UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Real-Time Object Pose Estimation

Author:
Denys MALETSKIY

Supervisor:
Dr. Taras KHAPKO

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Denys MALETSKIY, declare that this thesis titled, "Real-Time Object Pose Estimation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Education isn't something you can finish."

Isaac Asimov

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Real-Time Object Pose Estimation

by Denys MALETSKIY

Abstract

The primary purpose of this thesis is to create fast lightweight 3D CAD model-based tracking for 6 degrees-of-freedom object pose estimation and review current advancements in object pose tracking. The proposed approach is focused on running in real-time on low-performance devices with limited computation power.

Acknowledgements

I am very grateful to my supervisor Taras Khapko, who encourage me to do this research. He was very supportive during the whole project, and working with him was one of the most valuable experiences. Many thanks to the whole faculty of Ukrainian Catholic University for the most interesting 4 years of my life. Thanks to all the professors, whose enthusiasm inspired me and opened my eyes to so many new perspectives. I would like to thank my parents, who supported me during my whole life and made this thesis possible. Also, I would like to thank the Armed Forces of Ukraine for the possibility of finishing this thesis.

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **iv**

1 Introduction **1**
1.1 Problem Statement 1
1.2 Constrains 2
1.2.1 Textured object's 3D CAD model 2
1.2.2 Stationary objects 2
1.3 Requirements 2
1.3.1 Real-time performance 2
1.3.2 Memory and resources limits 2

2 Related Work **3**
2.1 Neural Network approaches 3
2.2 Classical Computer Vision 3

3 Background **4**
3.1 Camera Calibration 4
Camera Matrix 4
Lens Distortion 5
Projection 5

4 Proposed Method **6**
4.1 Initial object model preprocessing 6
4.2 Object pose tracking 7
4.2.1 Object localization 7
Template matching 7
4.2.2 Feature extraction 10
4.2.3 Pose estimation 11
4.3 Pose tracking pipeline 12

5 Experiments **13**
5.1 Experimental Setup 13
5.1.1 Dataset 13
5.1.2 Evaluation Metrics 13
5.2 Object Detection 14
5.3 Pose Estimation 14
5.4 Pose Detection Examples 15

6 Conclusion **16**

Bibliography **17**

List of Figures

1.1 Object pose estimation model . . . 1

3.1 Pinhole camera model. . . . 4
3.2 Lens distortions. . . . 5

4.1 Blender environment setup. . . . 6
4.2 Model preprocessing. Features extraction. . . . 7
4.3 Object Detection. Template matching. . . . 8
4.4 Object Detection. Template matching performance . . . 9
4.5 Feature Matching . . . 10
4.6 Pose estimation . . . 11
4.7 Object Tracking Pipeline . . . 12

5.1 Dataset objects . . . 13
5.2 Object Detection Recall . . . 14
5.3 Pose Estimation ADD . . . 15
5.4 Pose Detection Examples . . . 15

List of Tables

5.1 Pose Estimation Recall 14

List of Abbreviations

AR	**A**ugmented **R**eality
VR	**V**irtual **R**eality
AI	**A**rtificial **I**ntelligence
CAD	**C**omputer-**A**ided **D**esign
CPU	**C**entral **P**rocessing **U**nit
GPU	**G**raphics **P**rocessing **U**nit
DoF	**D**egrees **o**f **F**reedom
BVH	**B**ounding **V**olume **H**ierarchy
ROI	**R**egion **O**f **I**nterest
TM	**T**emplate **M**atching
PnP	**P**erspective-**n**-**P**oint
ADD	**A**verage **D**istance

Chapter 1

Introduction

One of the basic human cognitive skills is the ability to understand the 3D context of the environments around us. We can easily estimate the shape of the object, predict the distance to that object, do the object segmentation and instantly solve many other heavy computing tasks to understand the full context of the 3D world surrounding us. Researchers in the computer vision sphere try their best to learn how to solve these tasks with human accuracy or sometimes even overcome the limitations of the eye, however relying on human-built sensors and advances in artificial intelligence. Cognitive abilities that are natural to human beings are still a matter of research when it comes to AI but have the potential to unlock a variety of applications in spatial computing.

One of such cognitive abilities is object pose estimation, which is the task of reconstructing the object's position and rotation relative to the camera 3D coordinate system. The solution to this problem opens numerous possibilities for creating astonishing AR/VR applications with unique functionality and unlocks new frontiers in the robotics field.

1.1 Problem Statement

The main goal of this thesis is a 6 degrees-of-freedom object pose estimation using an available 3D CAD model of the object and an RGB image from the camera. Object pose consists of 6 parameters - 3-dimensional rotation vector and 3-dimensional translation vector - that uniquely define the object in the 3D camera coordinate system.

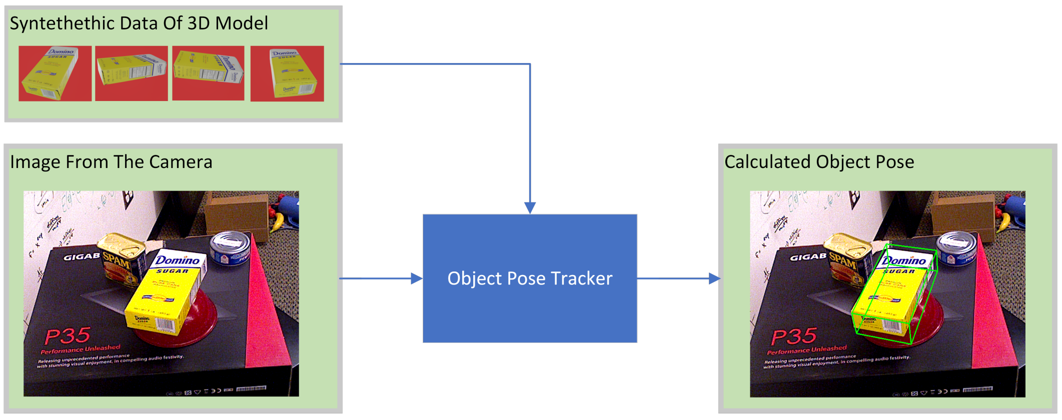


FIGURE 1.1: Object pose estimation model.

1.2 Constrains

1.2.1 Textured object's 3D CAD model

A high-quality textured object 3D CAD model must be provided for the high accuracy of object tracking. Based on the fact that the proposed approach uses the object's feature points (see more in 4.2.2), the 3D object itself should be rich with such feature points. Otherwise, the object will not be adequately detected (texture-less models cannot be used, and models without any high-contrast features, as well).

1.2.2 Stationary objects

The proposed in this thesis approach presumes that the object (which pose we try to estimate) stays stationary within the whole observation, and only the camera position is allowed to change. Support for moving objects is left for future consideration.

1.3 Requirements

The object pose estimation approach proposed in this thesis has 2 preliminary requirements: real-time performance and memory/resources limits.

1.3.1 Real-time performance

The proposed approach has to be fast enough for usage in real-time applications, even on low-performance devices with limited computational CPU and GPU power.

1.3.2 Memory and resources limits

As mentioned above, the tracking approach should be applicable to devices with limited computation resources, which means that approaches that depend on high-performance and memory-rich GPU are not being considered.

Chapter 2

Related Work

Object pose estimation is a well-studied field in computer vision. Nowadays, the approaches for object pose estimation can be divided into 2 categories: pure machine learning ones (which regress object pose directly from the image using Neural Networks) and mixed ones (where object descriptors/keypoints are being extracted first, using either Neural Networks or classical computer vision algorithms, and the 6 DoF object pose is being computed based on these descriptors).

2.1 Neural Network approaches

Several authors ([1], [2], [3]) propose state-of-the-art approaches with validation on well-know object pose datasets, such as LINEMOD, Occlusion-LINEMOD, and YCB-Video.

RNNPose ([1]) propose the recurrent neural network architecture that utilizes the 2D features from the camera image, 2D features of the reference object image (synthetic object's image), and the 3D features of the model (computed directly from the 3D point cloud of the object based on the 3D convolution described in [4]) and predicts the direct object pose. The proposed model is iterative so that after each iteration, the reference object image will be re-rendered with the corrected object pose.

SO-Pose ([2]) paper tries to solve the low accuracy limitation of end-to-end neural network approaches for pose estimation. The authors propose 2 stages architecture that combines an end-to-end neural network for direct pose estimation followed by the stage of pose refinement using a Perspective-n-Point (PnP) based algorithm.

DeepIM ([3]) model solves object-pose estimation using iterative pose refinement approach. Given the input of a test image with an initial pose, the model can calculate for the observed image the relative object pose. Later using that relative pose, the test image can be iteratively re-rendered with a corrected pose, improving pose estimation, and making the test and observed images more and more similar.

While these approaches provide great accuracy, they considerably lack generality. The models are re-trained for each specific object, with a significant dataset requirement in each case. Moreover, they typically require large amounts of GPU memory and resources to fulfill the real-time performance constraints.

2.2 Classical Computer Vision

On the other hand, solutions that are based on classical computer vision algorithms usually are independent of the chosen object model and are far less resource expensive. Therefore, they can be used in a variety of applications and support a larger range of devices.

Chapter 3

Background

3.1 Camera Calibration

Camera calibration is a process of estimating the camera lens and image sensor parameters. The estimated parameter, usually called the camera's intrinsics, consists of 4 parameters that define the camera matrix and 5 parameters that estimate the distortion of the camera lens.

Camera Matrix

The camera matrix is the matrix that can describe the projection of the 3D point $P = [X, Y, Z]^T$ in the camera coordinate system onto the image plane of the camera in the point $p = [u, v]^T$.

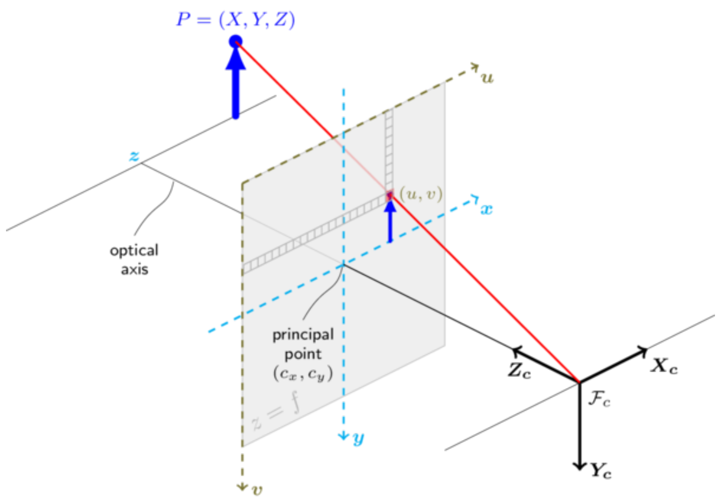


FIGURE 3.1: Pinhole camera model.

The camera's projection matrix that uses a pinhole camera model can be estimated using the camera's focal length and position of the principal point (intersection of an optical axis with the image plane).

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

Lens Distortion

The distortion of the lens is usually described by:

- radial distortion (that consist of barrel distortion and pincushion distortion) can occur if the camera lens are not equally bending the light
- tangential distortion can occur when the lens and sensor are not parallel but rotated with a slight angle.

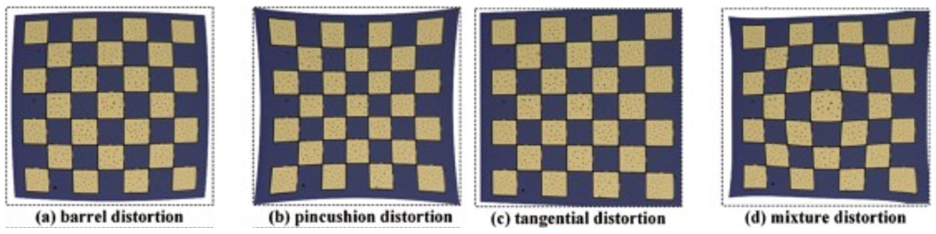


FIGURE 3.2: Lens distortions.

Mathematically lens distortion usually described using 3 parameters of radial distortion (k_1, k_2, k_3) and 2 parameters of tangential distortion (p_1, p_2):

$$\begin{aligned} X_{\text{undistorted}} &= X \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + [2 \cdot p_1 \cdot x \cdot y + p_2 \cdot (r^2 + 2 \cdot x^2)] \\ Y_{\text{undistorted}} &= Y \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + [p_1 \cdot (r^2 + 2 \cdot y^2) + 2 \cdot p_2 \cdot x \cdot y] \end{aligned} \tag{3.2}$$

where X and Y are the coordinates of the point ($P = [X, Y, Z]^T$) in the camera coordinate system.

Projection

The final pinhole projection model can be described using both lens distortion and camera matrix:

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X_{\text{undistorted}} \\ Y_{\text{undistorted}} \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{\text{undistorted}} \\ Y_{\text{undistorted}} \\ Z \end{bmatrix} \tag{3.3}$$

where u and v are the coordinates of a projected point in the camera image plane.

Chapter 4

Proposed Method

Object pose estimation approach described in this thesis consists of two parts:

- initial object model preprocessing;
- the actual object pose tracking.

The latter pose tracking part can be logically divided in three parts:

- object localization;
- feature extraction;
- pose estimation.

4.1 Initial object model preprocessing

Object model preprocessing aims to get the 3D keypoints and their projections at different object orientations and distances with respect to the camera.

The first step of model preprocessing is selecting the 3D viewing software. In this case, Blender was used because it has all the required 3D viewing functionality and rich support for Python scripting, which allows partially atomizing the preprocessing procedure.

After that, we can generate synthetic object images from different rotations and scales.

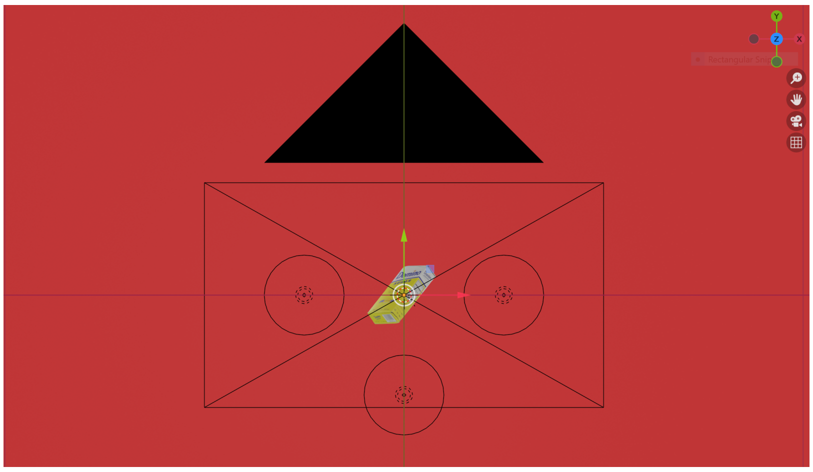


FIGURE 4.1: Blender environment setup.

Later the generated images will be processed to detect edges using a Canny edge detector [5]. After that SIFT feature detector [6] is used for extracting keypoints and corresponding descriptors.

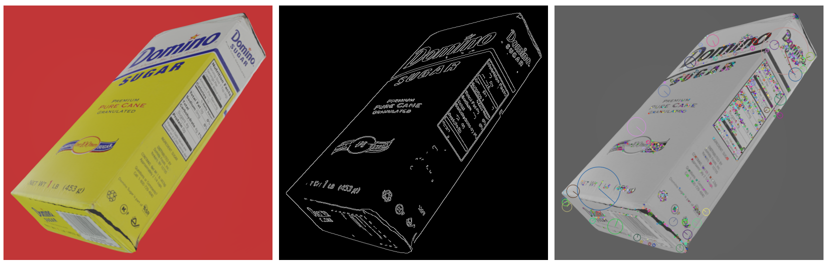


FIGURE 4.2: Model preprocessing. Features extraction.

After that, we need to calculate the 3D position for each 2D keypoint. In order to achieve this, a direction vector is calculated (starting at the camera origin and going through the point in the camera frame) for each 2D keypoint. That is done by unprojecting the 2D point in a homogeneous coordinate system using the inverse of the camera matrix (that is described in 3.1):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4.1}$$

Then, the intersection of each direction vector with the object model is calculated. That is done by constructing the Bounding Volume Hierarchy (BVH) Tree [7] from all the object's polygons and later searching for the nearest intersection in this tree.

After that, the preprocessing of the object's model is done. Now we stop working with the 3D CAD model and use only the generated templates (together with all the calculated data of the object) in the following sections of the pose estimation.

4.2 Object pose tracking

4.2.1 Object localization

The first iteration of the object localization is the hardest because there is no preliminary data on the object's location in the real world. Hence, a more exhaustive algorithm needs to be applied. Based on previous knowledge of the prepossessed model, we aim to find the initial crude pose of the model by comparing the image from the camera and our generated images of the object model from different orientations. That will give us a better starting point for object tracking.

Template matching

Template matching is the perfect technique for finding the object's position in the image using previously generated images during the preprocessing stage. During the matching procedure, the image from the camera will be matched with all existing templates and different scales of those templates. For object detection, 3 different approaches for template matching were tested to select the most suitable for this task.

RGB template is more computationally expensive but provides more data that can be used for better accuracy of the matching. The major limitation of this approach is the high dependency on the lighting as well as occlusions of the object. To partially overcome the lighting problem, correlation coefficients are used for template matching:

$$R(x,y) = \sum_{x',y'} ((T(x',y') - \mu_T) \cdot (I(x+x',y+y') - \mu_I)), \tag{4.2}$$

where T is the template, I is the image from the camera, μ_T is the template's mean value, and μ_I is the mean value of the camera's image.

Edges templates are binary images that contain knowledge of the shape of the object and its features. If the object's environment is clean and the edges of the object are well visible, the edges templates should be preferred for better matching performance. The matching score can be calculated using a cross-correlation of the template and the main image:

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y')), \tag{4.3}$$

where T is the template and I is the image from the camera.

Using the fact that both images are binary, the cross-correlation equation can be further optimized using bitwise AND operations:

$$R(x,y) = \sum_{x',y'} (T(x',y') \& I(x+x',y+y')), \tag{4.4}$$

Similarity measures is the another approach for template matching proposed in [8]. The proposed approach takes as an input the feature points of the template ($p'_i = (x'_i, y'_i)^T$) and main image ($p_i = (x_i, y_i)^T$) together with the direction vectors for each of those feature points ($d_i = (t_i, u_i)^T$ and $e_i = (v_i, w_i)^T$, corespondent). That allows to compute the score of the similarity between the 2 images as the average cosine of angles between the direction vectors, taking into account directions of the edges during matching:

$$\begin{aligned} s &= \frac{1}{n}\sum_{i=1}^{n} \frac{\langle d'_i, e_{q+p'} \rangle}{\|d'_i\| \cdot \|e_{q+p'}\|} \\ &= \frac{1}{n}\sum_{i=1}^{n} \frac{t'_i v_{x+x'_i,y+y'_i} + u'_i w_{x+x'_i,y+y'_i}}{\sqrt{t'^2_i + u'^2_i} \cdot \sqrt{v^2_{x+x'_i,y+y'_i} + w^2_{x+x'_i,y+y'_i}}} \end{aligned} \tag{4.5}$$

The similarity measurements approach was selected for template matching because it provides excellent accuracy, even in the situations of partial occlusions of the object, and does not depend on the intensity of the images.

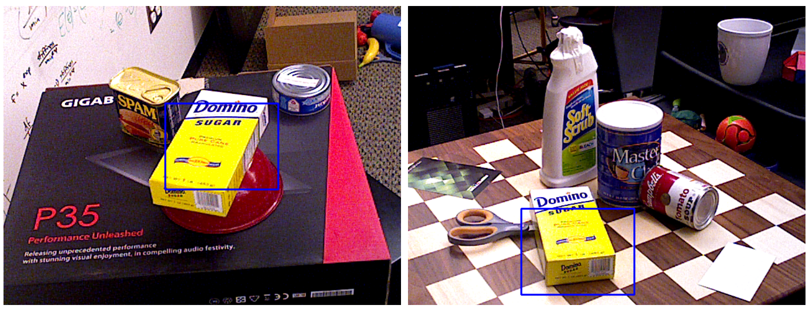


FIGURE 4.3: Object Detection. Template matching. For visualization the blue rectangular represent the location of the best match with template.

In order to use the template matching in a real-time application, the performance of the algorithm needs to be improved. The paper [8] proposes an additional criterion for early stopping the matching. Given the metric for similarity calculation:

$$s = \frac{1}{n} \sum_{i=1}^{n} \frac{\langle d'_i, e_{q+p'} \rangle}{\|d'_i\| \cdot \|e_{q+p'}\|} \tag{4.6}$$

The additional s_{min} score threshold can be added to construct constrain for all matching scores: $s > s_{min}$.

Therefore, for each s_j (the partial score for the first j direction vectors) the next constrain must be satisfied:

$$\frac{1}{n} \sum_{i=1}^{j} \frac{\langle d'_i, e_{q+p'} \rangle}{\|d'_i\| \cdot \|e_{q+p'}\|} > s_{min} - \frac{1}{n} \sum_{i=j}^{n} \frac{\langle d'_i, e_{q+p'} \rangle}{\|d'_i\| \cdot \|e_{q+p'}\|} \tag{4.7}$$

Using the fact that the cosine of the angle is in the range $[-1;1]$ the above constrain can be simplified to:

$$s_j > s_{min} - \frac{n-j}{n} \tag{4.8}$$

For even better performance, the template matching was further speed-up by resizing images (both camera's image and template image) using Gaussian pyramid [9]. Firstly, finding the object at the lowest level and then moving to the original image, but performing the search in a limited neighborhood instead of the whole image. In this case, we used just 1 level (effectively halving the image).

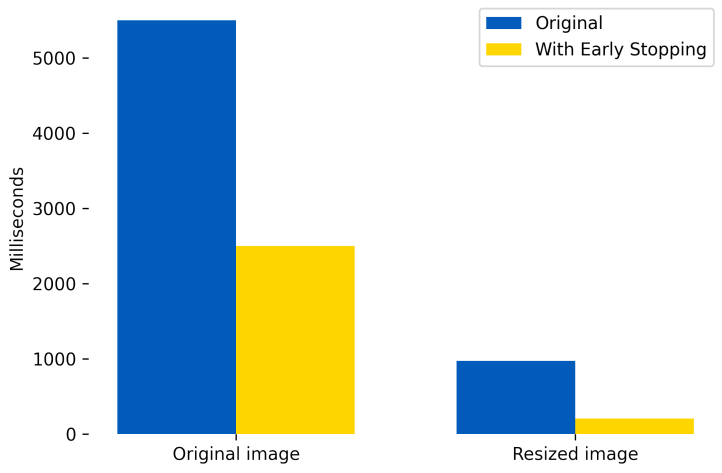


FIGURE 4.4: Object Detection. Template matching performance. The performance of original image (resolution is 640x480) matching with the template (resolution is 275x200) was tested with and without early stopping. The same was done for resized images.

After obtaining match scores for each template, the match with the maximum score is selected. If the similarity score exceeds the minimum threshold, the match is considered correct. This best match will be considered object location and used as ROI for object pose estimation in the following stages.

4.2.2 Feature extraction

The next pose estimation step is to extract the features in the previously detected (by template matching) region. The same algorithm for feature extraction must be used in this step as the one used in the preprocessing stage for generated templates. That is why the Canny edge detector [5] is used again to extract feature points (keypoints) and their correspondent descriptors.

After the keypoints with the correspondent descriptors are calculated, they need to be matched with previously calculated keypoints of the template image. In order to improve the performance of the feature matching, the k-nearest-neighbor-based algorithm was chosen. Afterward, matched features were filtered using Lowe's ratio test [6] to extract only the most accurate matches.

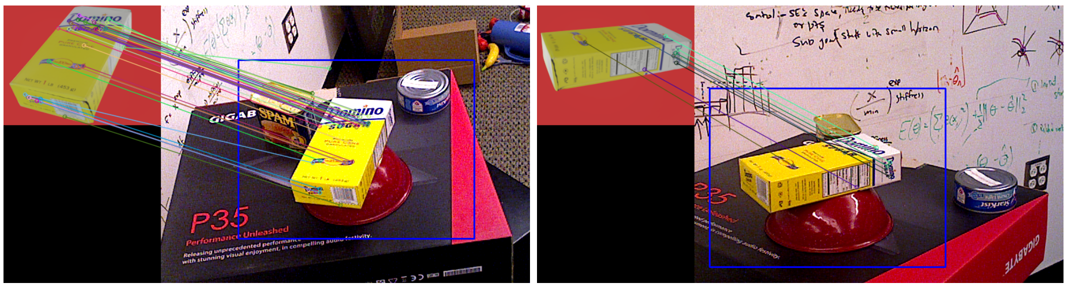


FIGURE 4.5: Feature Matching. For visualization the blue rectangular represent the enlarged bounding box computed in template matching step. The multi-color lines represents the matches of the features calculated in the template image (on the left) and image from the camera (on the right).

The result of the matching keypoints is a direct correspondence of 2D keypoints in the image (projections of the object points) to 3D keypoints of the 3D object model (with respect to the object's coordinate system). This can be achieved because, in the preprocessing stage, all the generated templates were saved with calculated 2D object features and correspondent 3D coordinates of those features in the object's coordinate system.

4.2.3 Pose estimation

For the pose estimation Perspective-n-Point (PnP) algorithm is used to estimate object pose. PnP is a well-known computer vision optimization problem that computes the pose (extrinsic) of the object using known 3D points of the object (in the object's coordinate system) and their 2D projections onto the camera image plane. The optimization problem can be formulated as the minimization of the distance between the projection of the 3D points using object pose ($\hat{p}_i = [\hat{u}, \hat{v}, 1]^T$) and their detected 2D points ($p_i = [u, v, 1]^T$):

$$\min \| \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \begin{bmatrix} \hat{u} \\ \hat{v} \\ 1 \end{bmatrix} \| \tag{4.9}$$

,

where

$$\begin{gathered}\begin{bmatrix} \hat{u} \\ \hat{v} \\ 1 \end{bmatrix} z = \mathbf{K}\,\mathbf{T}_{\mathbf{W}}^{\mathbf{C}} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \\ \begin{bmatrix} \hat{u} \\ \hat{v} \\ 1 \end{bmatrix} z = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}\end{gathered} \tag{4.10}$$

For better accuracy of the pose estimation, the modification of the PnP algorithm was used that firstly uses the RANSAC algorithm [10] to illuminate all the outlier features, followed by the Levenberg-Marquardt algorithm ([11], [12]) for the object pose optimization.

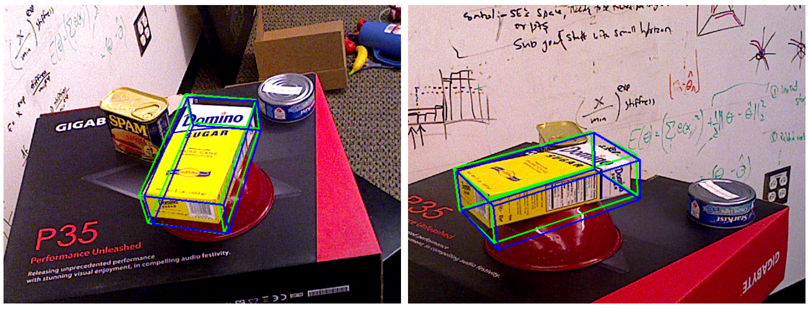


FIGURE 4.6: Pose estimation. For pose visualization the green boxes represents the ground-truth object poses and blue boxes are the detected object poses.

4.3 Pose tracking pipeline

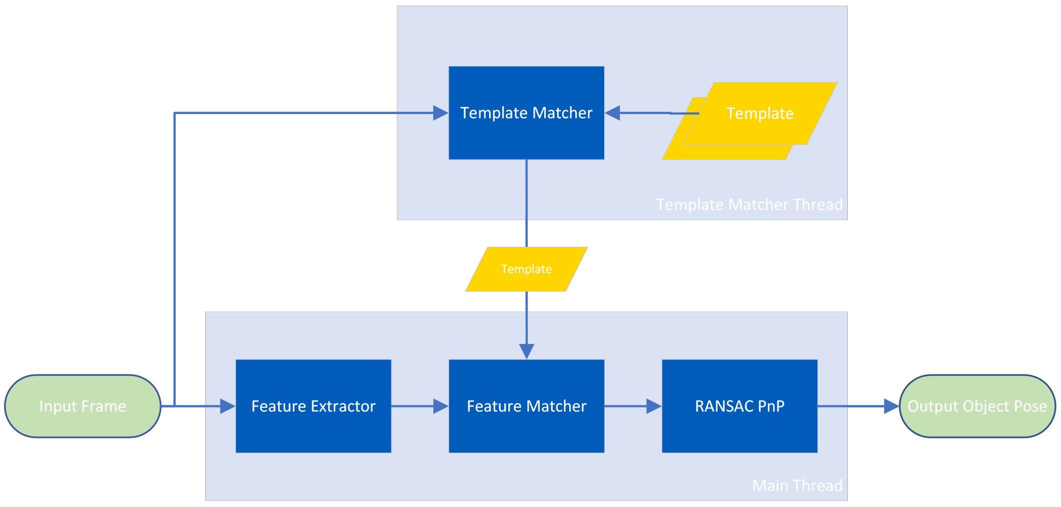


FIGURE 4.7: Object Tracking Pipeline. For visualization, the green ellipses represent the input and output of the pipeline, when dark blue rectangles represent the computation blocks, light blue rectangles represent the processor's threads, yellow parallelograms represent the pre-calculated data, and the arrows represent the flow of the data between the blocks.

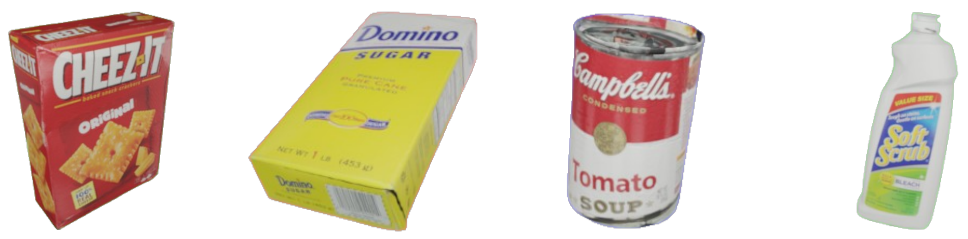
Chapter 5

Experiments

5.1 Experimental Setup

5.1.1 Dataset

The performance of the proposed method was tested on YCB-V (YCB-Video) dataset [13]. YCB-Video is a large-scale dataset for 6 degrees-of-freedom object pose estimation that contains accurate poses for 21 objects captured in 92 videos. For this experiment, a subset of 4 objects in 12 videos was manually chosen from the original 92 videos.



(A) Cheez-It (id 2) (B) Domino (id 3) (C) Campbell (id 4) (D) Soft Scrub (id 12)

FIGURE 5.1: Dataset objects with assigned names and id in YCB-V dataset.

5.1.2 Evaluation Metrics

The Object Detection accuracy was measured using Intersection-over-Union (IoU) metric. The detection was considered correct if the prediction and ground-truth IoU score exceeded 50%. The accuracy of the object pose estimation was measured using Average Distance (ADD) [14]. ADD calculates the average distance between the model points transformed with ground-truth object pose and estimated object pose. The estimated object pose is considered correct if the ADD metric stays below 50mm.

5.2 Object Detection

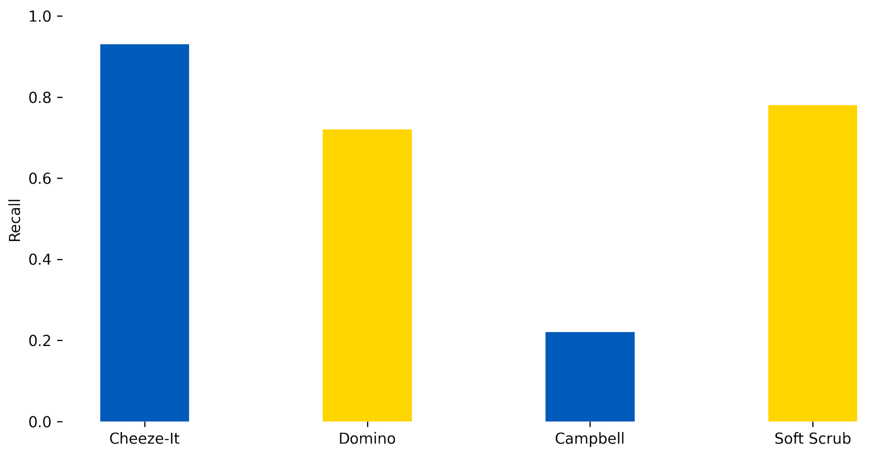


FIGURE 5.2: Object Detection Recall. Recall shows the ration of true-positive detections over all labeled ground-truth detections. For object detection IoU metric was used, the detection was considered correct if the prediction and ground-truth IoU score exceed 50%.

5.3 Pose Estimation

Object	**ADD (%) ↑**	**TP ADD (%) ↑**
Cheez-It	54.66	58.57
Domino	58.66	81.48
Campbell	2.66	11.74
Soft Scrub	30.66	38.98
Average	36.66	47.66

TABLE 5.1: Pose Estimation Recall. To measure the pose estimation accuracy Average Distance (ADD) [14] was used. The estimated object pose is considered correct if the ADD metric stays below the 50mm. ADD metric represent the ration of true-positive pose estimations over all the all labeled ground-truth poses. True-Positive ADD (TP ADD) metric represent the ration of true-positive pose estimations over all true-positive object detections.

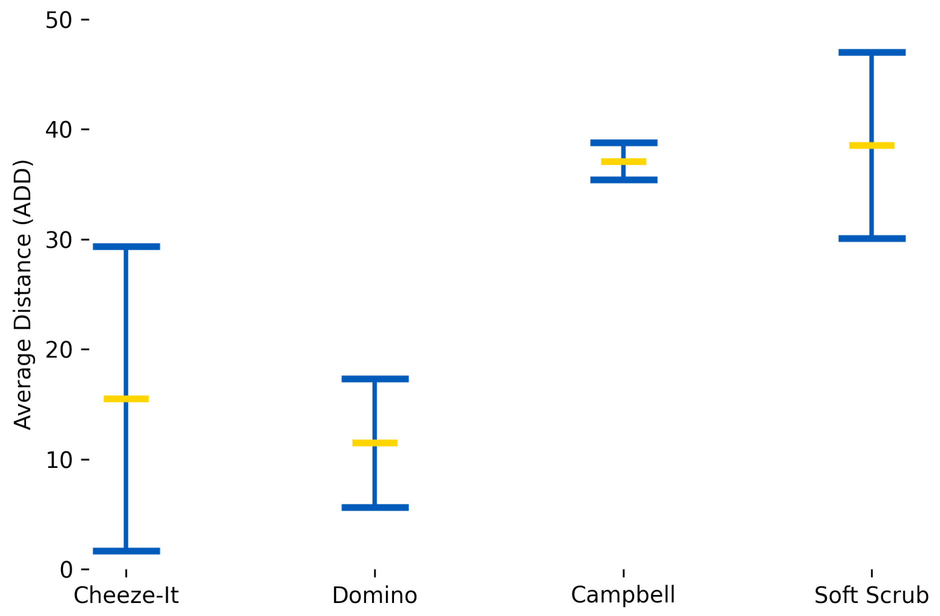


FIGURE 5.3: Pose Estimation ADD. The yellow lines represent the mean ADD error per each object (in mm). The blue error bar represent the standard deviation of ADD error per each object (in mm).

5.4 Pose Detection Examples

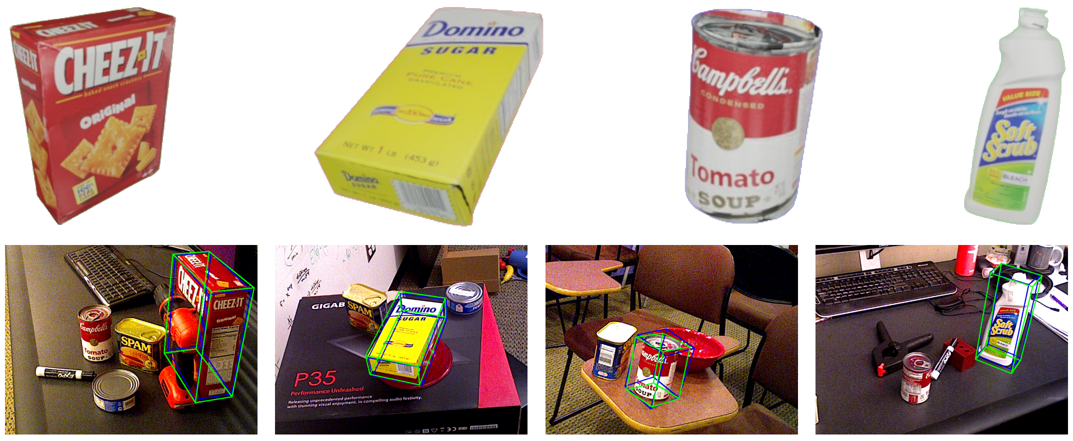


FIGURE 5.4: Pose Detection Examples. The green cuboid in images represents the ground-truth object pose. The blue cuboid represents the predicted object pose.

Chapter 6

Conclusion

This thesis proposes a lightweight real-time solution for 6 degrees-of-freedom (DoF) object pose estimation based on the object's textured 3D CAD model. The thesis demonstrates that classical computer vision algorithms can achieve high accuracy of object pose estimation while preserving real-time performance requirements.

Bibliography

[1] Y. Xu, K.-Y. Lin, G. Zhang, X. Wang, and H. Li, *Rnnpose: Recurrent 6-dof object pose refinement with robust correspondence field estimation and pose optimization*, 2022. DOI: 10.48550/ARXIV.2203.12870. [Online]. Available: https://arxiv.org/abs/2203.12870.

[2] Y. Di, F. Manhardt, G. Wang, X. Ji, N. Navab, and F. Tombari, *So-pose: Exploiting self-occlusion for direct 6d pose estimation*, 2021. DOI: 10.48550/ARXIV.2108.08367. [Online]. Available: https://arxiv.org/abs/2108.08367.

[3] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6d pose estimation," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 657–678, 2019. DOI: 10.1007/s11263-019-01250-9. [Online]. Available: https://doi.org/10.1007%2Fs11263-019-01250-9.

[4] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, *Kpconv: Flexible and deformable convolution for point clouds*, 2019. DOI: 10.48550/ARXIV.1904.08889. [Online]. Available: https://arxiv.org/abs/1904.08889.

[5] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986. DOI: 10.1109/TPAMI.1986.4767851.

[6] D. Lowe, "Distinctive image features from scale-invariant keypoints.," *International Journal of Computer Vision*, no. 60, 91–110, 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.

[7] I. Wald, S. Boulos, and P. Shirley, "Ray tracing deformable scenes using dynamic bounding volume hierarchies," *ACM Trans. Graph.*, vol. 26, Jan. 2007. DOI: 10.1145/1189762.1206075.

[8] C. Steger, "Occlusion, clutter, and illumination invariant object recognition," *International Archives of Photogrammetry and Remote Sensing*, vol. 34, Sep. 2003.

[9] S. L. Tanimoto, "Template matching in pyramids," *Computer Graphics and Image Processing*, vol. 16, no. 4, pp. 356–369, 1981, ISSN: 0146-664X. DOI: https://doi.org/10.1016/0146-664X(81)90046-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0146664X81900460.

[10] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. [Online]. Available: /brokenurl#http://publication.wilsonwong.me/load.php?id=233282275.

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

[12] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. DOI: 10.1137/0111030. eprint: https://doi.org/10.1137/0111030. [Online]. Available: https://doi.org/10.1137/0111030.

[13] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, *Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes*, 2017. DOI: 10.48550/ARXIV.1711.00199. [Online]. Available: https://arxiv.org/abs/1711.00199.

[14] S. Hinterstoisser, V. Lepetit, S. Ilic, *et al.*, “Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes,” in *Computer Vision – ACCV 2012*, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562.