3; 64; 2)$. The next two ResNet-like blocks consist of two parallel branches: first, $C(1; 64; 1)$, BN, $C(3; 256; 1)$, BN, $C(1; 64; 1)$, BN, and second $C(1; 64; 1)$, BN. Then tensors from these branches are added in the next layer. These blocks are followed by $C(3; 128; 2)$. The input quaternion path has the following architecture: The input is connected to $FC(256)$ which is followed by BN. Then, there is second $FC(256)$ and BN, followed by $FC(9216)$ and $C(1; 128; 1)$. Then, the two tensors $(32, 32, 128)$ from the image and quaternion branch are added, it is represented by a blue box on Figure 4.3, and then followed by $C(32; 256; 1)$. The output is fed to $C(1; 128; 1)$ and $CT(2; 128; 2)$. Then, there is a concatenation for a shortcut connection omitting the bottleneck layer. The output ends with $CT(2; 9; 2)$, which represents nine heatmaps in separate channels.

The neural network operates on RGB images of size 128×128 and delivers a single Gaussian blob on every of nine images of size 128×128. This resolution was chosen as best in terms of the size of a neural network, accuracy, and low resolution of YCB-Video dataset images (640×480), Section 2.4.2. The loss function calculates the MSE (Mean Square Error). Such a loss permits regression of fiducial keypoint locations. It is worth noting that owing to using separable convolutions [29] the number of trainable parameters has been reduced from $34,7 \times 10^6$ to 3×10^6. Thanks to using 1×1 convolutions to reduce the number of channels in the ResNet-like block the number of trainable parameters has been further reduced by almost one hundred thousand.

This end-to-end neural network for object pose tracking has been trained in 400 epochs with batch size set to 8, using RMSprop with $lr = 0.001$. A network for tracking objects from the OPT dataset has been trained on 2500 images, whereas the network for tracking objects from the YCB-Video dataset has been trained on 80 video sequences prepared for training, and an additional 80 000 frames of synthetic data.

Below, in Figure 4.4, a case study using a Y-Net trained in advance on data from a real experiment is presented. The heatmap in the first row is an output image that has been obtained when an empty image has been fed to the first input, whereas a quaternion determined in the previous frame has been fed to the second input. As can be

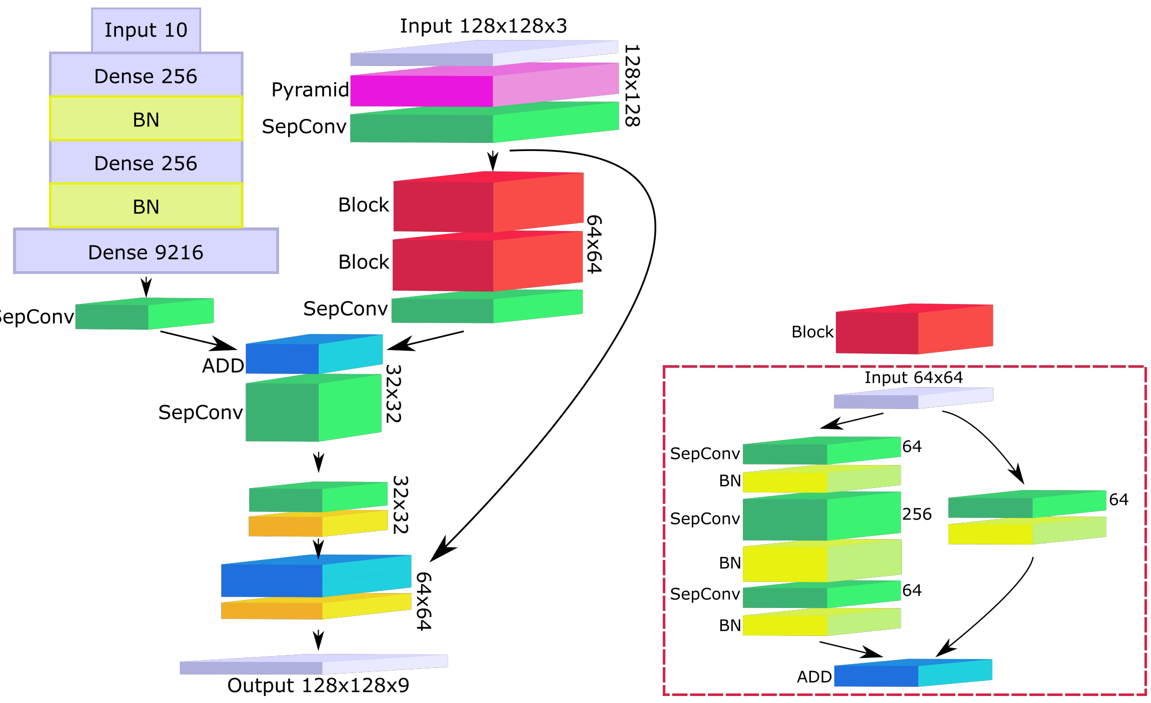


Figure 4.3: Architecture of proposed neural network for object pose estimation.

observed, a raw location of the keypoint location is well represented in the output in this case. The heatmap in the last row is an image output that has been obtained when an input image has been fed to the first input, whereas only zeros has been fed to the second input. There are two blobs at the output, representing two similar points on the drill. If the input image and quaternion representing the object pose in the previous frame are fed to the Y-Net then the maximum value is located in a blob corresponding to the considered keypoint, see the output map depicted in the middle row.

4.2. Geometric reasoning on keypoints and object shape

Estimation of a calibrated camera pose given 2D to 3D point correspondences and camera intrinsics is a standard computer vision problem, e.g. solved by a PnP algorithm [84], see Section 3.4. Non-linear optimization-based approaches permit pose refinement by minimizing the re-projection error between 3D points on the model corresponding to the detected feature points on the image. Although landmark regression occurs independently of object shape consideration, current two-stage methods have inherent limitations. Specifically, networks predict landmarks in isolation, and consistency is only ensured by the pose solver, which is external to the landmark regressor. To overcome these shortcomings, a novel approach is introduced that incorporates geometric reasoning for keypoints and object shape.

The keypoints acquired from Y-Net are fed to a PnP that determines the initial object pose, then utilized as an initial pose guess for the Levenberg-Marquardt (LM) [86][109], which iteratively refines the object pose using keypoints and information about object shape, see Figure 4.5. The current pose estimation yields a segmentation of the 3D object into k parts, which are then projected onto the 2D image plane. Each segment is assigned a corresponding k-th label (upper row). The shape of the object is calculated using a pre-trained neural network. The labeled

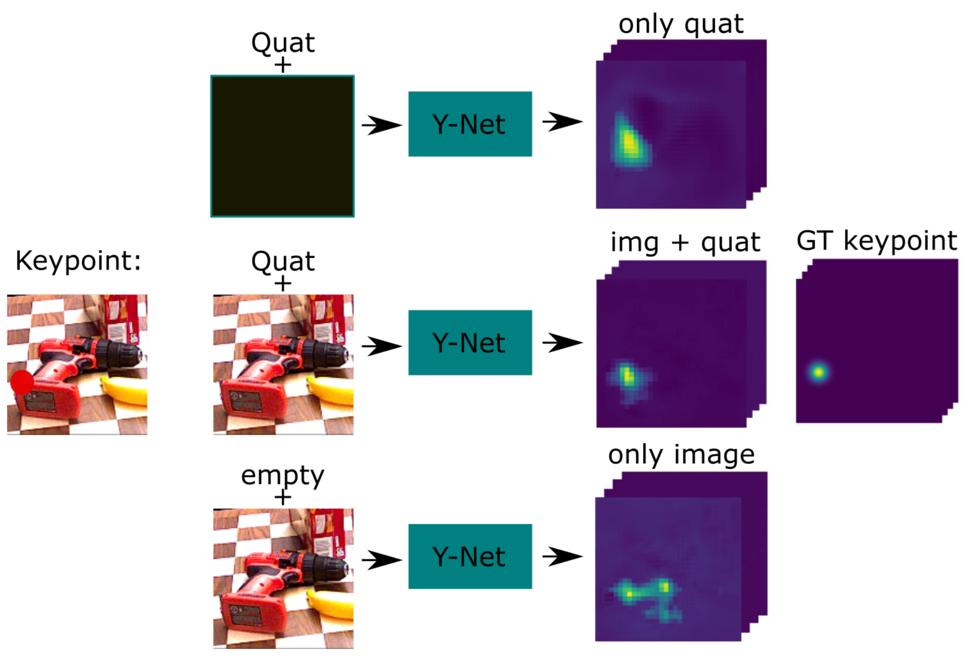


Figure 4.4: Pose proposal guided feature learning.

segments serve as masks to identify pixels belonging to the object's shape. Non-zero shape pixels vote for the closest keypoint, and the weights of these keypoints are computed based on these votes (bottom row). Meanwhile, the distance transform values on the projected boundary segments are utilized to determine the weights of the k boundary fragments.

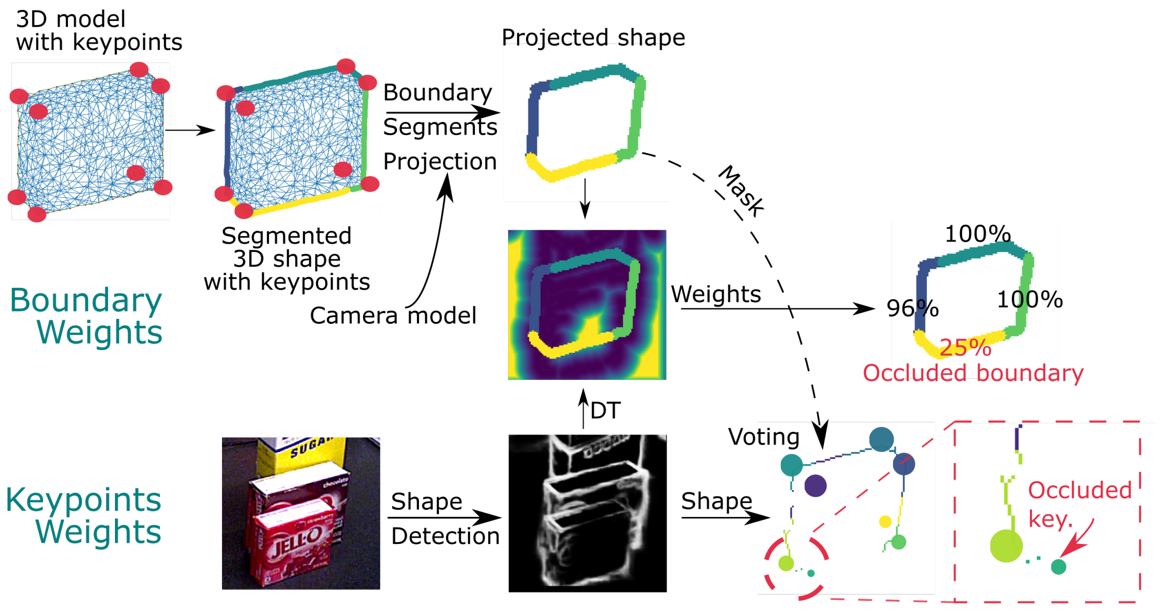


Figure 4.5: Calculation of objective function for object pose refinement.

The 3D model M describing the object can be defined as (V_M, F_M), where V_M is a finite set of model vertices and F_M is the set of triplets of vertices defining model faces. Those sets of model vertices from F_M are rotated using a quaternion from actual pose $\mathbf{x} = [\mathbf{q}\ \mathbf{z}]$. Then, the boundary detection is performed by choosing vertice with minimal value on the X-axis, which will be a starting point, and ignoring the Z-axis on all vertices. From this vertice, we follow the first edge, which is chosen from all edges for this node sorted clockwise, to the next vertice. In the case of intersections of two edges, a new node at the intersection is created and follows a new edge. It is repeated until vertices are found that already exist in our collection. The boundary of the model is represented as an ordered set of points $\mathbf{y} = [x, y, z]$: $\mathbf{y}^n_{Model\_Bound}$. Those sets of model vertices from the boundary are projected onto the image plane, using a matrix of intrinsic parameters of the camera $\mathbf{K}$ and pose $\mathbf{x}$:

$$\mathbf{y}^n_{Bound.} = \mathbf{K}(\mathbf{q}[0\ \mathbf{y}^n_{Model\_Bound}]\mathbf{q}^{-1} + \mathbf{z}) \tag{4.1}$$

The transition of $\mathbf{y}^n_{Bound}$ from resulted three element vector $\mathbf{y} = [x, y, z]$ into $\mathbf{p} = [x, y]$ is done by dividing all parts by z element: $[x/z, y/z, 1]$ and then cutting out the last part. Starting from a node with the minimum value on the X-axis, the boundary is divided into equal segments, see the first row in Figure 4.5. The boundary of projected points from the model and edges created between them, with use of OpenCV functions, will be defined as E and $\mathbf{e}$ will be a segment from this contour.

For each segment, the weight is acquired through an analysis of a distance transform. At the positions of pixels from the segment, the maximal values of the signed distance transform are detected. In cases of high values, a probable occlusion of the object is suspected and therefore the weight of the occluded segment is reduced proportionally to the percentage of exceeded values, see second row in Figure 4.5. $Edge_e$ stands for the weight of $\mathbf{e}$ segment of the boundary.

The weights of keypoints, representing the prediction of probable occlusion, are calculated on the basis of pixels from shape detection. Firstly, the acquired shapes from the neural network are filtered by a mask created by thickened edges from the projected boundary and the thickness of the remaining shapes is reduced to one. Then, each pixel from the shapes votes for the nearest detected keypoint. In the case of occlusion, the number of voting pixels in the occluded part is in most cases diminished, due to the vanished shape, so the number of votes for keypoints in the closest proximity to the occlusion is smaller, see last row in Figure 4.5. The weight of n keypoint, calculated on the basis of votes, will be defined as $Voting_n$.

The projection of keypoints of the 3D model is determined using the following equation:

$$\mathbf{y}^n_{Pose} = \mathbf{K}(\mathbf{q}[0\ \mathbf{y}^n_{Model}]\mathbf{q}^{-1} + \mathbf{z}) \tag{4.2}$$

where $\mathbf{K}$ stands for the matrix of intrinsic parameters of the camera, $\mathbf{q}$ is quaternion and $\mathbf{z}$ is translation acquired from $\mathbf{x} = [\mathbf{q}\ \mathbf{z}]$, and $\mathbf{y}^n_{Model}$ is a keypoint from the 3D model. The transition of $\mathbf{y}^n_{Pose}$ from resulted three element vector $\mathbf{y} = [x, y, z]$ into $\mathbf{p} = [x, y]$ is done the same way as previously, by dividing all parts by z: $[x/z, y/z, 1]$ and then omitting out the last part.

The objective function assumes the following form:

$$L = \min_p((1 - \gamma)L_{key}(p) + \gamma L_{Bound}(p)) \tag{4.3}$$

$$L_{key}(p) = \sum_{n=1}^{9} Voting_n^p * ||\mathbf{p}_{Pose}^n - \mathbf{p}_{Net}^n||_2^2 \tag{4.4}$$

$$L_{Bound}(p) = \frac{1}{|E|} \sum_{\mathbf{e} \in E} Edge_e^p * D(\mathbf{e})^2 \tag{4.5}$$

where $\mathbf{p}_{Net}^n$ 2D position of n keypoint detected by the Y-Net and rescaled to a reference plane, $\mathbf{p}_{Pose}^n$ is a projected keypoint of the 3D model in the actual pose, rescaled to the reference plane, whereas E is a set of pixels representing the boundary of object of actual pose and rescaled to the reference plane, $\mathbf{e}$ is segment of the boundary, $D(\mathbf{e})$ is a function which returns a value of the distance transform [24] for pixels in $\mathbf{e}$, and γ is a factor that has been determined experimentally. The size of the reference plane is calculated on the basis of the distance acquired from initial pose from the PnP algorithm.

4.3. Proposed method

Given an RGB image acquired by a calibrated camera and the object rotation determined in the previous frame, the goal is to estimate the 3D object pose. The 2D location of the keypoints is determined by Y-Net and the object shape is calculated using a pre-trained neural network, which operates on RGB images. The initial object pose is determined by the PnP, while the final pose is determined by the Levenberg-Marquardt (LM) [86] [109] algorithm, see Figure 4.6.

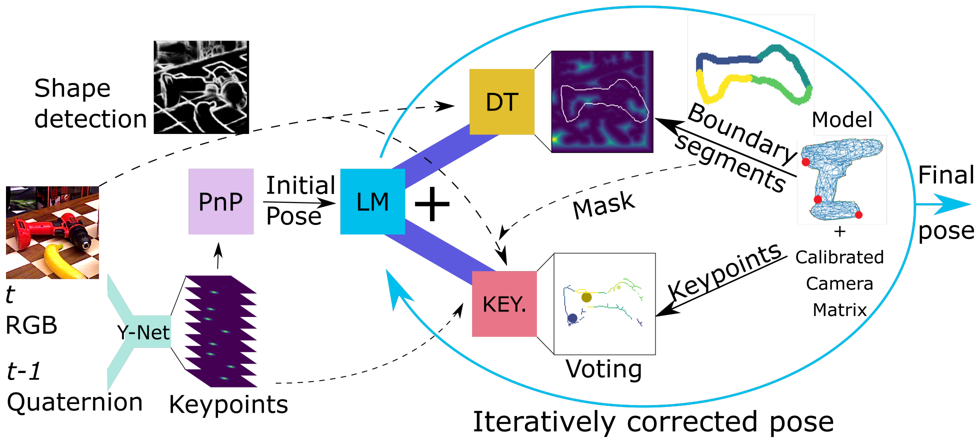


Figure 4.6: Visualization of data flow in the proposed approach to 6D object pose tracking on sequences of RGB images.

The objective function of LM contains two components: the object shape component (DT) and the keypoint component (KEY. on Figure 4.6). The pose estimation model at this stage utilizes the object's distinct topological information, i.e. sparse 3D keypoints (Model on Figure 4.6) that are projected onto a 2D image plane (Keypoints on Figure 4.6), distance transform representing object shape and shape-based voting for keypoint confidences (Voting on Figure 4.6). The objective function value is computed by aligning a coarse 3D geometric model with keypoints predicted by the proposed neural network and projecting it onto the object's actual 3D shape. Given an initial pose estimate, the 3D object shape is reconstructed and then divided into k segments (Boundary segmentation on Figure 4.6). The projected 2D image plane is obtained by mapping the 3D keypoints and shape segments using the camera matrix. The distance transform values along the boundary of the projected segments are used to determine the weights for each shape component. Meanwhile, the labeled shape segments are employed to compute the weights of the corresponding keypoints. The shape segments serve as a mask to identify object pixels, with non-zero pixels voting for their closest keypoint. The weights calculated in such a way are used for weight-matching results between the projected 3D points and keypoint heatmaps. In this way, the occluded parts of the object can be determined, and the effects of occlusions can be minimized. The object rotation determined at this stage is then fed to the next frame to one of the inputs of neural network responsible for the regression of object keypoints.

The system has been implemented in Python using the Keras API. The quaternion branch was first trained for 25 epochs, with batch size set to 16 using RMSprop optimizer, with lr set to 0.001. It has been trained on 40000 positions of nine keypoints and tested on 5000 positions. The keypoint positions were generated on the basis of projections of 3D keypoints of the objects. They were generated with the pose in the range $[-180°, 180°]$ with 10° step on every axis. The translation was sampled from the normal distribution in the interval $[-4$ cm, 4 cm$]$ for the X and Y axes. Changes in translation on the Z-axis were sampled from the normal distribution in the interval $[-10$ cm, 10 cm$]$. The parameters were chosen experimentally so the keypoins were not rendered outside of the images. The whole neural network has been trained for 400 epochs, batch size set to four, lr equal to 0.001, MSE loss function, using RMSprop. For $eps = 0.0001$ the average number of evaluations of the objective function by the LM is about eleven. The maximum number of iterations was set to twenty. The discussed parameters were determined experimentally.

4.4. Experiments and results

The proposed method was evaluated on sequences of real images from two challenging, freely available benchmark datasets for 3D object pose tracking, OPT discussed in Subsection 2.4.1 and YCB-Video discussed in Subsection 2.4.2. Both datasets were described in Section 2.4. In Subsection 4.4.1, experiments demonstrate the usability of the Y-Net with keypoints and shape pose refinement. Geometric reasoning on keypoints and object shape are presented in Subsection 4.4.2.

4.4.1. Experimental results on OPT dataset

In this section, experimental results that were achieved on the OPT dataset are presented. Due to the lack of occlusions between objects in the OPT dataset, the presented results were acquired by a simplified version of the proposed method, where the boundary segmentation and keypoints voting were not used.

The evaluation of the OPT dataset began with the comparison of the neural network with and without the quaternion branch. Table 4.1 presents reprojection errors in terms of 5 px scores, see Subsection 3.5.2, that were obtained on the OPT dataset for keypoints estimated by our network. The discussed error score expresses how close the 2D projected vertices are to the ground-truth. As can be observed, the largest reprojection error is for the House object. The errors achieved by the network with no information about object rotation in the previous frame are significantly larger.

The next step in the evaluation process consisted of calculating the ADD scores, see Subsection 3.5.1, that have

Table 4.1: Reprojection error 5 px scores [%] achieved by our algorithm on the OPT dataset.

OPT obj.	House	Iron.	Jet	Bike	Chest	Soda
2D-proj 5 px score (no quat.)	71.0	83.5	77.0	83.3	80.3	81.1
2D-proj 5 px score	74.8	90.2	82.9	90.0	85.9	86.7

been obtained on the OPT dataset for four different views of objects. The results are summarized in Table 4.2. As it can be observed, the ADD scores for the House objects assume high values despite low 5 px scores. The discussed ADD scores were achieved in the FreeMotion scenario.

Table 4.2: ADD scores [%] achieved by the proposed algorithm on the OPT dataset.

ADD	House	Iron.	Jet	Bike	Chest	Soda
beh., ADD 10%	76	74	63	76	76	55
beh., ADD 20%	90	93	81	94	94	72
left, ADD 10%	93	75	73	78	50	49
left, ADD 20%	100	94	95	98	90	83
right, ADD 10%	74	90	86	77	71	47
right, ADD 20%	98	98	94	99	96	100
front, ADD 10%	92	96	72	49	83	89
front, ADD 20%	100	100	91	84	98	97
Avg., ADD 10%	84	84	74	70	70	60
Avg., ADD 20%	97	96	91	94	95	88

Next, ADD scores achieved by three different modules for pose estimation based on keypoints determined by the proposed network were compared. The scores were achieved by (i) the PnP algorithm [84], (ii) LM initialized with a pose determined by the PnP algorithm and a fitness score calculated based on keypoints fitting, and (iii) LM initialized as previously with a fitness function calculated on the basis of keypoints and object shape. Experimental results are shown in Table 4.3. As can be observed, the algorithm using both keypoints and object shape achieves superior results for ADD 10% and ADD 20%.

Table 4.3: ADD scores [%] achieved by the proposed algorithm on the OPT dataset.

Avg. ADD	House	Iron.	Jet	Bike	Chest	Soda	Avg.
PnP, ADD 10%	76	80	65	62	55	53	65
key. LM, ADD 10%	81	81	71	66	59	45	67
key. + DT LM, ADD 10%	**84**	**84**	**74**	**70**	**70**	**60**	**74**
PnP, ADD 20%	**97**	95	88	93	85	84	90
key. LM, ADD 20%	95	95	88	93	86	84	90
key. + DT LM, ADD 20%	<u>97</u>	**96**	**91**	**94**	**95**	**88**	**94**

Table 4.4: AUC scores [%] on OPT dataset compared against results achieved by recent methods.

AUC score	scenario	House	Ironman	Jet	Bike	Chest	Soda	Avg.
PWP3D [131]	all tests	3.58	3.92	5.81	5.36	5.55	5.87	5.02
UDP [17]	all tests	5.97	5.25	2.34	6.10	6.79	8.49	5.82
ElasticFusion [181]	all tests	2.70	1.69	1.86	1.57	1.53	1.90	1.88
Reg. G-N. TPAMI [163]	all tests	10.15	11.99	13.22	11.90	11.76	8.86	11.31
region-based TIP [192]	all tests	13.61	11.21	**15.44**	**12.83**	**12.24**	9.01	12.39
contour energy & keyp. [21]	FreeM.	-	-	-	-	-	-	**13.91**
ph. enh. edge [168]	FreeM.	13.70	10.86	-	-	9.77	-	11.44
Quat-driven [100]	FreeM.	12.52	11.98	12.16	10.31	8.04	8.71	10.62
Proposed method	FreeM.	**13.98**	**13.99**	12.84	12.40	11.77	**10.48**	12.58

Finally, the AUC (Area Under Curve) ADD scores, c.f. Subsection 3.5.1, were calculated in order to compare with state-of-the-art algorithms for pose estimation. Table 4.4 presents the AUC scores achieved on the OPT dataset and compares them against results that were achieved by recent methods. As already mentioned above the FreeMotion scenario is far more challenging in comparison to other scenarios. As can be observed, the region-based TIP [192] that is the region-based algorithm achieves superior results for three objects. The average AUC score is slightly worse than AUC score achieved by the proposed algorithm. The best AUC score is achieved by [21], which is also a region-based algorithm. As mentioned previously, the performance of region-based algorithm degrades in real-scenarios in which, usually, there are strong variations of foreground and background statistics. By including in the objective function a component expressing fitness between object shape and the re-projected shape of the 3D model the AUC score is far better, cf. results in the last two rows.

4.4.2. Experimental results on YCB-Video dataset

In this section, experimental results that were achieved on the YCB-Video dataset are presented. Similarly as in Subsection 4.4.1, the evaluation of the method begins by comparing results achieved by the neural network with and without the quaternion branch. Table 4.5 presents reprojection errors in terms of 5 px scores, see Subsection 3.5.2, that were obtained on the YCB-Video dataset for keypoints estimated by the proposed network. As it can be observed, the use of a geometric prior in the form of object rotation from the previous frame permits achieving far better results in comparison to results achieved by an ordinary network for keypoints regression. For all YCB objects except for two objects the 5 px error scores achieved by the proposed network are far better in comparison to error scores achieved by algorithms discussed in [121], which will be called DHROP (Deep Heatmaps Robust to Partial Occlusions) in this thesis, and [66].

In order to verify the proposed method, the results achieved only by the PnP-based base algorithm were determined. Table 4.6 presents ADD and ADD-S, Subsection 3.5.1, that have been achieved on the YCB-Video dataset by PnP using keypoints determined by the proposed network and the proposed algorithm for 6D object pose tracking. ADD is calculated for non-symmetric objects, whereas ADD-S is determined for symmetric objects, and '*' stands for

Table 4.5: Reprojection errors, 5 px scores [%] achieved by the proposed network, compared with errors obtained by recent algorithms on the YCB-Video dataset.

Object	DHROP [121]	[66]	Y-Net w/o quat	Y-Net w/ quat
002_master_chef_can	29.7	21.0	79.4	**82.4**
003_cracker_box	64.7	12	79.1	**86.9**
004_sugar_box	72.2	56.3	73.0	**77.1**
005_tomato_soup_can	39.8	46.2	54.9	**73.8**
006_mustard_bottle	**87.7**	70.3	75.7	82.3
007_tuna_fish_can	38.9	39.3	67.0	**72.2**
008_pudding_box	78.0	17.3	68.2	**84.8**
009_gelatin_box	**94.8**	83.6	0.1	87.6
010_potted_meat_can	41.2	60.7	87.7	**90.8**
011_banana	10.3	22.4	59.5	**63.0**
019_pitcher_base	5.43	33.5	74.2	**78.8**
021_bleach_cleanser	23.2	43.3	60.9	**66.8**
024_bowl*	26.1	13.3	38.2	**62.7**
025_mug	29.2	38.1	51.1	**66.7**
035_power_drill	69.5	43.3	64.1	**73.0**
036_wood_block*	2.06	2.5	34.9	**51.2**
037_scissors	12.1	8.8	36.4	**45.7**
040_large_marker	1.85	13.6	72.3	**83.0**
051_large_clamp*	24.2	7.6	45.9	**51.6**
052_extra_large_clamp*	1.32	0.6	55.8	**73.1**
061_foam_brick*	75.0	13.5	65.6	**78.2**
Avg.	39.4	30.8	59.2	**72.9**

symmetric objects. In the PnP version of the algorithm, the keypoints were determined by the proposed network using rotation estimates from the previous frame. As can be seen, the results achieved by the proposed algorithm are much better.

Table 4.7 compares AUC ADD scores achieved by the proposed algorithm with AUC ADD scores achieved by recent algorithms. AUC ADD is calculated for non-symmetric objects, whereas AUC ADD-S (c.f. Subsection 3.5.1) is determined for symmetric objects, - denotes unavailable results, and '*' stands for symmetric objects. As can be observed, the proposed algorithm achieves competitive results on the challenging YCB-Video dataset. It achieved better results in comparison to PoseRBPF [36] and a slightly better average AUC ADD score than DeepIM [89], which, similarly to the proposed algorithm was developed for tracking the 6D object pose in sequences of RGB maps. For seven objects the AUC ADD scores achieved by the proposed algorithm were better, whereas for one object the AUC ADD score achieved by DeepIM was superior to the score achieved by the proposed algorithm. For eight objects proposed algorithm achieved second best results. The recently proposed GDR-Net achieved better AUC ADD scores for twelve objects. As can be noticed, the average AUC ADD score for all objects is lower compared to the average AUC ADD score achieved by the proposed algorithm. For several objects, including the pudding box and scissors, the discussed results were considerably worse in comparison to the results achieved by

Table 4.6: ADD scores [%] achieved by the proposed algorithm on the YCB-Video dataset.

Object	PnP	Proposed
002_master_chef_can	82.6	**95.6**
003_cracker_box	77.1	**88.8**
004_sugar_box	77.6	**92.6**
005_tomato_soup_can	45.0	**52.7**
006_mustard_bottle	84.5	**94.0**
007_tuna_fish_can	62.7	**69.2**
008_pudding_box	73.7	**89.1**
009_gelatin_box	90.6	**95.6**
010_potted_meat_can	45.5	**57.5**
011_banana	56.9	**61.5**
019_pitcher_base	91.6	**98.6**
021_bleach_cleanser	78.2	**85.5**
024_bowl*	69.0	**78.3**
025_mug	63.3	**74.5**
035_power_drill	66.7	**82.3**
036_wood_block*	80.5	**85.5**
037_scissors	46.3	**62.0**
040_large_marker	43.1	**51.6**
051_large_clamp*	69.5	**89.8**
052_extra_large_clamp*	66.0	**74.0**
061_foam_brick*	54.3	**59.3**
Avg.	67.8	**78.0**

the proposed algorithm. Most recent methods estimate the object pose from single RGB images without taking advantage of temporal consistency among video frames. Temporal consistency is a crucial factor, and it has been demonstrated that better results can be achieved by incorporating it in object tracking. The proposed pose refinement strategy allows for achieving competitive results.

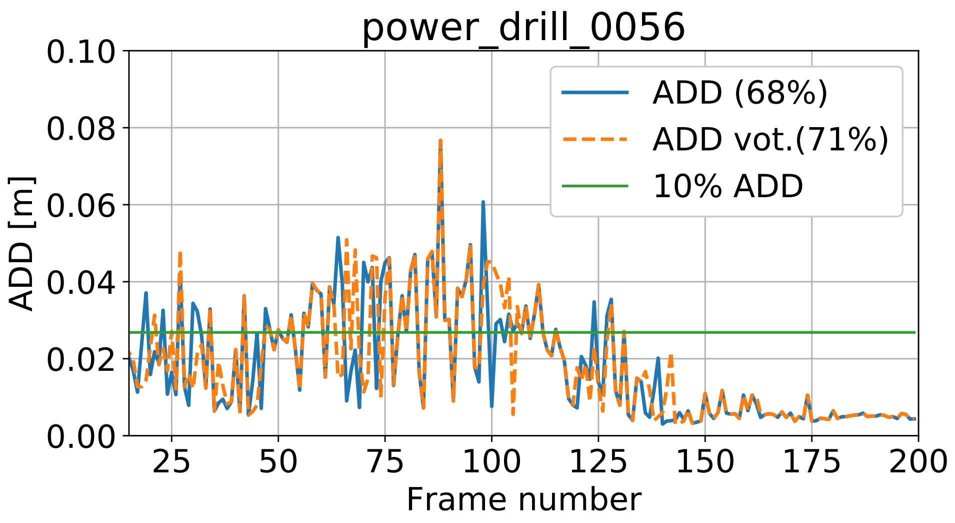
The proposed method was also tested by comparing results for individual frames. Figure 4.7 contains plots of ADD scores vs frame number for the power drill. These sample plots illustrate the ADD scores that have been achieved using voting in object pose refinement (upper plot), using boundary in object pose refinement (middle plot), and with both voting and boundary used in pose refinement (bottom plot). The blue ADD represents the version without voting or boundary. The green 10% ADD represents the maximum value of error that is acceptable as the correct object pose. As can be observed, the proposed pose refinement method achieved an average ADD score equal to 73%, whereas the average ADD scores for voting-based and boundary-based algorithms have been equal to 70% and 71%, respectively. Using voting in object pose refinement (upper plot), using boundary in object pose refinement (middle plot), and with voting and boundary used in pose refinement (bottom plot).

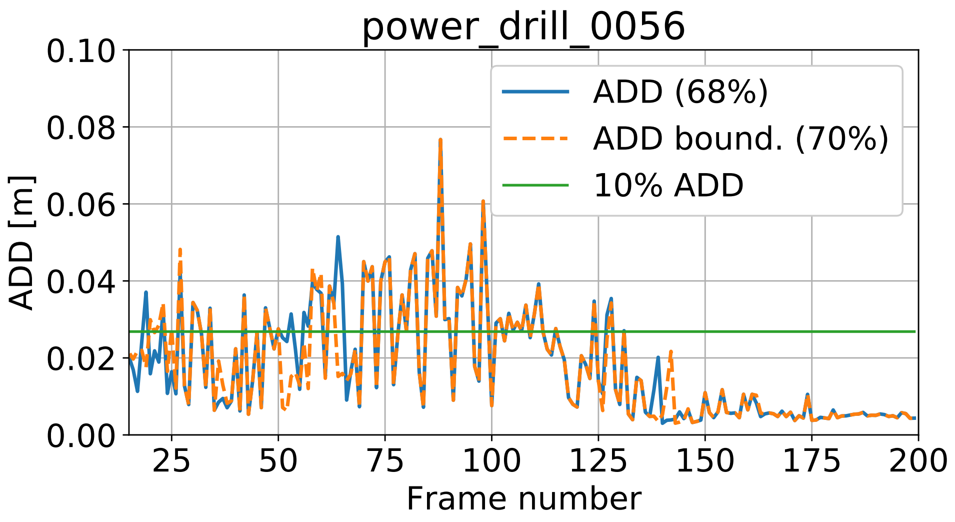
Figure 4.8 presents example qualitative results, which have been achieved by the proposed algorithm on the

Table 4.7: AUC ADD scores [%] (max. th. 10 cm) achieved by proposed algorithm on the YCB-Video dataset.

	Pose recovery on single RGB images					Pose tracking		
Object	PoseCNN [183]	DOPE [166]	DHROP [121]	[187]	GDR-Net [175]	PoseRBPF [36]	DeepIM [89]	Proposed method
002_master_chef_can	50.9	-	81.6	49.9	65.2	63.3	89.0	**90.5**
003_cracker_box	51.7	55.9	83.6	80.5	88.8	77.8	88.5	**89.1**
004_sugar_box	68.6	75.7	82.0	85.5	**95.0**	79.6	94.3	90.0
005_tomato_soup_can	66.0	76.1	79.7	68.5	**91.9**	73.0	89.1	80.0
006_mustard_bottle	79.9	81.9	91.4	87.0	**92.8**	84.7	92.0	92.0
007_tuna_fish_can	70.4	-	49.2	79.3	**94.2**	64.2	92.0	89.2
008_pudding_box	62.9	-	**90.1**	81.8	44.7	64.5	80.1	84.1
009_gelatin_box	75.2	-	93.6	89.4	92.5	83.0	92.0	**94.6**
010_potted_meat_can	59.6	39.4	79.0	59.6	80.2	51.8	78.0	**85.0**
011_banana	72.3	-	51.9	36.5	**85.8**	18.4	81.0	77.2
019_pitcher_base	52.5	-	69.4	78.1	**98.5**	63.7	90.4	92.3
021_bleach_cleanser	50.5	-	76.1	56.7	**84.3**	60.5	81.7	82.5
024_bowl*	69.7	-	76.9	23.5	85.7	85.6	**90.6**	89.2
025_mug	57.7	-	53.7	54.0	**94.0**	77.9	83.2	85.9
035_power_drill	55.1	-	82.7	82.8	**90.1**	71.8	85.4	82.1
036_wood_block*	65.8	-	55.0	29.6	82.5	31.4	75.4	**84.1**
037_scissors	35.8	-	65.9	46.0	49.5	38.7	70.3	**70.5**
040_large_marker	58.0	-	56.4	9.8	76.1	67.1	80.4	**81.8**
051_large_clamp*	49.9	-	67.5	47.4	**89.3**	59.3	84.1	86.9
052_extra_large_clamp*	47.0	-	53.9	47.0	**93.5**	44.3	90.3	90.8
061_foam_brick*	87.8	-	89.0	87.8	**96.9**	92.6	95.5	93.9
Avg.	61.3	65.8	72.8	61.0	84.4	64.4	86.3	**86.3**

YCB-Video dataset. These sample images depict that the proposed algorithm can cope with scene clutter, severe occlusions, reflection, different illumination, and various movements of objects. As shown, large errors can occur for symmetrical objects, see also bowl object (2nd row, 6th from left). Example images demonstrate how proposed algorithm handles occlusions, different arrangement, lighting in ordinary shots for the following objects: 002_master_chef_can, 003_cracker_box, 004_sugar_box, 005_tomato_soup_can, 006_mustard_bottle, 007_tuna_fish_can, 008_pudding_box, 009_gelatin_box, 010_potted_meat_can, 011_banana, 019_pitcher_base, 021_bleach_cleanser, 024_bowl, 025_mug, 035_power_drill, 036_wood_block, 037_scissors, 040_large_marker, 051_large_clamp, 052_extra_large_clamp, and 061_foam_brick. The blue bounding boxes show the ground truth poses, while the green ones correspond to the estimated poses. For better visualization, the regions of interest were cropped.





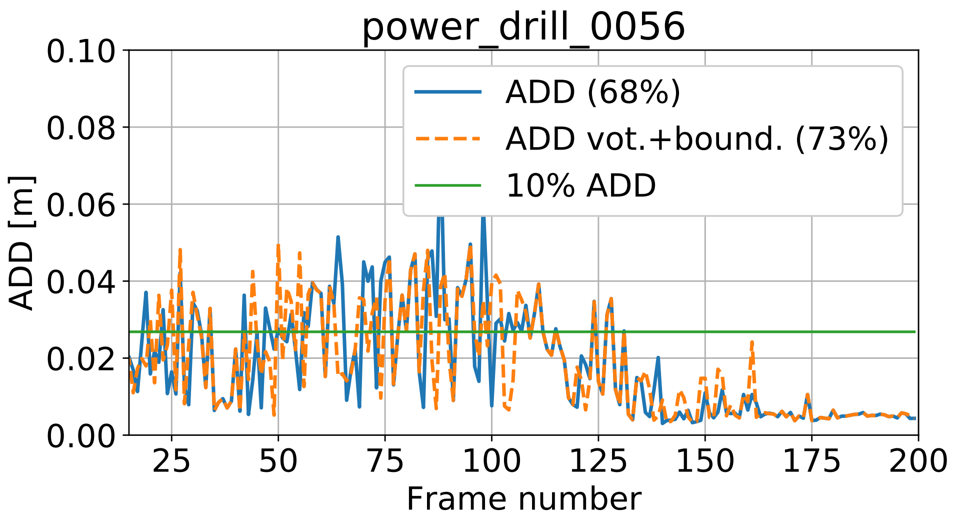


Figure 4.7: ADD scores vs time for power drill object from YCB-Video dataset.

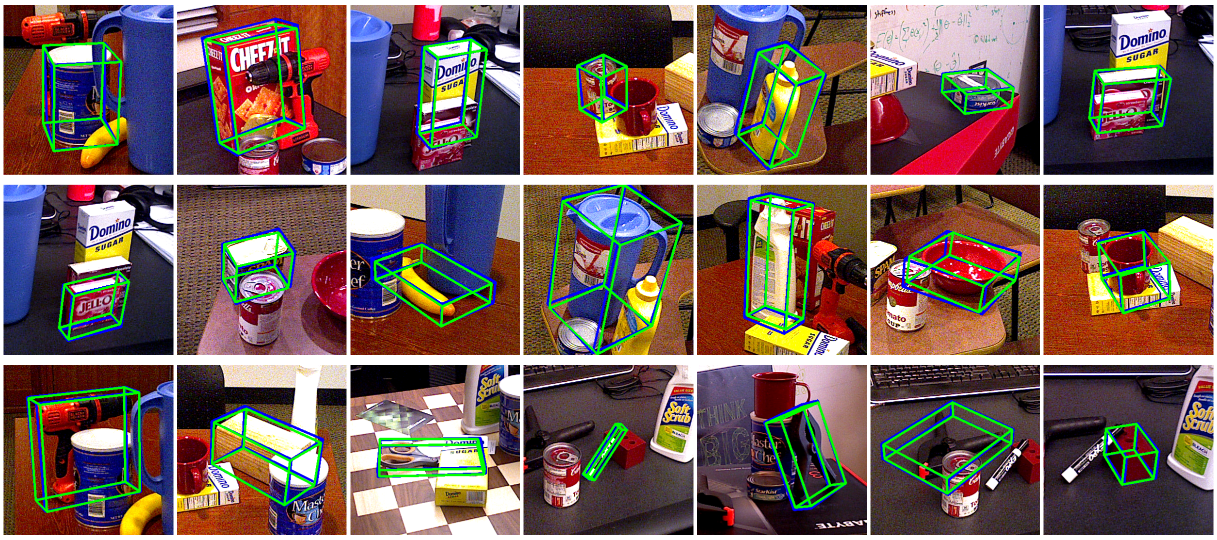


Figure 4.8: Qualitative results on YCB-Video dataset.

4.5. Summary

This chapter was devoted development, evaluation, and comparison with state-of-the-art to provide accurate pose tracking of objects using information from previous frame and geometric reasoning on keypoints and object shape. In order to achieve this goal, a novel algorithm for 6D object pose estimation on RGB images was created. The method was evaluated in several experiments, which included a thorough analysis of the proposed quaternion branch and geometric reasoning on two demanding datasets. The experiments confirmed that the proposed pose refinement algorithm working on the basis of geometry between keypoints and object shape permits improvement of accuracy in 6D object pose estimation. It was shown experimentally that a network that takes the current RGB image on the first input and quaternion representing object rotation in the previous frame in the second input permits achieving better results in comparison to the network operating on RGB images only. It indicates the possibility of using this proposed module in various algorithms both for object pose estimation as well as for object pose tracking. The key idea is to exploit the object rotation and feed it to a two-input neural network.

Chapter 5

Proposed X-Net

After introducing a new branch to the neural network to make object keypoints estimation more robust, as discussed in Chapter 4, this chapter focuses on adding another representation of the tracked object. Next to characteristic points on an object, another typical representation of an object in images is object segmentation. Image segmentation is the process of partitioning a digital image into multiple image segments [146], where pixels with the same label share certain properties. In the considered problem, it is desired to detect pixels belonging to the tracked object. In this chapter, a guided X-Net is proposed, which uses two inputs to encode data from the previous and the current frame, and encodes it by heatmaps with additional object masks, see Figure 5.1. This additional output in the proposed neural network, not only provides extra information about the tracked object but also improves the estimation of object keypoints. The proposed neural network was presented in an article published in [106].

The second part of the chapter presents different methods for pose refinement. In order to refine the determined pose, pose optimization and refinement strategy that apart from keypoints uses also signed distance maps and Hierarchical Optimistic Optimization was introduced. The experimental evaluation is performed on the OPT dataset and YCB-Video dataset.

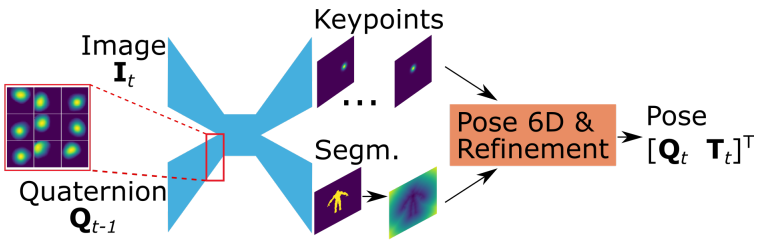


Figure 5.1: Pose guided feature learning for 6D object pose tracking.

This chapter is organized as follows. Section 5.1 discusses a new X-Net neural network, which is an extension of the Y-Net neural network proposed in Section 4.1. In Section 5.2 segmentation-driven pose refinement is described. Section 5.4 includes experimental results, discussion, and comparison with state-of-the-art results. A summary of the proposed method is given in Section 5.5.

In this chapter, the following contributions to the object pose tracking are discussed:

- rotation guided X-Net, which encodes current pose by heatmaps and object masks
- Hierarchical Optimistic Optimization to refine the object pose

5.1. X-Net architecture

Encouraged by the results achieved by Y-Net, described in Chapter 4.1, an expansion to this approach by adding segmentation of the tracked object was desired. A new X-Net neural network architecture for 6D object pose tracking in RGB image sequences was proposed, as an expansion of the Y-Net, see Figure 5.2. In addition to the Y-Net, which operates on two inputs, a second output delivering the segmentation of object was added, which makes this network multi-output. X-Net operates on RGB images of size 128×128 and produces multichannel maps of size 128×128, see Figure 5.2.

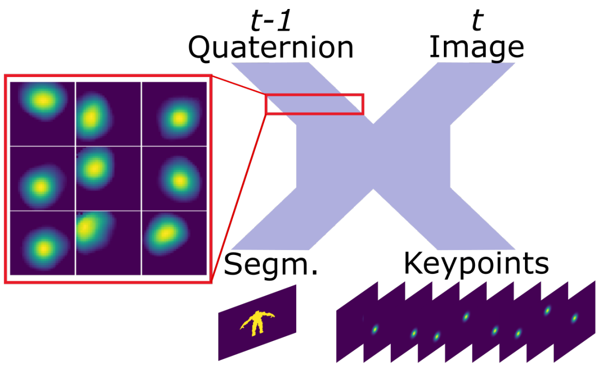


Figure 5.2: Neural model for quaternion driven object pose tracking.

The architecture of the proposed neural network is shown in Figure 5.3. It is an X-like neural network with two inputs and two outputs. The neural network is designed as follows: The input image is fed to a pyramid pooling module [190] to cope with multi-scale, which is followed by a convolution layer with kernel sized 3×3, 64 filters, and a stride of 2. Then, there are two resNet-like blocks composed of two parallel branches. The first branch consists of a $C(1, 64, 1)$, followed by BN and another $C(3, 256, 1)$. This branch continues with BN, $C(1, 64, 1)$, and BN. The second branch is as follows: $C(1, 64, 1)$, finished with BN. Then tensors from these branches are added in the next layer. These blocks are then followed by a $C(3, 128, 2)$. The input quaternion path has the following architecture: The input is connected to a FC layer with 256 neurons, which is followed by BN. Then, there is a second FC layer with 256 neurons and BN. Next is a FC layer with 9216 neurons and a $C(1, 128, 1)$. Then, the two tensors $(32, 32, 128)$ from the image and quaternion branch are added and then followed by a $C(32, 256, 1)$. The output is fed to two similar branches with a $C(1, 128, 1)$, followed by a $CT(2, 128, 2)$. Then, there is a concatenation for a shortcut connection omitting the bottleneck layer. The output branch for segmentation ends with a $CT(2, 1, 2)$, whereas the second output branch representing keypoints ends with a $CT(2, 9, 2)$.

Our experiments demonstrated that the 256 neurons in the FC layers permit achieving the best RMSE (Root Mean Square Error) for keypoints position. For training and testing, synthetic data of keypoints was generated with the pose in the range $[-180°, 180°]$ with $10°$ step on every axis. With translation defined as $\mathbf{z} = [x \ y \ z]$, it was sampled from the normal distribution in the interval $[-4$ cm, 4 cm$]$ for the X and Y axes. Change in translation on the Z-axis was sampled from the normal distribution in the interval $[-10$ cm, 10 cm$]$. The output of the quaternion branch is reshaped to $32 \times 32 \times 9$, so it can define nine heatmaps representing the position of the projected keypoint as Gaussian blobs with $\sigma = 5$. The 3D keypoints were projected to an image with a size adequate to the used camera matrix and then it was rescaled to 32×32. The discussed branch has been trained on 40000 quaternion representations of poses of nine keypoints and tested on 5000 representations. In order to determine the number of neurons in the discussed fully conected layer, the quaternion branch was pretrained on synthetic data and then compared using RMSE on the position of keypoints from heatmaps, which are shown in Figure 5.2, for the following

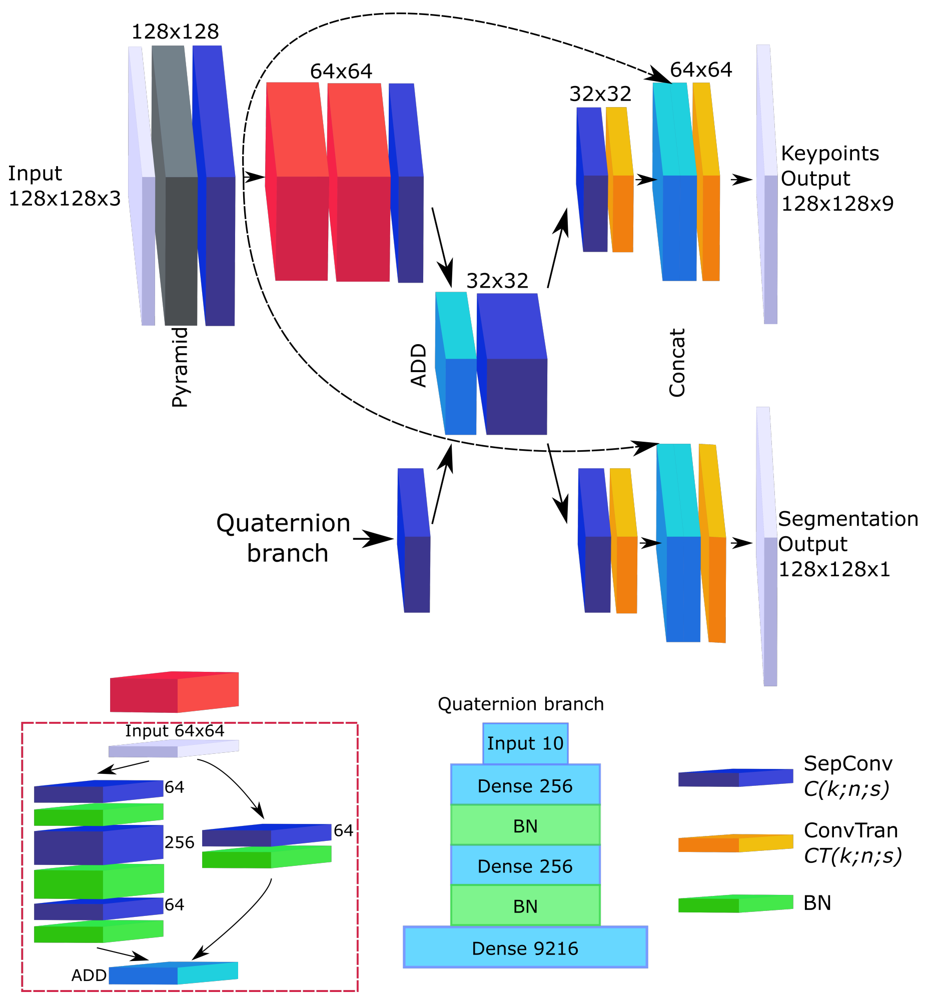


Figure 5.3: Architecture of the proposed X-Net.

number of neurons 64, 128, 256, and 512. Quaternion branch was trained in 25 epochs, batch size set to 16 using RMSprop optimizer with lr set to $1e-3$ and MSE loss function. This pretrained quaternion branch is then used in the training of the whole X-Net.

The loss function has the following form:

$$L = \alpha L_{MSE}(y_k) + \beta(L_{Bin\_cross}(y_s) + (1 - L_{Dice}(y_s))) \tag{5.1}$$

where $L_{MSE}(y_k)$ denotes mean square error on keypoint heatmap, $L_{Bin\_cross}(y_s)$ denotes binary cross-entropy and $L_{Dice}(y_s)$ stands for Dice score. L_{Bin_cross} and L_{Dice} have been determined using predicted segmentation masks and ground-truth object segmentation masks, whereas L_{MSE} has been calculated using predicted heatmap and ground truth map. The neural network was trained in 600 epochs. The α and β values were varied during the training of the network. In the first 50 epochs, they assume values $\alpha = 1$ and $\beta = 2$, then until the 300th epoch both factors have values equal to one and after the 300th epoch, $\alpha = 2$ and $\beta = 1$. The neural network has been trained with maximum epochs set to 600.

This neural network has two inputs, where an input RGB image is fed to the first input, whereas a quaternion representing the object rotation, for instance, initial object rotation or rotation determined in the former frame is fed to the second input. In practice, in tasks such as 3D object pose tracking, apart from the quaternion an object id is fed too in order to choose the type of object to track. Figure 5.4 shows the justification behind using such a mixed-input encoder-decoder. The red point on the most left image in Figure 5.4 represents the wanted keypoint on a symmetrical object. Below is presented a case study using an X-Net trained in advance on data from a real experiment. In the first row, the heatmap is an output image that has been obtained when an empty image has been fed to the first input, whereas a quaternion with object id determined in the previous frame has been fed to the second input. Using only the information about rotation from the previous frame, quaternion represents well a raw location of the keypoint at the output in this case.

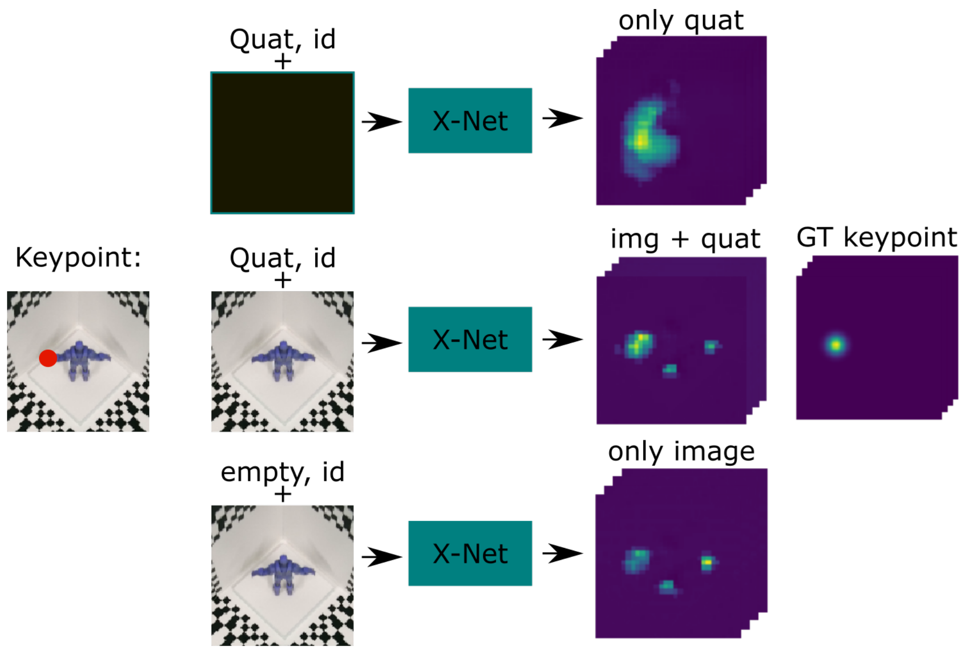


Figure 5.4: Pose proposal guided feature learning.

The heatmap in the last row is an image output that has been obtained when an input image has been fed to the first input, whereas only object id determined in the previous frame has been fed to the second input. As can be observed, there are three blobs at the output, and the maximum value is in the blob corresponding to the mistakenly identified hand. When all of the required inputs are provided, as can be seen in the middle row, then the maximum value is located in a blob corresponding to the considered keypoint.

5.2. Segmentation-driven pose refinement

The pose is optimized by Levenberg-Marquardt (LM) [86][109] using the reprojection error defined in Subsection 3.5.2 and optionally also the signed distance transform (SDT) [24] calculated on the basis of object segmentation. The value of the objective function in the last iteration is used to decide if the initial pose guess can be assumed as the final pose. If not, then less reliable keypoints are passed over and the remaining ones are used in further pose refinement, see Figure 5.5. After selecting such a set of points the LM is executed once again to optimize the object pose. If the pose is still determined to be weak, then the proposed Hierarchical Optimistic Optimization is executed to finally refine the object pose.

5.2.1. Hierarchical Optimistic Optimization

Monte Carlo Tree Search (MCTS) [78], [33] is an iterative algorithm that searches the state space and constructs statistical evidence about the decisions accessible from available states. It was used in AlphaGo developed by DeepMind [149] with a combination of deep reinforcement learning. More recently, it was used to improve 3D object pose estimation via physics-aware MCTS [115]. The majority of selection functions used in MCTS algorithms are not directly applicable in continuous action spaces, as they require exploring every possible action at least once. The usage of MCTS in problems with continuous-space has been relatively less studied. In such issues a progressive widening (PW) [8] is commonly utilized to discretize the action space and provide adequate exploitation. PW-based approaches maintain a finite list of available nodes to be searched and incrementally add children to the list according to number of the visits. The order of adding actions could be via exploiting domain knowledge or random. The algorithm proposed in [8] is not efficient as it draws an action uniformly at random. In contrast to [115], a more efficient Hierarchical Optimistic Optimization (HOO) [20] is employed instead of the MCTS. It is utilized to refine pose estimates calculated by the LM, which are weak, c.f. Figure 5.5.

MCTS can be used to perform global optimization. x is a sequence of actions for a given function $f(x)$. The Hierarchical Optimistic Optimization incrementally constructs a binary tree and recursively splits the continuous action space X into smaller candidate ranges at each depth. Each node in the tree covers a subset of X. The main idea is to make use of as much knowledge of f as possible around its maxima. It selects an action by starting at the root and then following a path from it to a leaf node, and picking at each node the child node that has the larger score, called the B-value. HOO adaptively splits the action space and rapidly concentrates on subspaces with potentially optimal actions.

Following the notation in [20], we index the nodes in the tree by pairs of integers (h, i), where $h0$ stands for the depth of the node, whereas $1 \leq i \leq 2^h$ stands for its index on depth h. Let $C(h, i)$ stand for the set of all descendants of node (h, i) (by convention, each node is a descendant of itself). Then

$$N_{h,i}(n) = \sum_{t=1}^{n} \mathbb{I}_{\{(H_t, I_t) \in C(h,i)\}} \tag{5.2}$$

is the number of times a descendant of (h, i) was played up to and including round n, where (H_t, I_t) denotes the node played by HOO at round t, and indexation by n is used to specify the value taken at the end of nth round. Let $\hat{R}_{h,i}(n)$ be the reward estimate of node(h, i) defined as follows:

$$\hat{R}_{h,i}(n) = \frac{1}{N_{h,i}(n)} \sum_{t=1}^{n} \mathbb{I}_{\{(H_t, I_t) \in C(h,i)\}} r_t \tag{5.3}$$

where r_t is a reward observed at round t. The upper bound on the estimate of the reward is defined in the following manner:

$$U_{h,i}(n) = \hat{R}_{h,i}(n) \sqrt{\frac{2 \ln n}{N_{h,i}(n)}} + v_1 p^h \tag{5.4}$$

where two constants $v_1 > 0$ and $0 < \; < 1$ characterize the reward function and the action domain, respectively. For nodes which have not been sampled, $\hat{R}_{h,i} = U_{h,i}(n) = \infty$. The B-value of a node is defined as follows:

$$B_{h,i}(n) = \min\{U_{h,i}(n), \max\{B_{h+1,2i-1}(n), B_{h+1,2i}(n)\}\} \tag{5.5}$$

The algorithm starts from root node $(0, 1)$ with the state determined by the LM algorithm. HOO picks an action by following a path from such a root node to a leaf node, and at each node it picks the child node that has a bigger B-value for the reward. The reward is calculated the same as in the LM.

5.3. Proposed method

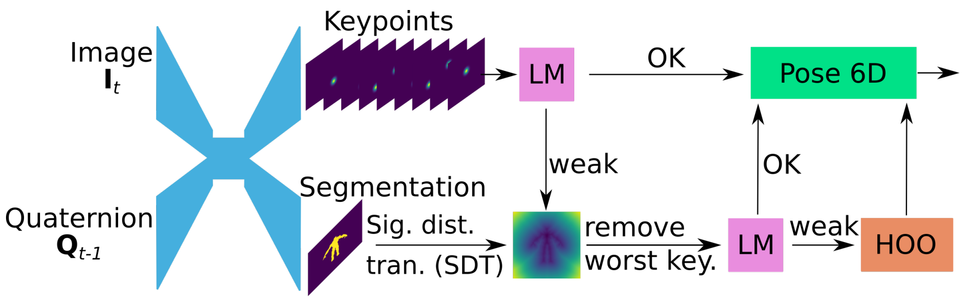


Figure 5.5: Pipeline of 3D object pose refinement.

The 3D object pose can be determined using an RGB image and information about object rotation acquired in the previous frame, see Figure 5.5. The 3D object locations together with estimated object sizes are projected to the image plane and then used to crop sub-images surrounding the tracked object. Next, X-Net provides us with 2D location of the keypoints, which are used by LM to estimate the initial pose of target object. The objective function of LM is based on keypoints that were determined by the X-Net. The reliability of keypoints is determined on the basis of values of SDT at projected 3D points. Next, the LM is executed to optimize the object pose for the point set selected in such a way. Using a value of the objective function in the last iteration the algorithm decides if the optimized pose can be assumed as the final pose. If not, then the Hierarchical Optimistic Optimization is executed to finally refine the object pose. The object rotation determined for this frame is then fed in the next one to the quaternion input of a neural network responsible for the regression of object keypoints.

5.4. Experiments and results

In this section, the experimental results that were achieved on the YCB-Video and OPT datasets are presented. Below different properties of the proposed algorithm are tested. In Subsection 5.4.1, experiments are focused on the use of various types of proposed algorithm with the X-Net. Results for YCB-Video are presented in Subsection 5.4.2.

5.4.1. Experimental results on OPT dataset

Table 5.1 presents ADD scores that have been obtained in 3D object pose tracking on the OPT dataset. The average ADD score for all objects is 84.8%. Every object has been observed in four camera views during the 6D pose tracking. The average scores from four views are presented in the last column. Due to the four parts of the FreeMotion scenario, the total number of evaluations on this dataset is even larger in comparison to a number of corresponding evaluations on the YCB-Video dataset. The scores have been acquired by the proposed X-Net using object segmentation for the pose refinement. The presented results have been achieved through training a separate neural network for each considered object and then using it for tracking its pose.

It can be noticed that the ADD scores for the House and Soda objects assume high values and for the Chest object are slightly worse. To better understand performance gains achieved by the proposed X-Net, an ablation study was conducted. Table 5.2 collects results from this ablation study. The ADD scores, as previously, are averages from four camera views.

Table 5.1: ADD scores [%] (for 10% threshold) achieved by the proposed algorithm (X-Net, single net per object, pose refinement) on the OPT dataset.

ADD	beh.	left	right	front	Avg.
Iron. X	76	99	88	85	87
House X	86	92	99	96	93
Bike X	88	76	94	75	83
Jet X	72	83	92	91	84
Soda X	75	89	100	98	90
Chest X	68	64	67	89	72

Table 5.2: ADD scores [%] (th. 10%) achieved on the OPT dataset.

Object	Net comm. w/o Q	Net comm.	X-Net comm.	X-Net comm. + ref.	X-Net sep. + ref.
Ironman	66	71	72	80	87
House	62	75	77	89	93
Jet	59	72	72	73	84
Bike	58	61	70	76	83
Chest	29	53	54	55	72
Soda	48	51	70	81	90
Avg.	53.7	63.8	69.2	75.7	84.8

A neural network without the quaternion branch as well as without the segmentation branch was trained. In Table 5.2, all columns except for the last one, present ADD scores that were achieved by a single neural network for all objects. The second column shows results for a simple neural network without a quaternion or segmentation branch. The third one presents a neural network with a quaternion branch. The next column contains ADD scores that were achieved after extending such a network about the segmentation branch. As can be observed, the proposed branch permits achieving considerable improvement of the ADD score. Experimental results that are collected in the last column demonstrate improvements in ADD scores through training a separate neural network per object.

AUC ADD scores achieved by the proposed algorithm on the OPT dataset were compared, in Table 5.3, against AUC ADD scores achieved by recent algorithms in the FreeMotion scenario. The FreeMotion scenario poses many challenges that 3D object pose estimators have to meet. As it can be seen, the proposed algorithm outperforms ORB-SLAM2 [118] as well as a recently proposed algorithm [168], which is based on photometrically-enhanced edges. The [21] that uses a contour energy and object keypoints obtained worse results than those achieved by the

Table 5.3: AUC scores [%] on OPT dataset compared to AUC scores achieved by recent methods. "-" stands for unavailable results.

AUC score [%]	scenario	House	Ironman	Jet	Bike	Chest	Soda	Avg.
ORB-SLAM2 [118]	FreeM.	-	-	-	-	-	-	9.10
CE&K [21]	FreeM.	-	-	-	-	-	-	13.91
ph. enh. edge [168]	FreeM.	13.70	10.86	-	-	9.77	-	11.44
Our X-Net. common+ref.	FreeM.	14.49	13.18	12.31	12.82	9.02	13.06	12.48
Our X-Net. sep+ref.	FreeM.	15.48	14.34	13.84	13.53	11.88	15.05	**14.10**

proposed algorithm. Figure 5.6 depicts sample qualitative results achieved on the OPT dataset. The blue bounding boxes show the ground truth poses, while the green ones correspond to the estimated poses. For better visualization, we cropped regions of interest.

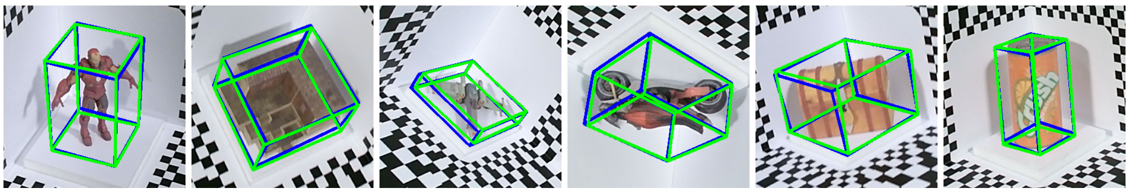


Figure 5.6: Qualitative results on OPT dataset achieved by the proposed method with X-Net.

5.4.2. Experimental results on YCB-Video dataset

In this section, experimental results that were achieved by the proposed method on the YCB-Video dataset are presented. First, experiments consisting of evaluation of the performance of the proposed X-Net on the reprojection error were conducted, see Subsection 3.5.2. Table 5.4 presents the mentioned measurement. As it can be observed, it outperforms by a large margin the PoseCNN [183] in the discussed task. It achieves also superior results compared to those obtained via more recent methods [121], [66].

Table 5.5 presents reprojection errors that have been obtained for three different versions of the algorithm achieved on the YCB Video dataset. From left to right: 3- 003_cracker_box, 4 - 004_sugar_box, 10 - 010_potted_meat_can, 21 - 021_bleach_cleanser, 35 - 035_power_drill, and 37 - 037_scissors. The results that have been obtained by the final algorithm, one neural network per object with pose refinement, are presented in the first row. The second row shows results, which have been obtained without pose refinement, i.e. without the use of object segmentation information. The third row presents reprojection errors obtained by one X-Net for all objects and without pose refinement. As it can be observed, thanks to using the object segmentation in the refinement of the object pose far better results can be obtained.

Next, different versions of the proposed algorithm were compared using ADD score on six objects from the YCB-Video dataset. An ablation study is performed to demonstrate the improvements obtained by the proposed X-Net in comparison to results achieved without the use of pose proposals. A version of the neural network without the quaternion branch was trained as well as without the segmentation branch. The second column in Table 5.6 presents

Table 5.4: Reprojection errors, 5 px scores [%] achieved by proposed algorithms, compared with errors obtained by recent algorithms (*errors from Supplementary Material in [121]).

Object	PoseCNN [183]*	DHROP [121]	[66]	X-Net sep. + ref.
002_master_chef_can	0.1	29.7	21.0	**30.8**
003_cracker_box	0.1	64.7	12	**86.6**
004_sugar_box	7.1	72.2	56.3	**73.8**
005_tomato_soup_can	5.2	39.8	46.2	**53.2**
006_mustard_bottle	6.4	87.7	70.3	**90.2**
007_tuna_fish_can	3.0	38.9	39.3	**39.8**
008_pudding_box	5.1	78.0	17.3	**78.2**
009_gelatin_box	15.8	**94.8**	83.6	94.6
010_potted_meat_can	23.1	41.2	60.7	**80.3**
011_banana	0.3	10.3	22.4	**32.2**
019_pitcher_base	0.0	5.4	**33.5**	13.1
021_bleach_cleanser	1.2	23.2	43.3	**60.3**
024_bowl	4.4	**26.1**	13.3	19.3
025_mug	0.8	29.2	38.1	**40.3**
035_power_drill	3.3	69.5	43.3	**80.8**
037_scissors	0.0	12.1	8.8	**44.7**
040_large_marker	1.4	1.9	13.6	**17.4**
051_large_clamp	0.3	24.2	7.6	**25.3**
052_extra_large_clamp	0.6	1.3	0.6	**11.8**
061_foam_brick	0.0	75.0	13.5	**78.6**
Avg.	5.8	47.2	37.4	71.1

Table 5.5: Reprojection error 5 px scores [%].

Version	3	4	10	21	35	37	Avg.
X-Net sep. + ref.	86.6	73.8	80.3	60.3	80.8	44.7	71.08
X-Net sep.	43.3	64.8	89.0	46.8	74.0	27.1	57.5
X-Net comm.	29.0	79.0	70.0	43.3	73.5	30.3	54.18

ADD scores that were achieved by a single neural network for all objects. After extending such a network about the pose proposal branch the achieved results were improved, see the third column. Experimental results that are collected in the 4th column demonstrate improvements in ADD scores through training a separate neural network per object. The ADD scores obtained by a single X-Net for all objects are presented in the fifth column of the table. In the sixth column of the table are scores obtained by the X-Net with no pose refinement. The discussed results have been achieved through training a separate neural network for each considered object and then using it for tracking its pose. As it can be observed the ADD scores achieved by a single X-Net trained for all objects are slightly worse in comparison to the scenario with separate X-Nets. Similar evaluations, i.e. with one pose estimator for all considered objects or 1 per object have been performed in a recent work [175]. The results achieved by the GDR-Net using a pose estimator per object were also better in comparison to the scenario with one pose estimator for all considered objects. The last column in the table contains ADD scores that have been achieved by the proposed X-Net using object segmentation for the pose refinement. A comparison of results with results presented in the last column shows that the segmentation branch permits achieving far better ADD scores. Based on the same evaluation method as used for the results presented in Table 5.2, it was confirmed that the input pose proposal as well as object segmentation permit achieving far better ADD scores. Below we perform several comparative evaluations and show that our algorithm surpasses other RGB-only methods.

Table 5.6: ADD scores [%] (10% th.) on the YCB-Video dataset.

Object	Net comm. w/o Q	Net comm.	Net sep.	X-Net comm.	X-Net sep.	X-Net sep. + ref.
003_cracker_box	48.3	52.3	58.3	57.0	63.7	91.6
004_sugar_box	69.6	74.8	81.2	77.4	81.4	77.6
010_potted_meat_can	46.3	60.3	64.6	66.6	68.0	70.0
021_bleach_cleanser	52.8	59.8	61.5	62.8	63.8	72.0
035_power_drill	57.5	72.2	73.3	74.0	79.3	89.5
037_scissors	00.0	30.7	38.3	35.0	39.0	68.0
Avg.	45.8	58.4	62.9	62.1	65.9	78.1

Table 5.7 presents the AUC ADD scores achieved by the proposed algorithm for pose tracking on the YCB-Video dataset in comparison against scores achieved by SOTA algorithms on the YCB-Video dataset. The AUC ADD score is calculated for non-symmetrical objects, whereas the ADD-S score [183] is determined for symmetrical ones. For five objects out of twenty-one objects our algorithm achieves the best results. GDR-Net algorithm [175] that has been published recently achieves the best results for nine objects, but the average AUC ADD score achieved by the discussed algorithm is smaller in comparison to those achieved by the proposed method. DeepIm [89] has achieved the best average AUC ADD score. Comparing results achieved by DeepIm against results achieved by the proposed method, it can be observed that the proposed algorithm achieved superior results on eleven out of twenty-one object classes, whereas DeepIM had better performance for ten object classes. In one case the DeepIM was better at about 0.1. As it can be noticed, the proposed algorithm surpasses by a large margin the PoseCNN [183]. It also outperforms [121] that focused on deep heatmaps robust to partial occlusions for 3D object pose estimation. Otherwise, it achieves superior results than PoseRBPF [36] that utilizes a Rao-Blackwellized particle filter for 6D object pose tracking. A recently proposed GDR-Net [175], which is a geometry-guided direct regression network achieves competitive results against results achieved by the proposed algorithm. Both algorithms are based on guiding mechanisms, where GDR-Net is a geometry-guided method, whereas the proposed method

is pose proposal-guided one. As it can be observed, owing to using the object segmentation information for pose refinement the results are significantly better, although in some cases the results have not been improved, for instance, c.f. ADD AUC for 004 sugar box.

Table 5.7: AUC ADD scores [%] (max. th. 10 cm) on the YCB-Video dataset compared to scores achieved by recent methods.

	Pose recovery on single RGB images								Pose tracking		
Object	PoseCNN [183]	DOPE [166]	DHROP [121]	[187]	GDR-Net [175]	PVNet [128]	YOLOPose [129]	[130]	PoseRBPF [36]	DeepIM [89]	Proposed
002_master_chef_can	50.9	-	81.6	49.9	65.2	81.6	71.3	66.7	63.3	89.0	**90.7**
003_cracker_box	51.7	55.9	83.6	80.5	88.8	80.5	83.3	86.0	77.8	88.5	**88.9**
004_sugar_box	68.6	75.7	82.0	85.5	**95.0**	84.9	83.6	89.1	79.6	94.3	84.8
005_tomato_soup_can	66.0	76.1	79.7	68.5	**91.9**	78.2	72.9	76.3	73.0	89.1	84.9
006_mustard_bottle	79.9	81.9	91.4	87.0	92.8	88.3	**93.4**	93.3	84.7	92.0	93.3
007_tuna_fish_can	70.4	-	49.2	79.3	**94.2**	66.2	70.5	67.4	64.2	92.0	91.9
008_pudding_box	62.9	-	90.1	81.8	44.7	85.2	87.0	**91.9**	64.5	80.1	86.4
009_gelatin_box	75.2	-	93.6	89.4	92.5	88.7	85.7	91.8	83.0	92.0	**93.9**
010_potted_meat_can	59.6	39.4	79.0	59.6	80.2	65.1	71.4	76.4	51.8	78.0	**86.8**
011_banana	72.3	-	51.9	36.5	85.8	51.8	90.0	**91.0**	18.4	81.0	74.8
019_pitcher_base	52.5	-	69.4	78.1	**98.5**	91.2	90.8	89.9	63.7	90.4	76.2
021_bleach_cleanser	50.5	-	76.1	56.7	**84.3**	74.8	70.8	73.9	60.5	81.7	72.8
024_bowl*	69.7	-	76.9	23.5	85.7	89.0	**93.4**	91.9	85.6	90.6	87.7
025_mug	57.7	-	53.7	54.0	**94.0**	81.5	90.0	89.3	77.9	92.0	91.8
035_power_drill	55.1	-	82.7	82.8	**90.1**	83.4	85.2	88.9	71.8	85.4	87.0
036_wood_block*	65.8	-	55.0	29.6	82.5	71.5	**93.0**	**93.0**	31.4	75.4	84.4
037_scissors	35.8	-	65.9	46.0	49.5	54.8	71.2	**76.2**	38.7	70.3	70.5
040_large_marker	58.0	-	56.4	9.8	76.1	35.8	77.0	77.4	67.1	80.4	**84.0**
051_large_clamp*	49.9	-	67.5	47.4	89.3	66.3	**94.7**	94.2	59.3	84.1	89.9
052_extra_large_clamp*	47.0	-	53.9	47.0	**93.5**	53.9	80.7	79.2	44.3	90.3	89.5
061_foam_brick*	87.8	-	89.0	87.8	**96.9**	80.6	93.8	95.0	92.6	95.5	88.0
Avg.	61.3	65.8	72.8	61.0	84.4	73.8	83.3	84.7	64.4	**86.3**	**86.3**

Several experiments were conducted to better understand the effect of using LM for 3D object pose estimation on the basis of keypoints estimated by the proposed X-Net. To achieve comparability of results the keypoint locations of YCB-Video objects were estimated in test sequences and then stored. The 3D object poses were estimated and compared to ground-truth poses. For each of the considered algorithms, LM, EPnP [84], and EPnP-RANSAC were counted on how many times they achieved the best estimates. In the first row of Figure 5.7, a bar plot illustrates results for 035_power_drill, 010_potted_meat_can and 003_cracker_box. As can be observed, the best results have been achieved by the LM algorithm. The second row of the Figure depicts ADD scores vs. frame number for the discussed set of objects. The ADD scores achieved by LM, EPnP, and EPnP-RANSAC were as follows: power drill - LM 70%, 55% 53%, potted meat can - 44%, 33%, 39%, and cracker box - 78%, 69%, 68%.

In order to verify the effect of joint detection of object keypoints and object segmentation for 3D object pose refinement, ADD scores vs. frame number that have been achieved with no pose refinement and with pose refinement were analyzed. Figure 5.8 depicts sample results, which have been obtained for the power drill, large clamp, and tuna fish can from the YCB-Video dataset. As it can be observed, using information about object segmentation

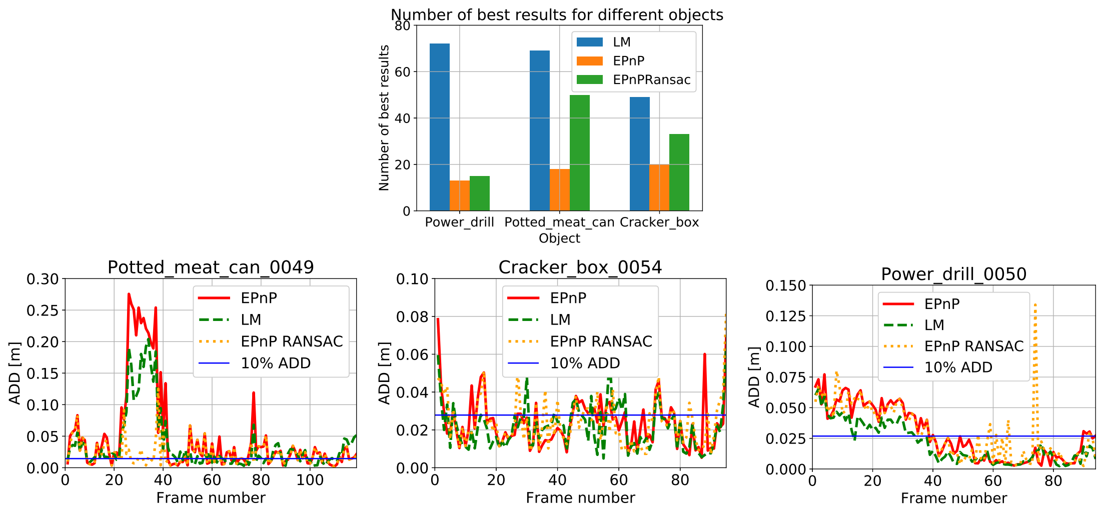


Figure 5.7: ADD scores vs. frame number for EPnP, EPnP/RANSAC and LM.

improves the ADD scores as well as the values of the objective in the last iteration of LM algorithm. With no object segmentation-based pose refinement (left), with object segmentation-based pose refinement (right).

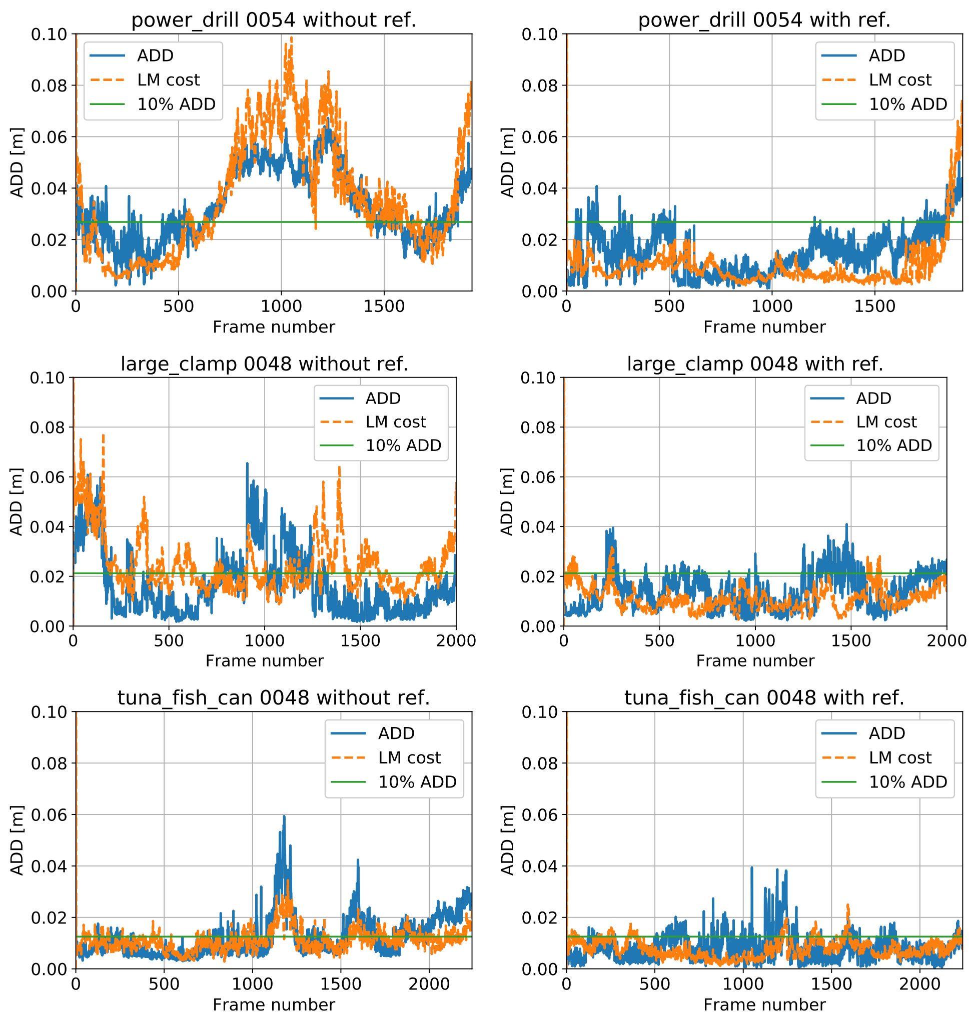


Figure 5.8: ADD scores for a power drill, large clamp, and tuna fish can from YCB-Video.

To better understand the usefulness of HOO in pose refinement the ADD scores were analyzed. Figure 5.9 depicts sample plots of ADD scores vs. frame number with and with no HOO-based pose refinement. As it can be observed, for the power drill the improvement of ADD is significant, whereas for the cracker box, the HOO also improved the average ADD, but sporadically in some frames the ADD scores can be worse.

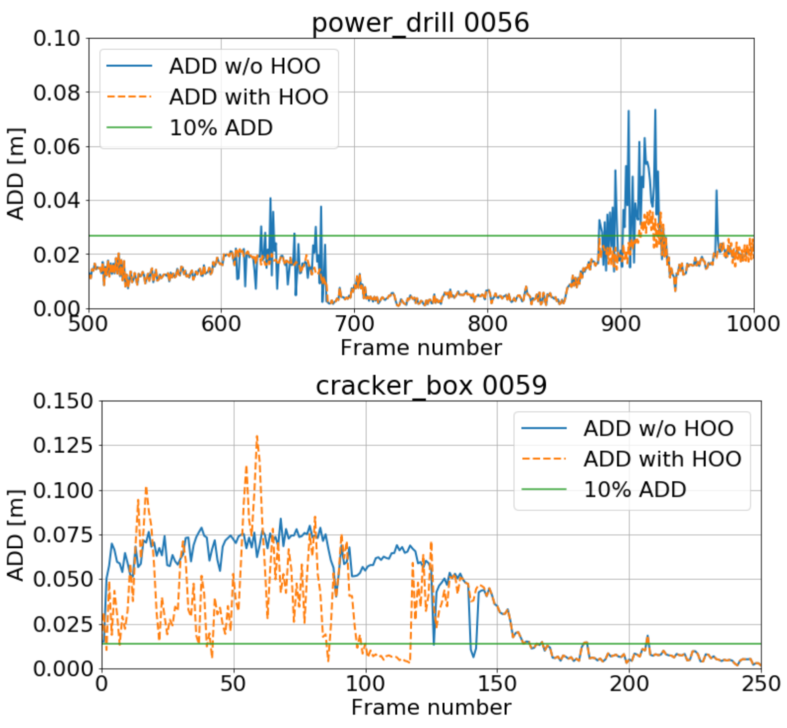


Figure 5.9: ADD scores for power drill and cracker box from YCB-Video with and with no HOO-based pose refinement.

Figure 5.10 depicts sample qualitative results for the YCB-Video dataset. These sample images show that the proposed method is robust to scene clutter, severe occlusions, different illumination, reflection, and different object motions. As shown, the large errors are often produced for symmetrical objects, see also 024 bowl (2nd row, 6th from left). Example images demonstrate how our algorithm handles occlusions, different arrangement, lighting in ordinary shots for the following objects: 002_master_chef_can, 003_cracker_box, 004_sugar_box, 005_tomato_soup_can, 006_mustard_bottle, 007_tuna_fish_can, 008_pudding_box, 009_gelatin_box, 010_potted_meat_can, 011_banana, 019_pitcher_base, 021_bleach_cleanser, 024_bowl, 025_mug, 035_power_drill, 036_wood_block, 037_scissors, 040_large_marker, 051_large_clamp, 052_extra_large_clamp, and 061_foam_brick. The blue bounding boxes show the ground truth poses, while the green ones correspond to the estimated poses. For better visualization, we cropped regions of interest.

To investigate the effect of object rotation on pose tracking for symmetrical objects, we trained two neural networks on the YCB-Video dataset: one without quaternion input (Net comm. w/o Q) and another with quaternion input (Net comm.). We compared their performance using ADD scores. As shown in Table 5.8, incorporating quaternion input significantly improves pose tracking results for all symmetrical objects in the dataset, as the network utilizing previous pose information achieves superior outcomes.

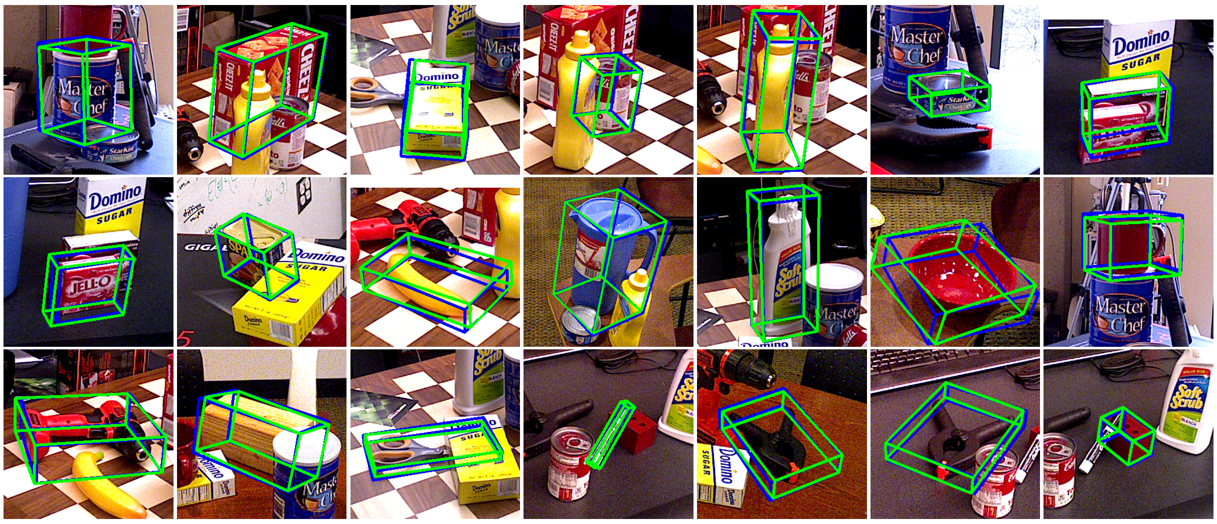


Figure 5.10: Qualitative results on YCB-Video.

Table 5.8: ADD scores [%] (for threshold 10%) achieved by the proposed algorithms on the YCB-Video dataset.

Symmetrical object	Net	
	comm. w/o Q	comm.
024_bowl	23	59
036_wood_block	36	56
051_large_clamp	48	61
052_extra_large_clamp	29	50
061_foam_brick	33	57
Avg.	33.8	56.6

5.5. Summary

This chapter addressed the problem of the selection of an appropriate set of keypoints and the expansion of the method proposed in the previous Chapter 4. The proposed X-Net for 3D object pose tracking extends the Y-Net by adding segmentation of tracked object. Simultaneous detection of keypoints and segmentation of an object enables the generation of object distance maps, thanks to which it is possible to eliminate outliers and suppress less reliable keypoints. A new keypoints selection algorithm was also introduced for the pose refinement strategy. It permits achieving better AUC ADD scores. The obtained results on YCB-Video and OPT datasets indicate that the proposed methods are effective and have some potential. It was demonstrated that the proposed methods achieve competitive results against SOTA.

Chapter 6

Proposed PF-PSO

In this chapter, a different approach for the usage of information about pose from the previous frame is proposed. A quaternion particle filter with a probabilistic motion model and a probabilistic observation model is employed to maintain 3D object pose distribution. It uses a matching between segmentation and a projected 3D model onto the image plane to acquire information about the probability of predictions. A Siamese neural network [18] is trained on keypoints from the current and previous frame in order to generate a particle representing the predicted 3D object pose. The filter draws particles to predict the current pose using it's a priori knowledge about the object velocity and includes the predicted 3D object pose by the neural network. Thus, the hypothesized 3D object poses are generated using both a priori knowledge about the object velocity in 3D and keypoint-based geometric reasoning as well as relative transformations in the image plane. A k-means++ [5] is used to maintain multimodal probability distributions. A multi-swarm particle swarm optimization [74] is executed afterward to find the finest modes in the probability distribution together with the best pose. The potential of quaternion locomotion models is demonstrated in several computer simulations. The author's proposed method based on a Siamese neural network for object pose estimation was proposed in [98]. The particle filter and particle swarm optimization were published in [97, 99] and the version of particle filter and particle swarm optimization that operates on quaternion particles was published in [98, 102, 105].

This chapter is organized as follows. Section 6.1 discusses the proposed quaternion particle filter. Then, in Section 6.2, there is an overview of particle swarm optimization. Next, Section 6.3 which includes a neural network for segmentation is discussed. Section 6.4 includes information about keypoints estimation. A Siamese neural network working on keypoints from the current and previous frame in order to generate a particle in the predicted 3D object pose is described in Section 6.5. The whole algorithm is discussed in Section 6.6. Section 6.7 includes experimental results and discussion about them. An additional version of the proposed method is discussed in Section 6.8. A summary is given in Section 6.9.

In this chapter, the following contributions to the object pose tracking are presented:

– a unit quaternion representation of the rotational state space for a particle filter and the particle swarm optimization

– Siamese neural network, which guides the particle representing object pose hypothesis toward the predicted pose of the object

6.1. Particle Filter

The idea of the particle filter (PF) [92] is based on Monte Carlo methods, which use particle sets to represent probabilities and can be used in any form of state space model.

The PFs have been applied in a diverse range of disciplines including control, computational physics, surveillance, defense, oceanography, geoscience, finance, autonomous vehicles, robotics, remote sensing, computer vision, biomedical research, and wireless communications [3, 132, 45, 80] to solve problems like object tracking, estimation, detection, equalization, and localization [172]. Particle filtering uses a set of particles to represent the posterior distribution of a stochastic process given the noisy and partial observations. Particle filters employ an approximate approach to update their predictions. A collection of particles represents the samples from the underlying distribution, each accompanied by a likelihood weight that denotes the probability of sampling this particle from the probability density function. To ensure efficient computation and prevent degeneracy, various adaptive resampling criteria can be employed [116]. These include the variance of the weights and the relative entropy with respect to the uniform distribution. In the resampling step, particles with negligible weights are replaced by new particles that are strategically located in the vicinity of those with higher weights. Particle filter techniques provide a well-established methodology for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions. These methods do not perform well when applied to very high-dimensional systems. Recently, a dual quaternion filter for recursive estimation of rigid body motions has been proposed [88]. A particle filter with quaternion-based state representation has been utilized for parameter estimation in electron cryo-microscopy (cryo-EM) in [64]. In [4], an operation known in geophysics as nudging has been applied to improve the sampling in the particle filters.

In this chapter, the unit quaternion representation of the rotational state space for a particle filter is proposed. The mentioned particle filter was published in [102]. The state vector describing the 6D object pose comprises two parts: a quaternion as a description of rotations and a translation vector in Euclidean space. The translation's origin is in the camera coordinate system. The quaternion is described in Section 3.2. In order to introduce the process noise in the quaternion motion of particle i, a three-dimensional normal distribution with zero mean and covariance matrix C_r in the tangential space is applied as follows:

$$\mathbf{q}_i(t+1) = expMap(\mathcal{N}([0,0,0]^T, C_r)) \star \mathbf{q}_i(t) \tag{6.1}$$

where $\mathbf{q}_i(t)$ - orientation of particle i at time t, C_r - covariance matrix for rotation with values $(\gamma_{r_1}\ \gamma_{r_2}\ \gamma_{r_3})$ on diagonal, $\star$ - quaternion product (Equation 3.4) and $expMap$ - exponential function (Equation 3.6).The probabilistic motion model for the translation is as follows:

$$\mathbf{z}_i(t+1) = \mathcal{N}([0,0,0]^T, C_z) + \mathbf{z}_i(t) \tag{6.2}$$

where $\mathbf{z}_i(t)$ - translation of particle i at time t and C_z - covariance matrix for translation with values $(\gamma_{z_1}\ \gamma_{z_2}\ \gamma_{z_3})$ on diagonal. Each particle i is represented as $s_i(t) = (\mathbf{x}_i(t), w_i(t))$, where $w_i(t)$ is the particle's weight. The weights are calculated on the basis of a probabilistic observation model and then used in the resampling of the particles. With the resampling, the particles with large weights are replicated and the ones with negligible weights are eliminated. In Algorithm 6.1, the measurement at time t is denoted by z_t, and $\mathbf{x}_t$ denotes the state of the system.

A new ensemble of particles is created between lines 2 and 10 by executing the probabilistic motion model, probabilistic observation model, and resampling techniques. This process recursively estimates the particle set X_t

Algorithm 6.1 Particle Filter

```
1   function PF($X_{t-1}$)
2       set $X_t = X_t^s = \emptyset$
3       for $j = 1$ to $m$ do
4           pick the $j$-th sample $\mathbf{x}_{t-1}^{[j]} \in X_{t-1}$
5           draw $\mathbf{x}_t^{[j]} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{[j]})$
6           set $w_t^{[j]} = p(z_t|\mathbf{x}_{t-1}^{[j]})$
7           add $\langle \mathbf{x}_t^{[j]}, w_t^{[j]} \rangle$ to $X_t^s$
8       endfor
9       for $i = 1$ to $m$ do
10          draw $\mathbf{x}_t^{[i]}$ from $X_t^s$ with prob. $\propto$ to $w_t^{[i]}$
11          add $\mathbf{x}_t^{[i]}$ to $X_t$
11      endfor
12      return $X_t$
```

based on its previous estimate X. As the number of particles (m) increases, the weighted particle set converges asymptotically to the desired posterior distribution.

The probabilistic observation model is defined as: $p(z_t|\mathbf{x}_{t-1}) = exp(-f_s/\sigma_0^2)$ where σ_0^2 is a value selected through experimental means. Particles are then propagated according to a Gaussian distribution characterized by σ_m^2, which was empirically established.

6.2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) [74, 144, 1] is a global optimization metaheuristic and stochastic method, which is based on swarm intelligence. It is a derivative-free, population-based computational method, which demonstrated a high potential in the optimization of unfriendly non-convex functions. The optimal solution is sought by a swarm of particles exploring candidate solutions, where each swarm member represents a potential solution of an optimization task. Each particle within this swarm represents a unique candidate solution to an optimization problem and iteratively refines its performance by striving to maximize a predetermined measure of quality. Every individual moves with its own velocity in the multidimensional search pace, determines its own best position, and calculates its fitness on the basis of a fitness function $f(x)$, see line #4 in Algorithm 6.2. Each particle's motion is influenced by its local best known position as well as the entire swarm's best known position. The objective function is used to determine the best particles' locations as well as the global best location. In the ordinary PSO, at the beginning, each particle is initialized with a random position and velocity [74].

During exploration of the search space every particle i updates its position, which is affected by the best position of this particle $\mathbf{p}_i = [\mathbf{q}_{i,best}\ \mathbf{z}_{i,best}]$ determined so far and the global best position $\hat{\mathbf{g}} = [\mathbf{q}_{gbest}\ \mathbf{z}_{gbest}]$ found by the entire swarm. The 3D position z and 3D velocity of the particle i in iteration k are determined, see lines #9-10 in Algorithm 6.2, as follows:

$$\mathbf{v}_i(k+1) = w\mathbf{v}_i(k) + c_1 r_1(\mathbf{z}_{i,best}(k) - \mathbf{z}_i(k)) + c_2 r_2(\mathbf{z}_{gbest}(k) - \mathbf{z}_i(k)) \tag{6.3}$$

$$\mathbf{z}_i(k+1) = \mathbf{z}_i(k) + \mathbf{v}_i(k+1) \tag{6.4}$$

where $\mathbf{v}_i(k)$ is velocity of particle i in iteration k, $\mathbf{z}_i(k)$ is position of particle i in iteration k, $\mathbf{z}_{i,best}(k)$ is the best hypothesis for particle i in iteration k, $\mathbf{z}_{gbest}(k)$ is the best hypothesis for all particles in iteration k. w is a positive inertia weight, c_1, c_2 are positive, cognitive and social constants, respectively, $r_{1,j}^{(i)}$ and $r_{2,j}^{(i)}$ are uniquely generated random numbers with the uniform distribution in the interval $[0.0, 1.0]$, generated in each iteration, for each dimension and independently for each particle.

Spherical linear interpolation (SLERP) [147], see Section 3.3, is used to calculate the object's angular velocity. The angular velocity update, see line #11 in Algorithm 6.2, for the i-th particle is defined as follows:

$$\begin{aligned}\omega_i(k+1) =& w\omega_i(k) + c_1 r_1[2logMap(\mathbf{q}_{i,best}(k) \star \mathbf{q}_i^*(k))] \\ &+ c_2 r_2[2logMap(\mathbf{q}_{gbest}(k) \star \mathbf{q}_i^*(k))]\end{aligned} \tag{6.5}$$

where $\mathbf{q}_i(k)$ is rotation of particle i in iteration k, $\mathbf{q}_{i,best}(k)$ is the best hypothesis for particle i in iteration k, $\mathbf{q}_{gbest}(k)$ is the best hypothesis for all particles in iteration k and $logMap$ is logarithmic map (Equation3.5). The rotation of the i-th particle is then updated, see line #12 in Algorithm 6.2, in the following manner:

$$\mathbf{q}_i(k+1) = [cos(\frac{\|\omega_i(k+1)\|T_c}{2}), sin(\frac{\|\omega_i(k+1)\|T_c}{2})\frac{\omega_i(k+1)}{\|\omega_i(k+1)\|}]\mathbf{q}_i(k) \tag{6.6}$$

where T_c is a parameter to scale angular velocity. After determining $\mathbf{x}_i(k+1)$ using Equations 6.4 6.6, $\mathbf{p}_i(k+1)$ is updated as follows:

$$\mathbf{p}_i(k+1) = \begin{cases} \mathbf{p}_i(k) & if\ f(\mathbf{x}_i(k+1)) \geq f(\mathbf{p}_i(k)) \\ \mathbf{x}_i(k+1) & if\ f(\mathbf{x}_i(k+1)) < f(\mathbf{p}_i(k)) \end{cases} \tag{6.7}$$

In a topology with the global best every particle is able to obtain information from the very best particle in the entire swarm population. The use of topology with the global best has been found to lead to fast convergence, though such optimizer is susceptible to getting trapped in local minima. In this version, the particles that will be updated in the same iteration can exploit the new best position immediately, instead of using the global best calculated in the previous iteration.

6.3. Object segmentation

To accurately estimate the pose of the tracked object and to detect keypoints with better precision, the segmentation of object on single images has been performed. The architecture of neural network for object segmentation has been based on the U-Net [139], see Figure 6.1. These networks consist of a contracting path (encoding) path and an expansive (decoding) path. In the contraction path, each block is composed of a series of 3×3 convolution layers followed by a 2×2 max pooling, where the number of cores after each block doubles. One distinctive aspect of the U-Net architecture is its expansive path, which mirrors the contracting path in structure. The expansive path comprises multiple expansion blocks, where conventional 2×2 max pooling operations are substituted with 2×2

Algorithm 6.2 Quaternion Particle Swarm Optimization

```
function QPSO($X, iter$)
    for $j = 1$ to $iter$ do
        for each particle $\mathbf{x}_i \in X$ do
            $fx_i = f(\mathbf{x}_i)$
            if $fx_i < f(\mathbf{p}_i)$ then
                $\mathbf{p}_i = \mathbf{x}_i$
            if $fx_i < f(\hat{\mathbf{g}}_i)$ then
                $\hat{\mathbf{g}}_i = \mathbf{x}_i$
            $\mathbf{v}_i \leftarrow$ update using 6.3
            $\mathbf{z}_i \leftarrow$ update using 6.4
            $\omega_i \leftarrow$ update using 6.5
            $\mathbf{q}_i \leftarrow$ update using 6.6
        endfor
    endfor
    return $\hat{\mathbf{g}}_i, X$
```

upsampling layers instead. The expanding path consists of the upsampling operation followed by a 2×2 convolution, combined with an appropriately cropped property map, and this is completed by two 3×3 convolutions, followed by ReLU. The feature maps between the encoder and decoder are directly connected by skip connections. The neural network was trained on RGB images. *BN* [68] was added after each Conv2D in order to reduce the training time, prevent overfitting, and increase the performance of the U-Net. As it improves gradient flow through the network, it reduces dependence on initialization and higher learning rates are achieved. To enhance the robustness of our model, we incorporated data augmentation techniques during training, specifically incorporating geometric transformations and color space transformations. The pixel-wise cross-entropy has been used as the loss function:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{6.8}$$

where N stands for the number of training samples, y is true value and $\hat{y}$ denotes predicted value. To segment all objects, a single U-Net was trained on 900 images from the OPT benchmark. Additionally, a separate U-Net was trained on 1800 images to accurately segment all six objects within the custom dataset. The Dice scores were higher than 95% for all objects. A method for object pose estimation using U-Net was published in [96].

6.4. Keypoint detection

One of the hypothetical poses in the swarm algorithm is estimated using a CNN, which delivers 2D locations of eight fiducial points of the object. The 2D locations of such points are then fed to a PnP algorithm, that delivers the 6D pose of the object. A segmented image of size 128×128 pixels is delivered from the U-Net CNN to the fiducial CNN as input, which determines the 2D locations of eight keypoints.

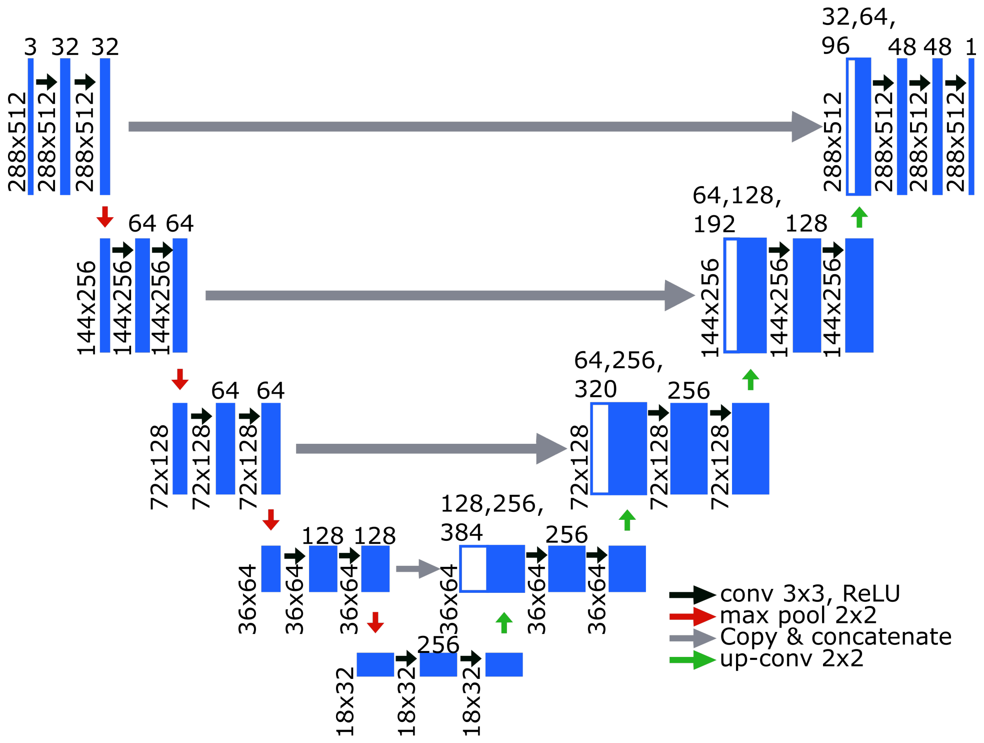


Figure 6.1: Architecture of U-Net for object segmentation.

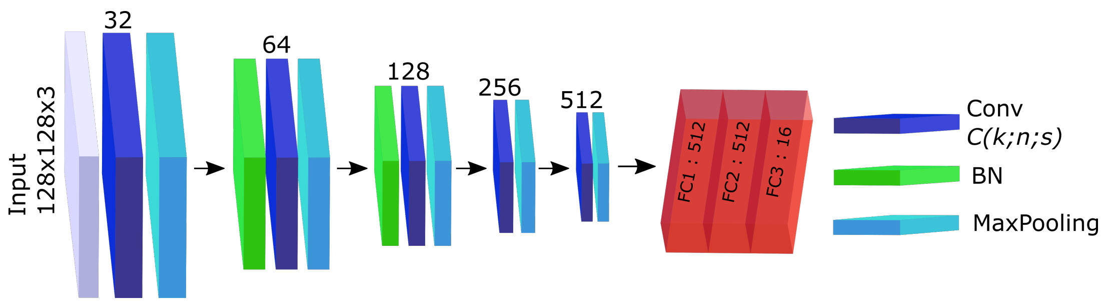


Figure 6.2: Neural network for estimation of fiducial points.

Figure 6.2 shows a schematic architecture of fiducial CNN.

The discussed network architecture has been designed to obtain sufficient precision in the estimation of the position of keypoints in the images and at the same time to avoid introducing unnecessary delays in the estimation of the position of the points during the pose tracking.

The neural network architecture is as follows: the input image is fed to a convolution layer with a kernel sized 3×3 and 32 filters, which is followed by max-pooling with a kernel of size 2×2. Then, there is a batch normalization and ReLU activation function. Next, there is a convolution layer with a kernel sized 3×3 and 64 filters, max-pooling with a kernel of size 2×2, and batch normalization. The next part consists of 3 consecutive convolutional layers with 3×3 kernels followed by max-pooling with a kernel of size 2×2. In the MLP part, in the first two layers, there are 512 neurons in each layer, and in the last there are 16 neurons that deliver the positions of 8 pairs of keypoints of the object. The loss function was a mean absolute error (MAE). This network is a result of several experiments aiming at finding possibly small neural network, while still achieving pose estimates with accuracy sufficient for object grasping. For each object, a separate network has been trained. Given class information determined by segmentation networks like mask-RCNN [56], the network for pose estimation can be selected automatically.

6.5. Siamese neural network for object pose prediction

In order to predict the pose of the tracked object, the location of characteristic keypoints is estimated using a neural network that was discussed in Section 6.4. Then, they are used by the PnP algorithm to determine the object pose, which is added as one of the hypotheses. Having in mind the significance of keypoints, we propose a Siamese-like neural network, which operates on keypoints in order to predict object pose and pull particles representing hypothetical poses in the space of available object poses.

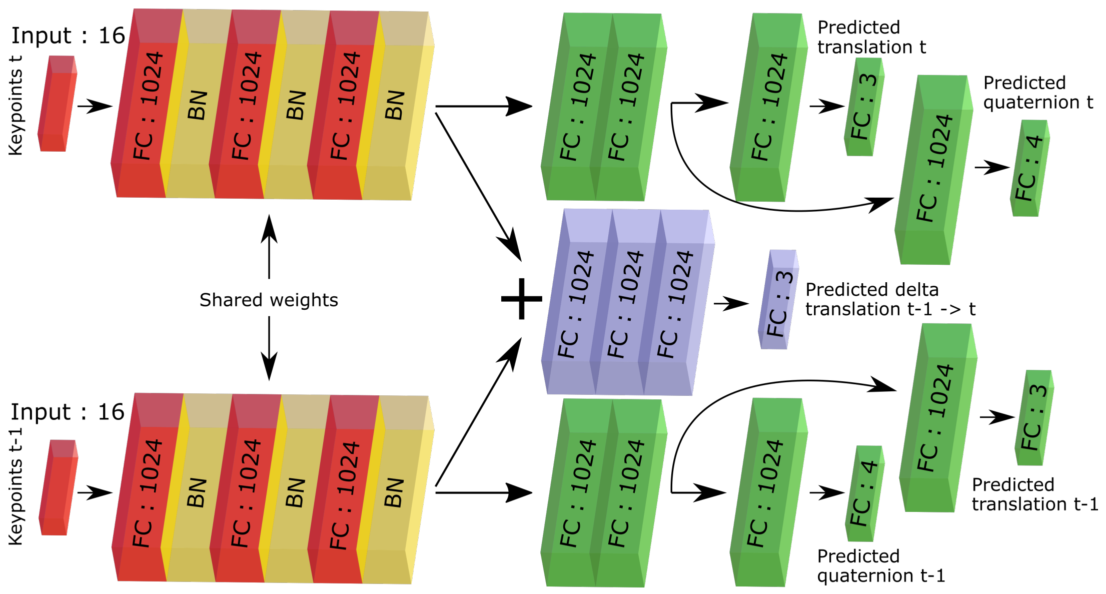


Figure 6.3: Architecture of Siamese neural network for prediction of 6D pose of the object.

Figure 6.3 depicts the architecture of the proposed neural network for the prediction of the object pose. It is a Siamese-like neural network, due to containing shared weights. The network takes as input pairs of positions of eight keypoints in the current frame and a previous one. The neural network predicts 3D positions and 3D rotations of the object in the next frame. The cost function is calculated using 3D positions of the object in time $t-1$ and t, quaternions representing object rotations in time $t-1$ and t, and delta translations from time $t-1$ to time t. The loss function is a sum of the following components:

$$L = L_{T,t-1} + L_{Q,t-1} + L_{T,t} + L_{Q,t} + L_{DT1} + L_{DT2} \tag{6.9}$$

where $L_{T,t-1}$ and $L_{Q,t-1}$ denote the position and rotation loss components in time $t-1$. Same, $L_{T,t}$ and $L_{Q,t}$ represent the position and rotation loss ingredient in time t. L_{DT1} and L_{DT2} stand for two delta translation loss components. The first of them expresses a discrepancy between ground truth translation with predicted translation. The second one calculates delta translation from predicted poses in time $t-1$, t and then calculates its discrepancy with the ground truth. The quaternions delivered by the network were normalized [98]. The mean squared error (MSE) metric was selected to quantify all loss values.

For the OPT dataset and proposed dataset, separate Siamese neural networks were trained in 15 epochs with batch size set to 256 using Adam optimizer with lr set to $1e-4$. Networks have been trained on 1000000 pairs of positions of eight keypoints. The keypoint positions were generated on the basis of projections of 3D keypoints of the objects. The pairs were generated with the starting pose in the range $[-180°, 180°]$ with 45° step on every axis. The translation was sampled from a uniform distribution in the interval $[-30$ cm, 30 cm$]$ with step 15 cm for all axes. The Z-axis was fixedly increased to fit the object in the image. The second pose of a pair was then generated by changing randomly the starting rotations on the basis of Equation 6.1 and changing randomly the translation through sampling from the uniform distribution $[-5$ cm, 5 cm$]$ with 2 cm step for all axes. It is obvious that the use of trajectories, or even keypoints from three consecutive frames might lead to better results or similar results.

6.6. Proposed method

Using Q-PF and Q-PSO as the base ingredients in this thesis, an algorithm for 6 DoF object pose estimation and tracking on RGB images acquired from a calibrated camera is proposed, see Figure 6.4. Using the U-Net neural network, the object of interest is segmented in a sequence of images. The 2D locations of eight fiducial points on the object are then estimated by a neural network that takes the segmented object as an input. Next, the 6D pose of the object is estimated by the PnP algorithm [84], which is then employed as a pose hypothesis during inference of the 6D pose. The Siamese neural network, which is trained on keypoints from current and previous frames, is used to predict the 6D object pose. Then, the object pose prediction is employed as a pose hypothesis in the next frame.

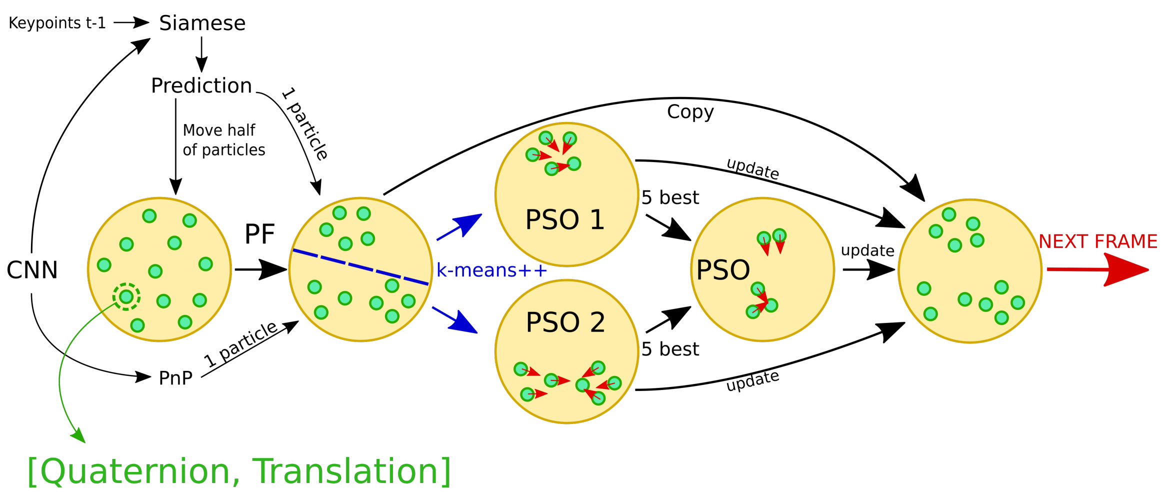


Figure 6.4: Q-PF-PSO supported with Siamese neural network-based object pose predictions.

Given the particle set representing the posterior probability distribution in the previous frame, the particles are propagated according to a probabilistic motion model, see Section 6.1. Then using the probabilistic observation model, the pose likelihoods are calculated, see Section 6.1. Afterward, the particle weights are determined and resampling is executed, as in an ordinary particle filter. After that, a particle with a small weight is replaced with a particle with a pose determined on the basis of keypoints and the PnP algorithm, see line #3 in below Algorithm 6.3. A particle with a pose predicted by the Siamese neural network is replacing a particle with the smallest weight. Next, samples are clustered using k-means++ algorithm [5] (blue lines on Figure 6.4), which applies a sequentially random selection strategy according to a squared distance from the closest center already selected. Afterward, there are three iterations of a two-swarm PSO (PSO 1 and PSO 2 on Figure 6.4) to find the modes in the probability distribution. Next, the ten best particles are selected from these two swarms to form a sub-swarm (PSO on Figure 6.4), see lines #8-9 in Algorithm 6.3. Such a sub-swarm then performs twenty iterations to find better particle positions. The number of particles and iterations were selected experimentally. The discussed sub-swarm returns the best global position and then it is used in visualization of the best pose. Finally, the estimate of the probability distribution is calculated by replacing the particle positions determined by the Q-PF with corresponding particle positions, which were selected to represent the modes in the probability distribution, see lines #6-7, and particles refined by the sub-swarm, see line #11. Ten particles with better positions found by the Q-PSO algorithms and ten particles with better positions found by the sub-swarm update the initial probability distribution, see line #12.

Algorithm 6.3 Quaternion PF-PSO

```
1   function select($n\_best$, $X$)
2       $X_{sorted} \leftarrow$ quicksort($X$) using $f(\mathbf{x})$
3   return $X_{sorted}[1...n\_best]$
    **Input:** $X_{t-1}$ - particle set ($\mathbf{x}$=[$\mathbf{q}$ $\mathbf{z}$])
1   $X_t \leftarrow$ propagate $X_{t-1}$ using (6.1, 6.2)
2   $X_t \leftarrow$ PF($X_t$) using $f(\mathbf{x})$
3   $\mathbf{x}_t^{PnP} \leftarrow PnP()$, replace worst $\mathbf{x} \in X_t$ with $\mathbf{x}_t^{PnP}$
4   $\mathbf{x}_t^{Siam} \leftarrow Siam()$, replace worst $\mathbf{x} \in X_t$ with $\mathbf{x}_t^{Siam}$
5   $X_t^{c1}, X_t^{c2} \leftarrow$ k-means++($X_t$)
6   $\sim, X_t^{c1} \leftarrow$ QPSO($X_t^{c1}, 3$) using (6.4,6.6)
7   $\sim, X_t^{c2} \leftarrow$ QPSO($X_t^{c2}, 3$) using (6.4,6.6)
8   $X_t^{c1\_best} \leftarrow$ select$(5, X_t^{c1})$
9   $X_t^{c2\_best} \leftarrow$ select$(5, X_t^{c2})$
10  $X_t^{best} \leftarrow X_t^{c1\_best} \bigcup X_t^{c2\_best}$
11  $\mathbf{x}_{gbest}, X_t^{best} \leftarrow$ QPSO($X_t^{best}, 20$) using (6.4,6.6)
12  **foreach** $\mathbf{x}_1 \in X_t^{c1\_best} \bigcup X_t^{c2\_best} \bigcup X_t^{best}$ **do** with $\mathbf{x}_1$ substitute corresp. $\mathbf{x}_2 \in X_t$
13  return $\mathbf{x}_{gbest}, X_t$
```

The probabilistic observation model and objective function in the Q-PSO are based on matching among the rendered object with the segmented object. The matching is calculated using object silhouette and distance transform-based edge scores. The fitness score is based on the ratio of overlap between the segmented object and the rasterized 3D model in the hypothesized pose. The overlap ratio is the sum of the overlap degree from the object shape to the rasterized model and the overlap degree from the rasterized model to the object shape. The larger the overlap ratio is, the larger is the fitness value. The fitness score is calculated as follows:

$$f = \left(0.5 * \frac{P_{outside}}{P_{model}} + 0.5 * \frac{P_{empty}}{P_{seg}}\right)^{w_1} * \left(\frac{K}{P_K}\right)^{w_2} \tag{6.10}$$

where $P_{outside}$ stands for the number of pixels projected from the model that are outside of the segmented object on the image, P_{model} denotes the number of pixels in the projected model, P_{empty} is the number of pixels of the segmented object on the image that are not covered by the projected model, P_{seg} stands for the number of pixels from the segmented object, K is a sum of signed distance transform values from projected model's edges to object edges on the image (which were generated using segmentation), P_K denotes the number of edge pixels in the projected object outline, and $w_1 = 0.4$, $w_2 = 0.6$ are exponents that were determined experimentally.

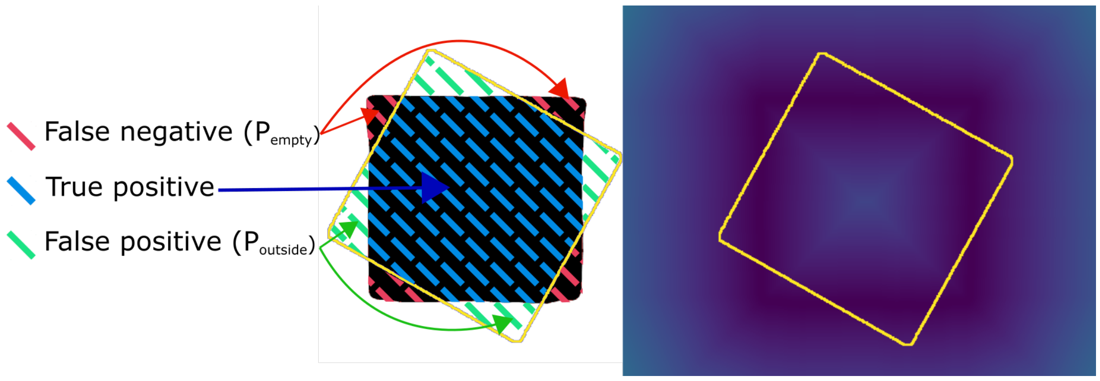


Figure 6.5: Illustration of the fitness function of the proposed method.

6.7. Experiments and results

At the beginning, computer simulations are discussed. Then proposed algorithm is evaluated on the freely available OPT benchmark dataset and the proposed dataset. The timing details for the execution of distinct components within the proposed method are presented in Subsection 6.7.4.

6.7.1. Unit quaternion-based motion model

The Q-PF-PSO presented above has been evaluated in various simulation experiments. Figure 6.6 depicts sample simulation results. In the discussed experiment a hypothetical situation is simulated with a rotating object about one of its axis. This object rotates according to a probabilistic motion model. The object orientations are represented by red dots. The second image depicts the probability density, which was estimated using Kernel Density Estimation (KDE) [125] after the prediction of the samples in the Q-PF, whereas the third one depicts the probability density after resampling. In the next images, the densities in the 1st, 2nd, and 3rd iterations of Q-PSO are shown. In the discussed experiment, the measurements were contaminated by Gaussian random noise of zero value and a given covariance matrix. For visualization purposes, relatively large standard deviations were utilized both in probabilistic motion and observation model. From left to right: ground-truth marked as red dots, predicted distribution, distribution after resampling, distribution after 1-st, after 2-nd, and after 3-rd iteration (best viewed in color).

The proposed Q-PF, Q-PSO, and Q-PF-PSO supported by k-means++ are general algorithms and can be used in a wide spectrum of applications. Figure 6.7 depicts sample plots that were obtained during the tuning of Q-PF-PSO

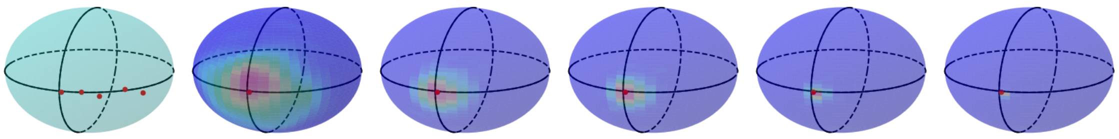


Figure 6.6: Simulation of Q-PF-PSO on toy data.

for tracking the rotation of a real object. Prior distribution, distribution after resampling, clusters determined by k-means++, locations of particles after execution of PSO on two sub-swarms.

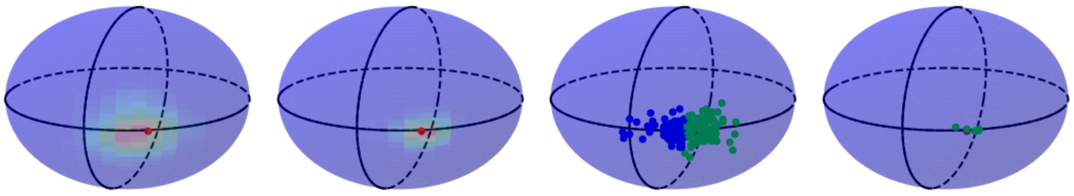


Figure 6.7: Simulation tests of Q-PF-PSO on real data.

6.7.2. Experimental results on OPT dataset

All tracking scores presented below are averages of three independent runs of the algorithm with unlike initializations. Table 6.1 presents the tracking scores that were achieved on the OPT dataset in the FreeMotion scenario with the use as well as with no use of pose predictions by the Siamese neural network. As it can be observed, considerable gains in the ADD scores can obtained thanks to Siemese neural network-based pose predictions. The discussed results were achieved using a common Siamese neural network that has been trained for all objects.

Table 6.1: ADD scores [%] achieved by our algorithm on OPT dataset in tracking 6D pose of House and Ironman in FreeMotion scenario (nS - no Siamese).

view, ADD	House (nS)	House	Ironman (nS)	Ironman
Behind, 10%	85 ± 2.07	82 ± 2.53	64 ± 2.29	77 ± 3.06
Behind, 20%	99 ± 0.51	97 ± 1.68	91 ± 1.62	95 ± 2.06
Left, 10%	81 ± 1.52	76 ± 2.58	35 ± 5.56	42 ± 3.37
Left, 20%	97 ± 1.34	98 ± 1.34	66 ± 6.77	69 ± 3.82
Right, 10%	55 ± 5.13	75 ± 2.02	46 ± 5.16	56 ± 4.34
Right, 20%	74 ± 7.20	96 ± 1.80	73 ± 6.73	79 ± 5.04
Front, 10%	53 ± 3.66	82 ± 1.14	55 ± 6.02	68 ± 3.45
Front, 20%	81 ± 5.40	97 ± 1.1	73 ± 5.94	83 ± 3.29
Average, 10%	68	79	50	61
Average, 20%	88	97	76	82

Afterward, the results achieved by the proposed algorithm with a neural network trained for all objects were compared with results that were obtained by a neural network trained for each object. Table 6.2 presents the tracking scores achieved in 6D pose tracking of House, Ironman, and Jet objects in the FreeMotion scenario.

Although the tracking scores achieved by the algorithm with a Siamese neural network trained for all objects are slightly worse than those achieved with the use of a separate network for each object, the difference between tracking scores is not high.

Table 6.2: ADD scores [%] achieved by the proposed algorithm in tracking pose 6D of House, Ironman, and Jet in FreeMotion scenario from the OPT dataset.

	Siamese for all			Siamese for each object		
tracking score	House	Ironman	Jet	House	Ironman	Jet
Behind, ADD 10%	82	77	74	85	73	80
Behind, ADD 20%	97	95	86	99	93	90
Left, ADD 10%	76	42	39	80	48	33
Left, ADD 20%	98	69	66	97	68	55
Right, ADD 10%	75	56	60	78	58	61
Right, ADD 20%	96	79	84	96	79	86
Front, ADD 10%	82	68	61	82	69	68
Front, ADD 20%	97	83	86	98	87	89
Average, ADD 10%	79	61	59	81	62	60
Average, ADD 20%	97	82	80	97	82	80

Table 6.3 contains AUC scores achieved by recent methods in 6D pose tracking of six objects in the FreeMotion scenario. As can be observed, the proposed algorithm achieves better results in comparison to results achieved by PWP3D [131], UDP [17], and ElasticFusion [181]. In comparison to results attained by algorithm [163] the proposed method outperforms it in many cases. However, the discussed method delivers a single guess of each object's pose, whereas the proposed method delivers the best poses together with probability distributions.

Table 6.3: AUC scores [%] achieved by in tracking 6D pose of House, Ironman and Jet in FreeMotion scenario.

AUC score	scenario	House	Ironman	Jet	Bike	Chest	Soda	Avg.
PWP3D [131]	all test	3.58	3.92	5.81	5.36	5.55	5.87	5.02
UDP [17]	all test	5.97	5.25	2.34	6.10	6.79	8.49	5.82
ElasticFusion [181]	all test	2.70	1.69	1.86	1.57	1.53	1.90	1.88
Reg. G-N. [163]	all test	10.15	11.99	13.22	11.90	11.76	8.86	11.31
w/o Siamese	FreeM.	11.52	9.26	9.19	10.81	6.93	6.61	9.05
Siamese for each	FreeM.	**13.68**	10.59	10.37	**12.36**	7.80	8.60	10.56
Siamese for all	FreeM.	13.27	10.32	10.33	11.88	7.60	**8.90**	10.38

6.7.3. Experimental results on custom dataset

Afterward, experiments on custom dataset [97] were conducted, c.f. Subsection 2.4.3. For evaluation of 6D pose tracking the sequences #2 was selected in which every object moves from left to right, simultaneously makes rotations from 0 to 180°, and after reaching the right side of the image, the object moves back from right to left, simultaneously makes full rotation about its axis. The training of neural networks both for object segmentation/detection and 6D pose estimation was done on synthetic images only. Table 6.4 presents experimental results that were obtained by the proposed algorithm.

Table 6.4: ADD scores [%] achieved by the proposed algorithm with and without Siamese.

tracking score	0°, ADD 10%	0°, ADD 20%	30°, ADD 10%	30°, ADD 20%	60°, ADD 10%	60°, ADD 20%	90°, ADD 10%	90°, ADD 20%	Avg., ADD 10%	Avg, ADD 20%
drill w/o Siam.	88	98	88	98	68	93	77	98	80	97
drill with Siam. sep.	92	98	87	100	70	93	83	93	83	96
drill with Siam. com.	90	98	90	99	78	93	77	84	84	94
frog w/o Siam.	71	82	87	98	66	79	38	57	65	79
frog with Siam. sep.	81	87	75	81	83	93	66	82	76	86
frog with Siam. com.	81	87	81	89	76	87	63	88	75	88
pig w/o Siam.	45	74	52	73	63	83	85	93	61	81
pig with Siam. sep.	53	73	61	70	64	79	93	100	68	80
pig with Siam. com.	53	73	56	83	64	79	91	93	66	82
duck w/o Siam.	52	94	78	86	73	79	86	100	72	90
duck with Siam. sep.	55	88	79	88	75	84	93	100	75	90
duck with Siam. com.	53	79	84	91	77	90	93	100	77	90
ext. w/o Siam.	23	35	58	71	62	75	75	98	54	70
ext. with Siam. sep.	37	50	58	77	70	83	86	97	63	76
ext. with Siam. com.	40	49	58	68	70	83	81	97	62	74
mult. w/o Siam.	15	26	75	93	85	96	94	100	67	79
mult. with Siam. sep.	17	26	76	96	84	98	98	99	69	80
mult. with Siam. com.	15	28	78	98	84	98	93	100	68	81

Figure 6.8 depicts ADD scores over time that were obtained by the Q-PF-PSO algorithm. As can be observed, slightly bigger errors were achieved for 0° camera view for the extension and the multimeter. The larger errors are due to similar appearances of the objects from the side view. For 10% ADD the pose predictions achieved by the Siamese neural network lead to considerable improvements in tracking scores. Although the tracking scores achieved by the algorithm with the Siamese neural network trained for all objects are smaller, the drop in the performance is relatively small. The advantages of using a single neural network for all objects are considerable. Moreover, this in turn opens several new research possibilities.

Figure 6.9 depicts rotation errors over time on real images, which have been obtained by the Q-PF-PSO with a neural network predicting the object pose over time. As can be observed, the largest errors were obtained for the extension, which is a symmetric and texture-less object. Somewhat larger errors were observed for 30° camera

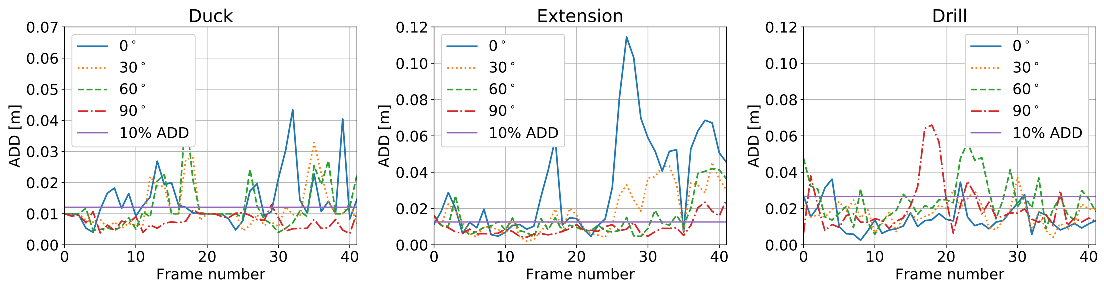


Figure 6.8: ADD scores over time for images from sequence #2, obtained by Q-PF-PSO with Siamese neural network predicting object pose.

view. The experimental results presented above were achieved on the basis of eight fiducial points, which were determined by a neural network, trained separately for each object. This could indicate the limited potential of the proposed approach. However, it is worth noting that the proposed approach is universal as it can be used for points that are discovered in semi-supervised or unsupervised learning.

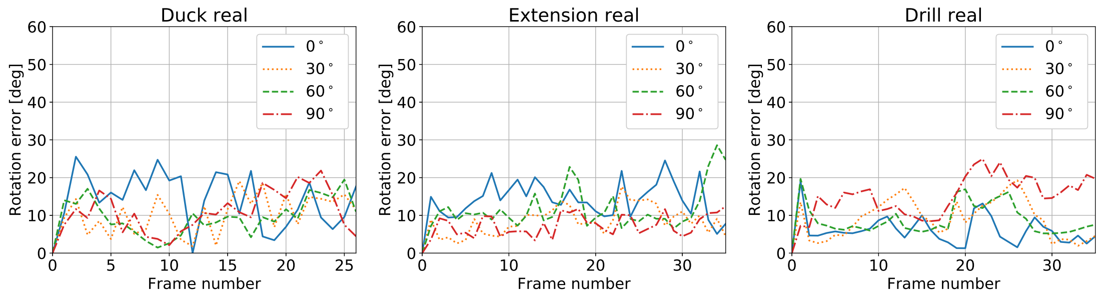


Figure 6.9: Rotation errors over time for real images, obtained by Q-PF-PSO using object poses that were predicted by the Siamese neural network.

6.7.4. Run times

The complete system for 6D pose estimation has been evaluated on the Nvidia Jetson AGX Xavier board. The Jetson AGX Xavier has a 512-core Volta GPU with Tensor Cores and it offers more than 20x the performance and 10x the energy efficiency of the Nvidia Jetson TX2. Running times that were compared with times obtained on a PC equipped with AMD Ryzen 7 2700, GeForce RTX 2060 are presented in Table 6.5. The Jetson AGX Xavier board was evaluated by setting to maximum performance mode (MAXN0). Jetpack 4.3, CUDA 10, and Tensorflow 2.1 were used. The weights of neural networks have been optimized with the TensorRT library (6.0) with default settings.

As it can be observed, thanks to optimized code the U-Net executes on the Jetson about two times faster. The speed-up for the Siamese neural network is even larger. The time needed for determination of the keypoints on the Jetson is somewhat larger in comparison to the time needed on the PC. The discussed operations are executed in 0.04 sec. on the Jetson, i.e. with about 25 Hz. Thanks to the implementation of the k-means++ in CUDA and executing it on the GPU, almost five times speed-up has been obtained in comparison to executing the k-means++ on NVIDIA Carmel ARMv8.2 CPU. The PSO is executed on the Jetson in almost two times larger time than on

Table 6.5: Running times on Jetson AGX Xavier [sec.].

	PC	Jetson
U-Net	0.040	0.020
Keypoints	0.035	0.040
Siamese	0.030	0.007
k-means++	0.005	0.010
PSO 200 p. 3 iter.	0.037	0.040
PSO 10 p. 10 iter.	0.026	0.023
overheads	0.017	0.023
Total	0.190	0.200

the PC due to the current unoptimized GPU implementation. The Jetson-based implementation of the algorithm runs at about 5 Hz, i.e. at about the same speed as on the PC. The most computationally demanding operation is matching between the projected 3D model and image observations, supported by an edge distance transform. The functions mentioned above are implemented using CUDA parallel programming environment. They can be further speed up with more advanced parallel processing such as CUDA-OpenGL interoperability, which has been shown to render one thousand images in 100 milliseconds [155].

6.8. Extended version of the method

In this section, experimental results are presented that were achieved on different versions of the algorithm. The U-Net and neural network for estimating keypoints were replaced with one neural network presented in Figure 6.10. In this version of the algorithm, the Siamese neural network was not used.

This neural network is a simplified version of X-Net presented in Chapter 5. It was reduced by the quaternion branch and convolution layers. The used neural network operates on RGB images of size 128×128, which contain the cropped object of interest. It is a two-output neural network, as X-Net, that on the first output returns the segmented object, whereas on the second output, it returns blobs representing the keypoints. The segmented object is represented by a mask on a single gray image, whereas each keypoint is represented on a separate gray image. The output of the first layer is fed to a pyramid pooling module, which is followed by a max pooling layer with 2×2 filters. The next block consists of two layers denoted as Block, which are followed by a max pooling layer with 2×2 filters. The first layer contains Conv2D with 1×1 filters, which are followed by BN layer. The output is fed to two parallel branches with Conv2D and filters of size 1×1 and 3×3, respectively, whose outputs are concatenated. The output of the next layer consisting of 128 Conv2D filters are fed to two branches that are responsible for object segmentation and keypoints estimation. The branch responsible for object segmentation contains 128 Conv2D filters of size 1×1 filters, which are followed by Conv2D with 4×4 filters. Similarly to the branch mentioned above, the second branch contains 128 Conv2D filters of size 1×1, which are followed by Conv2D with 4×4 filters.

The parameters of PF were without changes. The number of iterations for the first part of PSO was 15 and for the last part was 50. This version with additional iterations on PSO in the further part of the study is called "0.4 s.".

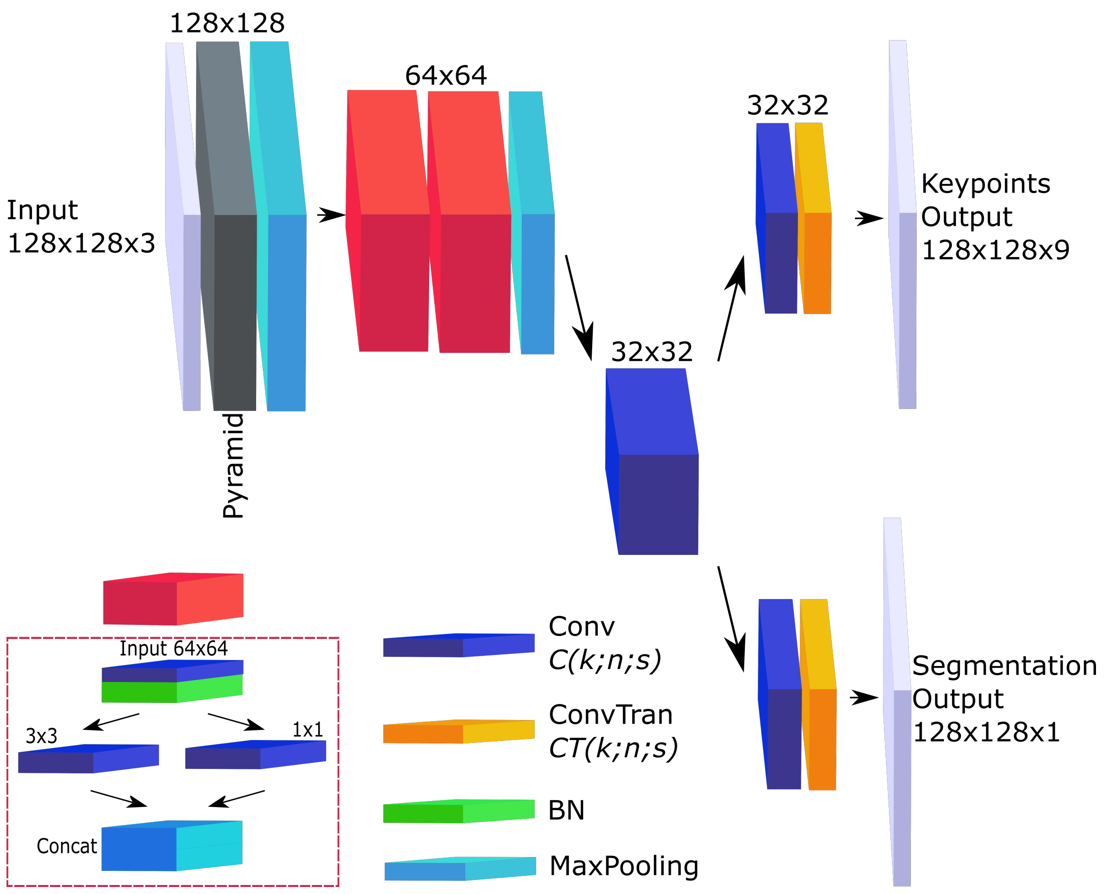


Figure 6.10: Architecture of proposed neural network.

Additionally, a subset of particles for final pose optimization is found in a multi-criteria analysis using the TOPSIS algorithm. The best particle of the swarm is selected using TOPSIS operating on:

- overlap degree between the rasterized 3D model of the object in the hypothesized pose and the segmented object.
- value of distance transform between the model's projected edges and the closest image edges.
- values of heatmaps on projected 3D keypoints of the model in hypothesized object pose.

Figure 6.11 illustrates the proposed approach. The fitness score was determined using not only the object segmentation and projecting 3D model but also using heatmaps representing the position of keypoints. This version of the method was published in [105].

6.8.1. Experimental results on OPT dataset

In this Section, experimental results are presented that were achieved on the OPT dataset. Table 6.6 contains AUC scores achieved by recent methods in 6D pose tracking of objects in FreeMotion scenario from the OPT dataset. If the results for this particular scenario were not available the AUC scores in the evaluation called all tests are reported instead.

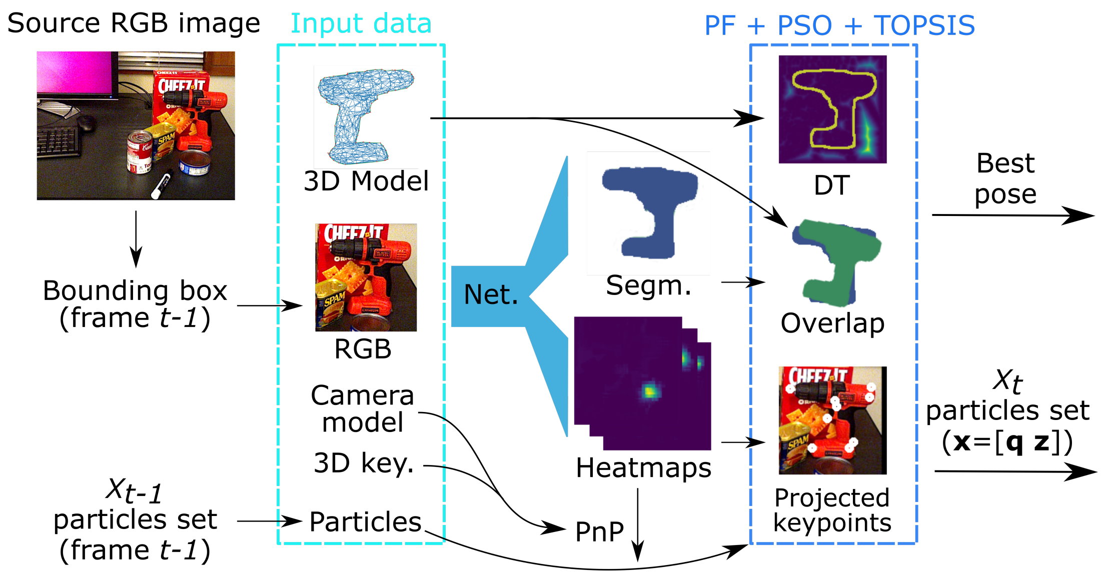


Figure 6.11: Proposed approach to 6D object pose tracking.

Table 6.6: AUC scores [%] achieved by in tracking 6D pose of House, Ironman and Jet in FreeMotion scenario.

AUC score	scenario	House	Ironman	Jet	Bike	Chest	Soda	Avg.
PWP3D [131]	all test	3.58	3.92	5.81	5.36	5.55	5.87	5.02
UDP [17]	all test	5.97	5.25	2.34	6.10	6.79	8.49	5.82
ElasticFusion [181]	all test	2.70	1.69	1.86	1.57	1.53	1.90	1.88
Reg. G-N. [163]	all test	10.15	11.99	13.22	11.90	11.76	8.86	11.31
CE&K [21]	all test	14.48	**14.71**	**17.17**	12.55	**14.97**	14.85	14.79
CE&K [21]	FreeM.	-	-	-	-	-	-	13.91
ph. enh. edge [168]	FreeM.	13.70	10.86	-	-	9.77	-	11.44
Siamese for all [98]	FreeM.	13.27	10.32	10.33	11.88	7.60	8.90	10.39
Propose method (0.2 s.)	FreeM.	14.13	12.49	12.59	12.38	11.98	12.34	12.65
Propose method (0.4 s.)	FreeM.	**15.82**	14.10	15.52	**14.45**	12.39	**15.16**	**14.57**

As can be observed, the proposed method achieves better results in comparison to results achieved by PWP3D [131], UDP [17], ElasticFusion [181], Reg. G-N. [163], and a recently proposed edge-based algorithm [168]. In the FreeMotion scenario it achieves better average results than the edge-based method [21]. It is worth noting that the discussed version of the proposed method is faster than the algorithm [21], which requires 680 ms on average to process a single frame. Experimental results achieved by the proposed method are averages from three runs of the algorithm with, unlike initializations.

6.8.2. Experimental results on YCB-Video dataset

In this section, experimental results are presented that were achieved on the YCB-Video dataset. Table 6.7 presents the AUC ADD scores achieved by the proposed algorithm for pose tracking on the YCB-Video dataset in comparison against scores achieved by SOTA algorithms. AUC ADD is calculated for non-symmetric objects, whereas AUC ADD-S is determined for symmetric objects, - denotes unavailable results, and '*' stands for symmetric objects.

Table 6.7: AUC ADD scores [%] (max. th. 10 cm).

Object	Pose recovery on single RGB images					Pose tracking		
	PoseCNN [183]	DOPE [166]	DHROP [121]	[187]	GDR-Net [175]	PoseRBPF [36]	DeepIM [89]	Proposed method
002_master_chef_can	50.9	-	81.6	49.9	65.2	63.3	89.0	**92.4**
003_cracker_box	51.7	55.9	83.6	80.5	**88.8**	77.8	88.5	88.2
004_sugar_box	68.6	75.7	82.0	85.5	**95.0**	79.6	94.3	89.0
005_tomato_soup_can	66.0	76.1	79.7	68.5	**91.9**	73.0	89.1	81.5
006_mustard_bottle	79.9	81.9	91.4	87.0	**92.8**	84.7	92.0	90.3
007_tuna_fish_can	70.4	-	49.2	79.3	**94.2**	64.2	92.0	87.8
008_pudding_box	62.9	-	90.1	81.8	44.7	64.5	80.1	**90.8**
009_gelatin_box	75.2	-	**93.6**	89.4	92.5	83.0	92.0	93.1
010_potted_meat_can	59.6	39.4	79.0	59.6	80.2	51.8	78.0	**82.9**
011_banana	72.3	-	51.9	36.5	**85.8**	18.4	81.0	80.8
019_pitcher_base	52.5	-	69.4	78.1	**98.5**	63.7	90.4	92.7
021_bleach_cleanser	50.5	-	76.1	56.7	**84.3**	60.5	81.7	83.6
024_bowl*	69.7	-	76.9	23.5	85.7	85.6	**90.6**	89.8
025_mug	57.7	-	53.7	54.0	**94.0**	77.9	83.2	90.3
035_power_drill	55.1	-	82.7	82.8	**90.1**	71.8	85.4	87.9
036_wood_block*	65.8	-	55.0	29.6	82.5	31.4	75.4	**84.0**
037_scissors	35.8	-	65.9	46.0	49.5	38.7	70.3	**74.2**
040_large_marker	58.0	-	56.4	9.8	76.1	67.1	**80.4**	80.1
051_large_clamp*	49.9	-	67.5	47.4	89.3	59.3	84.1	**89.4**
052_extra_large_clamp*	47.0	-	53.9	47.0	**93.5**	44.3	90.3	92.2
061_foam_brick*	87.8	-	89.0	87.8	**96.9**	92.6	95.5	90.9
Avg.	61.3	65.8	72.8	61.0	84.4	64.4	85.9	**87.2**

For six objects out of twenty-one objects, the proposed algorithm achieves the best results. It is worth noticing that the proposed algorithm has the best average result for all objects. As it can be observed, owing to using the PF and PSO for maintaining the probability of pose the results are significantly better. The discussed results were achieved by a version of the algorithm, in which the time of estimation is 0.4 sec. due to a larger number of calls of the objective function.

The experimental results presented in Table 6.8 demonstrate the effectiveness of the solutions introduced in this chapter. The proposed approach using the k-means++ algorithm to identify modes in multi-modal probability distributions outperforms both QPF-PSO and Q-PSO operating solely on the object keypoints, as reflected by significantly improved AUC ADD scores.

	Avg. AUC ADD
PnP	77.1
Q-PSO w/ keypoints	81.2
QPF-PSO w/o k-means++	81.9
QPF-PSO w/ k-means++	83.1
QPF-PSO w/ k-means++ and TOPSIS	87.2

Table 6.8: Average AUC ADD scores [%] (max. th. 10 cm) achieved by different versions of the proposed method on objects from the YCB-Video dataset.

6.9. Summary

This chapter was devoted to the proposed quaternion-based particle filter, particle swarm optimization, and hybrid quaternion particle filter with quaternion particle swarm optimization (Q-PF-PSO). It was demonstrated that the k-means++ algorithm which can maintain multimodal probability distributions, improves the object pose tracking. It was experimentally proven that the proposed Siamese neural network can deliver pose predictions over time, which considerably improves the performance of object tracking. The proposed algorithms are universal and can be used in a wide range of applications for object pose estimations. One of the advantages of the proposed approaches is that in contrast to recent approaches, the proposed algorithm delivers probability distribution of object poses instead of a single object pose guess. The proposed method has been studied and evaluated in more depth in terms of tracking 6D poses of the objects. The evaluations were done on the freely available OPT benchmark dataset, on the proposed dataset that contains both real and synthesized images of six texture-less objects, and on the YCB-Video dataset. On Nvidia Jetson AGX Xavier the neural networks for object segmentation, fiducial keypoints determination, and pose prediction are executed at 25 Hz, whereas the whole algorithm is running at about 5 Hz.

Chapter 7

Proposed Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) Object Pose Refinement

In this chapter, a pipeline for 6D object pose tracking on RGB images is proposed, wherein a fuzzy TOPSIS is employed. The proposed method calculates decision variables on features that are frequently utilized in relevant work including segmented masks, fiducial keypoints, and distance transform. Instead of relying solely on a single object pose prediction, it is desired to design a system that can recognize and acknowledge the inherent uncertainty in object pose estimation by actively considering multiple pose hypotheses. The decision variables are determined taking into account the object's non-occluded part and the object's occluded part. A two-stack hourglass network [185] to jointly estimate nine objects' keypoints and predict both the object's non-occluded part and the object's occluded part is proposed. The input RGB images are processed by an hourglass-based convolutional neural network that simultaneously performs three key tasks: identifying nine specific keypoints on objects, predicting the non-occluded portions of those objects, and estimating the occluded parts, see Figure 7.1. The loss function and training of the multi-task hourglass are discussed. The information about heatmaps, object segmentation, distance transform, and keypoints detected in the previous frame are fed to the TOPSIS algorithm. To verify proposed designs, thorough experiments on the challenging YCB-Video benchmark dataset are conducted. It is demonstrated experimentally that the algorithm achieves competitive results. The proposed TOPSIS-aided object pose tracking was published in [104].

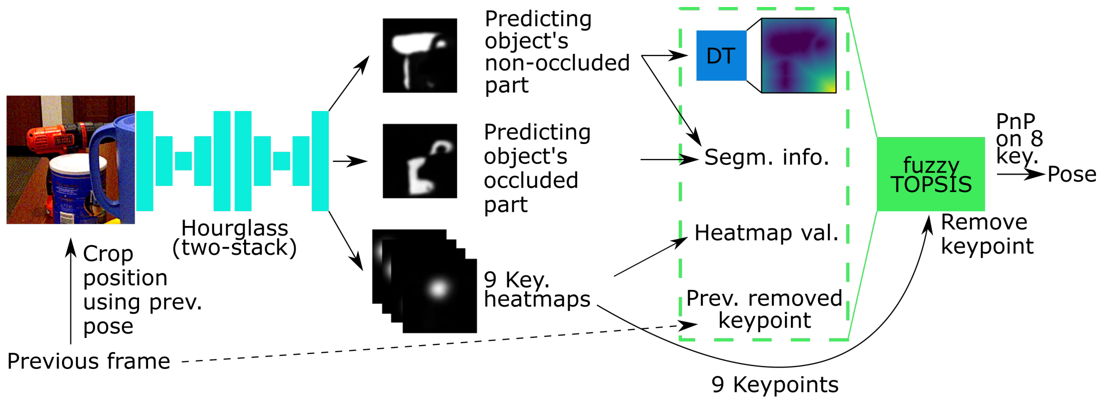


Figure 7.1: Overview of proposed TOPSIS aided algorithm for 6D object tracking in sequences of RGB images.

This chapter is organized in order of phases of the proposed method. First, the 2D location of the keypoints is determined by the hourglass neural network, which is described in Section 7.1. Then, in Section 7.2, the details of the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) are presented. Experimental results are presented and discussed in Section 7.4.

In this chapter, the following contributions to the object pose tracking are presented:

– pose optimization and refinement strategy that apart from keypoints uses also signed distance maps

– fuzzy TOPSIS algorithm that uses heatmaps, object segmentation, distance transform, and keypoints to remove the worst keypoint

7.1. Hourglass architecture

The architecture of the hourglass neural network is shown in Figure 7.2. It was designed to jointly detect nine object keypoints, predict the object's non-occluded part, and predict the object's occluded part. The neural network operates on RGB images of size 256×256 and delivers the object's non-occluded part and the object's occluded part on two separate output map channels. On nine output maps, the neural network generates blobs, whose centers represent 2D positions of the object keypoints. Each object keypoint is generated on a specific channel to provide 2D-3D correspondences. When training the network, the keypoints were represented by two-dimensional Gaussian functions with σ equal to five pixels and centered on the desired positions on the object.

The proposed neural network consists of two hourglass blocks, see Figure 7.2. Each basic block, see also read block on Figure 7.2, consists of two branches: one with 1×1, 3×3, and 1×1 convolutional blocks followed by BN, and the second one with a direct connection to calculate the residual. After the first part of basic blocks and max pooling, a residual block that is similar to the basic block except that instead of a direct connection a 1×1 convolution is utilized in order to calculate the residual. The first hourglass block is followed by a residual block with a 1×1 convolution followed by a 3×3 convolution in the first branch and a 1×1 convolution in the second one.

The proposed hourglass model executes multiple tasks, i.e. estimates keypoints, and determines both occluded and non-occluded parts of the object undergoing tracking. Heatmap regression and coordinate regression are two commonly used approaches in neural network-based landmark localization. Heatmap-based (also known as confidence map-based) methods aim at predicting heatmaps such that points with maximum values correspond to keypoints in the input image. In the proposed approach, the real keypoints are represented by Gaussian heatmaps, and the heatmap regression model was optimized with the pixel-wise MSE loss. The segmentation model was optimized using a Combo loss [157] that is a sum of binary cross-entropy (BCE) loss and Dice loss. Figure 7.3 details the process of training the proposed multi-task hourglass for joint object's keypoints estimation, object's non-occluded part segmentation, and object's occluded part segmentation. During the calculation of the loss, both feature maps determined by two-stack hourglass blocks were utilized.

The neural network has been trained in 1000 epochs with a batch size set to 16, using RMSprop with a lr set to $1e-4$. For each object in the YCB-Video dataset, a single neural network was trained. As already mentioned, this dataset consists of 92 video recordings, comprising a total of 133,827 frames. Specifically, 80 sets of RGB image sequences were used for training purposes, while 12 sets were reserved for evaluation and testing. The training subset

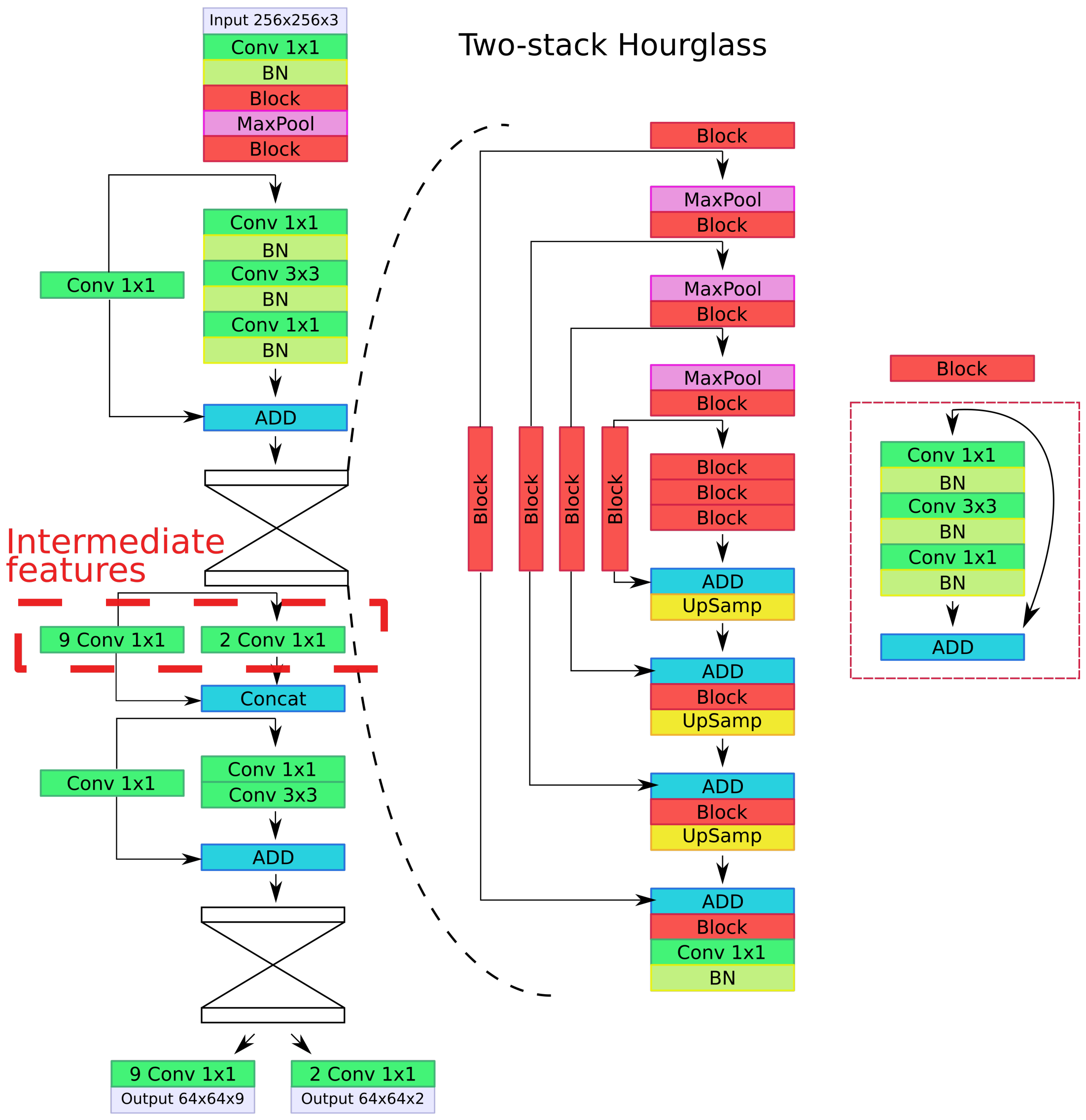


Figure 7.2: Two-stack hourglass neural network for jointly object keypoints detection, predicting the visible (not occluded) object part and predicting the occluded part of the object.

was expanded with 80k synthetically rendered images released in the dataset, similarly as in methods discussed in Chapters 4,5,6.

7.2. Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)

This thesis presents an implementation and evaluation of two variants of the TOPSIS-aided algorithms. The first variant employs crisp values, whereas the second variant incorporates fuzzy logic, utilizing fuzzy TOPSIS methodology to tackle decision-making problems. This section introduces the basic definitions and the procedure of the fuzzy TOPSIS method. Then, the proposed method is illustrated with numerical examples. The presented notation was based on publication [70] and only a single decision-maker is utilized in obtaining the appropriate decision. The decision-maker has to choose one of m possible alternatives a_i described by n criteria C_j measured

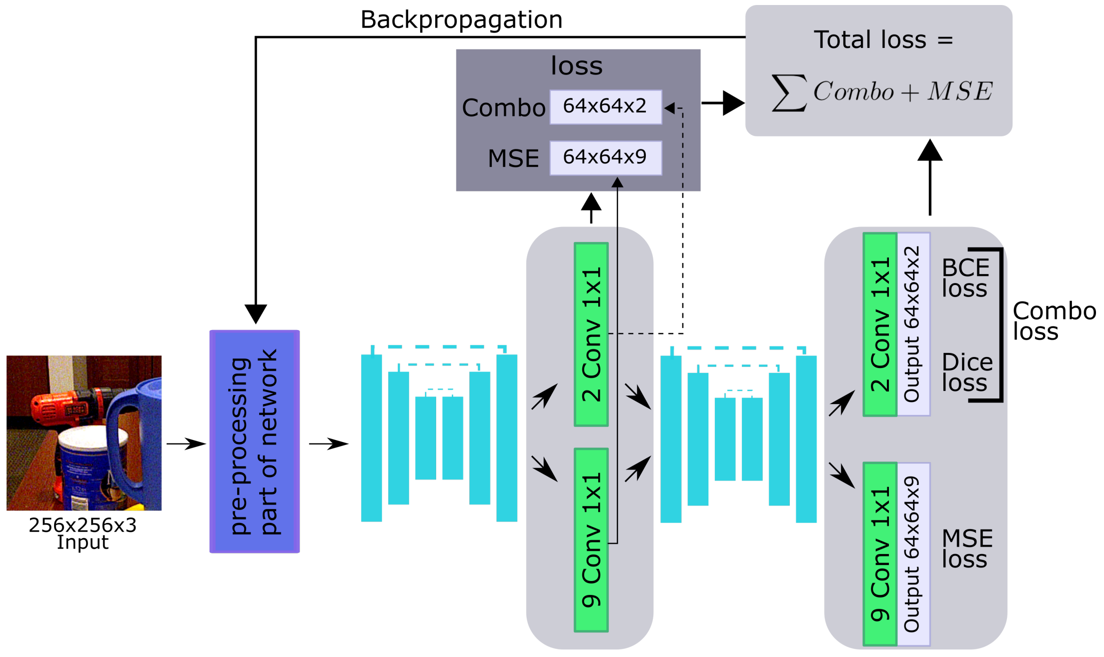


Figure 7.3: Details of training of multi-task hourglass for joint object's keypoint estimation and object's occluded/non-occluded part segmentation.

by triangle fuzzy numbers and linguistic values.

Let:

$$\tilde{w} = (\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_n) \tag{7.1}$$

be the vector of criteria weights, in which w_j is the weight of criterion C_j. This vector allows to emphasize on more important criteria and to reduce the influence of less important ones. The set of criteria is divided into two subsets: *BC* criteria (benefit criteria) whose greater value is better and *CC* criteria (cost criteria) whose lower value is better. A fuzzy multi-criteria problem can be concisely expressed in matrix form as:

$$X = \begin{array}{c c} & \begin{array}{cccc} C_1 & C_2 & & C_n \end{array} \\ \begin{array}{c} a_1 \\ a_2 \\ \\ a_m \end{array} & \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \ldots & \ldots & x_{ij} & \ldots \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix} \end{array} \tag{7.2}$$

The matrix consists of m possible alternatives a_i described by n criteria C_j. In fuzzy set theory, conversion scales are applied to transform linguistic terms into fuzzy numbers. The construction of a conversion scale is discussed for example by [9]. A linguistic variable is a variable whose value is given in linguistic terms [195]. The acquired linguistic values are then transformed according to the appropriate linguistic terms table into triangle fuzzy numbers $\tilde{e}_{ij} = (e^a_{ij}, e^b_{ij}, e^c_{ij})$ to get the processed decision matrix E.

$$E = \begin{array}{c} \\ a_1 \\ a_2 \\ \\ a_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & & C_n \end{array} \\ \begin{pmatrix} \tilde{e}_{11} & \tilde{e}_{12} & \dots & \tilde{e}_{1n} \\ \tilde{e}_{21} & \tilde{e}_{22} & \dots & \tilde{e}_{2n} \\ \dots & \dots & \tilde{e}_{ij} & \dots \\ \tilde{e}_{m1} & \tilde{e}_{m2} & \dots & \tilde{e}_{mn} \end{pmatrix} \end{array} \tag{7.3}$$

Then, a normalized decision matrix Z should be constructed using matrix E. The ranges of normalized triangular fuzzy numbers that belong to [0,1] are preserved by the linear normalization method. For the benefit criteria, normalization is expressed as follows:

$$\tilde{z}_{ij} = \left(\frac{e^a_{ij}}{e^+_j}, \frac{e^b_{ij}}{e^+_j}, \frac{e^c_{ij}}{e^+_j} \right), \text{where } e_{ij} \in BC, e^+_j = \max_i(e^c_{ij}) \tag{7.4}$$

While for the cost criteria, the normalization is performed as follows:

$$\tilde{z}_{ij} = \left(\frac{e^-_j}{e^a_{ij}}, \frac{e^-_j}{e^b_{ij}}, \frac{e^-_j}{e^v_{ij}} \right), \text{where } e_{ij} \in CC, e^-_j = \min_i(e^a_{ij}) \tag{7.5}$$

where BC and CC denote beneficial and cost criteria, respectively. Thus, the normalized decision matrix Z is created:

$$Z = \begin{array}{c} \\ a_1 \\ a_2 \\ \\ a_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & & C_n \end{array} \\ \begin{pmatrix} \tilde{z}_{11} & \tilde{z}_{12} & \dots & \tilde{z}_{1n} \\ \tilde{z}_{21} & \tilde{z}_{22} & \dots & \tilde{z}_{2n} \\ \dots & \dots & \tilde{z}_{ij} & \dots \\ \tilde{z}_{m1} & \tilde{z}_{m2} & \dots & \tilde{z}_{mn} \end{pmatrix} \end{array} \tag{7.6}$$

The weighted normalized fuzzy decision matrix V, consisting of $\tilde{v}_{ij}$, is computed by multiplying the normalized matrix Z with the criteria weight, see Equation 7.1, as follows:

$$\tilde{v}_{ij} = \tilde{z}_{ij} \times \tilde{w}_j \tag{7.7}$$

where $\times$ is a fuzzy product [71], $\tilde{w}_j$ is the fuzzy importance weight for criterion j and in a scenario involving more than one decision-maker in categorizing the degree of importance of criteria, it is referred to as a combined group criteria weight [43]. Due to the use of a single decision-maker in the decision-making process, the values of criteria weights are not aggregated from many decision-makers, as it is conventionally done, but are chosen by one single decision-maker from the linguistic terms table.

The positive ideal solution A^+ and the negative ideal solution A^- can be attained from the weighted normalized decision matrix V. The selection of A^+ and A^- is performed as follows:

$$A^+ = (\tilde{v}^+_1, \tilde{v}^+_2, ..., \tilde{v}^+_n), where\ \tilde{v}^+_j = \max_i(\tilde{v}_{ij}) \tag{7.8}$$

$$A^- = (\tilde{v}_1^-, \tilde{v}_2^-, ..., \tilde{v}_n^-), where\ \tilde{v}_j^- = \min_i(\tilde{v}_{ij}) \tag{7.9}$$

Applying the vertex method, the distances between each alternative a_m, acquired by Equation 7.7, and A^+ can be determined. Since $\tilde{v}_{ij} = (v_{ij}^a, v_{ij}^b, v_{ij}^c)$ and $\tilde{v}_j^+$ from Equation 7.8 can be described as $\tilde{v}_j^+ = (v_j^{a+}, v_j^{b+}, v_j^{c+})$, the distance can be expressed as follows:

$$d(\tilde{v}_{ij}, \tilde{v}_j^+) = \sqrt{\frac{1}{3}[(v_{ij}^a - v_j^{a+})^2 + (v_{ij}^b - v_j^{b+})^2 + (v_{ij}^c - v_j^{c+})^2]} \tag{7.10}$$

Similarly, the distances between each alternative and A^- can be expressed as follows:

$$d(\tilde{v}_{ij}, \tilde{v}_j^-) = \sqrt{\frac{1}{3}[(v_{ij}^a - v_j^{a-})^2 + (v_{ij}^b - v_j^{b-})^2 + (v_{ij}^c - v_j^{c-})^2]} \tag{7.11}$$

Therefore, the distance of each alternative to A^+ and A^- can be determined respectively as follows:

$$d^+(a_i) = \sum_{j=1}^{n} d(\tilde{v}_{ij}, \tilde{v}_j^+),\ i = 1, 2, .., m; j = 1, 2.., n \tag{7.12}$$

$$d^-(a_i) = \sum_{j=1}^{n} d(\tilde{v}_{ij}, \tilde{v}_j^-),\ i = 1, 2, .., m; j = 1, 2.., n \tag{7.13}$$

The closeness degree of each alternative a_i is defined as:

$$D_i = \frac{d^-(a_i)}{d^+(a_i) + d^-(a_i)} \tag{7.14}$$

The closeness determines the ranking order and indicates the best and worst of all alternatives.

7.3. Proposed method

The hourglass network detects nine key points of the object regardless of its pose, including poses in which some keypoints are on the invisible side of the object or in the case of partial occlusions of the object. The values of the heatmaps representing locations of the object keypoints are fed to the TOPSIS algorithm. Aside from the information on the estimated object keypoints the proposed TOPSIS-aided algorithms employ also information about object segmentation, distance transform, and keypoints detected in the previous frame. This way it can utilize temporal consistency among video frames. Proposed TOPSIS-aided algorithms, which were designed to cope with imperfect and uncertain attributes remove the worst keypoint out of nine object keypoints by ranking various alternatives. Finally, the object pose is estimated by the PnP algorithm in combination with RANSAC-based scheme. Information about the object location in the image is utilized to crop the object in the next input frame, see Figure 7.1.

In this work, the proposed TOPSIS-based algorithms operate on four decision variables. Figure 7.4 illustrates the main steps in determining decision variables. The decision variables that are determined for each object keypoint are as follows:

– Value of projected 3D keypoint on the corresponding heat map, which is determined by the hourglass neural network. The higher the value is, the better.

– Value depending on the type of the object segmentation on which the keypoint is located. If the keypoint is located on the image background the variable assumes a big value, if it is located on the occluded segment of the object the variable assumes a middle value, whereas in case it is located on the visible segment of the object the variable assumes a small value. The smaller the value is, the better.

– Value depending on the distance transform at which the keypoint is located to the closest visible pixel. The smaller the value of the distance transform at the location of the keypoint, the better.

– Value depending on whether the keypoint was omitted in the previous frame or was used by the PnP to determine the pose. If the keypoint was not omitted the variable assumes a higher value, otherwise the variable assumes a smaller value. The higher the value is, the better.

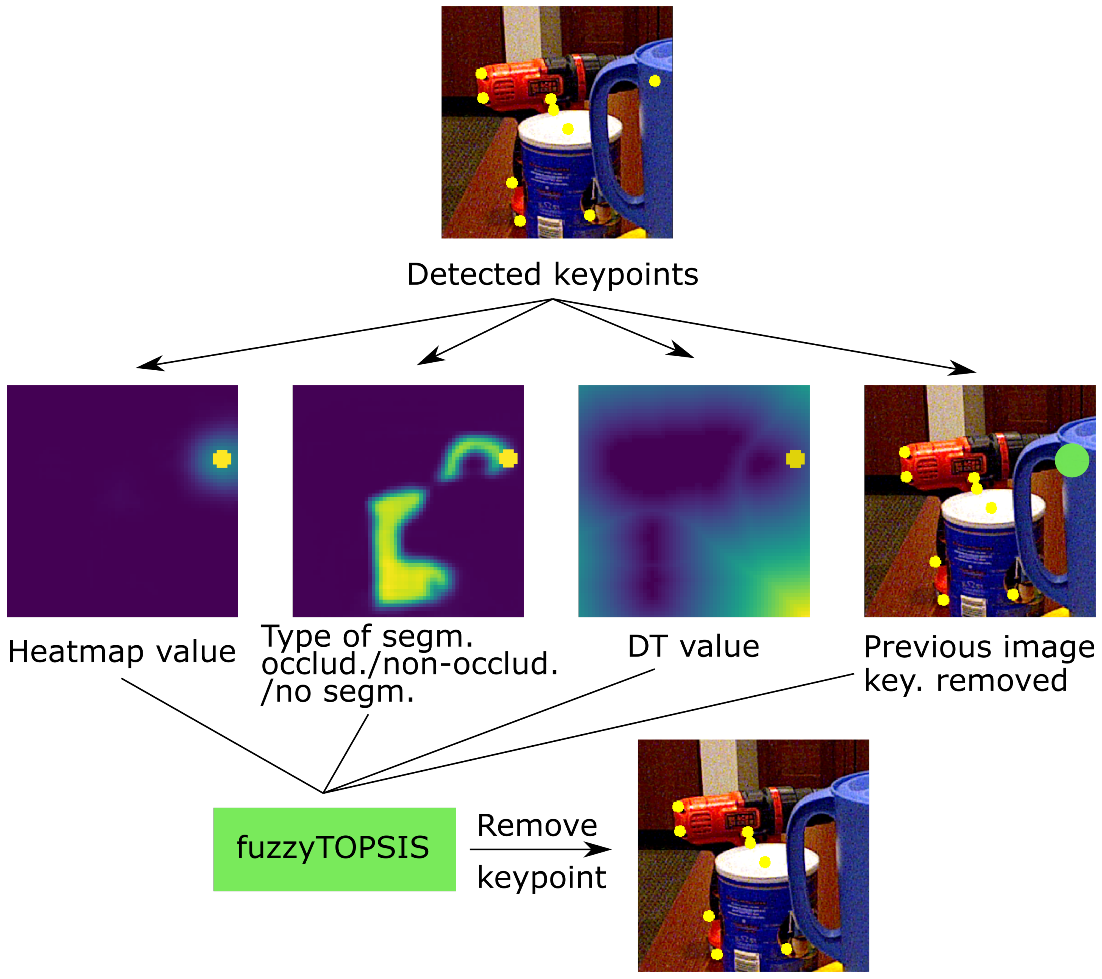

Detected keypoints
Heatmap value
Type of segm. occlud./non-occlud. /no segm.
DT value
Previous image key. removed
fuzzyTOPSIS
Remove keypoint

Figure 7.4: Scheme of decision variables used in TOPSIS.

7.3.1. Numerical example

Below a numerical example using toy data prepared using the YCB-Video dataset is presented. Figure 7.5 presents a toy example in which a fuzzy TOPSIS algorithm is executed in order to find the worst keypoint to remove for the pose tracking in frame t. The keypoint, which was removed in frame $t-1$ is indicated in red as prev. rem. To determine the worst keypoint, first, a fuzzy decision matrix with fuzzy weights needs to be prepared, see Equation 7.2. Weights, see Equation 7.1, indicate which criteria should have a more significant impact on the final decision.

The matrix consists of m possible alternatives A_m described by n criteria C_n. The alternatives represent each keypoints, therefore, $m = 9$. The $n = 4$, because four values were acquired from the neural network and previous frames.

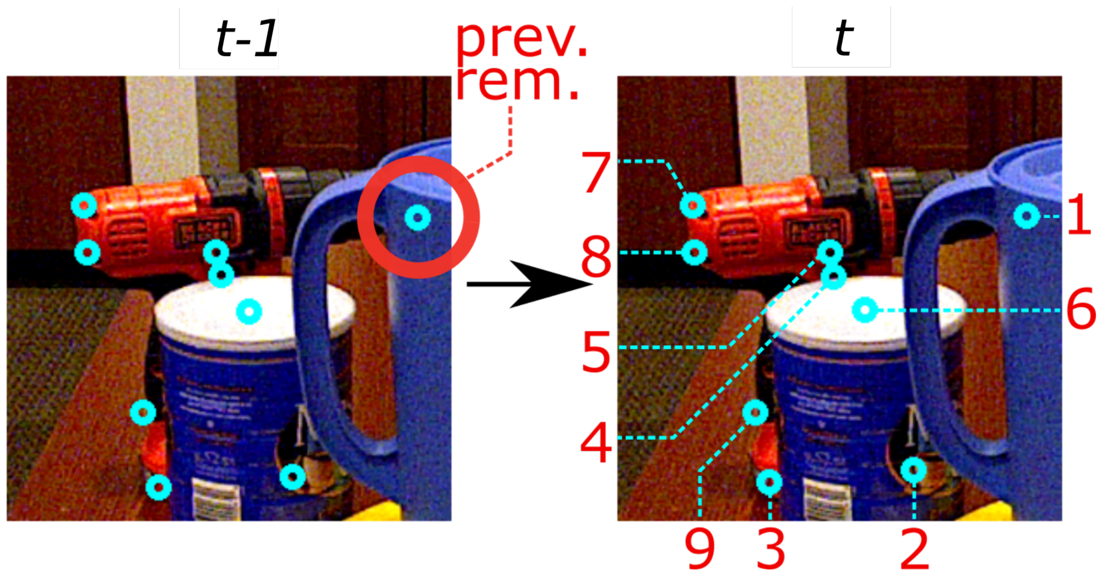


Figure 7.5: The toy example in which a fuzzy TOPSIS algorithm is executed to find the worst keypoint.

The decision variables, which are shown in Table 7.1, are the values of the heatmap for projected 3D keypoints, the type of segmentation on which the keypoint is located, the value depending on the distance transform, and a value depending on whether the keypoint was omitted in the previous frame. The C_1 and C_4 criteria belong to the BC subset whereas the C_2 and C_3 criteria belong to the CC subset. It means that the first two are expected to have a high value and the second two a low value. The C_2 and C_4 criteria are in a linguistic form.

Table 7.1: The values of decision variables for each keypoint acquired from hourglass and previous frame. Each keypoint is one of the alternatives A.

	C_1 : heatmap	C_2 : segm.	C_3 : DT	C_4 : prev. rem.
a_1	239	occluded	6	yes
a_2	207	occluded	2	no
a_3	251	occluded	1	no
a_4	226	occluded	1	no
a_5	242	non-occl.	1	no
a_6	248	occluded	5	no
a_7	251	non-occl.	1	no
a_8	253	non-occl.	1	no
a_9	239	non-occl.	1	no

To calculate fuzzy weights, see Equation 7.1, due to only one decision-maker, linguistic variables are chosen from the linguistic terms table, see Table 7.2, and then changed into corresponding fuzzy triangular numbers.

Table 7.2: Linguistic variables for the relative importance weights of criteria.

Linguistic variable	Fuzzy triangular number
Of little importance	(0.0, 0.0, 0.2)
Moderately important	(0.05, 0.25, 0.45)
Important	(0.3, 0.5, 0.7)
Very important	(0.55, 0.75, 0.95)
Absolutely important	(0.8, 1.0, 1.0)

The decision variables shown in Table 7.1 are converted into fuzzy numbers to construct the fuzzy decision matrix, see Table 7.5. The corresponding fuzzy triangular numbers for the segmentation linguistic variable and previously removed keypoints linguistic variable are shown in Table 7.3 and Table 7.4, respectively.

Table 7.3: Linguistic variables for the segmentation criteria.

Linguistic variable	Fuzzy triangular number
Non-occluded	(1, 1, 3)
Occluded	(1, 2, 3)
Background	(1, 3, 3)

Table 7.4: Linguistic variables for the previously removed keypoint criteria.

Linguistic variable	Fuzzy triangular number
Yes	(1, 1, 2)
No	(1, 2, 2)

The fuzzy triangular numbers for C_1 and C_3 are generated using information about the maximal and minimal values from the hourglass, which are different for each a_i due to separate channels for heatmaps. For each frame, the values vary slightly. The acquired fuzzy numbers for criteria weights and criteria for each alternative are shown in Table 7.5. The (normalized) fuzzy decision matrix is constructed using Equation 7.4 and 7.5, see Table 7.6.

The aim of the normalization method mentioned above is to preserve the property that the ranges of normalized triangular fuzzy numbers belong to [0,1]. Considering the different importance of each criterion, the weighted normalized fuzzy decision matrix is constructed using Equation 7.7, see Table 7.7.

Table 7.5: The fuzzy decision matrix and fuzzy weights.

	C_1	C_2	C_3	C_4
w	(0.05, 0.25, 0.45)	(0.55, 0.75, 0.95)	(0.05, 0.25, 0.45)	(0.3, 0.5, 0.7)
a_1	(178, 239, 239)	(1, 2, 3)	(1, 6, 21)	(1, 1, 2)
a_2	(160, 207, 207)	(1, 2, 3)	(1, 2, 21)	(1, 2, 2)
a_3	(200, 251, 251)	(1, 2, 3)	(1, 1, 21)	(1, 2, 2)
a_4	(169, 226, 232)	(1, 2, 3)	(1, 1, 21)	(1, 2, 2)
a_5	(191, 242, 243)	(1, 1, 3)	(1, 1, 21)	(1, 2, 2)
a_6	(182, 248, 251)	(1, 2, 3)	(1, 5, 21)	(1, 2, 2)
a_7	(189, 251, 253)	(1, 1, 3)	(1, 1, 21)	(1, 2, 2)
a_8	(176, 253, 253)	(1, 1, 3)	(1, 1, 21)	(1, 2, 2)
a_9	(177, 239, 241)	(1, 1, 3)	(1, 1, 21)	(1, 2, 2)

Table 7.6: The normalized fuzzy decision matrix.

	C_1	C_2	C_3	C_4
a_1	(0.7, 0.9, 0.9)	(0.3, 0.5, 1.0)	(0.04, 0.16, 1.0)	(0.5, 0.5, 1.0)
a_2	(0.6, 0.8, 0.8)	(0.3, 0.5, 1.0)	(0.04, 0.5, 1.0)	(0.5, 1.0, 1.0)
a_3	(0.7, 0.9, 0.9)	(0.3, 0.5, 1.0)	(0.04, 1.0, 1.0)	(0.5, 1.0, 1.0)
a_4	(0.6, 0.8, 0.9)	(0.3, 0.5, 1.0)	(0.04, 1.0, 1.0)	(0.5, 1.0, 1.0)
a_5	(0.7, 0.9, 0.9)	(0.3, 1.0, 1.0)	(0.04, 1.0, 1.0)	(0.5, 1.0, 1.0)
a_6	(0.7, 0.9, 0.9)	(0.3, 0.5, 1.0)	(0.04, 0.18, 1.0)	(0.5, 1.0, 1.0)
a_7	(0.7, 0.9, 1.0)	(0.3, 1.0, 1.0)	(0.04, 1.0, 1.0)	(0.5, 1.0, 1.0)
a_8	(0.6, 1.0, 1.0)	(0.3, 1.0, 1.0)	(0.04, 1.0, 1.0)	(0.5, 1.0, 1.0)
a_9	(0.7, 0.9, 0.9)	(0.3, 1.0, 1.0)	(0.04, 1.0, 1.0)	(0.5, 1.0, 1.0)

Table 7.7: The weighted normalized fuzzy decision matrix.

	C_1	C_2	C_3	C_4
a_1	(0.03, 0.23, 0.42)	(0.18, 0.37, 0.95)	(0.0, 0.04, 0.45)	(0.15, 0.25, 0.7)
a_2	(0.03, 0.20, 0.36)	(0.18, 0.37, 0.95)	(0.0, 0.12, 0.45)	(0.15, 0.5, 0.7)
a_3	(0.04, 0.24, 0.44)	(0.18, 0.37, 0.95)	(0.0, 0.24, 0.45)	(0.15, 0.5, 0.7)
a_4	(0.03, 0.22, 0.41)	(0.18, 0.37, 0.95)	(0.0, 0.24, 0.45)	(0.15, 0.5, 0.7)
a_5	(0.03, 0.23, 0.43)	(0.18, 0.75, 0.95)	(0.0, 0.24, 0.45)	(0.15, 0.5, 0.7)
a_6	(0.03, 0.24, 0.44)	(0.18, 0.37, 0.95)	(0.0, 0.04, 0.45)	(0.15, 0.5, 0.7)
a_7	(0.03, 0.24, 0.45)	(0.18, 0.75, 0.95)	(0.0, 0.24, 0.45)	(0.15, 0.5, 0.7)
a_8	(0.03, 0.25, 0.45)	(0.18, 0.75, 0.95)	(0.0, 0.24, 0.45)	(0.15, 0.5, 0.7)
a_9	(0.03, 0.23, 0.42)	(0.18, 0.75, 0.95)	(0.0, 0.24, 0.45)	(0.15, 0.5, 0.7)

Using positive ideal solution (Equation 7.8) and negative ideal solution (Equation 7.9) end Equation 7.14 the calculation of the relative closeness is shown in Table 7.8. In this case, the worst selection is candidate a_1.

Table 7.8: The relative closeness of the keypoints.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
D	0.35	0.36	0.38	0.37	0.40	0.37	0.41	0.41	0.40

According to the relative coefficient, we can determine the ranking order of all alternatives. We experimented also with the following distances: Euclidean, weighted Euclidean, Hamming [15], weighted Hamming, and L-R distance [176]. However, the improvement in performance was not statistically significant.

7.4. Experiments and results

The proposed algorithms were evaluated on the YCB-Video dataset. Additionally, an ablation study was conducted to analyze the impact of TOPSIS on the entire method.

7.4.1. Experimental results on YCB-Video dataset

The performance of proposed algorithms along with the performance of state-of-the-art algorithms is provided in Tab. 7.9. The AUC ADD scores achieved by proposed algorithms are compared with results attained by algorithms operating on single frame as well as algorithms for 6 DoF object pose tracking. The bold result in each row in the table indicates the highest AUC ADD value for the given object, whereas the second-best result is underlined. It can be observed that the TOPSIS aided algorithm permits achieving far better results in comparison to a base algorithm that determines the object pose on the basis of sparse object keypoints and the PnP in combination with RANSAC-based scheme. As can be seen, the results achieved by the fuzzy TOPSIS-based algorithm are far better than the results attained by TOPSIS-aided algorithm. Afterward, a simple voting-based decision algorithm was implemented. To alleviate the issue of keypoint selection, we employed the Borda rule on top-truncated preferences, as proposed in [162]. In the proposed approach the rules are truncated to two worst preferences. Interestingly, using an ordinary Borda count-based voting algorithm often yields a keypoint that was not initially ranked highest but received the best average ratings. This highlights the fact that keypoints with the highest average ratings do not necessarily guarantee optimal selections of keypoints, as they may instead result in smoother object pose trajectories. In addition to fuzzy TOPSIS, recently proposed fuzzy MABAC [170] and fuzzy MAIRCA [42] were used in the proposed voting-based decision committee (called VDC-fuzzyTOPSIS). By comparing results in the last two columns of Tab. 7.9, it can be seen that owing to using diverse algorithms for multi-criteria decision making with Borda count-based voting scheme it is possible to improve the AUC ADD scores. AUC ADD is calculated for non-symmetric objects, whereas AUC ADD-S is determined for symmetric objects, - denotes unavailable results, and "*" stands for symmetric objects. The value 128 indicates that the results were achieved by an hourglass neural network delivering output maps of size 128×128. The basic version of the hourglass network delivers output maps of size 64×64, c.f. Figure 7.2. Given the results in the discussed table, it is evident that TOPSIS algorithms operating on features that are frequently utilized in algorithms for object tracking including segmented masks, sparse keypoints, and distance transform permit achieving tangible performance gains. It is worth noting that for several objects from the YCB-Video dataset, the proposed voting-based method achieved the best and second-best results.

Table 7.9: AUC ADD scores [%] (max. th. 10 cm) achieved by the proposed algorithm on the YCB-Video dataset.

Object	PoseRBPF	DeepIM	[101]	PnP	TOPSIS	fuzzy TOPSIS	Proposed VDC-fuzzyTOPSIS	Proposed VDC-fuzzyTOPSIS 128
002_master_chef_can	63.3	89.0	**90.2**	87.7	88.7	90.1	**90.2**	88.8
003_cracker_box	77.8	**88.5**	85.4	83.3	86.4	86.7	87.4	87.1
004_sugar_box	79.6	**94.3**	87.5	82.7	85.7	86.2	88.4	89.3
005_tomato_soup_can	73.0	**89.1**	82.3	81.5	81.7	82.5	83.5	83.4
006_mustard_bottle	84.7	**92.0**	90.0	81.9	83.7	86.3	87.0	89.2
007_tuna_fish_can	64.2	**92.0**	87.9	80.3	81.7	82.6	84.4	90.4
008_pudding_box	64.5	80.1	83.3	80.0	82.4	**88.2**	84.4	87.1
009_gelatin_box	83.0	92.0	**94.4**	83.4	87.9	89.2	90.4	90.2
010_potted_meat_can	51.8	78.0	**82.3**	73.1	74.4	74.8	81.3	81.0
011_banana	18.4	**81.0**	73.6	63.6	67.4	67.0	67.5	78.1
019_pitcher_base	63.7	90.4	**90.7**	84.2	85.1	86.7	89.4	89.4
021_bleach_cleanser	60.5	81.7	80.6	77.3	78.8	79.9	81.7	**82.6**
024_bowl*	85.6	90.6	87.9	84.1	89.3	90.5	91.0	**91.1**
025_mug	77.9	83.2	85.8	80.5	83.5	83.7	84.7	**86.6**
035_power_drill	71.8	**85.4**	79.7	78.6	81.3	81.9	83.3	84.7
036_wood_block*	31.4	75.4	**82.6**	77.1	77.6	79.3	81.7	82.4
037_scissors	38.7	70.3	**71.4**	51.1	53.5	53.9	61.0	66.3
040_large_marker	67.1	80.4	80.6	58.8	60.2	64.6	62.4	**80.9**
051_large_clamp*	59.3	84.1	89.5	85.3	86.8	89.2	**90.8**	90.7
052_extra_large_clamp*	44.3	90.3	**91.8**	78.9	85.5	87.8	89.7	91.3
061_foam_brick*	92.6	**95.5**	87.4	83.3	84.1	87.7	86.8	92.9
Avg.	64.4	**85.9**	85.0	77.9	80.3	81.8	83.2	**85.9**

Table 7.10 compares results achieved by the proposed algorithm with recent algorithms estimating the object pose on the single frame [183], [166], [121], [187], [175] with results achieved by tracking-based methods [36], [89]. AUC ADD is calculated for non-symmetric objects, whereas AUC ADD-S is determined for symmetric objects. As can be observed the methods achieved by DeepIm and the proposed algorithm are better than the results achieved by PoseRBPF and all algorithms that estimate the pose without taking into account information from the previous frame, i.e. based on a single frame. Dealing with object occlusions is an important point in 6D object pose estimation. Since most objects in the visual world are partially obscured, this problem is encountered when estimating the object pose in most tasks.

Table 7.11 presents MSE errors for object keypoints and Dice scores for object segmentation, which were attained by the proposed hourglass-based neural network, that jointly detects nine object keypoints, predicts the object's non-occluded part, and also predicts the object's occluded part. To illustrate the capabilities of neural networks in detecting occluded parts of objects, we present Dice scores separately for predicting non-occluded and occluded object parts. The proposed approach has achieved promising results on selected objects from the YCB-Video

Table 7.10: Average AUC ADD scores [%] (max. th. 10 cm) achieved by the proposed algorithm on the YCB-Video dataset.

	PoseCNN	DOPE	DHROP	[187]	GDR-Net	PoseRBPF	DeepIM	Proposed VDC-fuzzyTOPSIS 128
Pose estimation	61.3	65.8	72.8	61.0	<u>84.4</u>	-	81.9	82.2
Pose tracking	-	-	-	-	-	64.4	**85.9**	**85.9**

dataset. As far as is known, this novel occlusion-aware pipeline is a first in object pose estimation, as it jointly detects keypoints, recovers non-occluded and occluded parts, and leverages this information to enhance pose estimation accuracy. The proposed method differs significantly from that of [124], which utilized a GAN network for object pose estimation. In contrast, the proposed method employs an hourglass neural network to jointly detect keypoints, predict non-occluded and occluded parts, and make decisions based on multi-criteria evaluation, incorporating uncertainty in observations and predictions.

7.4.2. Ablation study

In this section, the efficacy of TOPSIS-based components is analyzed and the superiority of the algorithm over the baseline one is proven. The impact of adding TOPSIS and fuzzy TOPSIS is inspected. The effect of the proposed extensions on the ADD scores is analyzed. The experiments were conducted on the power drill object from the 56th test video in the YCB-Video dataset. Figure 7.6 presents a plot of the occlusion degree of the power drill vs. frame number. As it can be observed, at the beginning of the sequence, the object of interest is the most occluded and with each frame, the degree of the occlusion gradually decreases. The change in the pose of the object between successive frames is small, particularly, there are no jerks, and the total changes in the object pose are not too large, i.e. up to 40 degrees over the entire sequence, see also Figure 7.7.

As can be observed, frames from the beginning of the sequence are better suited for achieving high ADD scores, even though the object is partially occluded, mainly because the wide side of the object is observed by the camera. On the other hand, the frames a the end of the sequence are less suitable for high ADD scores because the camera observes the narrower side of the object. The second plot presents ADD score errors vs. frame number achieved by the PnP algorithm operating on nine object keypoints. The ADD score on the whole video is equal to 0.9029. As can be observed, the biggest errors are in the beginning part of the sequence. The next plot presents ADD score errors over time that have been attained by the PnP algorithm operating on eight object keypoints. In the discussed experiment the hourglass neural network detected nine object keypoints and then the worst object keypoint was removed by the TOPSIS module. This means that eight object keypoints selected in this way were fed to the PnP algorithm with the RANSAC scheme. The ADD score achieved by such TOPSIS-aided algorithm was equal to 0.9584. As can be observed, the use of TOPSIS permitted to achieve a far better ADD score. The last plot presents ADD score errors over time that have been attained by the PnP algorithm operating on eight object keypoints, which were selected by the fuzzy PnP. The ADD score achieved by such fuzzy TOPSIS aided algorithm was equal to 0.9837. It can therefore be concluded that both TOPSIS algorithms permit achieving better ADD scores, but fuzzy

Table 7.11: MSE and Dice scores achieved by hourglass network for joint object segmentation and fiducial keypoints estimation.

	MSE (keypoints)	Dice sc. (obj. seg.)	Dice sc. (occlud. obj. seg.)
002_master_chef_can	3.65	0.96	0.88
003_cracker_box	1.83	0.94	0.81
004_sugar_box	3.96	0.97	1.00 (no occlusions)
005_tomato_soup_can	6.11	0.96	0.93
006_mustard_bottle	1.95	0.95	1.00 (no occlusions)
007_tuna_fish_can	1.96	0.96	0.46
008_pudding_box	6.56	0.93	0.91
009_gelatin_box	68.6	0.97	1.00 (no occlusions)
010_potted_meat_can	6.19	0.96	0.92
011_banana	3.39	0.95	0.83
019_pitcher_base	3.76	0.97	0.74
021_bleach_cleanser	2.47	0.97	0.94
024_bowl	2.25	0.97	0.92
025_mug	1.54	0.96	0.85
035_power_drill	1.49	0.94	0.81
036_wood_block	93.2	0.97	1.00 (no occlusions)
037_scissors	64.9	0.93	0.88
040_large_marker	2.90	0.92	0.42
051_large_clamp	1.92	0.95	0.33
052_extra_large_clamp	3.57	0.93	0.88
061_foam_brick	47.7	0.96	0.89

TOPSIS yields the largest gains in ADD scores. Following the initial simulation, we created additional scenarios in which the location of an object's keypoint was determined with increased error rates. It is not uncommon for keypoints situated on occluded or invisible parts of the object to deviate significantly from their true locations. To simulate situations involving occluded or invisible object keypoints, a single keypoint was randomly chosen and then displaced by a vector with values randomly sampled from a range of 5-15 pixels. In the experiment with the moved keypoint on the power drill object the ADD score achieved by the PnP-based algorithm was smaller by almost 3.5% in comparison to the ADD score of PnP-based algorithm with no keypoint movement, and it was smaller by about 6.4% for fuzzy TOPSIS. The fuzzy TOPSIS was capable of detecting this randomly selected and then translated keypoint in almost 79% of cases.

Table 7.12 summarizes experimental results that have been achieved on images with the power drill object. As it can be observed, the drop in the pose estimation performance for the proposed VDC-fuzzyTOPSIS method in the discussed experiment is far smaller than the drop in the performance for multi-criteria decision making

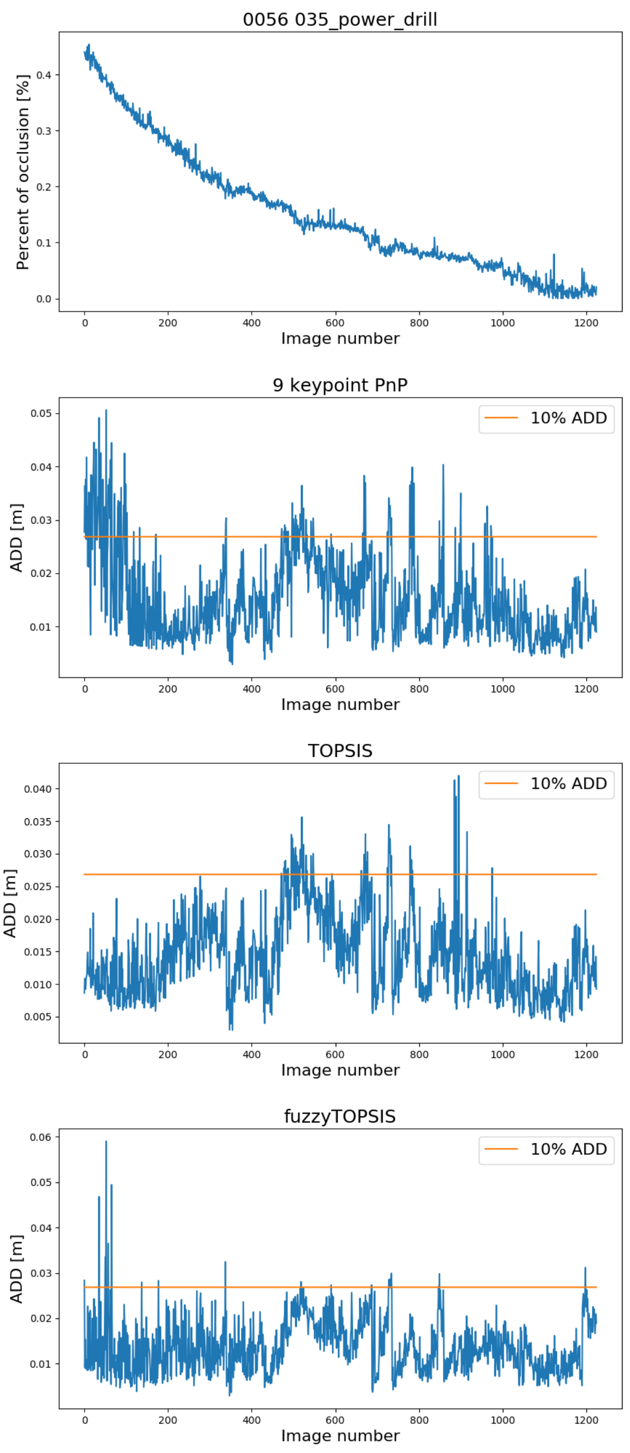


Figure 7.6: Occlusion degree of power drill in sequence #56 vs. frame number, ADD score vs. frame number achieved by PnP, ADD score vs. frame number achieved by PnP and TOPSIS, ADD score vs. frame number achieved by PnP and fuzzy TOPSIS.

with a single method, i.e. using only TOPSIS. The experimental results demonstrate that the fuzzy TOPSIS-based algorithm holds some potential as it is capable of decreasing errors in 6D object pose tracking. The proposed hourglass-based neural network, which jointly detects nine object keypoints, predicts the object's non-occluded part, and also predicts the object's occluded part delivers valuable decision variables.

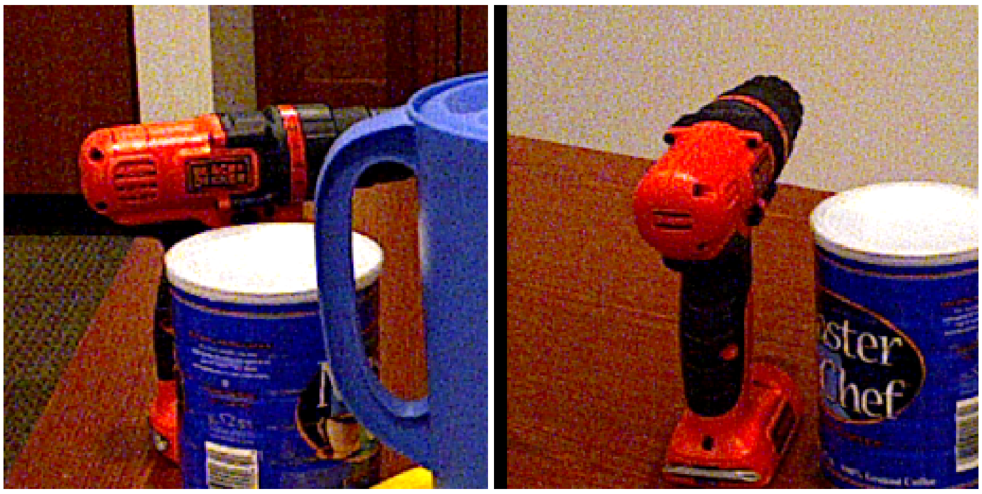


Figure 7.7: Appearance of the subject of interest in the first and last frames of sequence #56. from the YCB-Video dataset

Table 7.12: ADD scores [%] achieved on toy data.

Toy data	ADD
PnP	90.3
TOPSIS	95.8
fuzzyTOPSIS	98.4
VDC-fuzzyTOPSIS	98.4
Toy data with moved keypoint	**ADD**
PnP	86.5
fuzzyTOPSIS	92.0
VDC-fuzzyTOPSIS	97.8

7.5. Summary

In contrast to most state-of-the-art algorithms for 6D object estimation and tracking, which rely on a single hypothesis, the proposed approach leverages multi-criteria decision making under uncertainty to enhance PnP-RANSAC pose estimates. Unlike traditional methods that utilize sparse object keypoints for pose estimation, the proposed method takes a more comprehensive approach by considering the full range of available information. Experimental results demonstrated that TOPSIS algorithms, operating on features frequently utilized in pose estimation and tracking algorithms, including segmented masks, sparse object keypoints, and distance transforms, enable tangible performance gains. Experimental results demonstrated that fuzzy TOPSIS is capable of achieving far better results in comparison to TOPSIS operating on crisp values. The ablation study revealed that certain key components within the proposed algorithm are primarily responsible for enhancing performance, specifically in scenarios involving uncertain or noisy visual data. On a standard PC equipped with an NVIDIA GPU RTX 2060, the proposed method achieves nine frames per second.

Chapter 8

Conclusions

The final chapter provides an overview of the research findings, focusing on the experimental results that validate the research hypothesis posed in this dissertation, and discusses both the strengths and weaknesses of the employed methodologies. Additionally, potential directions for future research are explored.

8.1. Summary

This dissertation proposes innovative methods to overcome the main limitations and challenges associated with object pose tracking. Chapters 4, 5, 6, and 7 provide a detailed description of the proposed solutions and offer in-depth information on the results obtained on two challenging datasets. The proposed methods can be categorized into three distinct types, each with its unique characteristics and applications:

– neural network-based

– multi-hypothesis

– object pose refinement

The neural network-based category, comprising Y-Net and X-Net networks, leverages additional information about object rotation to improve keypoint estimation. Notably, X-Net also provides additional information about the segmentation of the object. A comparison of both approaches can be found in Table 8.1. Although they achieve similar average results on the YCB-Video dataset, it is evident that X-Net provides a significant improvement on the OPT dataset.

Table 8.1: Comparison of proposed methods on OPT dataset, YCB-Video dataset, and running times (NVIDIA GPU RTX 2060).

Proposed method	Avg. AUC on OPT	Avg. AUC ADD on YCB-Video	Running times sec.
Y-Net, Chapter 4	12.58	86.3	-
X-Net, Chapter 5	14.10	86.3	0.20
Q-PF-PSO, Chapter 6	12.65	-	0.19
Q-PF-PSO 0.4s, Chapter 6	14.57	87.2	0.40
TOPSIS, Chapter 7	-	85.9	0.11

The multi-hypothesis approach differs from traditional methods by providing a probability distribution of object poses rather than relying on a single pose estimate. The proposed hybrid quaternion particle filter with quaternion particle swarm optimization (Q-PF-PSO) has been found to produce the best results among the proposed methods when supplemented with k-means++ and TOPSIS refinement techniques. However, this approach requires additional computational resources due to the iterative character of PF and PSO iterations.

The object pose refinement category adopts a more comprehensive strategy by incorporating a wide range of available data. Experimental results have shown that TOPSIS-based methods operating on features commonly used in object pose estimation and tracking, such as segmented masks, sparse object keypoints, and distance transforms, lead to significant performance enhancements. Notably, despite achieving slightly inferior results, the proposed TOPSIS-based method is the fastest among all considered approaches.

8.2. Conclusions and thesis contribution

This thesis presents novel methods for 6D object pose tracking using only RGB images. The study focuses on two key research areas: (1) augmenting neural networks with supplementary information about an object's rotation in the quaternion form to improve the detection of distinctive points, and (2) refining multi-stage pose estimation by leveraging keypoints and geometric reasoning. The proposed methods were rigorously evaluated through an extensive series of experiments. Designed with object pose tracking in mind, these methods can be easily adapted or serve as a foundation for developing solutions to analogous problems across various disciplines.

The thesis of this dissertation, outlined in Section 1.3, states that a multi-input neural network with multi-key pose refinement can improve tracking of an object's pose on RGB images. A comprehensive review of current research advancements in pose estimation and tracking methods uncovered several challenges that have yet to be adequately addressed in this field. The simultaneous identification of keypoints and segmentation of an object allows for the creation of object distance maps, enabling the efficient removal of outliers and prioritization of more reliable keypoints. The proposed approaches achieve real-time, accurate, and robust monocular 3D object pose estimation. This study's novel approaches, tailored to pose tracking using only RGB images, aim to overcome the limitations of existing approaches and validate the research hypothesis.

The first challenge was utilizing information about the object's rotation from the previous frame. This issue is investigated in Chapters 4,5, which concerns using multi-input to encode data from previous and current frames. The results demonstrate that the proposed methods, which are built on an additional branch for neural networks, are an effective way to improve the detection of symmetric keypoints. This ability to estimate the position of keypoint inside the neural network could be used not only to help with poses with symmetric keypoints configuration but also when some keypoints are occluded. The proposed methods also demonstrate superior performance on the publicly available YCB-Video dataset.

The second challenge concerned the multi-stage pose refinement that uses keypoints and geometric reasoning. This is addressed in Chapters 5,6,7, by proposing novel methods for pose optimization and refinement strategy that apart from keypoints utilize also signed distance maps. As confirmed by the conducted experiments, estimating the potential occlusion of an object using the distance transform values on projected boundary segments, along with the detection of invisible or occluded keypoints through labeled segments of the object shape, leads to several improvements in pose tracking. Unlike traditional object pose estimation approaches that rely on sparse object keypoints, these novel algorithms leverage the power of multi-criteria decision-making under uncertainty to enhance pose estimation accuracy. Moreover, the Siamese neural network can be accurately integrated to guide the selected particle toward the predicted pose of the object. The effectiveness of the TOPSIS algorithm was experimentally validated when applied to features commonly used in object pose estimation techniques, such as segmented masks, sparse object keypoints, and distance transform. This resulted in notable accuracy improvements.

The findings of this thesis demonstrate that the primary objective of the research has been successfully achieved, namely, that a multi-input neural network with multi-stage pose refinement can improve tracking of the object's pose. The primary contribution of this thesis is a novel method for utilizing information from the previous frame to improve keypoint detection by neural networks. Two variants of deep neural networks, Y-Net and X-Net, were proposed, which achieved statistically significant improvements over their base variants. Another significant contribution is the multi-hypothesis Q-PF-PSO method, which achieved the best results and was thoroughly analyzed to demonstrate the potential of components of this method. Finally, the fuzzy TOPSIS-based method has some potential as it can reduce errors in 6D object pose tracking. The proposed hourglass-based neural network simultaneously detects nine object keypoints, and predicts both the non-occluded and occluded parts of the object, providing valuable decision variables. All proposed algorithms were verified on freely available benchmark datasets and achieved competitive results. Some of the proposed methods were evaluated on the Nvidia Jetson AGX Xavier board, where real-time execution was achieved.

8.3. Limitations and future work

This thesis presents various avenues for future investigations, offering numerous opportunities to explore and expand our understanding of 6 DoF tracking of transparent objects, symmetric ones, as well as occluded objects. While the proposed methods proved to be useful for object pose tracking, they also opened up several new research directions. First of all, the additional branch for neural networks presented in Chapters 4,5, can be further developed. The primary constraint of this method lies in its reliance on previously determined rotational data, which is used exclusively by the X-Net for 6D object tracking purposes. In future research, the current approach will be expanded by incorporating additional contextual information from the object's rotational dynamics. Furthermore, more comprehensive features from the preceding frame will be utilized to enhance understanding and modeling of the system.

In future research, consideration should be given to using the Q-PF-PSO for object pose refinement. Furthermore, the structure of the Q-PF-PSO should be optimized to reduce the computational overhead for real-time applications. The proposed approaches possess a versatility that enables their adaptability across various machine learning systems, transcending specific models or domains. This universality offers immense possibilities for their practical application in real-world scenarios.

List of Figures

List of Tables

Bibliography

[1] A. M. Abdel Tawab, M. Abdelhalim, and S.-D. Habib. Efficient multi-feature PSO for fast gray level object-tracking. *Applied Soft Computing*, 14:317–337, 2014.

[2] J. Abdi, A. Al-Hindawi, T. Ng, and M. P. Vizcaychipi. Scoping review on the use of socially assistive robot technology in elderly care. *BMJ Open*, 8(2), 2018.

[3] A. Akca and M. Önder Efe. Multiple Model Kalman and Particle Filters and Applications: A Survey. *IFAC-PapersOnLine*, 52(3):73–78, 2019. 15th IFAC Symposium on Large Scale Complex Systems LSS, 2019.

[4] Ö. D. Akyildiz and J. Míguez. Nudging the particle filter. *Statistics and Computing*, 30(2):305–330, Mar 2020.

[5] D. Arthur and S. Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[6] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

[7] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D Chairs: Exemplar Part-Based 2D-3D Alignment Using a Large Dataset of CAD Models. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3762–3769, 2014.

[8] D. Auger, A. Couëtoux, and O. Teytaud. Continuous Upper Confidence Trees with Polynomial Exploration – Consistency. In *Machine Learning and Knowledge Discovery in Databases*, pages 194–209, Berlin, Heidelberg, 2013. Springer.

[9] S. M. Baas and H. Kwakernaak. Rating and ranking of multiple-aspect alternatives using fuzzy sets. *Automatica*, 13(1):47–58, 1977.

[10] D. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[11] A. Behera, Z. Wharton, P. Hewage, and S. Kumar. Rotation axis focused attention network (rafa-net) for estimating head pose. In *Computer Vision – ACCV*, pages 223–240. Springer International Publishing, 2021.

[12] Y. Bengio, Y. Lecun, and G. Hinton. Deep Learning for AI. *Commun. ACM*, 64(7):58–65, Jun 2021.

[13] K. Berntorp and S. Di Cairano. Particle Filtering for Automotive: A survey. In *22th Int. Conf. on Information Fusion (FUSION)*, pages 1–8, 2019.

[14] J. Bidot, L. Karlsson, F. Lagriffoul, and A. Saffiotti. Geometric backtracking for combined task and motion planning in robotic systems. *Artificial Intelligence*, 247:229–265, 2017. Special Issue on AI and Robotics.

[15] A. Bookstein, S. Klein, and T. Raita. Fuzzy hamming distance: A new dissimilarity measure. pages 86–97, 01 2001.

[16] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *Computer Vision – ECCV*, pages 536–551. Springer, 2014.

[17] E. Brachmann, F. Michel, A. Krull, M. Yang, S. Gumhold, and C. Rother. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372. IEEE Computer Society, Jun 2016.

[18] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993.

[19] A. Brown and T. Green. Virtual Reality: Low-Cost Tools and Resources for the Classroom. *TechTrends*, 60(5):517–519, Sep 2016.

[20] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. Online Optimization in X-Armed Bandits. In *NIPS*, pages 201–208, 2008.

[21] B. Bugaev, A. Kryshchenko, and R. Belov. Combining 3D Model Contour Energy and Keypoints for Object Tracking. In *Computer Vision – ECCV*, pages 55–70. Springer, 2018.

[22] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar. Yale-CMU-Berkeley dataset for robotic manipulation research. *The Int. Journal of Robotics Research*, 36(3):261–268, 2017.

[23] D. Castelvecchi. Low-cost headsets boost virtual reality's lab appeal. *Nature*, 533(7602):153–154, May 2016.

[24] T. Chan and W. Zhu. Level set based shape prior segmentation. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1164–1170 vol. 2, 2005.

[25] B. Chen, A. Parra, J. Cao, N. Li, and T. Chin. End-to-End Learnable Geometric Vision by Backpropagating PnP Optimization. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8097–8106. IEEE Computer Society, Jun 2020.

[26] C.-M. Cheng, H.-W. Chen, T.-Y. Lee, S.-H. Lai, and Y.-H. Tsai. Robust 3d object pose estimation from a single 2d image. In *2011 Visual Communications and Image Processing (VCIP)*, pages 1–4, 2011.

[27] Q. Cheng, X. Han, T. Zhao, and V. S. S. Yadavalli. Improved particle swarm optimization and neighborhood field optimization by introducing the re-sampling step of particle filter. *Journal of Industrial and Management Optimization*, 15(1):177–198, 2019.

[28] C. Choi and H. I. Christensen. 3D textureless object detection and tracking: An edge-based approach. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3877–3884, 2012.

[29] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[30] P. Cipresso, I. A. C. Giglioli, M. A. Raya, and G. Riva. The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature. *Frontiers in Psychology*, 9, 2018.

[31] P. Cipresso, S. Serino, and G. Riva. Psychometric assessment and behavioral experiments using a free virtual reality platform and computational science. *BMC Medical Informatics and Decision Making*, 16(1):37, Mar 2016.

[32] T. Collins and A. Bartoli. Infinitesimal Plane-Based Pose Estimation. *Int. Journal of Computer Vision*, 109(3):252–286, Sep 2014.

[33] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Computers and Games*, pages 72–83, Berlin, Heidelberg, 2007. Springer.

[34] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. Robust 3D Object Tracking from Monocular Images Using Stable Parts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 40(6):1465–1479, 2018.

[35] Z. Dang, K. M. Yi, Y. Hu, F. Wang, P. Fua, and M. Salzmann. Eigendecomposition-Free Training of Deep Networks with Zero Eigenvalue-Based Losses. In *Computer Vision – ECCV*, pages 792–807. Springer, 2018.

[36] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox. PoseRBPF: A Rao-Blackwellized Particle Filter for 6-D Object Pose Tracking. *IEEE Trans. on Robotics*, 37(5):1328–1342, 2021.

[37] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3665–3671, 2020.

[38] Y. Dong, L. Ji, S. Wang, P. Gong, J. Yue, R. Shen, C. Chen, and Y. Zhang. Accurate 6DOF Pose Tracking for Texture-Less Objects. *IEEE Trans. on Circuits and Systems for Video Technology*, 31(5):1834–1848, 2021.

[39] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[40] G. Du, K. Wang, S. Lian, and K. Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 54(3):1677–1734, Mar 2021.

[41] F. Dunn and I. Parberry. *3D Math Primer for Graphics and Game Development*. Wordware game math library. Wordware Pub., 2002.

[42] F. Ecer. An extended mairca method using intuitionistic fuzzy sets for coronavirus vaccine selection in the age of covid-19. *Neural Computing and Applications*, 34(7):5603–5623, Apr 2022.

[43] I. Emovon and W. O. Aibuedefe. Fuzzy topsis application in materials analysis for economic production of cashew juice extractor. *Fuzzy Information and Engineering*, 12(1):1–18, 2020.

[44] Z. Fan, Y. Zhu, Y. He, Q. Sun, H. Liu, and J. He. Deep Learning on Monocular Object Pose Detection and Tracking: A Comprehensive Overview. *ACM Comput. Surv.*, Mar 2022. Just Accepted.

[45] P. Fearnhead and H. R. Künsch. Particle Filters and Data Assimilation. *Annual Review of Statistics and Its Application*, 5(1):421–449, 2018.

[46] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Readings in Computer Vision*, pages 726–740. Morgan Kaufmann, San Francisco (CA), 1987.

[47] L. Fortuna, G. Muscato, and M. G. Xibilia. A comparison between hmlp and hrbf for attitude control. *IEEE Tran. on Neural Networks*, 12(2):318–328, 2001.

[48] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003.

[49] Y. Gao, W. Du, and G. Yan. Selectively-informed particle swarm optimization. *Scientific Reports*, 5(1):9295, Mar 2015.

[50] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, D. D. Yuh, C. C. G. Chen, R. Vidal, S. Khudanpur, and G. Hager. Jhu-isi gesture and skill assessment working set (jigsaws) : A surgical activity dataset for human motion modeling. 2014.

[51] M. Garon and J.-F. Lalonde. Deep 6-DOF Tracking. *IEEE Trans. on Visualization and Computer Graphics*, 23(11):2410–2418, 2017.

[52] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[53] F. S. Grassia. Practical Parameterization of Rotations Using the Exponential Map. *J. Graph. Tools*, 3(3):29–48, Mar 1998.

[54] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[55] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[56] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[57] T. Hempel, A. A. Abdelrahman, and A. Al-Hamadi. 6d rotation representation for unconstrained head pose estimation. In *2022 IEEE International Conf. on Image Processing (ICIP)*, pages 2496–2500, 2022.

[58] J. A. Hesch and S. I. Roumeliotis. A Direct Least-Squares (DLS) method for PnP. In *Int. Conf. on Computer Vision*, pages 383–390, 2011.

[59] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient Response Maps for Real-Time Detection of Textureless Objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012.

[60] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Int. Conf. on Computer Vision*, pages 858–865, 2011.

[61] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Computer Vision – ACCV 2012*, pages 548–562, Berlin, Heidelberg, 2013. Springer.

[62] T. Hodan, P. Haluza, S. Obdrálek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. In *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pages 880–888, 2017.

[63] S. Hoque, M. Y. Arafat, S. Xu, A. Maiti, and Y. Wei. A comprehensive review on 3d object detection and 6d pose estimation with deep learning. *IEEE Access*, 9:143746–143770, 2021.

[64] M. Hu, H. Yu, K. Gu, Z. Wang, H. Ruan, K. Wang, S. Ren, B. Li, L. Gan, S. Xu, G. Yang, Y. Shen, and X. Li. A particle-filter framework for robust cryo-EM 3D reconstruction. *Nature methods*, 15(12):10831089, December 2018.

[65] Y. Hu, P. Fua, W. Wang, and M. Salzmann. Single-Stage 6D Object Pose Estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936. IEEE Computer Society, Jun 2020.

[66] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-Driven 6D Object Pose Estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3380–3389. IEEE Computer Society, Jun 2019.

[67] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. Journal of Computer Vision*, 5(2):195–212, Nov 1990.

[68] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the 32nd Int. Conf. on Int. Conf. on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015.

[69] S. Iwase, X. Liu, R. Khirodkar, R. Yokota, and K. M. Kitani. RePOSE: Fast 6D Object Pose Refinement via Deep Texture Rendering. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pages 3283–3292. IEEE Computer Society, Oct 2021.

[70] W. Jian-giang and W. Run-qi. Hybrid random multi-criteria decision-making approach with incomplete certain information. In *2008 Chinese Control and Decision Conf.*, pages 1444–1448, 2008.

[71] A. Kaufmann and M. Gupta. *Introduction to Fuzzy Arithmetic: Theory and Applications*. Electrical/computer science and engineering series. Van Nostrand Reinhold Company, 1985.

[72] T. Ke and S. I. Roumeliotis. An Efficient Algebraic Solution to the Perspective-Three-Point Problem. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4618–4626, 2017.

[73] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1530–1538. IEEE Computer Society, Oct 2017.

[74] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. of ICNN'95 - Int. Conf. on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[75] N. U. Khan and W. Wan. A review of human pose estimation from single image. In *2018 Int. Conf. on Audio, Language and Image Processing (ICALIP)*, pages 230–236, 2018.

[76] S. Kim and M. Kim. Rotation representations and their conversions. *IEEE Access*, 11:6682–6699, 2023.

[77] E. Koch, F. Baig, and Q. Zaidi. Picture perception reveals mental geometry of 3D scene inferences. *Proc. of the National Academy of Sciences*, 115(30):7807–7812, 2018.

[78] L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer.

[79] A. Kutschireiter, S. C. Surace, H. Sprekeler, and J.-P. Pfister. Nonlinear Bayesian filtering and learning: a neuronal dynamics for perception. *Scientific Reports*, 7(1):8722, Aug 2017.

[80] B. Kwolek. Model based facial pose tracking using a particle filter. In *Geometric Modeling and Imaging–New Trends (GMAI'06)*, pages 203–208, 2006.

[81] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation. In *Computer Vision – ECCV*, pages 574–591. Springer, 2020.

[82] J. Lambek. If hamilton had prevailed: quaternions in physics. *The Mathematical Intelligencer*, 17(4):7–15, Dec 1995.

[83] G. Lecuyer, M. Ragot, N. Martin, L. Launay, and P. Jannin. Assisted phase and step annotation for surgical videos. *International Journal of Computer Assisted Radiology and Surgery*, 15(4):673–680, Apr 2020.

[84] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *Int. Journal of Computer Vision*, 81:155–166, 2009.

[85] V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *Proc. of the 2004 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II, 2004.

[86] K. Levenberg. A method for the solution of certain non - linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[87] C. Li, J. Bai, and G. D. Hager. A Unified Framework for Multi-view Multi-class Object Pose Estimation. In *Computer Vision – ECCV*, pages 263–281. Springer, 2018.

[88] K. Li, F. Pfaff, and U. D. Hanebeck. Unscented Dual Quaternion Particle Filter for SE(3) Estimation. *IEEE Control Systems Letters*, 5(2):647–652, 2021.

[89] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. *Int. Journal of Computer Vision*, 128(3):657–678, Mar 2020.

[90] Z. Li, G. Wang, and X. Ji. CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6-DoF Object Pose Estimation. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pages 7677–7686. IEEE Computer Society, Nov 2019.

[91] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV*, pages 740–755. Springer, 2014.

[92] J. S. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[93] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV*, pages 21–37. Springer, 2016.

[94] V. Lopatin and B. Kwolek. 6D Pose Estimation of Texture-Less Objects on RGB Images Using CNNs. In *Artificial Intelligence and Soft Computing*, pages 180–192. Springer, 2020.

[95] D. Lowe. Object recognition from local scale-invariant features. In *Proc. of the Seventh IEEE Int. Conf. on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[96] M. Majcher and B. Kwolek. 3D Model-Based 6D Object Pose Tracking on RGB Images. In *Intelligent Information and Database Systems*, pages 271–282, Lecture Notes in Computer Science, vol. 12033, Cham, 2020. Springer.

[97] M. Majcher. and B. Kwolek. 3D Model-based 6D Object Pose Tracking on RGB Images using Particle Filtering and Heuristic Optimization. In *Proc. of the 15th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pages 690–697. INSTICC, SciTePress, 2020.

[98] M. Majcher and B. Kwolek. Deep Quaternion Pose Proposals for 6D Object Pose Tracking. In *2021 IEEE/CVF Int. Conf. on Computer Vision Workshops (ICCVW)*, pages 243–251, IEEE, 2021.

[99] M. Majcher and B. Kwolek. Fiducial Points-supported Object Pose Tracking on RGB Images via Particle Filtering with Heuristic Optimization. In *Proc. of the 16th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*, pages 919–926. INSTICC, SciTePress, 2021.

[100] M. Majcher and B. Kwolek. Quaternion-driven CNN for Object Pose Tracking. In *IEEE Int. Conf. on Visual Communications and Image Processing (VCIP)*, pages 1–4, IEEE, 2021.

[101] M. Majcher and B. Kwolek. Multiple-criteria-Based Object Pose Tracking in RGB Videos. In *(Eds.): ICCCI 2022, LNAI 13501*, pages 1–14. Springer, 2022.

[102] M. Majcher and B. Kwolek. Pose Guided Feature Learning for 3D Object Tracking on RGB Videos. In *Proc. of the 17th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2022, Volume 5: VISAPP, Online Streaming, February 6-8, 2022*, pages 574–581. SCITEPRESS, 2022.

[103] M. Majcher and B. Kwolek. Shape Enhanced Keypoints Learning With Geometric Prior for 6D Object Pose Tracking. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2986–2992, IEEE, June 2022.

[104] M. Majcher and B. Kwolek. TOPSIS Aided Object Pose Tracking on RGB Images. *IEEE Access*, 11:139498–139508, 2023.

[105] M. Majcher and B. Kwolek. Multi-Modal Pose Representations for 6-DOF Object Tracking. *Journal of Intelligent & Robotic Systems*, 110(4):149, Oct 2024.

[106] M. Majcher and B. Kwolek. Object pose tracking using multimodal knowledge from RGB images and quaternion-based rotation contexts. *Applied Soft Computing*, page 112699, 2025.

[107] F. Manhardt, D. Arroyo, C. Rupprecht, B. Busam, T. Birdal, N. Navab, and F. Tombari. Explaining the Ambiguity of Object Detection and 6D Pose From Visual Data. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pages 6840–6849. IEEE Computer Society, Nov 2019.

[108] F. Manhardt, W. Kehl, N. Navab, and F. Tombari. Deep Model-Based 6D Pose Refinement in RGB. In *Computer Vision – ECCV*, pages 833–849. Springer, 2018.

[109] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[110] Z. C. Márton, S. Türker, C. Rink, M. Brucker, S. Kriegel, T. Bodenmüller, and S. Riedel. Improving object orientation estimates by considering multiple viewpoints. *Autonomous Robots*, 42(2):423–442, Feb 2018.

[111] G. Marullo, L. Tanzi, P. Piazzolla, and E. Vezzetti. 6D object position estimation from 2D images: a literature review. *Multimedia Tools and Applications*, 82(16):24605–24643, Jul 2023.

[112] María-Blanca Ibáñez and Carlos Delgado-Kloos. Augmented reality for STEM learning: A systematic review. *Computers & Education*, 123:109–123, 2018.

[113] N. Matsui, T. Isokawa, H. Kusamichi, F. Peper, and H. Nishimura. Quaternion neural network with geometrical operators. *Journal of Intelligent & Fuzzy Systems*, 15(3-4):149–164, 2004.

[114] M. Maurer, J. Gerdes, B. Lenz, and H. Winner. *Autonomous Driving. Technical, Legal and Social Aspects*. Springer Berlin Heidelberg, 05 2016.

[115] C. Mitash, A. Boularias, and K. E. Bekris. Improving 6D Pose Estimation of Objects in Clutter Via Physics-Aware Monte Carlo Tree Search. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3331–3338, 2018.

[116] P. D. Moral, A. Doucet, and A. Jasra. On adaptive resampling strategies for sequential Monte Carlo methods. *Bernoulli*, 18(1):252 – 278, 2012.

[117] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5632–5640. IEEE Computer Society, Jul 2017.

[118] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. on Robotics*, 33(5):1255–1262, 2017.

[119] K. Murphy and S. Russell. *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks*, pages 499–515. Springer New York, New York, NY, 2001.

[120] S. Obdrálek, G. Kurillo, F. Ofli, R. Bajcsy, E. Seto, H. Jimison, and M. Pavel. Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population. In *Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, pages 1188–1193, 2012.

[121] M. Oberweger, M. Rad, and V. Lepetit. Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. In *Computer Vision – ECCV*, pages 125–141. Springer, 2018.

[122] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. LF-Net: Learning Local Features from Images. In *Proc. of the 32nd Int. Conf. on Neural Information Processing Systems*, NIPS'18, pages 6237–6247, Red Hook, NY, USA, 2018. Curran Associates Inc.

[123] C. Papaioannidis and I. Pitas. 3D Object Pose Estimation Using Multi-Objective Quaternion Learning. *IEEE Trans. on Circuits and Systems for Video Technology*, 30(8):2683–2693, 2020.

[124] K. Park, T. Patten, and M. Vincze. Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pages 7667–7676. IEEE Computer Society, Nov 2019.

[125] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[126] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-DoF object pose from semantic keypoints. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2011–2018, 2017.

[127] J. Peddie. Augmented reality. 01 2017.

[128] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4556–4565, 2019.

[129] A. S. Periyasamy, A. Amini, V. Tsaturyan, and S. Behnke. Yolopose v2: Understanding and improving transformer-based 6d pose estimation. *Robotics and Autonomous Systems*, 168:104490, 2023.

[130] A. S. Periyasamy, V. Tsaturyan, and S. Behnke. Efficient Multi-Object Pose Estimation Using Multi-Resolution Deformable Attention and Query Aggregation . In *2023 Seventh IEEE Int. Conf. on Robotic Computing (IRC)*, pages 247–254. IEEE Computer Society, Dec. 2023.

[131] V. A. Prisacariu and I. D. Reid. PWP3D: Real-Time Segmentation and Tracking of 3D Objects. *Int. Journal of Computer Vision*, 98(3):335–354, Jul 2012.

[132] M. Pulido and P. J. van Leeuwen. Sequential Monte Carlo with kernel embedded mappings: The mapping particle filter. *Journal of Computational Physics*, 396:400–415, 2019.

[133] M. Rad and V. Lepetit. BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 3848–3856. IEEE Computer Society, Oct 2017.

[134] K. Ramnath, S. N. Sinha, R. Szeliski, and E. Hsiao. Car make and model recognition using 3D curve alignment. In *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pages 285–292. IEEE Computer Society, Mar 2014.

[135] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[136] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza. A Dataset for Improved RGBD-Based Object Detection and Pose Estimation for Warehouse Pick-and-Place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[137] S. Riedel, Z.-C. Marton, and S. Kriegel. Multi-view orientation estimation using Bingham mixture models. In *IEEE Int. Conf. on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–6, 2016.

[138] R. Rios-Cabrera and T. Tuytelaars. Discriminatively Trained Templates for 3D Object Detection: A Real Time Scalable Approach. In *IEEE Int. Conf. on Computer Vision*, pages 2048–2055, 2013.

[139] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer, 2015.

[140] S. Rosa, G. Toscana, and B. Bona. Q-PSO: Fast Quaternion-Based Pose Estimation from RGB-D Images. *Journal of Intelligent & Robotic Systems*, 92(3):465–487, Dec 2018.

[141] C. Sahin, G. Garcia-Hernando, J. Sock, and T.-K. Kim. A review on object pose recovery: From 3D bounding box detectors to full 6D pose estimators. *Image and Vision Computing*, 96:103898, 2020.

[142] C. Sahin and T.-K. Kim. Recovering 6D Object Pose: A Review and Multi-modal Analysis. In *Computer Vision – ECCV Workshops*, pages 15–31. Springer, 2019.

[143] A. P. Sánchez, J. Andrade-Cetto, and F. Moreno-Noguer. Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35:2387–2400, 2013.

[144] S. Sengupta, S. Basak, and R. A. Peters. Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. *Machine Learning and Knowledge Extraction*, 1(1):157–191, 2019.

[145] B.-K. Seo, H. Park, J.-I. Park, S. Hinterstoisser, and S. Ilic. Optimal Local Searching for Fast and Robust Textureless 3D Object Tracking in Highly Cluttered Backgrounds. *IEEE Trans. on Visualization & Computer Graphics*, 20(01):99–110, Jan 2014.

[146] L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, 2001.

[147] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conf. on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 245–254, New York, NY, USA, 1985. Association for Computing Machinery.

[148] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304, 2011.

[149] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016.

[150] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd Int. Conf. on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conf. Track Proc.*, 2015.

[151] G. Slyusarev. *Aberration and Optical Design Theory,*. Taylor & Francis, 1984.

[152] C. Song, J. Song, and Q. Huang. HybridPose: 6D Object Pose Estimation Under Hybrid Representations. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 428–437. IEEE Computer Society, Jun 2020.

[153] P. Sturm, S. Ramalingam, J.-P. Tardif, and S. Gasparini. Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6:1–183, 01 2011.

[154] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2686–2694, 2015.

[155] Z. Sui, L. Xiang, O. C. Jenkins, and K. Desingh. Goal-directed robot manipulation through axiomatic scene estimation. *The Int. Journal of Robotics Research*, 36(1):86–104, 2017.

[156] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Proc. of the Thirty-First AAAI Conf. on Artificial Intelligence*. AAAI Press, 2017.

[157] S. A. Taghanaki, Y. Zheng, S. Kevin Zhou, B. Georgescu, P. Sharma, D. Xu, D. Comaniciu, and G. Hamarneh. Combo loss: Handling input and output imbalance in multi-organ segmentation. *Computerized Medical Imaging and Graphics*, 75:24–33, 2019.

[158] F. Tang, Y. Wu, X. Hou, and H. Ling. 3D Mapping and 6D Pose Computation for Real Time Augmented Reality on Cylindrical Objects. *IEEE Trans. on Circuits and Systems for Video Technology*, 30(9):2887–2899, 2020.

[159] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-class hough forests for 3D object detection and pose estimation. In *Computer Vision–ECCV*, pages 462–477. Springer, 2014.

[160] B. Tekin, S. N. Sinha, and P. Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 292–301. IEEE Computer Society, Jun 2018.

[161] G. Terzakis and M. Lourakis. A Consistently Fast and Globally Optimal Solution to the Perspective-n-Point Problem. In *Computer Vision – ECCV*, pages 478–494. Springer, 2020.

[162] Z. Terzopoulou and U. Endriss. The borda class: An axiomatic study of the borda rule on top-truncated preferences. *Journal of Mathematical Economics*, 92:31–40, 2021.

[163] H. Tjaden, U. Schwanecke, E. Schömer, and D. Cremers. A Region-Based Gauss-Newton Approach to Real-Time Monocular Multiple Object Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 41(8):1797–1812, 2019.

[164] F. A. Tobar and D. P. Mandic. Quaternion reproducing kernel hilbert spaces: Existence and uniqueness conditions. *IEEE Tran. on Information Theory*, 60(9):5736–5749, 2014.

[165] S. Todorovic and N. Payet. From contours to 3D object detection and pose estimation. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 983–990. IEEE Computer Society, Nov 2011.

[166] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. In *CoRL*, volume 87 of *Proc. of Machine Learning Research*, pages 306–316. PMLR, 2018.

[167] H.-Y. Tseng, S. D. Mello, J. Tremblay, S. Liu, S. Birchfield, M.-H. Yang, and J. Kautz. Few-Shot Viewpoint Estimation. In *BMVC*, 2019.

[168] L. Valenca., L. Silva., T. Chaves., A. Gomes., L. Figueiredo., L. Cossio., S. Tandel., J. Lima., F. Simões., and V. Teichrieb. Real-time Monocular 6DoF Tracking of Textureless Objects using Photometrically-enhanced Edges. In *Proc. of the 16th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pages 763–773. INSTICC, SciTePress, 2021.

[169] A. Veeraraghavan, R. Chellappa, O. Tuzel, and M. Liu. Fast directional chamfer matching. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1696–1703. IEEE Computer Society, Jun 2010.

[170] R. Verma. Fuzzy mabac method based on new exponential fuzzy information measures. *Soft Computing*, 25(14):9575–9589, Jul 2021.

[171] J. Vidal, C.-Y. Lin, and R. Martí. 6D pose estimation using an improved method based on point pair features. In *4th Int. Conf. on Control, Automation and Robotics (ICCAR)*, pages 405–409, 2018.

[172] P. Vishak, P. Sudheesh, and M. Jayakumar. A survey on nonlinear applications of modified particle filter. In *Int. Conf. on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1059–1063, 2017.

[173] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3338–3347, 2019.

[174] D. Wang, D. Tan, and L. Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, 22(2):387–408, Jan 2018.

[175] G. Wang, F. Manhardt, F. Tombari, and X. Ji. GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 16606–16616. IEEE Computer Society, Jun 2021.

[176] H. Wang, X. Lu, Y. Du, C. Zhang, R. Sadiq, and Y. Deng. Fault tree analysis based on topsis and triangular fuzzy number. *Int. Journal of System Assurance Engineering and Management*, 8(4):2064–2070, 2017.

[177] P. Wang, G. Xu, Z. Wang, and Y. Cheng. An efficient solution to the perspective-three-point pose problem. *Computer Vision and Image Understanding*, 166:81–87, 2018.

[178] Z. Wang, X. Li, D. Navarro-Alarcon, and Y.-h. Liu. A unified controller for region-reaching and deforming of soft objects. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 472–478, 2018.

[179] B. Wen, C. Mitash, B. Ren, and K. E. Bekris. se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 10367–10373, 2020.

[180] Y. Wen, H. Pan, L. Yang, and W. Wang. Edge Enhanced Implicit Orientation Learning With Geometric Prior for 6D Pose Estimation. *IEEE Robotics and Automation Letters*, 5(3):4931–4938, 2020.

[181] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. ElasticFusion. *Int. J. Rob. Res.*, 35(14):1697–1716, Dec 2016.

[182] P. Wu, Y. Lee, H. Tseng, H. Ho, M. Yang, and S. Chien. A Benchmark Dataset for 6DoF Object Pose Tracking. In *IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 186–191. IEEE Computer Society, Oct 2017.

[183] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *Proc. of Robotics: Science and Systems*, Pittsburgh, June 2018.

[184] D. Xu, Y. Xia, and D. P. Mandic. Optimization in quaternion dynamic systems: Gradient, hessian, and learning algorithms. *IEEE Tran. on Neural Networks and Learning Systems*, 27(2):249–261, 2016.

[185] T. Xu and W. Takano. Graph stacked hourglass networks for 3d human pose estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 16100–16109, 2021.

[186] S. Zakharov, I. Shugurov, and S. Ilic. DPOD: 6D Pose Object Detector and Refiner. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pages 1941–1950. IEEE Computer Society, Nov 2019.

[187] M. Zappel, S. Bultmann, and S. Behnke. 6D Object Pose Estimation Using Keypoints andăPart Affinity Fields. In *RoboCup 2021: Robot World Cup XXIV*, pages 78–90. Springer, 2022.

[188] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *The Int. J. of Robotics Research*, 41(7):690–705, 2022.

[189] S. Zhang, W. Zhao, Z. Guan, X. Peng, and J. Peng. Keypoint-graph-driven learning framework for object pose estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1065–1073. IEEE Computer Society, Jun 2021.

[190] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239. IEEE Computer Society, Jul 2017.

[191] W. Zhao, S. Zhang, Z. Guan, W. Zhao, J. Peng, and J. Fan. Learning Deep Network for Detecting 3D Object Keypoints and 6D Poses. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14122–14130. IEEE Computer Society, Jun 2020.

[192] L. Zhong, X. Zhao, Y. Zhang, S. Zhang, and L. Zhang. Occlusion-Aware Region-Based 3D Pose Tracking of Objects With Temporally Consistent Polar-Based Local Partitioning. *IEEE Trans. on Image Processing*, 29:5065–5078, 2020.

[193] H. Zhou, T. Zhang, and J. Jagadeesan. Re-weighting and 1-Point RANSAC-Based P*n*P Solution to Handle Outliers. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 41(12):3022–3033, Dec 2019.

[194] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3D object detection and pose estimation for grasping. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3936–3943, 2014.

[195] H.-J. Zimmermann. *Fuzzy Set Theory - and Its Applications*, volume 2001. 01 2001.