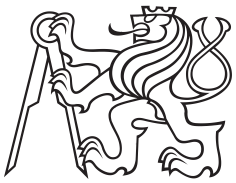
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Object Localization by Spatial Matching of High-level CNN Features

Bc. Lukáš Majer

Supervisor: Ing. Viktor Kozák
Supervisor–specialist: Ing. Libor Preučil CSc.
Field of study: Cybernetics and Robotics
May 2021

Acknowledgements

I want to thank my supervisor Ing. Viktor Kozák, for his patience and guidance, especially at the most trying times. My work could never be accomplished without the support and equipment provided by the Intelligent and Mobile Robotics Group. Moreover, none of this would be possible without the work of Luis Gomez Camara, PhD. Last but not least, I am grateful for all the support provided by my girlfriend.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 16, 2021

.....

signature

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 16. května 2021

.....

podpis autora práce

Abstract

This work deals with an application of matching of high-level CNN features for object pose estimation from camera images. First, research of the current state-of-the-art in object pose estimation is conducted. Then a solution based on an improved version of an existing semantic spatial matching framework is proposed. The result is a functional pipeline for object pose estimation, which is then compared against the current state-of-the-art methods.

Keywords: object pose estimation, object localization

Supervisor: Ing. Viktor Kozák
CIIRC B-324,
Jugoslávských partyzánů 3,
16000 Praha 6

Abstrakt

Tato práce se zabývá aplikací porovnávání aktivací konvolučních neuronových sítí pro odhad pozice objektu z kamerových obrázků. Nejprve je proveden výzkum současného stavu poznání pro odhad pozice objektu. Poté je navrženo řešení založené na vylepšené verzi existujícího softwarového rámce pro sémantické a prostorové porovnání. Výsledkem je funkční software pro odhad pozice objektu, který je poté porovnán se současnými nejmodernějšími metodami.

Klíčová slova: odhad pozice objektu, lokalizace objektu

Překlad názvu: Lokalizace objektů pomocí prostorové informace vyšších vrstev CNN

Contents

Figures

Tables

Chapter 1

Preliminaries

1.1 Introduction

This thesis addresses the problem of object pose estimation from camera images. Object pose estimation sometimes denoted object localization, is the task of detecting the 6D pose of an object, including its location and orientation. Object localization is a highly relevant research topic in robotics. A fast, robust, and scalable solution will significantly impact the industry, such as production lines, human-robot interaction, and more. Calculating pose from RGB images is considered a complex problem, mainly because of the loss of information about depth. Another option is to use RGB-D cameras. The problem is the high cost of industrial-grade equipment or low resolution for Kinect-like devices. Moreover, introducing depth also brings the curse of dimensionality and thus is not always possible to perform in real-time.

Traditionally, the problem has been tackled by matching local 2D features such as SIFT[36], SURF[8] or ORB[47] with the corresponding 3D model. This leads to a well-known problem of Perspective-n-Point [17]. The main problem with these methods is that they are posed to fail with texture-less objects. Another option is to use template matching [39], which work by creating a database of template images and finding the most similar one in the database. This has been proven to work sufficiently well, and we will use some core ideas to devise our own approach.

In recent years many solutions have been proposed with varying degree of success. One of the most prominent groups of algorithms is the ones based on machine learning, which have been deployed to solve such a task with promising results. Convolutional neural networks [33] being by far the most popular one. Still, general pose estimation of objects from a single RGB

image remains an unsolved problem and have been a subject of multiple international contests as an underlying problem [4], or main objective [25].

1.2 Goal

The primary goal of our work is to research state-of-the-art of object pose estimation and utilize such knowledge to design and develop a pipeline for object pose estimation based on an SSM-VPR framework [11], which is, in fact, state-of-the-art for the task of image retrieval. This framework was initially used for visual place recognition, but we hope to utilize the core principles and transfer them to the task of object pose estimation. Moreover, it is desirable to explore the advantages and limitations of using SSM for object pose estimation.

The secondary goal is the software development and advancement of the former SSM-VPR framework. As mentioned before, it was primary designed for visual place recognition. Thus it is necessary to re-implement or change several functionalities. From this point of view, it is desirable to make the re-implementation as general as possible to allow the usage of the said framework for multiple computer vision problems.

1.3 Structure

This thesis is divided into three main parts. The first part provides a rigorous mathematical definition of the task while pinpointing underlying assumptions. A brief overview of state-of-the-art algorithms follows a problem statement with an emphasis on similar approaches.

The next part deals with pose estimation pipeline design and implementation. Core ideas and considerations are presented here, as well as related principles. Moreover, the implementation is briefly analyzed with emphasis on the core functionalities. We also discuss changes in the SSM-VPR framework and highlight its improvements.

The last section contains an evaluation of the proposed algorithm and a comparison against state-of-the-art approaches. This also includes discussing a choice of metrics and their respective properties. Furthermore, these results are commented on and put into context.

Chapter 2

Problem definition

2.1 An opening example

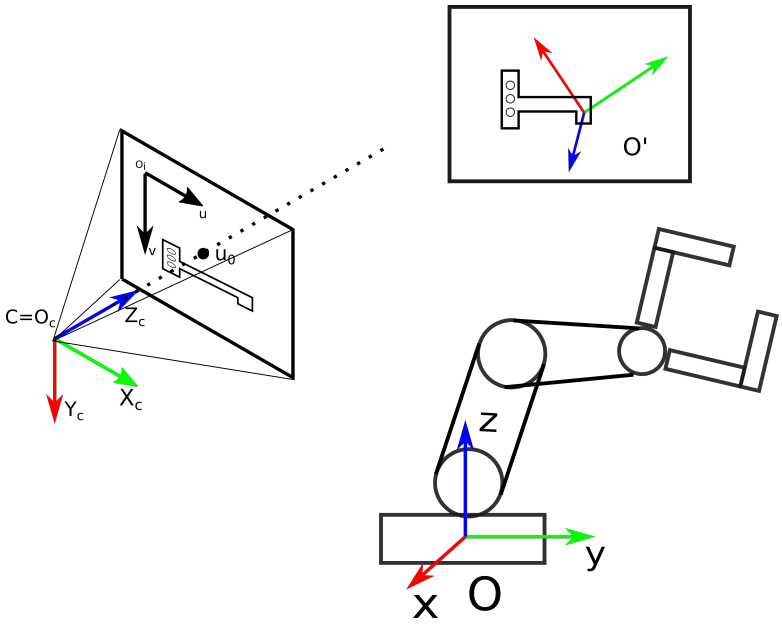
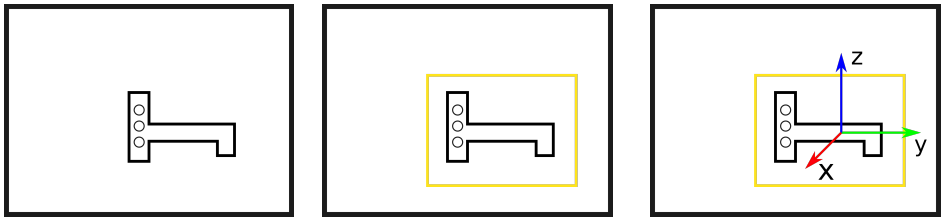


Figure 2.1: Robot arm is required to pick-up an object.

In this section, we aim to emphasize the importance of object pose estimation with an opening example. Let us start with a situation illustrated in figure 2.1. We have a robot manipulator with a gripper, and we want to pick up an object, but we have no way of determining transformation from O to O'. This is a relatively common robotic problem, called random bin picking in the literature. A possible solution is to add a sensor with a predetermined location (position and orientation of C relative to the robot base O are known). One such sensor can be an inexpensive RGB camera.

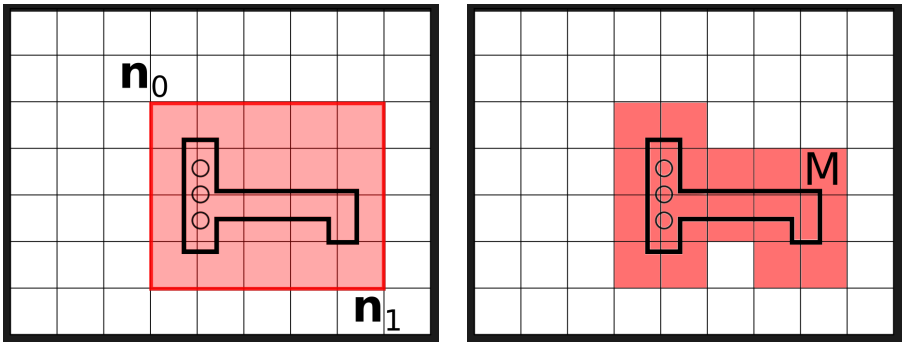
This leads us to the problem of estimating the position and orientation of the object relative to the camera. An illustration can be seen in figure 2.2 and can be divided into three separate stages. First we have to capture the image 2.2a, then we need to detect object 2.2b and based on that we can estimate object position 2.2c. Such decoupling separates the problem of finding an object and determining its relative localization. This separation allows us to focus on the task of estimating object poses. We will not concern ourselves with the task of capturing images as we will almost exclusively make use of public datasets. The main focus of our work will the task object pose estimation, but we will briefly mention algorithms used to solve object detection.



(a) : image capture **(b) :** object detection **(c) :** object localization

Figure 2.2: Example of object localization procedure.

Input to the image capture stage is a 3D scene, which is then projected by the camera sensor onto an image. Throughout the thesis, we will denote the image as I and represent it either as a set of pixels or a matrix of intensities interchangeably. The object detection stage takes the image I as an input and outputs instances of detected objects. Object instance consists of the object identifier and its detection inside the image I. Two common ways to represent detection mask and bounding box as illustrated in figure 2.3. A mask is a binary image of the same size as I with ones at the places where an object was detected. Throughout the thesis, we will denote it as a set of pixels M. A bounding box is a tuple of two points representing upper left and lower right corners. Throughout the thesis we will denote bounding box as $(\mathbf{n}_0, \mathbf{n}_1)$. The problem of converting between these two representations is straightforward. Converting the mask M to bounding box $(\mathbf{n}_0, \mathbf{n}_1)$ is equivalent to finding axis-aligned minimum bounding box within which all the points $\mathbf{p} \in M$ lie. Bounding box $(\mathbf{n}_0, \mathbf{n}_1)$ already defines a set of pixels M and thus the conversion is trivial.



(a) : bounding box **(b) :** mask

Figure 2.3: Illustration of object detection representation.

2.2 Formal definition

2.2.1 Overview

The main objective is to estimate the orientation and translation of an object from a single image. An input is a single image RGB image with detected objects (detection can be a bounding box or binary mask). Position and orientation in space for each detected object is the desired output. Additional knowledge which can be utilized is either textured 3D models or labelled images with known poses of said objects.

We pose no restriction on the uniqueness or number of occurrences of objects of the same class. Current literature labels this task as ViVo, which stands for *varying number of instances of a varying number of object*. Moreover, we allow a low degree of occlusion, which is caused when two objects overlap or a view of an object is obstructed.

2.2.2 Mathematical formulation

Let us have an image I taken with camera C and a set $L = \{o_1, o_2, ..., o_n\}$, where o_i is an object instance and n is a number of objects present in the image I. The goal of object pose estimation is to find a mapping K such as $K : o_i \rightarrow P_i \quad \forall o_i$, where $P_i \in SE(3)$ is the respective pose of object o_i. Object instance o_i is defined as tuple (k, M), where k is object class and M is a set of pixels, such as $M \in I$. Also if $o_i = (k_i, M_i)$, $o_j = (k_j, M_j)$ and $k_i \neq k_j$, then $M_i \neq M_j$.

2.2.3 Machine learning context

In order to unify our formal definition with the current state-of-the-art, we provide another definition from the machine learning point of view. Let us define two phases, the training phase and the test phase.

During training/offline phase each object class k_i gets assigned a data collection $T_i = \{(I_1, M_1, P_1), (I_2, M_2, P_2), ..., (I_m, M_m, P_m)\}$, where I_j is a training image and P_j is associated object pose, M_j is the object detection and m is number of training images. Optional input consists of a 3D object model S_i. Throughout this thesis, the tuple (I_m, M_m, P_m) will be referred to as "an annotated image" for the sake of readability. Such a process is often called supervised learning in literature.

During test/online phase an input is a pair of I and L. Expected output are object poses P_i $\forall o_i \in L$. Throughout this thesis, the I will be interchangeably referred to as "an (unannotated) image".

2.2.4 Constraints

While the definition provided in the previous section is sufficient to formulate the task, it is desirable to fully constraint the problem and include/exclude borderline cases, e.g. pose estimation of pliable objects. For this reason, we provide the following list of assumptions.

- Camera C with the same focal length is used for training and testing.
- Objects o_i are in focus.
- Image I was taken without any or negligible motion, thus we do not need to consider the effect of rolling shutter.
- Object o is rigid, non deformable.
- The list of detected objects L is assumed to be correct.
- The training data collection T is also assumed to be correct.
- Object marker M can only have small errors ($< 10\%$ pixels).

2.2.5 Challenges

As mentioned before, estimating object pose from a single RGB image is considered to be a difficult problem. The main issue is the lack of depth

measurement when capturing the image, which has to be compensated for with prior knowledge. There has been research effort [32] put into directly estimating depth from monocular camera images, but the resulting precision is not sufficient for 6D pose estimation.

Another challenging problem is an inconsistency between training and testing images. Any successful algorithm has to tackle changes in illumination, relative distance and orientation. Furthermore, the mapping K between input and output is highly non-linear. Because of its nature, it is difficult to come up with a general yet effective mathematical model.

Mentioned issues have been partly, but not completely, solved by the use of machine learning with heavy use of artificial neural networks. Such advancement comes at the price of introducing a *black-box* model. This is such a model that studying its structure will not provide any insights on the structure of the mapping being approximated. Using an algorithm with an unknown structure is problematic in areas where reliability and predictability are critical.

Chapter 3

State of the art

This chapter gives an overview of state-of-the-art methods for object pose estimation from single RGB images. Although multiple types of algorithms are mentioned, emphasis is given to approaches based on convolutional neural networks to reflect the current state-of-the-art. Because CNNs are heavily featured in the following parts of the thesis, we only mention the core ideas in this chapter and leave the detailed explanation for later.

3.1 Context

Because of the potential industrial impact, object pose estimation has been a target of multiple challenges over the years. Apart from yearly challenges at the *International Workshop on Recovering 6D Object Pose* we want to explore other notable examples. One of the first competition where pose estimation has been featured as a sub-task was Amazon Picking Challenge[4]. Another challange was SIXD Challenge 2017 [5], although challenging it covered only the task of a **s**ingle **i**nstance of a **s**ingle **o**bject (SiSo) localization.

One of the most recent one is BOP [25]. Organized two consecutive years in a row between 2019 and 2020, it is considered the benchmark of the current state-of-the-art. It comprises several challenging datasets [9, 23, 15, 27, 14] and features multiple categories for submission. Results of the said challenge were then presented on *European Conference on Computer Vision 2020*. A noteworthy mention is that most of the best-performing methods were based on deep convolutional networks.

Datasets are an important part of the state-of-the art research. As mentioned before, the dataset consists of annotated images and 3D models. Examples of the widely known datasets are T-LESS [23], and YCB-Video

introduced in [53]. Because capturing annotated images is time-consuming and sometimes even impractical, a lot of researchers started using synthetic images for training and real images for evaluation. One of the most common synthetic datasets used to compare object pose estimation frameworks is LINEMOD [10, 9]. A more in-depth overview of the used datasets will be provided in chapter 6.

3.2 Methods

The most recent state-of-the-art is dominated by the use of machine learning as supported by [25] and such methods will be the main focus of our overview. Nevertheless, we will also provide a brief description of algorithms not based on machine learning, mainly because our own proposed solution draws core ideas from some of them.

3.2.1 Traditional approaches

As mentioned in chapter 1.1, there are two kinds of approaches. The first one is based on the idea of finding distinct 2D key-points inside an input image I and matching them to 3D key-points from object model S_i. With matches found, the task reduces to a well-known Perspective N Point problem [17]. The solution of PnP is well studied and can be solved by, among others, employment of **Ran**dom **Sa**mple **C**onsensus (RANSAC) paradigm. The main challenge is to find a suitable matching model for 2D-3D correspondences. This is not an easy task and can be made even more difficult with occlusion. Moreover, a lot of handcrafted features require rich surface texture. As a result, they fail with texture-less objects.

Other approaches rely on template matching, where the object pose is estimated by comparing the input image I with templates from a reference database based on a predefined similarity metric. These templates encode information about the viewpoint of an object, which is then utilized to determine the final pose. One shortcoming is that similarity metrics are not robust against heavy occlusions, which can lead to false viewpoint classification. An example of such an approach is [21], where templates are created by sampling the viewpoints from the hemisphere. Selected features used for template creation are colour gradients and surface normal. Another option is to use 2D key-points for template generation as shown in [37], where images were clustered by SIFT[36] feature similarity.

3.2.2 Machine learning approaches

Current leading approaches solve pose estimation by training deep networks to either regress both rotation and translation from image directly e.g. CosyPose[31] and PoseCNN[53] or to construct 2D-3D correspondences with the use of machine learning and solve it as a PnP problem e.g Pix2Pose[41] and DPOD [54]. Estimating pose directly requires an elaborate refinement step. On the other hand, indirect approaches are more prone to fail with heavy occlusion and degenerate configurations. For training, networks can either use labelled image, 3D model or a combination of both, as mentioned in chapter 2.2.3. Important or otherwise representative cases will be described in the following sections.

A typical deep learning technique utilized is called transfer learning. It consists of taking a model trained for a specific task and reusing it as the starting point for a model on another task. General, although slightly abstract architecture can be divided into two parts backbone and head. The backbone part extracts features from the input image, and the head part performs regression of either rotation, orientation or a combination of both. A common approach is to take some off the shelf network, which is pre-trained on a large dataset [13] and use its backbone part. The backbone is then connected to a new head part, specifically designed for object pose estimation.

CosyPose

CosyPose[31] is highly regarded as one of the best frameworks for object pose estimation, mainly because of its performance in BOP challenge[25]. It an example of the direct approach. It is based on DeepIM [34] and consists of two networks, where rotation and translation are estimated simultaneously. Both of these networks share the same architecture but differ in parameters used during training. The first one is used for coarse object localization and the second one iteratively refines the pose estimates.

The backbone of the network is the EfficientNet[52] followed by spatial average pooling and a fully connected layer. Final layer outputs translation vector t and two vectors e_1, e_2 used to parameterize rotation, as described in [56]. The training phase utilizes ADD-S loss [53], with minor adjustments. Training is performed with standard Adam optimizer [29].

DPOD

Dense **P**ose **O**bject **D**etector [54] is an indirect approach. Pipeline architecture can be seen in figure 3.1. During the training phase, an annotated input image is fed to the encoder-decoder network. The encoder-decoder part learns to estimate 2D-3D correspondences between the input and corresponding 3D model. Proposed correspondences are then used to solve the PnP problem using the RANSAC matching scheme. This initial estimate is then further refined by an independently trained regression network.

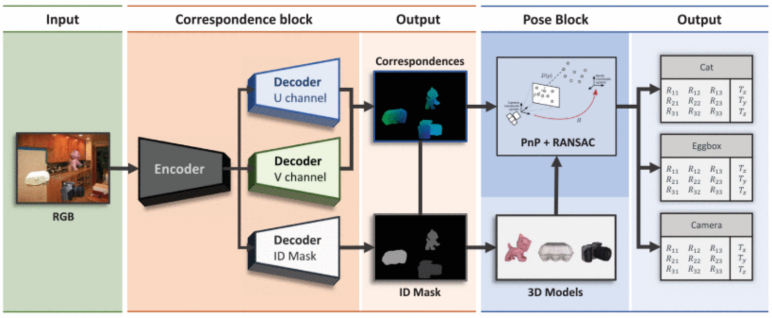


Figure 3.1: DPOD architecture. [54]

CDPN

CDPN [35], short for **C**oordinates-Based **D**isentangled **P**ose **N**etwork, is a combination of direct and indirect approach. The network consists of one shared backbone and two independent heads, one for translation and the other one for rotation. Translation head learns to directly regress translation vector. The rotation head is trained to estimate 2D-3D correspondences, which is then used to solve PnP via RANSAC.

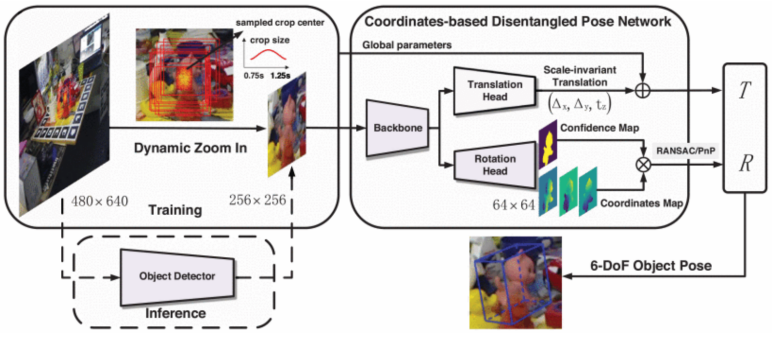


Figure 3.2: CDPN architecture. [35]

Chapter 4

Prerequisites

This chapter will go over the necessary knowledge required to understand our proposed solution.

4.1 Camera model

This section will go over the camera sensor model we will use throughout the whole thesis. It is a standard pinhole camera model as defined in [49] with the notation we devised in our previous work [38]. As illustrated in figure 4.1 the origin O_c, which stands for camera coordinate system, This coordinate system is positioned in a way that Z_c axis is aligned with camera optical axis with X_c axis facing to the right. The coordinate system with origin O_o represents the object coordinate system, and the coordinate system with origin o_i represents a normalized image plane. The last coordinate system represents an image influenced by intrinsic camera parameters, with origin o_a located on the image plane. The extrinsic camera matrix representing transformation between O_o and O_c can be expressed using homogeneous coordinates as

$$[\mathbf{R}\ \mathbf{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

where parameters r_{ij} form a rotation matrix and parameter t_k form a translation vector. The projection from the camera coordinate frame O_c to normalized image plane o_i can be written as

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.2}$$

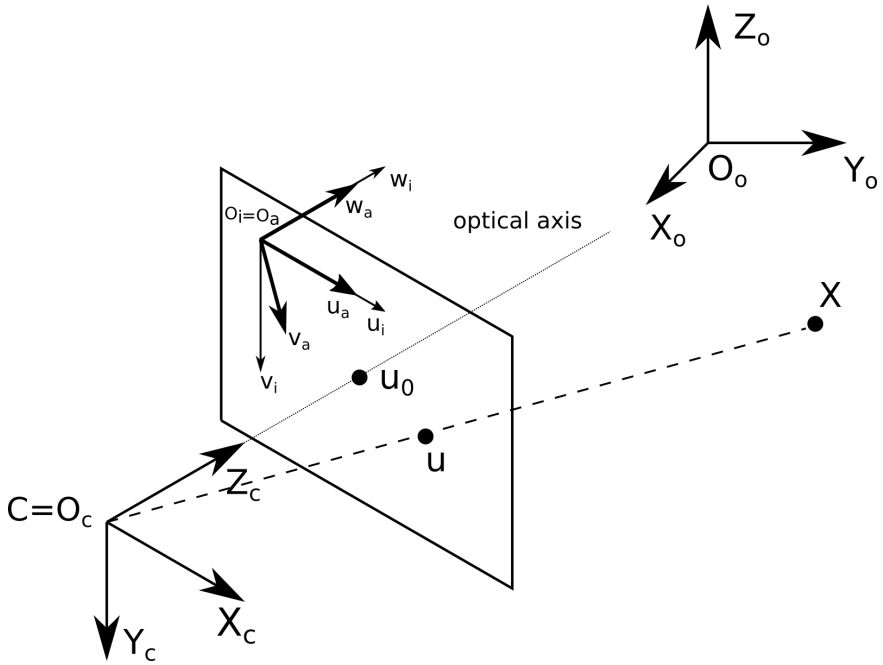


Figure 4.1: Coordinate system definition for pinhole camera model. [38]

The intrinsic camera matrix representing the transformation from the coordinate system o_i to the coordinate system o_a is defined as

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

where f_x and f_y are focal lengths of respective axis. Parameters u_0 and v_0 together form the principal point. Equations 4.1,4.2 and 4.3 can be written as a single camera matrix defined as

$$\mathbf{P} = \mathbf{KE}[\mathbf{R}\ \mathbf{t}]. \tag{4.4}$$

4.2 Neural Networks

A basic building block of artificial neural network is a neuron, defined as

$$y = f(\mathbf{w}^T\mathbf{x} + b) \tag{4.5}$$

where $\mathbf{w}^T \in R^n$, $b \in R$ and $\mathbf{x} \in R^n$ is an input vector. Function f is called an activation function in the literature. One of the typical choices for an activation function is sigmoid function

$$f(z) = \frac{1}{1 + e^{-z}}. \tag{4.6}$$

The main drawback of a single neuron model is that it can only approximate linear mapping. This can be overcome by results of the universal approximation theorem [12], which states that given any $f \in [0,1]^n$ and $\epsilon > 0$, there is a sum

$$G = \sum_{j=1}^{n} \alpha_j \sigma \left(\mathbf{w}_j^T \mathbf{x} + b_j \right) \tag{4.7}$$

for which

$$||G(\mathbf{x}) - f(\mathbf{x})|| < \epsilon \tag{4.8}$$

holds true. In other words, any continuous real function $f(x)$ can be approximated by a neural network with a single hidden layer of sufficient width.

Combining several neurons together results in a creation of artificial neural network. An example can be seen in figure 4.2. Output of j-th layer is defined by following equation

$$\mathbf{x}_j = f(\mathbf{w}_j^T \mathbf{x}_{j-1} + \mathbf{b}_j). \tag{4.9}$$

This kind of connection is typically called *fully connected layer* in the literature.

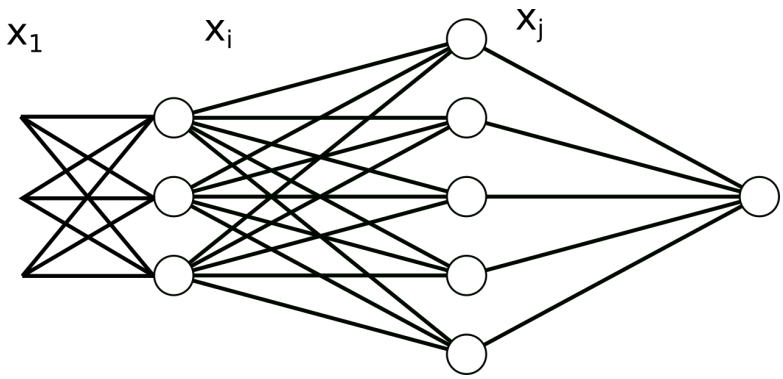


Figure 4.2: An example of neural network. Circles represent individual neurons, while lines mark connections between them.

4.2.1 Convolutional Neural Networks

Convolutional neural networks are a special kind of artificial neural networks designed specifically for digital image processing tasks. Input into the network is a tensor $\mathbb{U} \in \mathbb{R}^{h \times w \times n}$, where h is input height, w is input width and n denotes a number of channels. It should be noted that this is much different than in regular neural networks, where the input is a one-dimensional vector of flattened image rows/columns. The absence of flattening enables better preservation of spatial relationships between adjacent pixels.

Convolutional layer

Convolutional layer is the basic building block of convolutional neural networks. It is based on two dimensional discrete convolution operation. Discrete one dimensional convolution of two discrete functions $f[k]$ and $g[k]$ is defined as

$$(f * g)[k] = \sum_{-\infty}^{\infty} f[m]g[k-m], \tag{4.10}$$

where $k \in Z$ and $m \in Z$. This formula can be extended into the two-dimensional case as

$$(f * g)[k,l] = \sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty} f[m,n]g[k-m,l-n], \tag{4.11}$$

where $n, l, m, n \in Z$.

When working with neural networks, it is more convenient to express convolution using matrices and tensors. Let us define a matrix $\mathbf{X}$ and $\mathbf{W}$, then conv($\mathbf{X}$,$\mathbf{W}$) operation performed by sliding $\mathbf{W}$ across $\mathbf{X}$ and computing dot product of each step. An illustration of this procedure is displayed in figure 4.3. This idea can be easily extended for additional matrix channels, i.e. three to represent an RGB image as illustrated in figure 4.4.

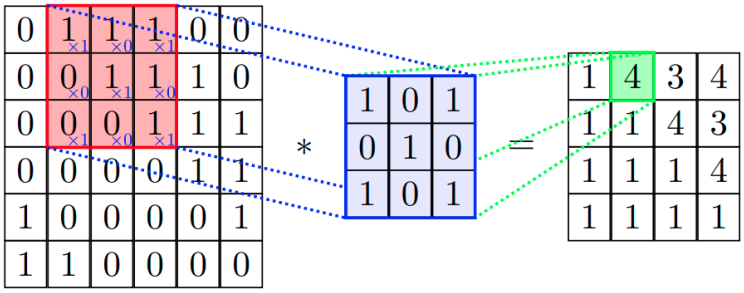


Figure 4.3: An illustration of one channel convolution. [40]

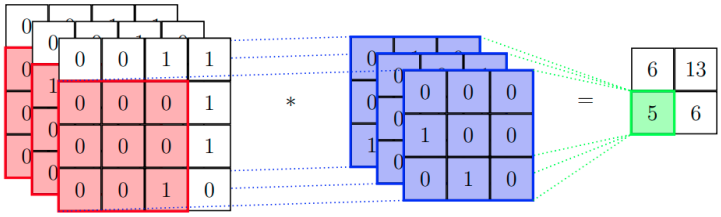


Figure 4.4: An illustration of three channel convolution. [40]

Typical convolutional layer consists of multiple filters. Let us define a tensor $\mathbb{D}_i \in \mathbb{R}^{h \times w \times n}$ which is output of i-th layer. Then let us define another

tensor $\mathbb{F} \in \mathbb{R}^{k \times l \times m}$ which is called bank of filters. Then output of the j-th layer can be expressed as

$$\mathbb{D}_j = \mathbb{D}_i * \mathbb{F}. \tag{4.12}$$

An illustration of this operation can be seen in figure 4.5.

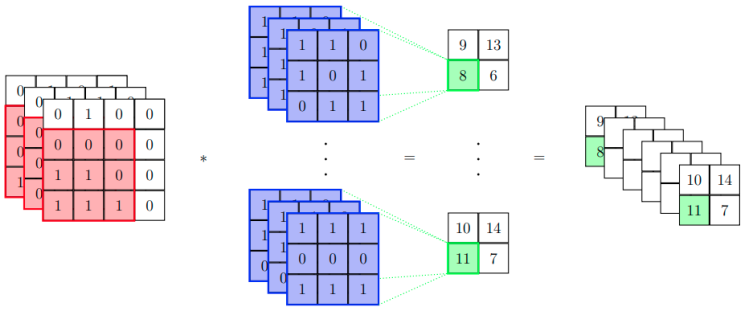


Figure 4.5: An illustration of convolution by a bank of filters. [40]

Pooling layer

The pooling layer is designed to reduce the number of parameters inside a network, prevent over-fitting and reduce computational time. Commonly used types are minimum, maximum and average pooling. Let us define a matrix $\mathbf{X}$ and pooling filter of size MxN. The output of the pooling layer is then obtained by diving matrix $\mathbf{X}$ into MxN regions and computing the maximum, average or minimum of these regions. This process is illustrated in figure 4.6 for a pooling filter of size 2x2.

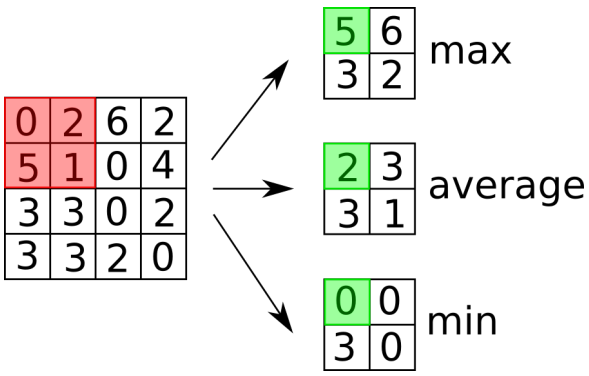


Figure 4.6: An illustration of pooling.

VGG-16

The pre-trained network heavily utilized in this thesis is VGG-16 [48]. This network was originally designed for the task of image classification but is commonly utilized for a variety of other tasks based on transfer learning principles. In our work we use VGG-16 pretrained on Places365 [55] dataset. Looking at the figure 4.7 we see that there are multiple layer groups denoted conv1, conv2, ..., conv5. These layer groups consist of convolutional layers with the same dimensions. Throughout this work, we will use the following notation of convX-Y, where X is a layer group and Y is the order of the convolutional layer inside the group. For example, conv4-3 would denote the third layer from the fourth group with dimensions 28x28x512. Moreover, the original VGG-16 has three fully connected layers, which are not used in our work.

In our proposed solution, we do not use the VGG-16 output but use activation of respective layers instead. Activation is represented by a tensor. In the following sections, we will work with conv 4-2 and conv 5-2 layers. These layers are represented by two tensors $\mathbb{D}_{4-2} \in \mathbb{R}^{52\times52\times512}$ and $\mathbb{D}_{5-2} \in \mathbb{R}^{14\times14\times512}$ respectively.

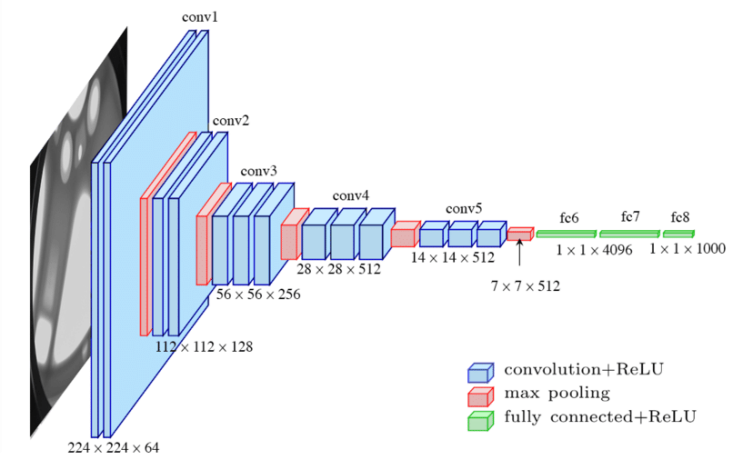


Figure 4.7: Architecture of VGG-16. [16].

4.3 Principal Component Analysis

Principal **C**omponent **A**nalysis is a method for data transformation, which is achieved by a change of basis. It expresses the data in a new coordinate system in which basis vectors follow the greatest variance inside data. We utilize this approach for dimensionality reduction.

Let us have two real spaces R^d and R^n, where $d \leq n$. Then mapping from R^n to R^d can be defined as

$$\mathbf{Y}^{d\times m} = \mathbf{W}^{d\times n}\mathbf{X}^{n\times m}, \tag{4.13}$$

where $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_m]\,\mathbf{y} \in R^d$, $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_m]\,\mathbf{x} \in R^n$. Matrix $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_d]$ is orthogonal and $\mathbf{w}_i$ are called principal components. While n is defined by the dimension of the input data, m can be chosen arbitrarily, as long as $m \leq n$ and $m \leq k$. The goal of PCA is to find the $\mathbf{W}^{d\times n}$, constrained by the choice of m. We can define sample covariance matrix as

$$\mathbf{P}_X = \frac{1}{m}\sum_{j=1}^{m}(\mathbf{x}_j - \mu_x)(\mathbf{x}_j - \mu_x)^T, \tag{4.14}$$

where

$$\mu_x = \frac{1}{m}\sum_{j=1}^{m}\mathbf{x}_j. \tag{4.15}$$

Vector $\mathbf{w}_1$ is then defined as

$$\mathbf{w}_1 = \arg\max_{\mathbf{w}} \left\{\mathbf{w}^T\mathbf{P}_X\mathbf{w}\right\} \quad ||\mathbf{w}|| = 1 \tag{4.16}$$

Each subsequent $\mathbf{w}_i$, where $i \leq d$ is then defined as

$$\mathbf{w}_i = \arg\max_{\mathbf{w}} \left\{\mathbf{w}^T\mathbf{P}_X\mathbf{w}\right\} \quad ||\mathbf{w}|| = 1 \quad \mathbf{w}\cdot\mathbf{w}_l = 0\ \forall l < i. \tag{4.17}$$

There are multiple ways to solve equations 4.16 and 4.17. It can be shown that the task of solving them is equivalent to finding eigenvectors corresponding to the d largest eigenvalues.

4.4 Semantic Spatial Matching

4.4.1 Image retrieval

Lets us define a set apriori known images $S = \{I_1, ..., In\}$ and have a previously unseen input image J. Then image retrieval task comes down to solving

$$I^* = \arg\max_{I} h(I, J), \tag{4.18}$$

where term $h(\cdot)$ is a similarity function, which is obtained either explicitly or implicitly. The main difficulty of this task is obtaining the similarity function. There has been proposed CNN architectures able to learn this similarity function in an end to end fashion. One of the examples being NetVLAD [7], which uses VLAD layer inspired by **V**ector of **L**ocally **A**ggregated **D**escriptors. SSM, on the other hand, works in a different manner. Instead of directly trying to learn $h(\cdot)$, spatial matching of CNN activation is utilized.

4.4.2 Semantic Spatial Matching for Visual Place Recognition

Spatial **S**emantic **M**atching is a framework for solving the task of visual place recognition. SSM-VPR treats the task as an image retrieval one. In the following paragraphs, we will analyze SSM in a top-down manner.

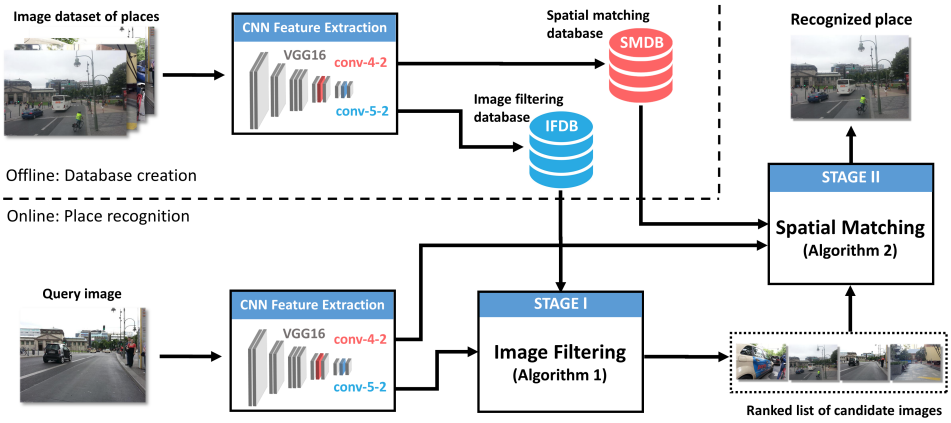


Figure 4.8: Overview of SSM image retrieval pipeline [11].

The basic architecture can be seen in figure 4.8. There are two phases training/offline and testing/online phase. Input to the offline phase is a set of images $S = \{I_1, ..., In\}$. Output of the offline phase are two databases called **I**mage **f**iltering **d**ata**b**ase (IFDB) and **S**patial **m**atching **d**ata**b**ase (SMDB). Both of them are created in a similar manner, which will be described below. The online phase takes one image J as an input and outputs a ranked set of candidates C.

Offline phase

First, the image I is run through VGG-16 and activation from conv 4-2 and conv 5-2 layers are cached. Output of this step are two activation tensors $\mathbb{D}_{4-2} \in \mathbb{R}^{52\times52\times512}$ and $\mathbb{D}_{5-2} \in \mathbb{R}^{14\times14\times512}$, which will be sometimes refered to as raw activation for the sake of readability. Tensors $\mathbb{D}_{4-2}$ and $\mathbb{D}_{5-2}$ are then converted into a set of smaller tensors by sliding selection cubes across tensors as illustrated in figure 4.9. These selection cubes are defined by their dimensions and stride, which in our case is equal to one. Output of this step are two sets of tensors $D_{5-2} = \{\mathbb{D}_1, ..., \mathbb{D}_{36}\}$ $\mathbb{D}_i \in \mathbb{R}^{9\times9\times512}$ and $D_{4-2} = \{\mathbb{D}_1, ..., \mathbb{D}_{625}\}$ $\mathbb{D}_i \in \mathbb{R}^{9\times9\times512}$.

The next step consists of flattening tensors along the third dimension and applying L2 normalization. Output of this step are two sets of vectors $D'_{5-2} = \{\mathbf{d}_1, ..., \mathbf{d}_{36}\}$ $\mathbf{d}_i \in \mathbb{R}^{41472}$ and $D'_{4-2} = \{\mathbf{d}_1, ..., \mathbf{d}_{625}\}$ $\mathbf{d}_i \in \mathbb{R}^{4608}$. The

final step consists of dimensionality reduction of vectors through PCA. After applying PCA transformation we are left with two sets of compressed vectors $D''_{5-2} = \{\mathbf{d}_1, ..., \mathbf{d}_{36}\}$ $\mathbf{d}_i \in \mathbb{R}^{125}$ and $D''_{4-2} = \{\mathbf{d}_1, ..., \mathbf{d}_{625}\}$ $\mathbf{d}_i \in \mathbb{R}^{100}$. Each image from the set S is the represented as D''_{5-2} inside IFDB and D''_{4-2} inside SMDB.

Principal Component Analysis is trained on a randomly chosen subset of S from D'_{5-2} and D'_{4-2}. One of the shortcomings of this approach is its memory intensity during training. Sets D'_{5-2} and D'_{4-2} are represented by 1 492 992 and 2 880 000 numbers respectively. The original article [11] uses around 50-250 samples depending on the other hyperparameters. This poses a serious challenge on a machine with limited memory resources and will be further addressed in the following chapter.

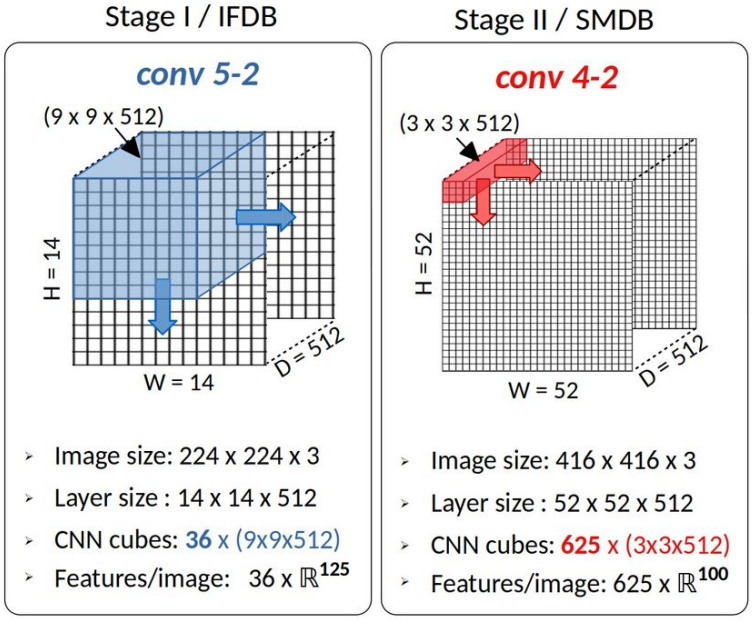


Figure 4.9: Feature extraction process for IFDB and SMDB creation. [11].

Online phase

The online phase takes an image J as an input. This image is first processed as described in the previous section. Thus we have J and corresponding D''_{5-2} and D''_{4-2}. Set D''_{5-2} is compared against all entries in IFDB and k best matches $S_1 \subset S$ are returned, where k is one of the hyper parameters. The process of ranking is summarized in algorithm 1. The closest matches of $\mathbf{d}_i$ are retrieved in terms of Euclidean distance.

During the second stage, J and corresponding D''_{4-2} is compared against a subset of SMDB defined by S_1. The process of ranking is summarized in

Algorithm 1: Image filtering stage

Data: $D''_{5-2} = \{\mathbf{d}_1, ..., \mathbf{d}_{36}\}$
Result: ranked list of candidates S_1
cand_hist ← Initialize histogram of candidate images
for $i \leftarrow 1$ **to** 36 **do**
    Find closest k matches of $\mathbf{d}_i$ in database IFDB
    Increment the corresponding k bins in *cand_hist*
S_1 ← Select k highest bins from *cand_hist*

algorithm 2. The output of this step is a list of ranked candidates S_2 and the member of S_2 with the highest score is returned as the retrieved image.

Algorithm 2: Spatial matching stage

Data: $D''_{5-2} = \{\mathbf{d}_1, ..., \mathbf{d}_{625}\}$, S_1
Result: ranked list of candidates S_2
scores_hist ← Initialize score histogram for candidates
foreach *s in* S_1 **do**
    $\{\mathbf{c}_{i,j}\}$ ← Get spatially-aware candidate vectors from SMDB
    for ***all*** $i, j \in \{1 \ ... \ 25\}$ **do**
        $\mathbf{d}_{k,l}$ ← Find closest match of $\{\mathbf{c}_{i,j}\}$ in query array
        Set (i, j) and (k, l) as anchor points
        Define patch P around location (i, j). It may be truncated
        Define identical (possibly truncated) patch around (k, l)
        for ***all*** $k', l' \in P$ **do**
            $\mathbf{c}_{i',j'}$ ← Find closest match of $\mathbf{d}_{k',l'}$ in candidate array
            if (i', j') - *(i, j)* = (k', l') - *(k, l)* **then**
                Increment bin in *scores_hist* for s

Following paragraphs explain algorithm 2 in more detailed fashion. The process of spatial matching is done independently for every member of S_1. The first step consists of finding so-called anchor points. Anchor points are positions of $\mathbf{d}_i \in D''_{4-2}$ inside D_{4-2}. The next step is to pair anchor points from candidate image to query image. This is done by retrieving the closest match of $\mathbf{d}_i$ of the query from the candidate array of spatially aware vectors. Once the correspondence (i, j) and (k, l) is decided, we can set them as anchor points.

We will now go over the process of creating the patch P based on anchor points (i, j) and (k, l) and demonstrate it for a patch of dimensions 9x9. Figure 4.10 show two images: candidate with highlighted anchor point (i, j) and query with highlighted anchor point (k, l). Patch P is defined around (i, j), but has to be truncated to account for finite candidate image dimensions. The same patch P is then defined around (k, l) as P', with possible further

truncation.

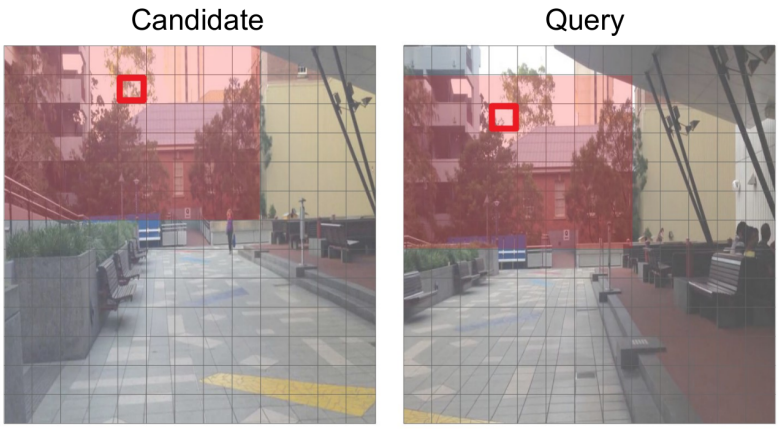


Figure 4.10: Illustration of patch definition around anchor points [11].

Geometrical comparison is the main defining element of the SSM approach. The general idea comes from the assumption that vectors around anchor points should have a similar location in each patch P and $P^{'}$ relative to anchor points.

Chapter 5

Proposed solution

This chapter will go over our proposed solution for object pose estimation based on SSM. We will explain and formally describe our design. The final implementation description is also present in this chapter. Contributions of our work are following:

- Development of object localization pipeline.
- Comparison of the proposed approach against the state-of-the-art solution.
- Software development of the former SSM-VPR framework, which we dubbed SSM v.2.

5.1 Pipeline

5.1.1 Overview

An overview of our proposed pipeline for object estimation can be seen in figure 5.1. The input is an un-annotated image I with list of object instances $L = \{(k_i, M_i)\}$ as defined in 2.2.2, where M_i are bounding boxes. Output is a list of object instances and their respective poses $\mathbf{P}_i = (\mathbf{R}, \mathbf{t})$. Each object instance is treated independently, and the process of obtaining the location of the object will be explained for a single instance without a loss of generality. The input image is first cropped by a bounding box and scaled to a predefined dimension. The processed input image is fed through the VGG-16 to obtain CNN features. These features are then used to estimate the orientation of the object. CNN features are used as an input into SSM, and the closest match is

retrieved. Unlike in SSM-VPR we also store bounding box $(\mathbf{m_0}, \mathbf{m_1})$ camera calibration $\mathbf{K}_d$, object orientation $\mathbf{R}_d$ and translation $\mathbf{t}_d$. Object orientation $\mathbf{R}_d$ is then used as $\mathbf{R}$. Translation $\mathbf{t}$ is then computed based on retrieved $(\mathbf{m_0}, \mathbf{m_1})$, $\mathbf{K}_d$ and $\mathbf{t}_d$.

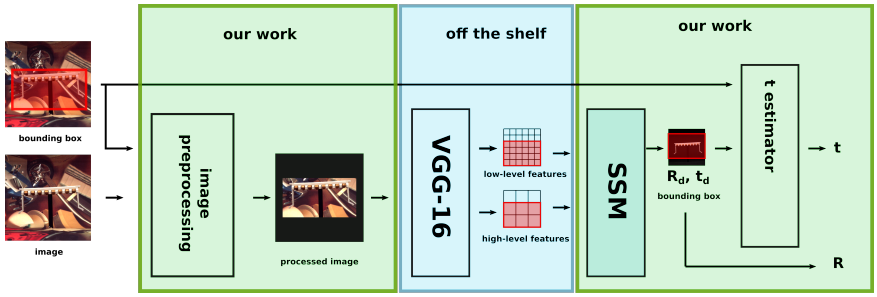


Figure 5.1: Our proposed pipeline for object pose estimation.

The way our problem is formulated in chapter 2 we consider object detection an input into our method. This means we use mostly pre-computed bounding boxes or detection masks already included in the dataset. These detections are generally either hand-selected by a human or outputted by a trained object detector. In case there is no object detection provided, it is necessary to train an object detector ourselves. Training the object detector is generally a straightforward but time-consuming task, which is why we have chosen to decouple object pose estimation into two separate problems. Because our method is not dependent on a specific object detector provided, the same one is used for training and evaluation. A recommended approach would be to use one of the state-of-the-art object detectors based on CNN. An example of a suitable bounding box detector would be YOLO [44] or Faster R-CNN [45]. For mask detection, a potential reader can use Mask R-CNN [19].

5.1.2 Offline phase

Each object o_i has a unique and independent SSM database. We will now describe how this database is created. First arises the task of picking the "correct" set of images for the training of the database. Let us first define a term called *view sphere*. Figure 5.2 shows a graphical representation of the set of all possible camera views of an object from a fixed distance $|\mathbf{r}|$ (scale) and camera centred on the object centre. The task of choosing the training samples is then equivalent to sampling the view sphere and scale. While it is certainly desirable to sample as uniformly as possible, it is usually not feasible when working with real data. We have decided to use one defined scale, which is one of the hyperparameters. Another option would be to store the database for a defined sequence of scales. This has proven to be computationally infeasible during our testing, and instead, we re-scale the input image.

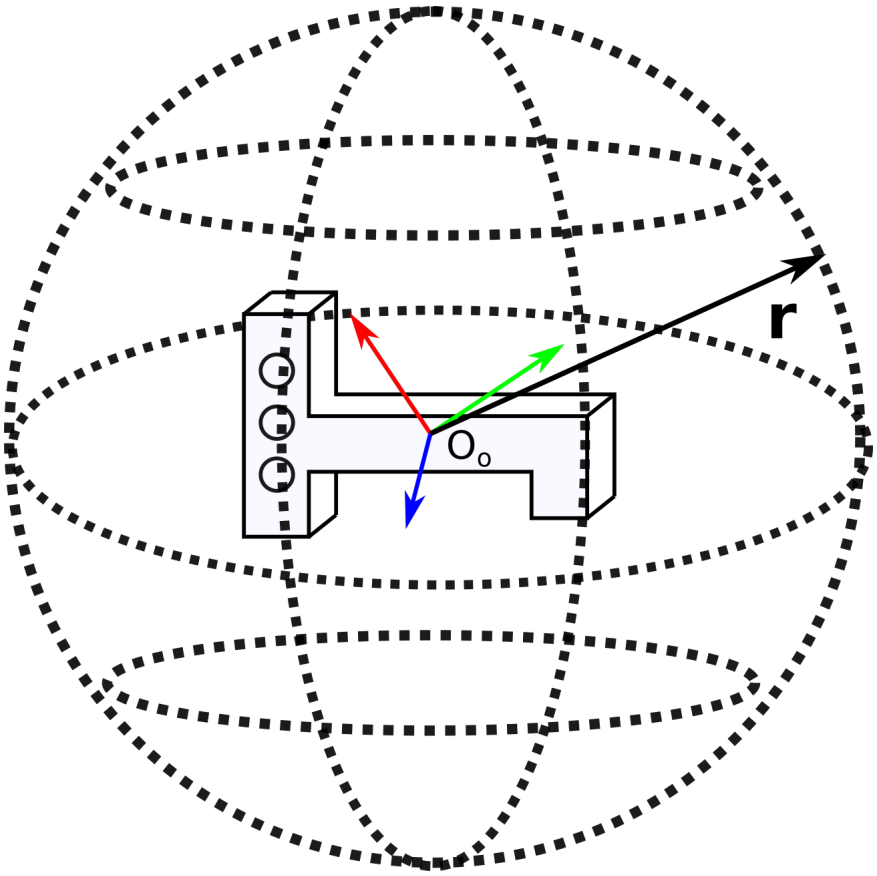


Figure 5.2: View sphere around object. Sampling also includes the rotation around **r**.

Once the correct set S of annotated images is obtained, we can proceed to create a database in a similar manner as described in 4.4.2. There are few modifications, though. First instead of just an image and processed activation from conv 4-2 and conv 5-2 layers, we also store bounding box $(\mathbf{m_0}, \mathbf{m_1})$ camera calibration $\mathbf{K}_d$, object orientation $\mathbf{R}_d$ and translation $\mathbf{t}_d$.

Instead of computing PCA on a randomly chosen subset, we use the so-called Incremental PCA, which will be described in the following chapter. This mainly beneficial on machines with limited resources.

The third modification comes from an optional training input. Let us define first a set of images S_o which **is not** be used for SSM database creation and a set of images S, which **is** used for SSM database creation and $S \cap S_o = \{\}$. Then we train Incremental PCA on $S \cup S_o$, but create the database itself only from S. This is especially useful while combining real and synthetic training images. If we used a mixture of real and synthetic training images for SSM database creation, it would lead to worse performance. This is due to real images being too similar to each other even though taken from different viewpoints, while the same goes for synthetic images. This way, we

can still extract useful information from real and synthetic images while not compromising system performance. We refer to this optional training input PCA injection.

5.1.3 Online phase

Image pre-processing

The input of the pre-processing phase in an image I with dimensions $h \times w$. Output of the pre-processing phase in an image I''' with dimensions $h''' \times h'''$. Pre-processing is captured in figure 5.3. Image I is cropped by bounding box $(\mathbf{n_0}, \mathbf{n_1})$, such as $I' = I \cap (\mathbf{n_0}, \mathbf{n_1})$. Resulting image I' is then scaled to I'' by a factor s defined by the following equation

$$s = \frac{h'''}{\max(h_b, w_b)}. \tag{5.1}$$

Dimensions of I'' are denoted $h'' \times w''$ and either $h'' = h'''$ or $w'' = h'''$ holds true. Image I'' is then centered and zero padded to the final dimensions $h''' \times h'''$, which results in I'''.

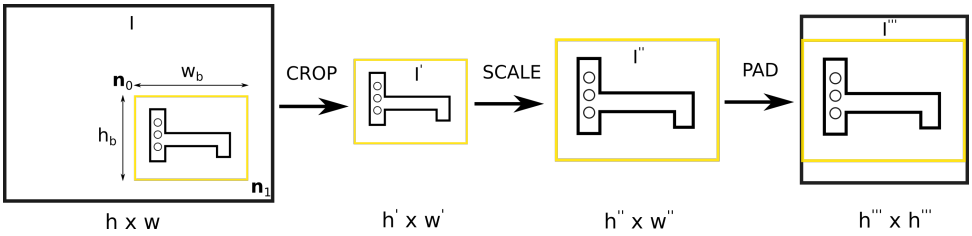


Figure 5.3: Image pre-processing illustration.

Rotation estimation

Processed image I''' is fed to the **VGG-16** and activation from conv 4-2 and conv 5-2 layers are processed in the same manner as described in 4.4.2. From **SSM** we retrieve the best match, bounding box $(\mathbf{m_0}, \mathbf{m_1})$ camera calibration $\mathbf{K}_d$, object orientation $\mathbf{R}_d$ and translation $\mathbf{t}_d$. Object orientation $\mathbf{R}_d$ is then used as $\mathbf{R}$. One thing to note, that rotation is represented in discrete and sometimes not uniform steps. Also, this part of our pipeline is the most computationally expensive one as it involves one feed-forward through the CNN and then requires several expensive operations for CNN feature processing and comparison.

Translation estimation

Object translation is computed in **t estimator** module and it is determined by three coordinates x, y and z. Starting with figure 5.4 we will now derive a formula for z, starting with an assumption that $(\mathbf{n_0}, \mathbf{n_1})$, $(\mathbf{m_0}, \mathbf{m_1})$ have same width to height ratio. Using the triangle similarity rule we get a set of

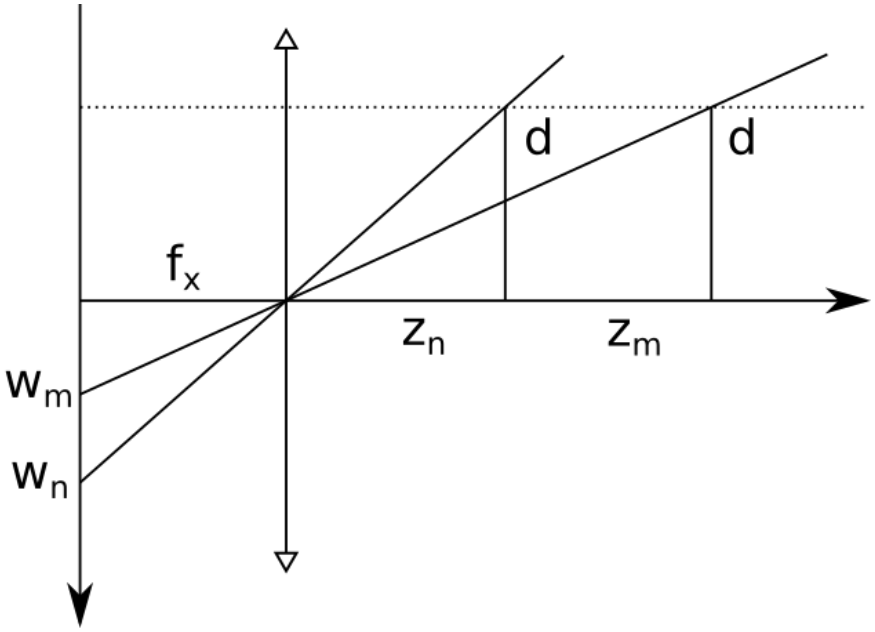


Figure 5.4: Pinhole camera model with focal length f_x schematic used for distance calculation. Object of width d is captured from two different distances z_n and z_m, while z_m is known.

following set of equations

$$\frac{d}{z_n} = \frac{f_x}{w_n} \tag{5.2}$$

$$\frac{d}{z_m} = \frac{f_x}{w_m}, \tag{5.3}$$

where f_x is a camera parameter as in 4.3, d is object width.

This is generally not a case, so in order to approximate, we will now derive a formula to estimate z by using an arithmetic average. Let us start with

$$z = \frac{\frac{w_n}{w_m} z_m + \frac{h_n}{h_m} z_m}{2}, \tag{5.4}$$

where h_n and h_m object heights in pixel captured in the same context as w_n and w_m as described in figure 5.4. After a several trivial steps we arrive at the following formula

$$z = \frac{1}{2} \frac{w_n h_m + h_n w_m}{w_m h_m} z_d. \tag{5.5}$$

We will now derive formula to compute x and y from input bounding box $(\mathbf{n_0}, \mathbf{n_1})$, database bounding box $(\mathbf{m_0}, \mathbf{m_1})$ and database camera matrix $\mathbf{P}_d = \mathbf{K}_d\mathbf{E}[\mathbf{R}_d\mathbf{t}_d]$. Based on this information we can obtain a projection $\mathbf{m}$ of object coordinate system center $\mathbf{O}_o$ as illustrated in figure 5.5 with

$$\mathbf{m} = \mathbf{P}_d\mathbf{0}_0. \tag{5.6}$$

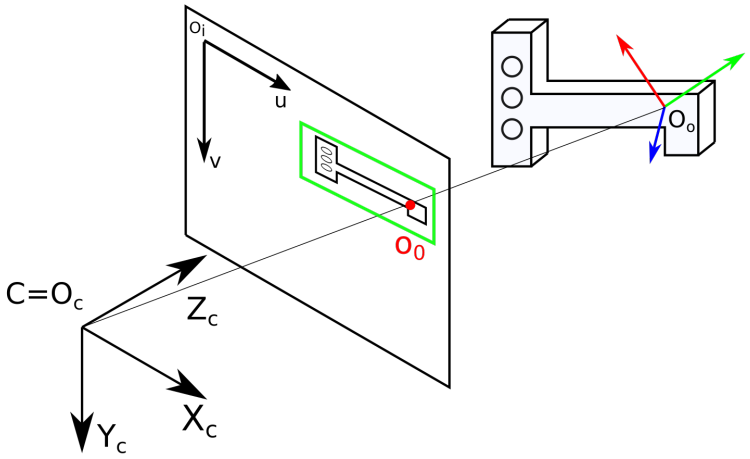


Figure 5.5: Illustration of object center projection.

Looking at figure 5.6 we will now derive a way to estimate $\mathbf{n}$, which is an estimated projection of object centre inside input image. In order to determine $\mathbf{n}$ we will use the fact, that $\mathbf{m}$ is already known from equation 5.6. We can obtain $\mathbf{n}$ as a solution of following equations

$$n_x = \frac{w_n}{w_m}(m_x - m_{0x}) + n_{0x} \tag{5.7}$$

and

$$n_y = \frac{h_n}{h_m}(m_y - m_{0y}) + n_{0y}. \tag{5.8}$$

The last step consists of converting $\mathbf{n}$ from 2D image coordinates to 3D coordinates. Based on equation 4.3 we can obtain x as

$$x = \frac{(n_x - u_0)z}{f_x} \tag{5.9}$$

and y as

$$y = \frac{(n_x - v_0)z}{f_y}. \tag{5.10}$$

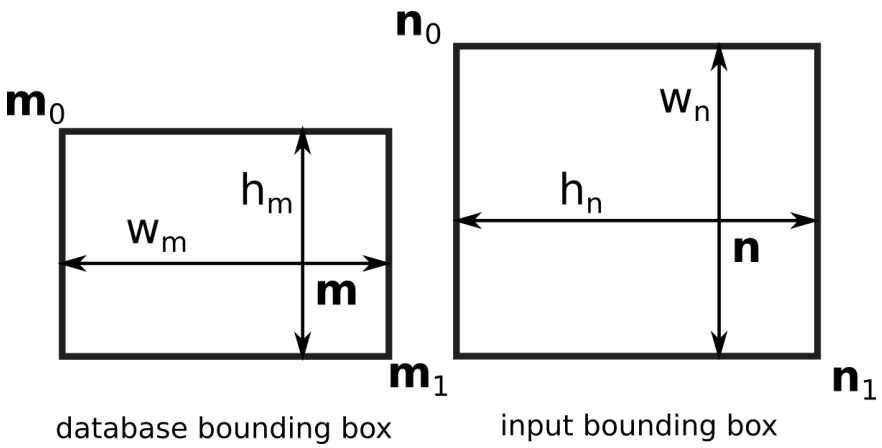


Figure 5.6: Illustration of translation estimation.

5.2 Implementation

5.2.1 SSM v.2

The original code base for SSM-VPR [11] is publicly available on [6] under MIT license. As mentioned before, one of the goals of this thesis is the development of the SSM framework. We have been able to identify the following shortcomings

- SSM-VPR data structure is not suitable for different tasks.
- Code was not easily extendable.
- Large RAM requirements.

Our contribution consists of the following:

- Data abstraction and reworked object oriented design.
- The use of Incremental PCA.
- Update of the graphical user interface functionalities.

The new version SSM v.2 can be found in the directory *ssm_v2* inside the attached DVD. Furthermore, the results of this work, SSM v.2 in particular, has been used as a part of the UAV Teach and Repeat framework [30], which is currently under review for ECMR2021.

Code and data structure

This section will go over changes made to the code structure. The codebase has been modified to be more modular and versatile. The image retrieval part can now be run independently without GUI.

One of the most important changes is the update of the input file format. Previously SSM-VPR expected images in format *imageXXXX.png* in two folders *Live* and *Reference*. Images in these two folders were then paired with *GroundTruth.csv*, a spreadsheet containing *Reference* and *Live* columns. While this representation was convenient for working with VPR datasets, it proved to be unsatisfactory for working with object pose estimation datasets.

Our updated input file format works in the following way. The database is defined by a single *poses.csv* file. This file contains a list of all images inside the database, their system path and annotation. The annotation can be of any of the supported types and is specified inside *metadata.txt* file. Some of the supported types are 6D pose, used in this thesis, and ID, which simulates the original functionality of the SSM-VPR.

Memory Usage

The problem with RAM usage stems from the use of PCA. In an ideal case, PCA would be performed on the whole dataset, but this is just infeasible on the average consumer computer. As mentioned before, PCA in SSM-VPR is trained on the random subset S_R of S all at once. The problem is that S_R needs to be large enough to infer meaningful dimensionality reduction, which, especially for larger datasets, is still problematic. Instead of performing PCA on a large sample all at once, we propose to perform PCA in an iterative fashion with an approach called IPCA [46], short for **I**ncremental **PCA**. We used implementation provided in *scikit-learn* [42] library.

GUI

To reflect the changes in the code base and to make the usage of GUI more efficient, several functionalities have been added to GUI. Overview of the most important changes can be seen in figure 5.7. Input fields have been updated to reflect new/changed parameters. For PCA, we have added an option to split the input dataset into multiple batches and an additional optional input to be used during PCA analysis, but not the database creation.

Furthermore, we have reworked database manipulation. The GUI now

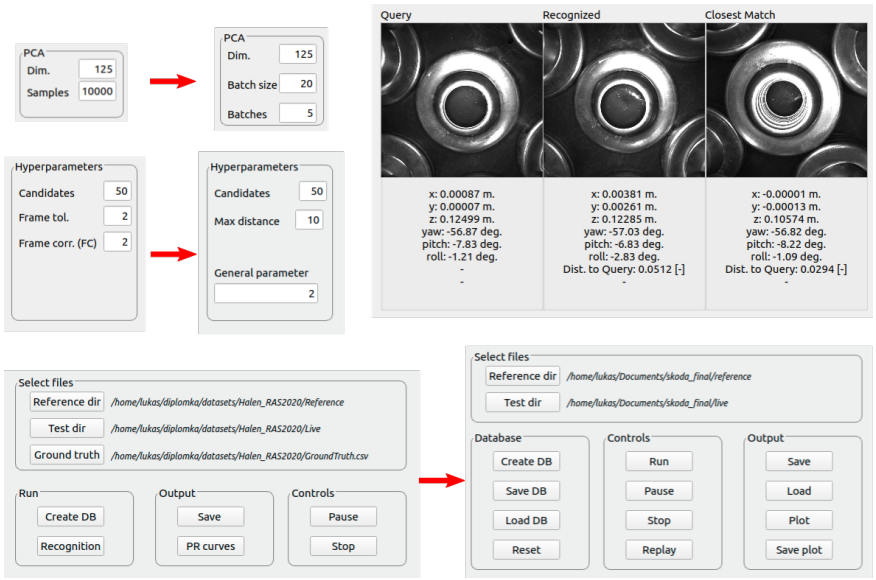


Figure 5.7: Overview of updated GUI.

allows to load and save multiple different databases to the desired location. Also, we have added a replay feature to review the image retrieval results. We have changed frame tolerance to the maximum distance parameter to reflect the domain abstraction. This, along with other changes, makes it easy to implement the custom distance metric used for evaluation.

5.2.2 SSM Pose

Pipeline presented in 5.1 was implemented as a standalone application in Python 3 and can be found in *ssm_pose* inside the attached DVD. We have also developed a standalone GUI independent of SSM v.2, which can be seen in figure 5.8.

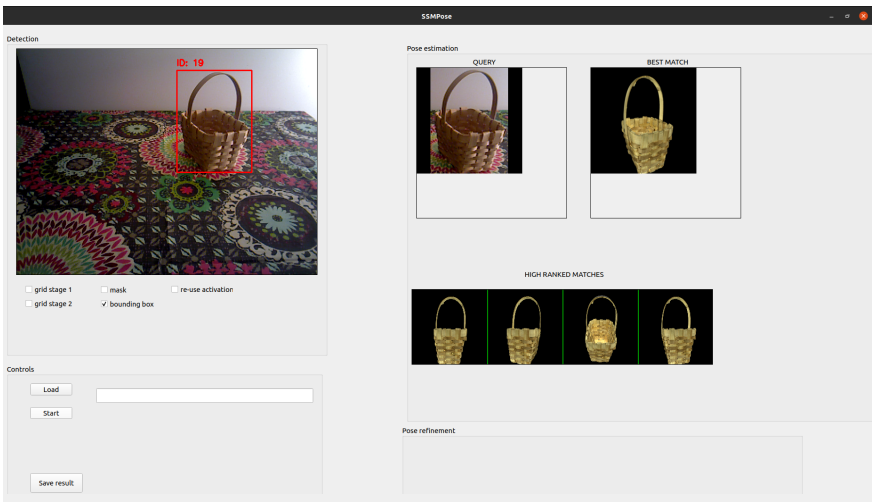


Figure 5.8: An illustration of GUI implemented specifically for the task of object pose estimation.

Chapter 6

Evaluation

6.1 Methodology

Evaluating object pose estimation is generally not a straightforward task. This is mainly because of the inherent ambiguity. There can be poses, which are indistinguishable due to object symmetries, occlusions and even self-occlusion. This characteristic is called "pose ambiguity". It is desirable for metric to "pose ambiguity invariant". However, metrics showing this property are generally quite complex both in definition and computation required.

Therefore we have decided to include two methodologies of evaluation. First, we will conduct a throughout analysis of our framework performance on a single asymmetrical and well-defined object without pose ambiguities, i.e. SiSo task. Using such an object will allow us to use more rudimentary metrics, which are, on the other hand, much more informative for a potential reader not well versed in the problem of object pose estimation.

The second methodology will use more advanced metrics, which are invariant to pose ambiguity. We will conduct an analysis of our framework performance on a whole dataset with multiple objects present, i.e. ViVo task. This approach will allow us to compare our method against other state-of-the-art algorithms. This is well suited for the relative comparison of object pose estimation algorithms.

6.1.1 Metrics

Let us first define the notation used in the following subsections. The model of the object, which is being evaluated, will be denoted M. Estimate of object

pose will be denoted $\hat{\mathbf{P}}$ and ground truth pose $\bar{\mathbf{P}}$. Metric will be denoted as $e(.)$ with an appropriate subscript. One of the measured parameters will also be a run time and training time.

We will now formally introduce pose ambiguity invariance. Let us define P^0 set of all poses which are ambiguous with each other and $\hat{\mathbf{P}} \in P^0$, then metric $e(.)$ is said to be pose ambiguity invariant if and only if $e(\hat{\mathbf{P}}, \bar{\mathbf{P}}) \approx e(\hat{\mathbf{P}}_i, \bar{\mathbf{P}})\ \forall \hat{\mathbf{P}}_i \in P^0$. It is important to note that one object can have more than one P^0 set.

We will use two metrics which are not pose ambiguity invariant translational and rotational error. We will use three pose ambiguous invariant metrics to evaluate the performance of our proposed method visible surface discrepancy, maximum symmetry-aware surface distance and maximum symmetry-aware projection distance. Tho choice of metrics was inspired by authors of the before-mentioned BOP in [24].

Translational and rotational error

We will use two metrics which are not pose ambiguity invariant translational and rotational error. Translational error is defined as

$$e_{trans} = ||\hat{\mathbf{t}} - \bar{\mathbf{t}}||^2. \tag{6.1}$$

rotational error is defined as

$$e_{rot} = \arccos\left(\left(trace\left(\hat{\mathbf{R}}\bar{\mathbf{R}}^{-1}\right) - 1\right)/2\right). \tag{6.2}$$

Visible Surface Discrepancy

Visible **S**urface **D**iscrepancy aims to calculate the error only over the visible part of the model surface. It is ambiguity invariant metric and thus will be used for all tested datasets. It defined by following equation

$$e_{VSD}(\hat{D}, \bar{D}, \hat{V}, \bar{V}, \tau) = avg_{p \in \hat{V} \cup \bar{V}} \begin{cases} 1 & if\ p \in \hat{V} \cap \bar{V} \wedge |\hat{D}(p), \bar{D}(p)| < \tau \\ 0 & otherwise, \end{cases} \tag{6.3}$$

where $\hat{D}, \bar{D}$ are distance maps obtained by rendering the object model S and $\hat{V}$, $\bar{V}$ are visibility masks. Parameter τ denotes a misalignment tolerance and is usually dependent on the precision of object model S.

Maximum Symmetry-Aware Surface Distance

Maximum **S**ymmetry-Aware **S**urface **D**istance is based on distance of corresponding points and can be described by

$$e_{MSSD}(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_S, V_S) = \min_{\mathbf{S} \in S_S} \max_{\mathbf{x} \in V_S} ||\hat{\mathbf{P}}\mathbf{x} - \bar{\mathbf{P}}\mathbf{S}\mathbf{x}||_2, \tag{6.4}$$

where S_S is a set of global symmetry transformations and V_S is a set of mesh vertices of object model S. The results in meters.

Maximum Symmetry-Aware Projection Distance

Maximum **S**ymmetry-Aware **P**rojection **D**istance is very similar to MSSD, but instead of

$$e_{MSPD}(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_M, V_M) = \min_{\mathbf{S} \in S_M} \max_{\mathbf{x} \in V_M} ||pro(\hat{\mathbf{P}}\mathbf{x}) - proj(\bar{\mathbf{P}}\mathbf{S}\mathbf{x})||_2, \tag{6.5}$$

where $proj(.)$ is the 2D projection operation. This operation is performed by projecting the object model M onto the image using the $\hat{\mathbf{P}}$ and $\bar{\mathbf{P}}$, which means the results is in pixels.

6.1.2 Datasets

Datasets are an important part of state-of-the-art research. This section will give a brief overview of common datasets used to compare object pose estimation frameworks. We aim to give a potential reader an idea of problem difficulty and illustrate the problems of the task of pose estimation from images.

TYOL

TYOL dataset introduced in [53] contains 21 objects. Real testing images and 3D models are provided. An example of the testing image can be seen in figure 6.1. There is almost no occlusion present. This dataset also contains an object without global symmetries (figure 6.1). All of the reasons above make this dataset a suitable candidate for parameter tuning.

LM/LM-O

LINEMOD dataset with its two versions [10, 9] has been the most used dataset for object pose estimation. It consists of 15 texture-less objects with

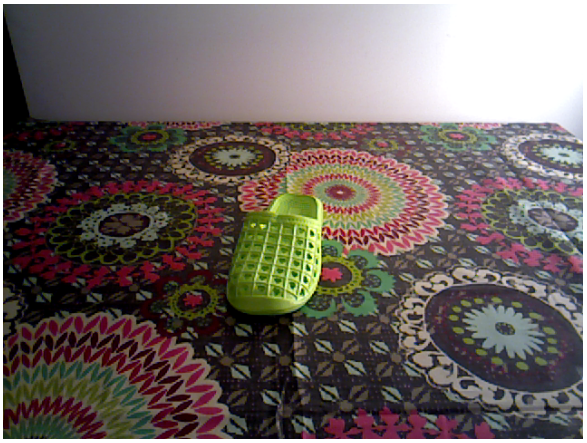


Figure 6.1: Sample image from TYOL dataset. [25]

a varying level of occlusion. This dataset is split into two subsets LM and LM-O. The LM part features only a mild level of occlusion and the LM-O has some testing images almost fully occluded. Only 3D models and annotated synthetic images are available for training. One of the reasons LINEMOD is considered challenging is the absence of real training images, which can lead to over-fitting during training. An example of the testing image can be seen in figure 6.2 and rendered synthetic images of structured models can be seen in figure 6.3.

Figure 6.2: Sample image from LINEMOD dataset. [10]

T-LESS

T-LESS [23] named after the fact, that it contains texture-less objects. It provides 3D models, synthetic and real training images of 30 industrial objects. An example of the testing image can be seen in figure 6.2 and rendered synthetic images of structured models can be seen in figure 6.3.

Figure 6.3: Sample views of rendered synthetic images from LM-O dataset.

6.2 Training and parameter tuning

Let us first start with a summary of all hyperparameters, which need to be tuned. Overview and description of parameters for SSM v.2 is presented in table 6.1 and for SSM Pose in table 6.2.

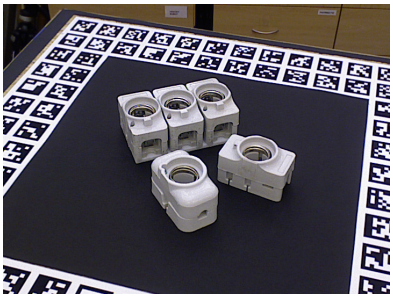


Figure 6.4: Sample image from T-LESS dataset.[23]

Figure 6.5: Sample views of rendered synthetic images from TLESS dataset.

Parameter	Meaning
image_size	Dimension of an input image.
pca_dim	Dimension of database descriptors after PCA compression.
batch_size	Batch size used for one iteration of IPCA.
number_of_batches	Number of IPCA iterations.
candidates	Number of candidates retrieved.

Table 6.1: Hyper parameters of SSM. Every parameter is set for each stage independently.

Parameter	Meaning
view_sphere_samples	Number of samples from the view sphere.
scale	Distance from which the object is captured during training.

Table 6.2: Hyper parameters of SSM Pose.

Because all of our experiments in the next section are centred around BOP challenge [25] and its BOP toolkit [2], we will base parameter tuning around them.

6.2.1 Hardware

For training and evaluation we have used laptop **ASUS ROG GL702VT** with the following specifications **IntelCore i76700HQ** processor, **NVIDIA GeForce GTX 970M** dedicated graphics card and Linux **Ubuntu 20.04 LTS** operating system. The graphic card was used with **NVIDIA CUDA V11.2** library and graphic card driver version **460.73**.

6.2.2 Training data

The BOP challenge provides several datasets consisting of object models, annotated synthetic, and real training images. Our approach creates the SSM database from synthetic rendered images of the object model. We render images by sampling the view-sphere at a single scale. There are two sampling schemes implemented in the BOP toolkit Fibonacci [18] and Hinterhosser [20]. We have tried both and found no difference in performance but opted for Hinterhosser one as it is the most common approach among BOP participants. Optional PCA injection is obtained from provided real images, if available, which are subjected to the same pre-processing as described in 5.1.3.

6.2.3 Parameter tuning procedure

The first parameter to be tuned is the image size for each stage of SSM. Because we decided to use only one forward pass through the network, we use the same dimension for each stage. The optimal image size was inferred from the distribution of training images dimensions, bounding box sizes and computational speed. In BOP training and testing, images are of dimensions 640x480, and bounding boxes cover around 25 % of the image surface. It is not desirable to upscale images too much because there is no additional information and the image entropy stays the same. Using an image dimension of 224x224 proved to be a good compromise, which was then verified empirically.

The initial value of *pca_dim* was set according to [11] and then iteratively adjusted by small amounts, similar to the grid search methodology. With this approach have arrived at the value of 100 for each stage. Because of our improvement in the form of IPCA, we can use the whole database for training in an iterative fashion. Parameter *number_of_batches* is then set to the maximum possible value. The last parameter to be tuned is the *batch_size*. Here we have to consider our hardware, which provides 8 GB of RAM, and thus the upper limit was around 30 samples per batch. The theoretical minimum is different for each stage. Generally, if we want to compress our feature vector to *pca_dim* dimension space we need atleast *pca_dim* or more feature vectors. This minimum is then defined as

$$batch\_size_{min} = \left\lceil \frac{pca\_dim}{n\_vectors} \right\rceil, \tag{6.6}$$

where *n_vectors* corresponds to the number of feature vectors extracted from one image and can be inferred from figure 4.9. For the first stage of SSM $n\_vectors = 36$ and for the second stage $n\_vectors = 625$ respectively.

Parameter tuning for SSM Pose is much more straightforward. Parameter *view_sphere_samples* has a direct trade-off between angular resolution and training/run time. The excellent compromise between accuracy and speed is around 100-200 samples at a single scale. Hyperparameter *scale* is set depending on the average height of an object and camera parameters manually. It is desirable for the object to cover most of the image. The list of hyperparameters used in all of the following experiments is included in table 6.3.

The training time takes around 5-15 minutes per object, with a database of 100-200 images. We have been able to train our method with as low as 50 images, at which point the performance deteriorates significantly for some objects. While more training images from different viewpoints mean finer view-sphere sampling, we have found that the database of size around 150-300 is sufficient, provided the sampling is close to uniform.

Parameter	Value
SSM	stage 1
image_size	224x224 [pixel x pixel]
pca_dim	100 [-]
batch_size	30 [-]
number_of_batches	MAX
candidates	20 [-]
SSM	stage 2
image_size	224x224 [pixel x pixel]
pca_dim	100 [-]
batch_size	30 [-]
number_of_batches	MAX
candidates	1 [-]
SSMPose	
view_sphere_samples	161 [-]
scale	500 mm

Table 6.3: Settings used in following experiments.

6.3 Results

6.3.1 In-depth analysis

For a more exhaustive analysis, we have selected one object from the TYOL dataset, which can be seen in figure 6.6. The testing split consists of a single scene with a single object and around real 80 images. The selected object exhibits no symmetries. Thus there are no pose ambiguities. We have conducted three experiments, one with only synthetic training dataset (Synt), one with a combination of synthetic and real training data for the PCA injection (Synt + Real) and one with synthetic training dataset and object detector trained by ourselves (Synt + Det). Pre-computed bounding boxes for training and testing have been obtained from BOP. Because these bounding boxes are generally of high quality, we have decided to also train our own detector for comparison to investigate the dependency of our method on the quality of the object detector. We have chosen Faster R-CNN [45] object detector with implementation provided in Detecto object detection library [3].

Results can be seen in table 6.4. We also provide error histograms in 6.7, 6.8 and 6.9. The average time needed to estimate the pose of a single object was around 300 ms. It is common for a monocular camera to have different accuracy in the x/y axis versus the z-axis, which is why we evaluate each axis individually and provide histograms for the x/y and z-axis separately. Let us start discussing rotational and translational error. The way we

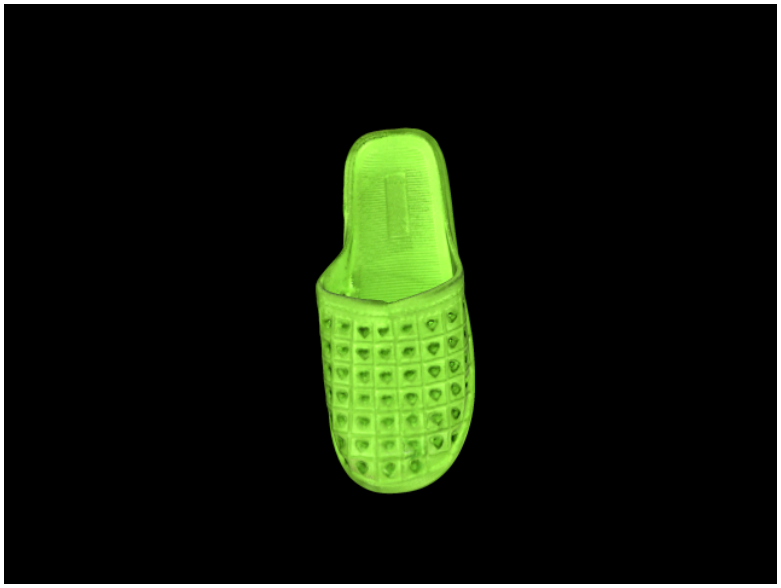


Figure 6.6: Top view of structured model. [25]

compute translation vector is directly influenced by the precision we achieve in estimating orientation. Error in the z-axis is approximately two times the other axis, which is the expected result. Angular error is slightly higher than desired but can be easily explained. Looking at figure 6.7c, we see that a lot of error comes with estimates, which are around 180° off. If we have a look at the figure 6.6, we see no real symmetries. Nevertheless, the object is close to being symmetrical, and for low resolution, the rotation of 180 ° around the top view can be indistinguishable. This shows that our method is insensitive to small details in object shape. This information is not really surprising considering the low spatial resolution of CNN features as illustrated in figure 4.9. On the other hand, this can also be a desirable feature in some applications.

For comparison, the best-performing state of the art algorithms achieves an MSSD, MSPD and VSD score of around 0.5-0.8. As mentioned in [24] a high MSSD score indicates the suitability of the method for object grasping by a robotic arm. A slightly lower VSD score can be explained by frequent 180° rotational miss-alignments.

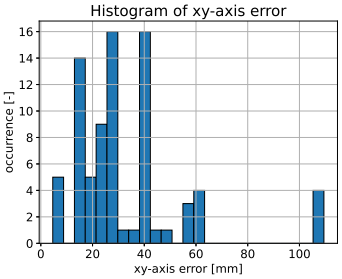
Interesting results come from comparing synthetic and synthetic/real training split. There is little to no improvement from adding real training image samples. This can be explained by the relatively low resolution of input images, quality of rendered synthetic training images and lack of colour diversity of the object. The main difference between synthetic and real images comes from colour differences and not the shape. Our object exhibits a simple colour to render, and most of the information about its orientation is encoded in its shape. This also shows that our method is easily trained on a synthetic

dataset with little to no degradation in performance.

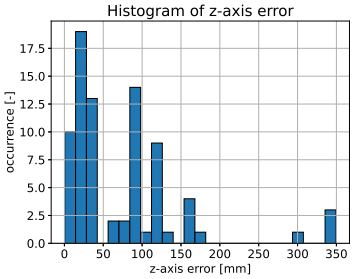
Metric	Synt	Synt + Real	Synthetic + Det
Translational error (median)	48.712 mm	49.180 mm	82.549 mm
Trans. error x-axis (median)	20.586 mm	24.153 mm	17.965 mm
Trans. error y-axis (median)	15.776 mm	16.551 mm	18.932 mm
Trans. error z-axis (median)	40.860 mm	39.273 mm	74.085 mm
Rotational error (median)	21.564 °	22.828 °	22.281 °
MSSD (average recall)	0.309	0.306	0.208
MSPD (average recall)	0.321	0.319	0.195
VSD (average recall)	0.193	0.169	0.116

Table 6.4: Results of SISO experiment from synthetic training data.

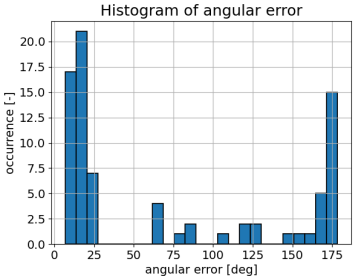
Overall results of these experiments clearly demonstrate that our proposed solution works. Nevertheless, it is obvious that most of the error comes from the wrong result of the image retrieval stage. An obvious solution to this would be to try a different image retrieval pipeline, which is beyond the scope of this thesis. On the other hand, it should be noted that the SSM provides a scoring function for retrieved images, and we were able to observe that a lows score oftentimes correlates with retrieving an image with a wrong orientation. This information could be utilized while building, for example, a random bin picking pipeline to simply skip frames with a low level of confidence.



(a) : Histogram of xy-axis error.

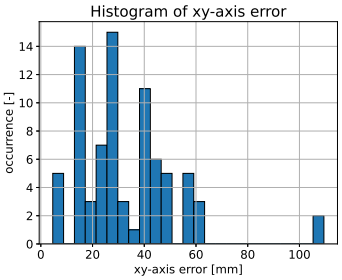


(b) : Histogram of z-axis error.

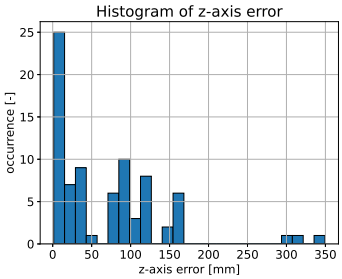


(c) : Histogram of angular error.

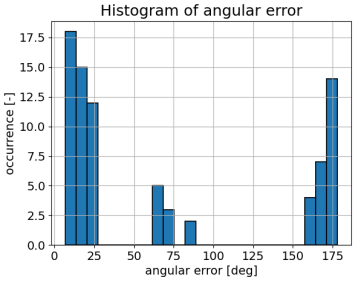
Figure 6.7: Histogram of error for synthetic training dataset.



(a) : Histogram of xy-axis error.

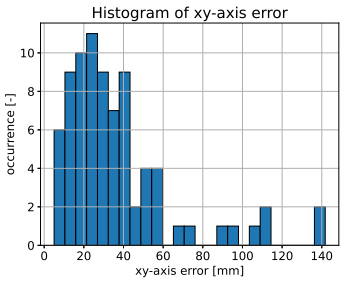


(b) : Histogram of z-axis error.

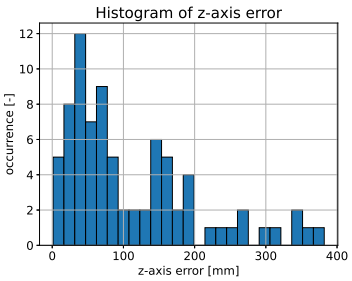


(c) : Histogram of angular error.

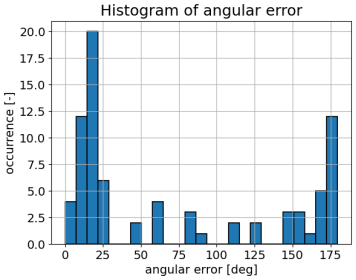
Figure 6.8: Histogram of error for synthetic/real training dataset.



(a) : Histogram of xy-axis error.



(b) : Histogram of z-axis error.



(c) : Histogram of angular error.

Figure 6.9: Histogram of error for synthetic training dataset with trained detector.

6.3.2 State of the art comparison

LM/LM-O

For comparison of our method against the state of the art algorithms, we have chosen LM-O dataset. The main challenge of this dataset comes from a heavy occlusion and thus is a good indicator of how well the method handles occlusion. The testing split consists of a single scene with around 200 images. Results can be seen in table 6.5. All bounding boxes for training and testing have been obtained from BOP. Scores of other methods have been extracted from the BOP challenge leaderboard [1]. Our method has placed in the seventh spot. It is interesting to note that our approach has a lower VSD score compared to other methods. This would suggest that our main error is in the estimated rotation because VSD is more sensitive to it. Another thing to consider is the relatively short training time of our method. It took around 2 hours on our consumer-grade laptop to train. Other algorithms like CosyPose take multiple days on GPU clusters to train. We have also included the run time for illustration. Nevertheless, a direct comparison is inherently biased as we have used a consumer laptop for evaluation, while most of the other authors use high-end desktops.

Place	Method	AR	AR_{MSSD}	AR_{MSPD}	AR_{VSD}	time[s]
1	CosyPose [31]	0.633	0.480	0.606	0.812	0.550
2	CDPNv2 [35]	0.624	0.445	0.612	0.815	0.163
3	PVNet [43]	0.575	0.428	0.543	0.754	x
4	EPOS [22]	0.547	0.389	0.501	0.750	0.468
5	Jinhuiliu [x]	0.525	0.350	0.444	0.781	0.939
6	Pix2Pose[41]	0.363	0.233	0.307	0.550	1.310
7	SSMPose	0.224	0.154	0.377	0.141	2.250
8	SMPE [50]	0.217	0.150	0.153	0.346	0.200
9	DPOD [54]	0.169	0.101	0.126	0.278	0.172
10	Sundermeyer [51]	0.146	0.090	0.095	0.254	0.550
11	SSD-6D [28]	0.139	0.047	0.083	0.285	x

Table 6.5: Results of BOP challenge for LM-O dataset. For methods with multiple submissions only the best performing one was considered. Entries marked with x were not made public.

TLESS

As a supplementary comparison, we have chosen the TLESS dataset. The testing split consists of 20 scenes with multiple objects present. Each scene has around 200 images. Results can be seen in table 6.6. We used the same methodology as in the last section. We have placed in ninth place out of ten, which is slightly worse than in the case of the LM-O dataset. This can indicate two things. Our method is generally more suitable for objects with more visual texture, and it handles occlusion reasonably well. One thing, which can be noted is higher run time. This has an obvious explanation. An average scene in the TLESS dataset has more objects than in the LM-O dataset, and the run time of our algorithm scales linearly with the number of objects present in the scene.

Place	Method	AR	AR_{MSSD}	AR_{MSPD}	AR_{VSD}	time[s]
1	CosyPose [31]	0.839	0.773	0.836	0.907	0.969
2	CDPNv2 [35]	0.490	0.377	0.418	0.674	0.708
4	EPOS [22]	0.476	0.369	0.423	0.635	1.177
5	Jinhuiliu [x]	0.403	0.225	0.271	0.712	0.611
6	Pix2Pose[41]	0.344	0.261	0.296	0.476	1.084
7	SMPE [50]	0.310	0.211	0.224	0.496	0.193
8	Sundermeyer [51]	0.304	0.196	0.211	0.504	0.194
9	SSMPose	0.130	0.145	0.192	0.053	4.712
10	DPOD [54]	0.081	0.048	0.055	0.139	0.206

Table 6.6: Results of BOP challenge for TLESS dataset. For methods with multiple submissions only the best performing one was considered. Entries marked with x were not made public.

Chapter 7

Conclusion

7.1 Evaluation

This thesis set out to achieve two main goals to design and implement an object pose estimation pipeline based on the SSM-VPR framework pipeline and software development of the aforementioned framework. Both of these goals have been successfully completed. From chapter 6 it is clear that our proposed solution works and even outperforms some of the state-of-the-art solutions with some limitations. The new version SSM v.2 has been proven to work as expected.

In our work, we have performed an analysis of state-of-the-art approaches for object pose estimation and familiarized ourselves with the SSM-VPR framework. We then utilized this knowledge to design and implement our object localization pipeline. We then conducted multiple experiments to verify our design and compared it against other state-of-the-art approaches. Results of the conducted experiments clearly show that our solution is viable and even outperformed some of the state-of-the-art methods. Moreover, we have clearly demonstrated that our proposed solution handles synthetic training images reasonably well, which is something which other approaches struggle with most often. Nevertheless, it is obvious that our solution is far from perfect. The general accuracy of translation in the z-axis offers the biggest room for improvement. Also, the variance of error in orientation is slightly higher than desirable.

Now we need to ask ourselves if using the SSM for object pose estimation brings any benefit over other already existing solutions. Results of section 6.3.2 already tell us that it does. Moreover, our solution offers some nontrivial advantages. It is reasonably fast to train, requiring around 5-15 minutes on a consumer computer per object. The next advantage is the independence of

object databases. Adding a new object to the already pre-trained pipeline brings no issues and requires merely an addition of another database. Most of the state of the art approaches require around 500-1000 training images per object, while our approach is able to produce reasonable results for as low as 50-100 images.

Another contribution was the development of SSM v.2. One of the underlying requirement for the new version of the SSM framework was to make it viable for more general problems. Not only was this version of the spatial matching framework successfully used in this work, but it was also used during the development of the UAV localization framework, with our contribution. Results of this contribution are to be released in an independent research article [30].

7.2 Future work

The results show that SSM, while usable, is not a sufficient solution for object orientation estimation. There are two directions to be taken to implement an additional refinement step, which is common among object pose estimation pipelines, or to replace SSM with a different approach. Both of these directions have their reasoning. The only problems with adding an additional refinement step are the computational overhead and the absence of image features with a reasonable spatial resolution. Another option would be to replace SSM with another image retrieval pipeline or to forgo the idea of template matching as orientation estimation altogether. The next logical step would be to try the aforementioned NetVLAD image retrieval pipeline [7].

Another thing which could improve our system performance is a bounding box correction. It is obvious that our translation suffers greatly from wrong object detection. The results of our experiments clearly show that the main problem is with object distance estimation, i.e. the z coordinate. One of the possible solutions could be to implement a bounding box refinement procedure, e.g. as proposed in [26].

The last thing we would consider exploring is to implement a random bin picking solution based on our framework. Sometimes high variance inaccuracy could be overcome by capturing the object from multiple angles and choosing the respective pose with the best score, the so-called multi-view approach. Potential advantages of our approach would the training time and the possibility to use only synthetic training data. Training time is shorter than most of the other state-of-the-art solutions, which leads to faster deployment times, a crucial industrial requirement. The idea of using structured object models to generate synthetic data is becoming the more and more prevalent method of training, with the main motivator being the time saved.

Bibliography

[1] BOP challenge leaderboards. https://bop.felk.cvut.cz/leaderboards/. Accessed: 2021-05-08.

[2] BOP toolkit. https://github.com/thodan/bop_toolkit. Accessed: 2021-05-09.

[3] Detecto library. https://github.com/alankbi/detecto. Accessed: 2021-04-12.

[4] Picking challenge. http://pwurman.org/amazonpickingchallenge/2015/index.shtml. Accessed: 2021-03-01.

[5] SIXD challenge 2017. http://cmp.felk.cvut.cz/sixd/challenge_2017/. Accessed: 2021-03-01.

[6] SSM-VPR. https://github.com/Chicone/SSM-VPR/. Accessed: 2021-04-24.

[7] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition, 2016.

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer Vision – ECCV 2006 Lecture Notes in Computer Science*, page 404–417, 2006.

[9] Eric Brachmann. 6D Object Pose Estimation using 3D Object Coordinates [Data], 2020.

[10] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. volume 8690, pages 536–551, 09 2014.

[11] Luis G. Camara and Libor Přeučil. Visual place recognition by spatial matching of high-level cnn features. *Robotics and Autonomous Systems, accepted for publication*, 2020.

[12] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd, 2016.

[15] B. Drost, M. Ulrich, P. Bergmann, P. Härtinger, and C. Steger. Introducing mvtec itodd — a dataset for 3d object recognition in industry. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2200–2208, 2017.

[16] Max Ferguson, Ronay ak, Yung-Tsun Lee, and Kincho Law. Automatic localization of casting defects with convolutional neural networks. pages 1726–1735, 12 2017.

[17] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[18] Álvaro González. Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, 42(1):49–64, 2009.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[20] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. *Procedings of the British Machine Vision Conference 2008*, 2008.

[21] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. *Computer Vision – ACCV 2012*, page 548–562, 2013.

[22] Tomáš Hodaň, Dániel Baráth, and Jiří Matas. EPOS: Estimating 6D pose of objects with symmetries. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[24] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. On evaluation of 6d object pose estimation. pages 606–619, 2016.

[25] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. BOP challenge 2020 on 6d object localization. *Computer Vision – ECCV 2020 Workshops*, page 577–594, 2020.

[26] Yifan Jiang, Hyunhak Shin, and Hanseok Ko. Precise regression for bounding box correction for improved tracking based on deep reinforcement learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1643–1647, 2018.

[27] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. *International Conference on Computer Vision (ICCV) Workshops*, 2019.

[28] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1530–1538, 2017.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[30] Viktor Kozák, Pavlos Avgoustinakis, Tomáš Pivoňka, Lukáš Majer, Luis Gomez Camara, and Libor Přeučil. Robust visual teach and repeat navigation for unmanned aerial vehicles. *European Conference on Mobile Robots*, 2021. Under review.

[31] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. *Computer Vision – ECCV 2020*, page 574–591, 2020.

[32] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks, 2016.

[33] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. *Shape, Contour and Grouping in Computer Vision*, page 319–345, 1999.

[34] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. *International Journal of Computer Vision*, 128(3):657–678, Nov 2019.

[35] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[36] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[37] D.G. Lowe. Local feature view clustering for 3d object recognition. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.*

[38] Lukas Majer. Slam localization using stereo camera. *Czech Technical University*, 2019.

[39] E. Munoz, Y. Konishi, V. Murino, and A. Del Bue. Fast 6d pose estimation for texture-less objects from a single rgb image. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[40] Jakub Paplham. Convolutional neural networks with local context masks. *Czech Technical University*, 2019.

[41] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019.

[44] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[46] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2007.

[47] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, 2011.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[49] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Thompson Learning, 3 edition, 2008.

[50] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, Narunas Vaskevicius, Kai O. Arras, and Rudolph Triebel. Multi-path learning for object pose estimation across domains. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[51] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[52] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[53] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2018.

[54] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: Dense 6d pose object detector in rgb images. *International Conference on Computer Vision (ICCV)*, 2019.

[55] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[56] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks, 2020.

Appendix A

List of Mathematical Notation

Symbol	Meaning
$\mathbf{r}$	Column vector.
$\mathbf{A}$	Matrix of numbers.
X	Set.
$\hat{x}$	Estimated value.
$\mathbb{X}$	Tensor.
$f(x_1, x_2, .., x_n)$	Scalar function of n variables.
$\mathbf{f}(x_1, x_2, .., x_n)$	Vector function of n variables.

Appendix B

List of Abbreviation

Symbol	Meaning
SSM	Semantic Spatial Matching
DOF	Degrees of Freedom
PnP	Perspective N Point
RANSAC	Random Sample Consensus
GPU	Graphical Processing Unit
RGB	Red Green Blue
RGB-D	Red Green Blue - Depth
SE(3)	The Lie group
PCA	Principal Component Analysis
IPCA	Incremental Principal Component Analysis
SiSo	Single Instance of a Single Object
ViVo	Varying number of Instances of a Varying number of Object
VPR	Visual Place Recognition
AR	Average Recall
UAV	Unmanned Aerial Vehicle

Appendix C

DVD Content

File	Content
F3-MP-2021-Lukas-Majer.pdf	This thesis in PDF format.
ssm_pose/	Implementation of object pose estimation.
ssm_v2/	Implementation of SSM v.2.
results/	Results of SSM Pose on testing datasets.

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Majer Lukáš** Personal ID number: **457193**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Object Localization by Spatial Matching of High-level CNN Features

Master's thesis title in Czech:

Lokalizace objektů pomocí prostorové informace vyšších vrstev CNN

Guidelines:

1. Get acquainted with methods for object detection and localization.
2. Get acquainted with the current state of the SSM-VPR (Semantic and Spatial Matching for Visual Place Recognition) framework for image retrieval [1], [2].
3. Explore the application possibilities of SSM for object detection and localization and modify the current SSM-VPR framework for the task.
4. Select suitable datasets and verify the behavior of the proposed solution [3]. Describe and discuss the obtained results. Focus on the benefits and limitations of the proposed solution.

Bibliography / sources:

[1] http://imr.ciirc.cvut.cz/Downloads/Software - SSM-VPR
[2] Camara L. G.,Přeučil L., Visual Place Recognition by Spatial Matching of High-level CNN Features, Robotics and Autonomous Systems, 2020
[3] Hodaň, T., Michel, F., Brachmann, E., Kehl, W., Glent Buch, A., Kraft, D., Drost, B., Vidal, J., Ihrke, S. Zabulis, X., Sahin, C., Manhardt, F., Tombari, F., Kim, T.K., Matas, J., Rother, C. BOP: Benchmark for 6D Object Pose Estimation, European Conference on Computer Vision (ECCV) 2018.

Name and workplace of master's thesis supervisor:

Ing. Viktor Kozák, Intelligent and Mobile Robotics, CIIRC

Name and workplace of second master's thesis supervisor or consultant:

Ing. Libor Přeučil, CSc., Intelligent and Mobile Robotics, CIIRC

Date of master's thesis assignment: **26.01.2021** Deadline for master's thesis submission: **21.05.2021**

Assignment valid until:
by the end of summer semester 2021/2022

Ing. Viktor Kozák	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student's signature