Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Magistrale in Computer Engineering

Master's Thesis

A Multi-view Pixel-wise Voting Network for 6DoF Pose Estimation

Advisor: Alberto Pretto

Author: Ivano Donadi - 2019147

July 18, 2022

Abstract

6DoF pose estimation is an important task in the Computer Vision field for what regards robotic and automotive applications. Many recent approaches successfully perform pose estimation on monocular images, which lack depth information. In this work, the potential of extending such methods to a multi-view setting is explored, in order to recover depth information from geometrical relations between the views. In particular two different multi-view adaptations for a particular monocular pose estimator, called PVNet, are developed, by either combining monocular results on the individual views or by modifying the original method to take in input directly the set of views. The new models are evaluated on the TOD transparent object dataset and compared against the original PVNet implementation, a depth-based pose estimation called DenseFusion, and the method proposed by the authors of the dataset, called Keypose. Experimental results show that integrating multi-view information significantly increases test accuracy and that both models outperform DenseFusion, while still being slightly surpassed by Keypose.

Sommario

La stima della posa a 6 gradi di libertà è un importante problema nel campo della Visione Artificiale per quanto riguarda applicazioni di robotica e guida autonoma. Esistono diversi modelli recenti che eseguono con successo la stima della posa su immagini monocolari, che non forniscono informazioni riguardo la profondità degli oggetti inquadrati. In questa tesi si esploreranno i vantaggi di estendere tali metodi al caso di camere multiple, che rendono possibile la stima della profondità degli oggetti inquadrati sfruttando relazioni geometriche tra le camere stesse. In particolare sono stati sviluppati due diversi adattamenti di un metodo di stima della posizione di oggetti, chiamato PVNet: uno che combina i risultati del sistema monocolare ottenuti indipendentemente dalle diverse camere, e l'altro che modifica invece il metodo originale fornendo in ingresso l'intero insieme di immagini catturate dalle camere presenti nel sistema. Entrambi i modelli sono stati testati sul dataset TOD di oggetti trasparenti, e confrontati con l'implementazione originale di PVNet, un metodo chiamato DenseFusion che sfrutta anche le letture di un sensore di profondità, ed il metodo proposto dagli autori del dataset, chiamato Keypose. Le sperimentazioni hanno mostrato che integrare le informazioni di più camere migliora considerevolmente il risultato della stima della posa, ed entrambi i modelli superano l'accuratezza di DenseFusion, pur non raggiungendo i risultati di Keypose.

Contents

Introduction

Object pose estimation, which is the task of detecting and estimating the position and orientation of an object, is a fundamental problem in many Computer Vision applications involving robotic manipulation, augmented reality or autonomous driving. Pose estimation on monocular images is particularly challenging due to the loss of depth information incurred when projecting a 3D scene into a 2D image. Despite this limitation, several recent deep learning based approaches have achieved impressive results in the task of pose estimation from pure-RGB monocular images [1, 2, 3, 4, 5]. All those models have to "work around" the missing depth information and must learn some bias, implicit in the data they are trained on, to be able to recover the depth.

The problem of missing depth in pose estimation can be mitigated by employing depth cameras, which associate each pixel in the RGB image with the reading from a depth sensor, as is done by [6, 7, 8, 9]. These systems, however, are restricted to indoor settings, might be imprecise when dealing with specular or transparent objects, and may require too much power for mobile applications [10, 11]. For these reasons, the problem of depth loss could be solved instead by leveraging geometrical constraints in a multi-camera system, which allows to recover depth as a function of the difference between 2D projections of the same 3D scene point in the different images.

This work aims to adapt a known pose estimation model working on monocular images, called PVNet [1], to the case of multiple camera systems, with a particular focus on stereo (dual cameras) setups. PVNet performs pose estimation in three steps:

1. define a set of 3D control points with respect to the object to detect. Examples of control points may be 3D bounding box corners or vertices extracted from the object's model with furthest point sampling. The centroid of the object is added as an additional control point;
2. use a residual neural network (see section 1.4.3) to perform object detection and, for each pixel in the input image belonging to the object, cast a vote for the location of each control point projected in the image plane;
3. perform a post-processing step in which votes from each pixel are aggregated to extract the final hypothesis for the location of control points in the image; at this point a PnP problem (see section 1.3.2) is solved to obtain the object's 3D position and orientation, by relating the 3D control points to their 2D correspondences in the input image.

In this thesis, three models are proposed to adapt PVNet to the multi-view setup:

- A "parallel networks" setup which runs multiple PVNet instances in parallel, one for each camera in the system, and then uses a novel multi-view iterative PnP solver in the post-processing step to aggregate the results;

- A novel "stereo network" setup which stacks images from all cameras in a single tensor and uses a modified version of PVNet to perform object detection and cast votes for each image in the stack; in contrast with the model above, this architecture allows the network to potentially extract features aggregating information from multiple images in the multi-view system; the post-processing step is performed as in the model above;
- Another "stereo network" approach but this time using a more powerful residual network backbone (from ResNet18 to ResNet34 [12]).

The performance of the models described above, and of the original PVNet model, are evaluated on the TOD dataset [13], consisting on images of 15 different transparent objects framed by a stereo camera system. All image pairs in this dataset frame a single object, so the models should return one and only one detection for each input image pair. To reflect this bias, two original post-processing steps are introduced: one which enforces object detection in each input image by redistributing detection probabilities for inputs with no detected objects, and one which enforces the presence of no more than one object in each input by creating an histogram of object centroid hypotheses and discarding votes not related to the "winning" one. The authors of the dataset also provided an original pose estimation model, called KeyPose [13], and its performance on the TOD dataset, which is used in this work as a baseline against which results can be compared.

This thesis is structured as follows:

- in Chapter 1 the background material related with the thesis is presented, including projective geometry, robust estimation frameworks, pose representation and neural network architectures;
- in Chapter 2 several approaches for image-based pose estimation are presented, starting from classical pre-deep learning methods up to modern state-of-the-art approaches; particular attention is given to PVNet [1] and KeyPose [13], which are used as baselines in the experimentation;
- in Chapter 3 the original contributions of this thesis are reported and described in detail,
- in Chapter 4 the dataset [13] and metrics used during the experimentation are presented, along with experimentation results and their discussion;
- finally, in Chapter 5 the thesis results are summed up and a discussion of strengths and weaknesses of the models developed is provided, along with ideas for future improvements.

Chapter 1

Background

In this chapter the theoretical background material needed to follow the contents of this work is presented, starting from the necessary background on perspective geometry and estimation algorithms, and ending with some insights on modern neural network models.

1.1 Pose

The pose of a rigid object is the description of the translation and rotation of a reference frame (O) attached to it with respect to a fixed "world" reference frame (W).

The translation component is represented in $\mathcal{R}^3$ as a 3-dimensional vector $t = [t_x, t_y, t_z]^T$, representing the translation components respectively along the x, y and z axis of the origin of one reference frame with respect to the other. For example t_O^W, the vector representing the translation of the object reference frame with respect to the world reference frame, is composed by the coordinates of the origin of reference frame O expressed in the world reference frame.

There are several possible representation for the rotation component, but only two are referenced in this work:

- **Rotation matrix:** $R \in \mathcal{R}^{3\times3}$. Each column represents an (unit) axis of the source reference frame, expressed in the target reference frame. Let A and B be two reference frames differing only by a rotation. Then it is possible to write a point expressed in frame B's coordinates (p_B) in frame A's coordinates as $p_A = R_B^A \cdot p_B$, where the first column of R_B^A is the vector expressing the x axis of frame B in frame A's coordinates, the second column of R_B^A is the vector expressing the y axis of frame B in frame A's coordinates, and the same for z in the third column of R_B^A. Due to the properties of rotation matrices, the inverse transformation can be obtained with the matrix $R_A^B = {R_B^A}^{-1} = {R_B^A}^T$. Due to the unit norm of each column, rotation matrices can be represented by 6 parameters instead of 9.

- **Axis-angle representation:** the transformation is represented as a single rotation by an angle $\theta \in \mathcal{R}$ about an unit axis $r \in \mathcal{R}^3$. Since the magnitude of r is constrained, it can actually be represented with a vector in $\mathcal{R}^2$. This representation is more compact than rotation matrices, using only 3 parameters instead of

6. The dimensionality difference is due to redundant information inside rotation matrices and not to approximations.

Let now t_O^W describe the translation of the object reference frame with respect to the world reference frame, and let R_O^W be the rotation matrix representing the rotation between frames O and W. Then, a generic point in the generic frame O can be represented in frame W with a single linear operation as $\hat{p_W} = T_O^W \hat{p_O}$, where:

$$T_O^W = \begin{bmatrix} & R_O^W & & t_O^W \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathcal{R}^{4\times4},$$

$$\hat{p_W} = [p_{W_x}, p_{W_y}, p_{W_z}, 1]^T, \hat{p_O} = [p_{O_x}, p_{O_y}, p_{O_z}, 1]^T$$

In this work, describing the pose of an object is synonym with providing the transformation matrix T_O^W, unless it is explicitly stated otherwise.

1.2 Projective geometry

1.2.1 Pinhole camera model

The simplest way to model a camera is through the pinhole camera model, where no lenses are used and the camera aperture is modeled as a single point (pinhole). The infinitesimal size of the aperture allows the camera to focus on objects at any distance from it. When light rays hitting objects in the scene framed by the camera are reflected and pass through the pinhole, they are projected into a plane called image plane. The distance between this plane and the camera aperture is a fundamental parameter called focal length (f). In real cameras, the image plane is usually identified with the one where the camera sensors are located. The axis connecting the pinhole camera with the aforementioned plane is called optical axis.

To transform 3D scene points into 2D image coordinates, a standard camera reference frame is introduced, centered at the intersection between the optical axis and the image plane, whose Z axis is aligned with the optical axis, going out of the camera. Usually the x axis is placed on the horizontal direction from left to right, while the y axis is in the vertical direction going from top to bottom. Given a 3D point $X_p = (X, Y, Z)^T$, it is possible to find the coordinates of its 2D projection $x_p = (x, y)^T$ using the similar triangles rule as $x = \frac{fX}{Z}$ and $y = \frac{fY}{Z}$. If we represent X_p and x_p with their homogeneous representations $\tilde{X}_p = (X, Y, Z, 1)^T, \tilde{x}_p = (xZ, yZ, Z)^T$, it is possible to formulate the transformation as a matrix multiplication:

$$\tilde{x}_p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{X}_p$$

The projection described above maps 3D points to 2D points, so it is possible to see the camera as a dimensionality reduction machine. Recovering the 3D coordinates of a point just by ts 2D projection is an ill posed problem, since multiple 3D points are mapped to the same 2D image plane coordinate.

Modern digital cameras use discrete sensors to acquire an image, thus performing an additional transformation between continuous image plane coordinates to discrete pixel coordinates. Since pixels coordinates conventionally are centered in the top-left corner of the image, the offsets between the image center and the top-left corner are needed as parameters: u_c in the x coordinate and v_c in the y coordinate. Let h_p and w_p be the height and width of the discrete sensor cells; then the mapping from image plane coordinates to pixels can be expressed as: $u = u_c + \frac{x}{w_p}$, $v = v_c + \frac{y}{h_p}$, where x and y are the image plane coordinates as described above. In order to go directly from 3D scene points to pixel coordinates with a single matrix multiplication, let's introduce the parameters $\alpha_u = \frac{f}{w_p}$ and $\alpha_v = \frac{f}{h_p}$. Then defining the camera matrix A as:

$$A = \begin{bmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

allows to obtain the pixel coordinates in homogeneous representation (up to a scale factor s) $\tilde{u}_p = (us, vs, s)^T$ as:

$$\tilde{u}_p = A\tilde{X}_p$$

Finally, if the 3D coordinates of the scene points are specified in a different reference frame compared to the camera one, it is possible to transform them into the camera frame coordinates prior to the projection using a transformation matrix $T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$:

$$\tilde{u}_p = AT\tilde{X}_p^{'}$$

where $\tilde{X}_p^{'}$ represents the same point as $\tilde{X}_p$ but in the other reference frame. Due to the nature of the matrix A, it is possible to rewrite the previous equation as:

$$\tilde{u}_p = K[R|t]\tilde{X}_p^{'}$$

where K is a re-definition of the camera matrix as:

$$K = \begin{bmatrix} \alpha_u & 0 & u_c \\ 0 & \alpha_v & v_c \\ 0 & 0 & 1 \end{bmatrix}$$

1.2.2 Stereo cameras

A pair of cameras mounted on a rigid frame is called stereo rig, and is useful to simulate human vision, which can estimate depth by using two cameras sharing a portion of their field of view instead of just one. Assume the cameras are mounted in such a way that it is clearly possible to differentiate between a left and right camera, and also that the camera matrices of both are known (K_l, K_r). Also assume that the rigid body transformation between the two cameras is known (T_{lr}), mapping points in the right camera frame to points in the left camera frame. It is then possible to relate the projections of the same 3D scene point in the two cameras using epipolar geometry. Let the baseline be the segment connecting the origin of the two camera frames, and P be a generic scene point. The two origins of the camera frames and the point P define a plane called epipolar plane. The intersections between the baseline and the two image planes are called epipoles, and the same name is given also to the projections of P in the two image planes (p_l and p_r). Due to the geometry of this system, all points that are projected in the same pixel as P in the left camera can only appear along a specific line in the right camera, called epipolar line. Thanks to this, the search of corresponding points in the two images is restricted to a line rather than to the whole image.

Let t_{lr} be the translation component of T_{lr} and let R_{lr} represent its rotation component. Let also $\tilde{p}_l$ and $\tilde{p}_r$ be the homogeneous representations of p_l and p_r. Since t_{lr} corresponds to the baseline, then t_{lr}, p_l and p_r lie all in the epipolar plane, and thus it holds that $\tilde{p}_l \cdot (t_{lr} \times R_{lr}\tilde{p}_r) = 0$, since the cross product will yield a vector orthogonal to the epipolar plane. This constraint, called epipolar constraint, can be re-formulated by transforming the cross product into a matrix multiplication by employing the skew-symmetric representation for $t_{lr} = (x_{lr}, y_{lr}, z_{lr})^T$, defined as:

$$\hat{t}_{lr} = \begin{bmatrix} 0 & -z_{lr} & y_{lr} \\ z_{lr} & 0 & -x_{lr} \\ -y_{lr} & x_{lr} & 0 \end{bmatrix}$$

which yields:

$$\tilde{p}_l^T \hat{t}_{lr} R_{lr} \tilde{p}_r = \tilde{p}_l^T E \tilde{p}_r = 0$$

where E is called essential matrix and encapsulates the relative rotation and translation between the two cameras in 5 degrees of freedom. The same relation holds going from the right camera to the left: $\tilde{p}_r^T E^T \tilde{p}_l = 0$.

Given a point p_l in the left image and the essential matrix E, the line in the right camera where the corresponding point lies can be obtained in homogeneous coordinates as $\tilde{l}_r = E^T \tilde{p}_l$. The same can be done in the opposite direction as $\tilde{l}_l = E\tilde{p}_r$.

1.2.3 Rectified stereo cameras

It is possible to apply a transformation to the images of both cameras so that the images appear as taken from two cameras with image planes parallel to the baseline and in the same plane. This is done by applying a pure rotation transformation to both images. In order to do so, the optical axes of both cameras need to be projected onto a plane perpendicular to the baseline, and their bisector is taken as the new optical axis. At this point a rotation transformation is applied to both images so that their new optical axis coincides with the one just found. Since the two new image planes intersect the baseline at infinity (given that they are parallel to it), any 3D point projected in the left image is projected in the same row in the right image and vice versa. Any stereo rig which employs this transformation is called a rectified stereo rig.

Let now $p_l = (x_l, y_l)^T$ and $p_r = (x_r, y_r)^T$ be the projections of the same point $P = (X, Y, Z)^T$ in the left and right image respectively. Given the discussion above, it follows that $y_l = y_r$. Let now B be the length of the baseline and f be the focal length of both cameras. Let also $d = x_l - x_r$ be a quantity called disparity. Knowing both projections, and thus the disparity, it is possible to backproject the two 2D points in the original 3D scene point, expressed in the left camera frame, by $Z = \frac{fB}{d}$, $X = \frac{Zx_l}{f}$ and $Y = \frac{Zy_l}{f}$. This same problem is ill-posed when performed on single images, but as it has just been showed, it has a closed form solution when employing at least two cameras. In the rest of this work, whenever stereo images are mentioned, it is assumed that they are a pair of images taken by a rectified stereo rig.

1.3 Estimation background

1.3.1 RANSAC

The Random sample consensus algorithm (RANSAC) [14] is a robust method to estimate the parameters of a mathematical model from the model definition and a set of input readings. The main strength of this approach is the ability to identify and filter out outliers in the input. RANSAC is an iterative algorithm which runs for a pre-determined number (N) of iterations. For each each iteration:

1. a minimum set of points is sampled from the input data, and is used to fit the model;
2. the extracted model is compared with each input data point, to find the number of inliers (points which are in agreement with the model). The maximum allowed distance from an input point to the corresponding one predicted by the model, in order to be considered an inlier, is a parameter of the algorithm and can be used to define the amount of noise tolerated in the input data.

After executing all the iterations, the N models are compared, and the one with the most inliers supporting it is chosen as the output. Usually the output model is fitted again on the whole inlier set to increase precision. The number of iterations should be tuned according to the input data quality, and is usually chosen based on the desired probability (p) of getting a good model in output: let ϵ be the estimated number of outliers in in the input data, and S be the size of a minimum point set needed to estimate the model's parameters, then the number of iterations can be derived in closed form as:

$$N = \frac{log(1-p)}{log(1-(1-\epsilon)^S)}$$

RANSAC is often used inside voting schemes, where the model parameters represent candidates and the input data represent the votes, as can be seen in section 2.4.

1.3.2 Perpective-n-Point problem

The perspective-n-point problem (PnP) [14] has the goal of estimating the pose of a camera from a set of n known 3D points and their corresponding 2D projections in an image captured from aforementioned camera.

There are several algorithmic solutions to this problem, like the P3P algorithm [15] for the case $n = 3$, or EPnP (efficient PnP) [16] for the general case $n \geq 4$. Another solution is to treat this problem as a non-linear optimization problem, where the target is to minimize the reprojection error between the original 2D data points and the 3D points reprojected using the camera matrix K and the current estimation of the transformation given by R, t:

$$R^*, t^* = \arg\min_{R,t} \sum_{i=1}^{n} \rho(||\pi(P_i) - p_i||_2)$$

where $\pi(x)$ denotes the camera reprojection operation, P_i is one of the known 3D points, p_i is one of the input 2D correspondencies and finally ρ is a loss function, like least squares or Cauchy. Typically, during the optimization, the rotation is represented with the axis-angle notation for compactness. This minimization problem can be solved with any non-linear solver like Levenberg-Marquardt [17].

1.4 Deep Learning background

1.4.1 Artificial Neural Networks

An artificial neural network (ANN) in the context of machine learning is a class of predictors described by directed graphs, whose nodes are called "neurons". The most basic type of ANN is called feed-forward neural network, where neurons are organized in subsequent layers. Each neuron in one layer is connected by an outgoing edge to every neuron in the next layer. In this architecture each neuron receives in input the weighted sum of the outputs of the neurons connected to its incoming edges and its output is the result of a function σ called "activation function" computed on its input ([18] pg. 228-229). The input to this model is set as the output of the first layer, while the output of the last layer constitutes the model's prediction. Feed forward neural networks can be trained with heuristic methods, like stochastic gradient descent (SGD) [19], to learn the parameters used to weight the outputs of each neuron. Artificial Neural Networks can be used to solve both classification and regression problems [18]. A neural network is called deep when there is more than one layer in the graph, excluding the input and output layers.

1.4.2 Convolutional neural networks

Convolutional neural networks (CNNs) [20] are a specialized kind of neural networks which encode the prior that the input data has a grid-like topology. These networks are very popular in the field of Computer Vision due to images being 2D grids of pixels and the fact that CNNs are able to exploit the spatial relationship between neighboring pixels.

Convolutional neural networks get their name from the convolution operation which substitutes the regular weighted sum described in section 1.4.1 in certain layers, called convolutional layers. In the case of 2-dimensional images, each convolutional layer is composed by a regular 2D grid of neurons. Let t and $t+1$ be the indexes of two subsequent layers in a CNN connected by the convolution operation, and let $I(i,j)$ denote the output of the neuron at index (i,j) in layer t. Let now K be a $n \times m$ matrix of learnable weights called "kernel". The input to the neuron at the generic index (i,j) of layer $t+1$ is then computed as:

$$(K * I)(i,j) \coloneqq \sum_{r=0}^{n} \sum_{s=0}^{m} I(i+r, j+s) K(r,s)$$

where $*$ denotes the convolution operation [21]. Convolution can work with inputs of any size, since it does not specifically describe each edge in the graph, as long as the kernel is smaller than the input. Since the kernel size is usually significantly smaller than the input size, and since the convolution uses the same kernel to compute each output value at the same layer, this operation is dramatically more efficient than dense matrix multiplication both in terms of memory requirements and statistical efficiency [21]. It is possible to apply a stride to a convolutional layer, which specifies when to apply convolution. A stride of 1 means that the operation is applied to every pixel position that allows the kernel to fall inside the input matrix, while a stride of 2, for example, applies the operation to a feasible pixel position and skips the next one.

By its nature, the convolution operation reduces the size of the input, creating a smaller representation of the input data. There are cases when it is desired to revert this process, like in generative convolutional networks, where the task is to produce in output an image possibly of the same size of the input. In order to do so, the smaller feature matrices obtained by convolution need to be re-mapped to the original size. This is done by an operation called transposed convolution, sometimes used interchangeably with deconvolution. Let n be a feature matrix, n_h and n_w be its height and width and let K be a kernel and k_h and k_w be its height and width. The result is a feature map of height $n_h + k_h - 1$ and width $n_w + k_w - 1$, which is the sum of a set of intermediate maps computed by multiplying the whole kernel by each element in the original feature matrix. Formally, it can be defined as:

$$\sum_{i=0}^{n_h} \sum_{j=0}^{n_w} Y(K, n[i,j])$$

where $Y(K, n[i,j])$ is a matrix of size $(n_h + k_h - 1) \times (n_w + k_w - 1)$ initialized with zeros and containing the kernel values multiplied by $n[i,j]$ starting from index i, j and ending at index $i + k_h, j + k_w$. Transposed convolution can be applied in strides to further enlarge the size of its output, and in this case the stride is applied on the output matrix, contrary to classical convolution where stride is applied to the input [22].

A convolutional network employing both convolution and transposed convolution usually has the shape of an hourglass: first the convolutional layers compress the input data into compact, abstract feature representations, and then those features are used in the transposed convolutional layers to generate the output, forming an encoder-decoder architecture, useful for example for predicting semantic segmentation maps [23]. Sometimes the image given in input to the first network layer is also used as an additional input to the last decoder layer, with the goal of producing in output an image which is a modification of the input one, for example in image restoration [24].

1.4.3 Residual networks

Residual networks [12] are deep neural networks in which the output of some layers is not only used as input to the subsequent one, but also as an additional input to a layer deeper down in the graph, “skipping” all the intermediate layers. For this reason these are called “skip connections”. Residual networks facilitate information flow across many layers [25] and ease optimization [12]. A particular residual network model, called “Resnet18” since it is 18 layers deep, presented in [12], will be referenced later in this work. Similarly, “Resnet34” and “Resnet50” are terms used to refer to residual networks which are, respectively, 34 and 50 layers deep.

1.4.4 Transformers

A transformer is a deep learning model which is able to weight differently the significance of each part of the input data through a mechanism called “self-attention”. The network realizing the transformer is shaped in an “encoder-decoder” model, where the encoder compresses a possibly structured input into intermediate vector representations and the decoder uses this representations as its input to provide a possibly structured network’s output. Both the encoder and decoder are organized in layers: each encoder layer learns attention (weight) matrices to understand which parts of the input are relevant to each other and should be encoded together, while each decoding layer learns attention matrices to understand which parts of the input encodings and of the previous decoder layer are useful to generate the desired output.

If the output of the encoder is a single vector, it is possible to make the transformer learn to group relevant information together in a restricted portion of the output vector. For example, if the network should make a fixed number of concurrent predictions, it is possible to divide the output vector in virtual segments and make the transformer focus each prediction in the dedicated virtual segment by employing an appropriate loss function, thus enabling parallelization, as done in [3].

Transformers are widely used in the field of natural language processing [26] and are recently gaining traction in the field of computer vision [27].

Chapter 2

Image-based 6DoF pose estimation

2.1 Problem definition

Pose estimation is a well known general problem in the field of computer vision, where it is required to detect the position and orientation of an object framed by one or multiple cameras and possibly also by depth sensors. Usually the object is known and its 3D CAD model is available as an additional input to the problem. Currently, solutions to this task can be divided into two main categories:

- **Holistic models** try to solve the pose estimation problem directly, taking in input the original images/depth readings and giving in output the relative transformation between a fixed reference frame (usually the frame of the camera) and a reference frame fixed on the object, without using intermediate representations. Examples of this approach are [28, 29, 30].
- **Two-step methods** try to solve the pose estimation problem by first computing a set of intermediate representations from the input images/depth readings, and then recovering the object's transformation from these intermediate representations. Most commonly, the intermediate representation used is a set of 2D interest points locations in the input images, which are then related to their corresponding 3D points on the object model using PnP algorithms, as seen in section 1.3.2. Examples of this approach are [10, 1, 2]. Some modern two-step methods also use additional intermediate representations, like edges or symmetry information [2].

2.2 Geometry-based methods

Prior to the rise in popularity and effectiveness of deep learning methods, the pose estimation problem was solved by leveraging geometrical constraints between the input images and the known 3D model of the target object.

Template-based methods: In the context of pose estimation from a sequence of video frames, the problem of extracting a 3D pose can be formulated as finding the parameters p of some deformation f mapping a known template T, of the object to be located, into a patch of the input image, thus working globally on aligning the whole object template rather than some of its local features. The solution to this problem can be found with the Lukas-Kanade [31] algorithm: let I denote the current input image and m_j a location inside the template. The matching error between an image patch and the template, given the current parameters, can be formulated as:

$$E(p) = \sum_j (I(f(p; m_j)) - T(m_j))^2 \tag{2.1}$$

By minimizing $E(p)$ one can find the best set of parameters matching a deformed version of the template to an image patch. In the Lukas-Kanade algorithm in particular, the aim is to start from given initial deformation parameters p and at each step finding the best parameter update Δ such that $p+\Delta$ is the best set of parameters for the current frame, which is reasonable for the purpose of tracking in a sequence of video frames [32]. The original problem is computationally expensive but it has been effectively used for 3D pose estimation under additional constraints [33, 34, 35]. When the object to track is planar, its pose can be extracted by an homography estimation once the deformation parameters have been found. In the same way, the object can be modeled with a simplified surface (like a generalized cylinder modeling a human head), in order to be able to recover the pose using geometrical constraints [33].

Other, more recent, template-based approaches for single-image pose estimation rely on other cues (like image gradient orientations) to match templates with the input images [29].

Edge-based methods: One approach to pose estimation in a sequence of video frames, proposed in [36] is to identify a set of "control points" along the edges of an object's 3D model and recovering their position in the input image. In order to perform control points detection, the system starts from a known object's pose and, for each video frame, it uses the previous pose to estimate which points are going to be visible in the current frame and where their location should be. After extracting image contours, the predicted control points positions are matched against the image edges and the pose is retrieved from the correspondences [32]. However, image contours are affected by shadows, clutter and occlusions, so additional precautions should be taken to avoid incorrectly detecting a non-existing edge and using points along it to predict the pose.

Another edge-based approach [37, 38, 39] is to extract image primitives, like straight edges or parametric curves, and match them directly with the same primitives extracted from the 3D model. The pose of the object is iteratively estimated by minimizing the Mahalanobis distance between primitives using the Levenberg-Marquardt solver [32].

Lastly, [30] is a template matchng method based on edge information which relies on the computation of the Hausdorff distance between the template and input image to find the image patch which best matches the template or a portion of it.

Feature-based methods: In 1997, [40] proposed using local features robust to illumination and contrast changes to describe a database of known images picturing the objects to be located. In particular, such work made use of the Harris corner detector [41] to detect interest points in the images and described them using "local jet" image derivatives [42], to achieve invariance to similarity transformations. At test time a new query image is fed to the system and its local salient points are extracted and compared with the images in the database in order to find the best match. Since each feature could match well with different images in the database, the system requires that the p best matches in both the query and retrieved image are spatially distributed in a similar way, by adding edges between the features and imposing that the angles between pair of edges have to be locally consistent. This allows the retrieval of images where the correct object has a different pose than the in the query image. A degree of multi scale invariance is obtained by computing the features at different scales only for the query images. This solution proved to be able to detect objects even under partial occlusions. In the context of pose estimation, knowing the correspondences of keypoints between a query image and a known database image would allow to extract the pose with methods such as PnP. In 1999, [43] improved the previous model by introducing SIFT features for object keypoint detection, which are intrinsically invariant to translation, rotation, scale and partially to noise and small distortions.

Fiducial-based methods: In a controlled environment, it is possible to add "fiducials" (easily recognizable markers with known 3D location) in the background of the scene in which a camera is located. They can be designed in such a way that they can be easily detected and identified by an "ad-hoc" method. Once a set of fiducials has been identified in an image, their correspondences with their known 3D locations can be used to solve a PnP problem to retrieve the camera pose [32]. These systems are only able to detect the camera pose, and need additional assumptions to be able to locate an object within the scene, so they are not flexible and not suitable for the general case of unknown background environment.

2.3 Deep learning methods

In this section, several neural models are shown, starting from early attempts and ending with modern, state-of-the-art proposals.

2.3.1 RGB-D input

Some deep learning architectures for 3d pose estimation use depth information together with color images to improve estimation quality. Their applicability may be restricted due to depth sensors failing outdoor or on specular objects; also depth sensors may be too taxing on the batteries of mobile devices to be used extensively on mobile applications [10, 11]. Nonetheless, in situations where depth readings are available they can be leveraged to increase the model's accuracy [6, 7, 8, 9].

This work will focus on methods that only require one or more RGB images as input, without the need for depth information, which is a more challenging problem. However, an approach using RBG and depth information is still discussed briefly since it will be used as a baseline comparison for the experiments in chapter 4.

DenseFusion: This recent method tries to combine RGB and depth information in dense pixel-wise feature embeddings using a novel dense fusion network [9]. The reasoning behind this approach is that depth information is very useful in cluttered or otherwise adversarial environments, but it resides in a different space than RGB information, so there is the need for a way to combine features extracted from these heterogeneous data sources. The model proposed uses an heterogeneous architecture to process RGB and depth data differently, and then a dense pixel-wise fusion network performs color-depth fusion for pose estimation. Finally, the pose is refined with a lightweight differentiable iterative module. Experimental results show that training a dedicated fusion network outperforms simple feature concatenation by a large margin.

2.3.2 Pure RGB input

As mentioned above, this work focuses on pose estimation methods relying only on RGB color images as input, without any explicit depth measure. This makes the problem more complex to solve in monocular pose estimation on opaque objects but has a broader range of application, since it can be also applied to images acquired with cheap and mobile devices, like a cellphone camera.

2.3.3 First approaches

Learning descriptors: in 2015, [44] proposed a convolutional architecture with the task of learning images descriptors for object identification and pose estimation. In contrast with the methods described in 2.2, the descriptors are not manually defined with specific invariances in mind, but rather are directly extracted from the input data by enforcing dissimilarity between images of different objects, while the distance between descriptors of images framing the same object should be related to the difference in the poses of the framed object. Such descriptors are able to outperform hand-crafted ones like SIFT or SURF significantly [44]. Pose estimation is then achieved by comparing the input image with images in a database containing different views of the object and finding the most similar one, in terms of euclidean distance between descriptors. Employing euclidean distances allows using fast nearest-neighbor search methods for image retrieval.

The network is trained with a triplet loss: each input sample consists in two similar images and a dissimilar one. The loss brings the descriptors of the two similar ones close to each other, while distancing them from the descriptor of the dissimilar one. In addition to the triplet loss, a pair-wise term is added, which considers a pair of images framing the same object ideally in the same or in a very similar pose, and minimizes the distance between the two descriptors, to add robustness to noise and artifacts such as illumination changes.

Pose estimation as a classification task: In 2016, [45] built upon a deep object detector [46] to obtain a pose estimator. Pose estimation is treated as a classification problem by discretizing the pose space in disjoint bins and predicting the correct one. The network used is obtained by substituting the last layers of an image classification network such as VGG [47] with convolutional and max-pooling layers. The network takes in input a single RGB image and produces scores for: each image category, a constant number of detected objects bounding boxes and the discretized poses. The results are then filtered with non-max suppression to produce the final output. Experiments showed that this method provided results comparable with the state-of-the-art at that time while providing a significant speedup, due to the object detection and pose estimation being done in a single shot inside the same network. Moreover, discretizing the pose space allows to obtain a confidence metric on the predicted pose, at the cost of accuracy.

2.3.4 Modern holistic methods

PoseCNN: to handle the complex task of pose estimation, the PoseCNN model [4] proposes to use a deep convolutional network which is able to decouple the translational part of the pose, which reflects an object's position and scale in an image, from the rotational part, which instead affects the image appearance of the object. To achieve this, a two-stage network is used: in the first stage general image features are extracted using convolutional layers, while in the second stage there are three different branches performing specific tasks leading to the pose estimation:

- **semantic labeling:** the first branch classifies each pixel in to an object class using softmax and is trained with cross-entropy loss;
- **3D translation estimation:** the second branch estimates the object's 3D translation by making each pixel in the image vote for the direction of the 2D object's centroid and its depth, and is trained with a smooth l_1 loss; then a Hough voting layer is integrated in the network, taking in input both the pixels' votes and their semantic label and giving in output the most likely 2D centroid location + depth for each class; finally, a bounding box is created by surrounding all the pixels that voted for the winning centroid location;
- **3D rotation regression:** the bounding boxes predicted in the second branch are used to filter the features extracted by the first network stage. The filtered features are fed to a sequence of fully connected layers which predict the object's orientation represented by a quaternion; this branch is trained with a squared loss on the 3D point to point difference between the object model's vertices rotated with the ground truth and predicted quaternion.

SSD-6D: This model improves the SSD paradigm described in paragraph 2.3.3 to cover the full 6D pose space, rather than discretizing it [5]. Since direct 6D pose regression is a difficult problem, the approach proposed is to predict, for each object, the viewpoint under which it is framed and an in-plane rotation w.r.t. the viewpoint axis. From these two informations, it is possible to use projective geometry to obtain a 6D pose hypothesis.

SSD-6D takes in input an RGB image framing some objects of interest and uses the SSD object detection paradigm [46] to predict bounding boxes, their associated object ID and prediction confidence, the object's viewpoint and the in-plane rotation. To increase robustness, instead of predicting just one bounding box for each detected location, multiple ones are created with different aspect ratios. At this point all bounding boxes with a confidence above a certain threshold are kept and non maxima suppression is used as an additional filter. This means that for each detected object the method outputs a pool of views and in-plane rotations with the highest confidence, from which a pool of 6D poses can be estimated. To refine the pose estimation, each hypothesis is used to render the object into the scene and extract a series of 3D contour points, which are projected into the image to find their closest scene edges. The pose refinement step is then treated as a weighted least squares problem minimizing the distance between control points projections and their closest scene edge. This procedure gives multiple poses for each object instance and the best one is chosen by rendering again the object into the scene and keeping the pose which minimizes the average deviation of contour gradients between the render and the scene via absolute dot product.

At training time, only synthetic images are used, which are created by rendering the object's CAD model over images from the MS COCO dataset [48]. When rendering, the closest discrete viewpoint and in-plane rotation are taken as the ground truth, in addition to all the possible bounding boxes with an IoU metric above 0.5 with the rendered object's mask. The training loss is computed as follows: given a set of positive boxes, created with the method mentioned above, and hard-mined negative boxes, the sum of the class probabilities loss for both positive and negative examples is computed, with additional terms for bounding box tightness, viewpoint and in-plane rotation classification for the positive samples. Classification losses are computed as softmax cross-entropy, while the boxes' tightness is computed as a smooth l_1 loss. The negative examples are created by choosing unassigned boxes with high class probabilities. To avoid symmetry ambiguity, symmetrical object's samples are only taken from viewpoints which produce unique projections.

This method proved to be quite effective and able to match and surpass RGB-D methods that were state-of-the-art at the time (2017). It is crucial for this method to be effective that the training viewpoints are sampled with a fine grain to avoid missing viewponts and that the rendered images are as close as possible in appearance to the real image instances.

2.3.5 Modern two-step methods

BB8: one of the first instances of two-step keypoint based object detection is the "BB8" network [10], whose name is an acronym which stands for the 8 corners of a 3D bounding box. The main idea is to train a convolutional neural network to perform object segmentation and to predict the 2D projections of the 8 bounding box corners around the object. The output of the network is then given in input to a PnP solver which returns the object's pose. The main advantages of such method are that it does not require a depth map in input and that 2D keypoint prediction is a simpler task than regressing a continuous pose representation but, on the other side, the pose estimation part comes after the network's prediction, so it is not possible to train the model directly on a loss defined on the pose.

The model is the composition of two convolutional networks:

- the first one takes in input the original RGB image and performs object segmentation, giving in output the 2D centroid of the object, computed from the segmentation mask;
- the second one takes in input an image patch surrounding the object's centroid and outputs the 2D locations of the 8 bounding box corners for the object. This network is trained with a squared loss on the keypoints' reprojection.

To handle objects with a symmetry of rotation of α degrees about an axis β, the network's training is done by restricting the range of rotations about such axis between 0 and $\alpha/2$. At inference time a simple CNN classifier is used to predict whether the object's rotation about axis β is in the range $[0; \alpha/2[$ or $[\alpha/2; \alpha[$. In the latter case, the input image is mirrored so that the network still operates in the usual range, then the output keypoints are mirrored again to predict the correct pose.

YOLOPose: a very recent and promising approach to deep pose estimation is given by YOLOPose [3]. In contrast with the other methods mentioned in this section, YOLO-

Pose leverages the transformer architecture after a Resnet50 backbone. The main idea of this approach is to:

- use a transformer-based real time single stage model to predict the 2d projections of a set of 3D keypoints defined on the object's CAD model;
- go through a learnable rotation estimation module (fully connected network) which predicts the object's rotation from the 2D keypoints;
- a third and last independent fully connected network is used as a translation estimator.

The architecture described above allows end-to-end learning without the hassle of going through a non-differentiable PnP module. Although differentiable PnP modules do exist, the authors were discouraged by the additional training overhead they added, and opted for the simpler architecture described above.

YOLOPose formulates the keypoint regression problem as a set prediction problem. Given an input RGB image, the model predicts a fixed size set of elements, each of which contains 2D bounding boxes, class probability, translation and keypoints. The translation component is regressed directly following the same procedure described in paragraph 2.3.4. The rotation estimation module takes in input the predicted keypoints and outputs the rotation estimation in the form of a regressed rotation matrix.

During training the transformer module is trained with a 4 component loss:

- the class probability loss is modeled with negative log likelihood;
- the bounding boxes loss is a combination of generalized IoU and l_1 loss;
- the keypoint loss combines l_1 loss and a cross-ratio component proposed in [49];
- finally the pose loss is the the l_1 loss between the 3D keypoints rotated with the ground truth rotation matrix and with the predicted rotation, plus the l_1 loss of the translation component alone.

For what concerns the choice of keypoints, YOLOPose uses 32 keypoints consisting of the 8 bounding box corners and their interpolations, as described in [49]. This choice has the advantage of always having 4 collinear points when projected into an image, and the distance ratio between them can be exploited in the cross-ratio loss component to improve the results.

Performance-wise this architecture proved to perform similarly to state-of-the-art approaches in the YCB-video dataset [4], while being real-time capable due to the parallelization abilities given by transformers.

2.4 PVNet

The work presented on this thesis is built upon the PVNet architecture, which stands for pixel-wise voting network for 6DOF pose estimation [1]. As the other two-step methods for pose estimation, the main idea is to predict the 2D position of keypoints in the input RGB image and match them with their corresponding 3D model keypoints using PnP. What's particular about PVNet is that every pixel in the image corresponding to the target object votes for the position of each keypoint; then the votes are aggregated using RANSAC to obtain the final 2D coordinates. The PnP problem solution is also improved by providing in input the candidate keypoint positions together with their variance, which can be computed during the voting process, in order to give more weight to correspondencies with low variance.

Keypoints selection: The PVNet architecture requires a 3D CAD model of the object to be located, together with K 3D keypoints. Usually K is chosen equal to 9, where the first eight are computed with farthest point sampling [50], while the ninth is the centroid of the object.

Voting scheme: PVNet is based on the "ResNet18" (1.4.3) architecture, and it takes in input a single 3-channel RGB image and outputs a segmentation mask localizing the object, together with unit vectors connecting each pixel to each keypoint. Let p be a pixel in the input image, and let x_k be one of the K 2D keypoint position to be predicted. Then, the network's prediction for p will be a semantic label, identifying the object it belongs to, and K unit vectors $v_k(p)$ representing the direction of the line connecting p and each x_k: $v_k(p) = \frac{x_k - p}{||x_k - p||_2}$. From the segmentation mask and unit vectors, keypoint location hypotheses can be generated taking the intersections of the vectors belonging to two random pixels in the object. A set of N hypotheses $h_{k,1} \dots h_{k,N}$ is generated and each one is weighted according to the following formula:

$$w_{k,i} = \sum_{p \in O} \mathbb{1}\left(\frac{(h_{k,i} - p)^T}{||h_{k,i} - p||_2} v_k(p) \geq \theta \right) \tag{2.2}$$

where $\mathbb{1}$ is the indicator function, O is the set of 2D projected object points, and θ is a threshold usually set to 0.99. This weighting function measures the percentage of object points whose predicted direction match the one of the vector connecting them to the hypothesis. The final keypoint position is computed as the weighted average between the hypotheses:

$$\mu_k = \frac{\sum_{i=1}^{N} w_{k,i} h_{k,i}}{\sum_{i=1}^{N} w_{k,i}} \tag{2.3}$$

while the keypoint variance can be obtained as:

$$\Sigma_k = \frac{\sum_{i=1}^{N} w_{k,i} (h_{k,i} - \mu_k)(h_{k,i} - \mu_k)^T}{\sum_{i=1}^{N} w_{k,i}} \tag{2.4}$$

Uncertainty-driven PnP: since the keypoints are predicted together with their variance, we can use the latter to formulate the PnP problem as weighted minimization:

$$\arg\min_{R,t} \sum_{k=1}^{K} (\tilde{x_k} - \mu_k)^T \Sigma_K^{-1} (\tilde{x_k} - \mu_k) \tag{2.5}$$

where $\tilde{x_k}$ is the reprojection of the k-th 3D keypoint X_k on the image: $\tilde{x_k} = \pi(RX_k + t)$, with π being the perspective reprojection function.

Performance: the PVNet architecture proved to be very effective, surpassing the performance of the state-of-the-art at that time on two popular datasets for 3D pose estimation: LINEMOD [51] and Occlusion LINEMOD [52]. The authors also created a new test dataset by randomly cropping images in the LINEMOD dataset so that only 40% to 60% of the object surface is still visible. The voting scheme described above proved to be robust against occlusions and truncations due to its ability to potentially predict keypoints outside the object's mask and even outside the image.

2.5 Keypose

The dataset which will be used in this thesis (see section 4.1) was introduced in 2019 by [13], which also described a novel 3D pose estimation network called Keypose, which exploits stereo images to let the system learn an implicit depth representation. This method uses keypoints to predict objects' poses but is not really a two-step method since the network directly predicts the 2D keypoints' positions along with their depth, so the 3D keypoints can be directly extracted by back-projection in a single step. The set of 3D keypoints is used to define the object's pose, rather than a direct representation through a transformation matrix.

Network structure Keypose takes in input the two stacked stereo images and puts them through a set of dilated convolutional layers to predict a 2D location heatmap for each keypoint, which is used to obtain a spatial probability distribution for each keypoint, together with a disparity heatmap. The probability map for each keypoint is integrated to obtain the final 2D keypoint position, and is also convoluted with the disparity heatmap to obtain disparities, from which depths can be directly inferred with epipolar geometry constraints. The output of the network is the set of UVD keypoints locations (2D spatial position + depth)

Training The network is trained with three combined losses, from the keypoints' ground truth UVD values:

- a keypoint loss, measuring the squared loss difference between predicted and ground truth UVD values;
- a projection loss, measuring the squared difference between the 2D projections of both predicted and ground truth UVD values in the original stereo images;
- a locality loss which encourages the 2d keypoints' position heatmap to be unimodal and localized around the true keypoint location.

For some symmetric objects, there could be indistinguishable keypoints from the point of view of the pose estimation. For this reason, when training the network, the loss is computed on the best permutation of keypoints, obtained according to the object's symmetry. For some objects it is also possible to avoid symmetry ambiguity with a smart keypoint choice, like just a keypoint in its center is sufficient to define a sphere.

Performance: The experiments were conducted on the TOD transparent objects dataset, created by the same authors (see section 4.1 for more details). Keypose's performance was tested against the DenseFusion [53] architecture. The results of the tests showed that Keypose's ability to leverage stereo images allowed it to overcome the difficulties in predicting transparent object's poses, outperforming DenseFusion, which struggled with some more challenging objects, despite its ability to exploit depth maps.

Chapter 3

Multi-view Pixel-wise Voting Network for 6DoF Pose Estimation

In this thesis, several variations of the PVNet network presented in section 2.4 are proposed, with the intent of extending the network's domain to rectified stereo images. The purpose of this chapter is to introduce the various models, the ideas behind them, and the original contributions provided for their realization.

3.1 Pose estimation from stereo keypoints

As discussed in section 2.4, the main idea behind PVNet is to predict the 2D positions of a set of keypoints in an image, and then use PnP to estimate the object pose. Assume now to have two images which are the output of a rectified stereo rig, and assume to be given the 2D positions of the same set of keypoints in both images, either as the result of PVNet or by any other mean. It would obviously be possible to extract two pose estimations by applying PnP independently to the two inputs. What is presented in this section is a modification of the PnP problem in order to leverage the knowledge of both sets of 2D keypoints at the same time, assuming that the geometry of the stereo rig is known.

Let $K = \{K_1, K_2, \ldots, K_N\}$ be a set of 3D keypoints, of cardinality N, and let $k_l = \{k_{l,1}, k_{l,2}, \ldots, k_{l,N}\}$, $k_r = \{k_{r,1}, k_{r,2}, \ldots, k_{r,N}\}$ be the sets of 2D keypoints projections respectively in the left and right image. It is then possible to rewrite the PnP problem seen in section 1.3.2 as a non-linear optimization problem of the type:

$$R, t = \arg\min_{R,t} \left(\sum_{i=1}^{N} ||\pi(RK_i + t) - k_{l,i}||^2 + ||\pi(RK_i + \bar{t}) - k_{r,i}||^2 \right) \tag{3.1}$$

where $\pi(\cdot)$ is the perspective projection function and $\bar{t} = t + (0, -b, 0)^T$ is the translation associated with the right image, with b being the length of the baseline of the rectified stereo rig.

In this thesis, an iterative solver for this problem was developed using the Ceres optimization library for C++ [54]. Experimental results showed that the solver converges easily with good 2D keypoint predictions even from an empty initial guess for R and t. During testing, the effects of providing a more accurate initial transformation, for example by back-projecting the 2D stereo keypoints, were negligible, so the problem is always initialized with null initial rotation and translation.

This approach to solving the stereo PnP problem was also proposed in [11], under the name of "Object Triangulation".

3.1.1 Integrating variance information

If the 2D keypoints positions are computed using the method proposed by PVNet, then they also come with their associated variance, which can be used to transform PnP in a weighted optimization problem, as in Eq. 2.5. Let N, K, k_l, k_r be as above. Let also $\Sigma_l = \{\Sigma_{l,1}, \ldots, \Sigma_{l,N}\}$ and $\Sigma_r = \{\Sigma_{r,1}, \ldots, \Sigma_{r,N}\}$ be the covariance matrices of the N 2D keypoints in the left and right images respectively. Then, Eq. 3.1 can be modified to obtain:

$$\begin{aligned} R, t = \arg \min_{R,t} & \sum_{i=1}^{N} ||(\tilde{k_{l,i}} - k_{l,i})^T \Sigma_{l,i}^{-1} (\tilde{k_{r,i}} - k_{l,i})||^2 + \\ & \sum_{i=1}^{N} ||(\tilde{k_{r,i}} - k_{l,i})^T \Sigma_{r,i}^{-1} (\tilde{k_{r,i}} - k_{l,i})||^2 \end{aligned} \tag{3.2}$$

where $\tilde{k_{l,i}}$ and $\tilde{k_{r,i}}$ are the projections of keypoints K_i respectively in the left and right images, obtained as in Eq. 3.1.

An iterative solver for this problem was also developed using the Ceres library [54]. Contrary to the previous case, experimental results proved that this problem required accurate initialization to converge to a good solution. What worked best in the experiments was to initialize the transformation by solving the classic PnP problem from the 4 keypoints with lowest variance, as suggested in the original PVNet paper. To select the view from which the initialization keypoints are extracted, the views are ranked with a score defined as the sum of the trace of the covariance matrices of the 4 keypoints with lowest variance. Such score can also be used to discard noisy views that would compromise the optimization: if the score of the worst view is greater than three times the score of the best view, then all the weights associated with the worst view are set to zero, so that they don't interfere with the optimization procedure in the better view.

3.2 Parallel network

A straightforward way to adapt PVNet to stereo RGB input is to predict keypoints for the two images separately and then combining the result solving the stereo version of PnP described above. This approach will be referred to as the "parallel network" approach in the remainder of this work.

This method was introduced in [11], solving the PnP problem without the variance information. In this work, the variance is instead considered since it has shown to significantly increase precision during experimentation.

3.3 Stereo network

This approach changes the input and output layers of the original PVNet network while keeping the core untouched. The new input to the network is composed by stacking the left and right RGB images on top of each other, and the output contains segmentation masks and pixel votes for both images. Essentially, this approach is similar to the parallel one of the previous section, but the network is made aware of both images and is allowed to share weights when processing the two images. The goal of this model is to see whether the network is able to implicitly learn how to merge information coming from both inputs. The resulting network is referred to as "stereo network 18" since it has stereo inputs and uses the same ResNet18 backbone as the original PVNet.

Note: Since keypoints are obtained by post-processing the network's output, it is not possible to directly relate keypoints from both images in the network's loss. A number of attempts were made to predict parameters tying together the votes for both images in order to include them in the loss (like predicting angle differences between vote vectors belonging to corresponding pixels in the two images), but all proved to be too complex for the network to learn. A possible way to overcome this problem would be to integrate into the model a differentiable RANSAC module, and modify the loss to include both a term adjusting the votes and one enforcing epipolar constraints on the resulting keypoints.

3.3.1 Stereo network 34

The stereo network proposed in the previous section uses the same backbone network as the original PVNet, i.e. ResNet18, while performing a more complex task. For this reason, another model was tested by substituting the backbone network with a ResNet34, with the goal of providing stronger feature extraction abilities. Current state-of-the-art methods like YOLOPose [3], presented in paragraph 2.3.5, are using ResNet50 as their backbone while still retaining real-time capabilities, so the added training and inference complexity was deemed not to be detrimental to the cause of fast object pose detection.

3.4 Single object assumption

As is detailed in section 4.1, the dataset used in this work is composed of images containing one and only one object at a time. This prior can be leveraged to provide robustness to the estimation process.

Enforcing object detection: As will be discussed in Chapter 4, the dataset used in the experiments contains a small number of very challenging instances, where objects are difficult to detect even to the human eye. In such cases, the system could fail to detect the object, providing an empty segmentation mask. Given the prior that all images frame at least one object, it is possible to force the system to make a prediction by redistributing class probabilities. This is done by increasing the probability that each pixel belongs to the object by a small fixed amount, until a minimum threshold of object pixels is reached (computed empirically by extracting the minimum number of pixels for a mask in the training set). In order not to burden the system with unnecessary additional noise, this process is performed only in the case of completely empty masks; this means that the system may perform detections in which the number of object pixels in the mask is lower than a very small threshold, to include images with virtually (but not fully) empty detections.

Enforcing unimodal masks: Let $\mathcal{M}$ be the segmentation mask given in output by PVNet applied on an image. If $\mathcal{M}$ has multiple connected components, it is possible that multiple plausible object center hypotheses are found during the RANSAC voting phase. This phenomenon may cause keypoint predictions to be influenced by the high inliers count of a wrongly detected region of $\mathcal{M}$. To suppress the influence of such areas, the following algorithm is proposed:

1. use a RANSAC-based voting scheme to predict hypotheses only for the objects' centers;
2. aggregate the resulting hypotheses in a 2D histogram (in the experiments the bin size is always 15x15 pixels);
3. find the two bins with the highest number of votes;
4. if the bin with the second most voted bin has a vote count higher than a set percentage of the top bin's (0.01 in the experiments) then a noisy center is detected in the location of the second best bin;
5. if a noisy center is detected, alter the segmentation mask by silencing all pixels which are closer to the noisy center than to the top hypothesis.

During the experimentation, a smaller bin size (5x5) with a higher noise detection threshold (0.6 percent the winning bin votes) has been explored, but larger bin sizes have resulted to be more reliable.

Chapter 4

Experiments

4.1 TOD Dataset

The transparent object dataset (TOD) was introduced by [13] along with a novel 3D pose estimation network (Keypose) analyzed in section 2.5. The dataset contains 15 different transparent objects, including a sphere, 3 bottles, 2 cups, 7 mugs, a heart and a Christmas tree, displayed in Figures from 4.1 to 4.8. The dataset includes, for each object:

- rectified stereo RGB images (1280x720 pixels);
- a depth-map acquired framing the transparent object;
- a depth-map acquired by substituting the transparent object with an identical opaque counterpart;
- 3D keypoints definitions;
- 2D projections of each keypoint in the images plus their depths;
- the segmentation mask for the left stereo image.

The RGB images for each object are divided into 10 "textures", each of which represents a different background image over which the transparent object is placed, as shown in Figures from 4.9 to 4.13. The authors of the dataset mention that the suggested use of this dataset is to perform training on images taken from 9 textures, leaving out one. The images taken using the texture that is left out are to be considered the test set.

This is a non-trivial dataset to perform pose estimation on, due to the transparent nature of the objects and the complexity of the background textures, which make object detection hard when pictures are taken under some challenging points of view.

In this work, the dataset was augmented by:

- extracting a ground truth pose by solving an Orthogonal Procrustes problem [55] on the 3D model's keypoints and ground truth 2D projections + depth;
- producing the segmentation masks for the right camera images by reprojecting object vertices.

It is to note that all images in this dataset contain one and only one object, simplifying the task.

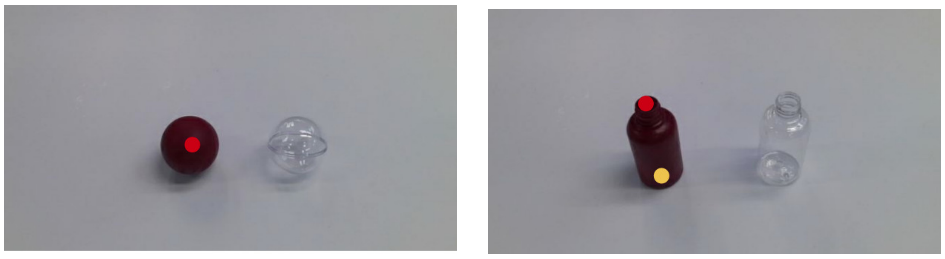

Figure 4.1. ball_0 and bottle_0 objects, with the keypoints used by Keypose, next to an opaque counterpart

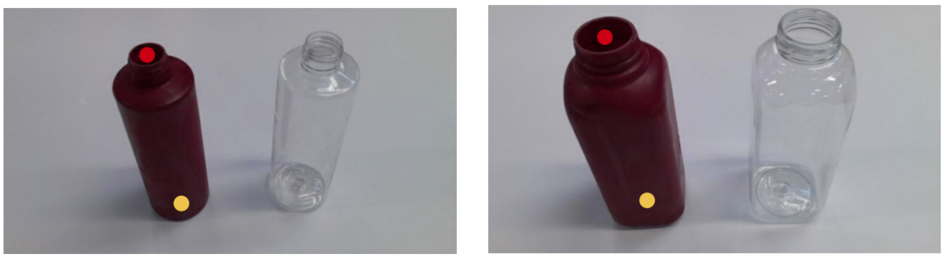

Figure 4.2. bottle_1 and bottle_2 objects, with the keypoints used by Keypose, next to an opaque counterpart

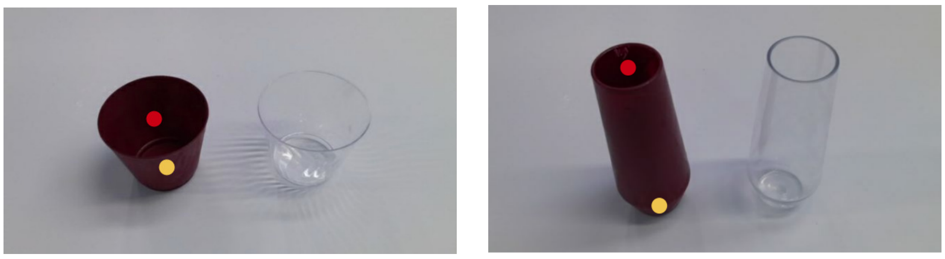

Figure 4.3. cup_0 and cup_1 objects, with the keypoints used by Keypose, next to an opaque counterpart

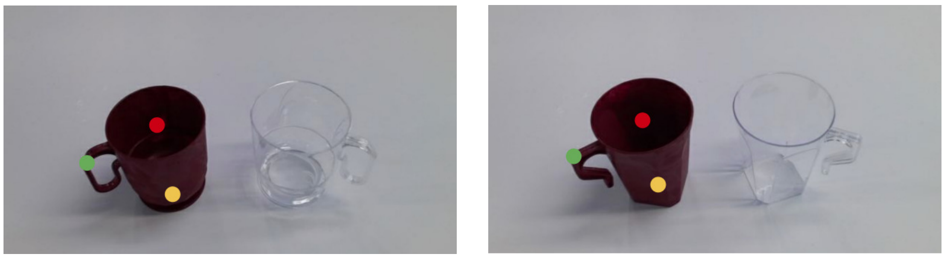

Figure 4.4. mug_0 and mug_1 objects, with the keypoints used by Keypose, next to an opaque counterpart

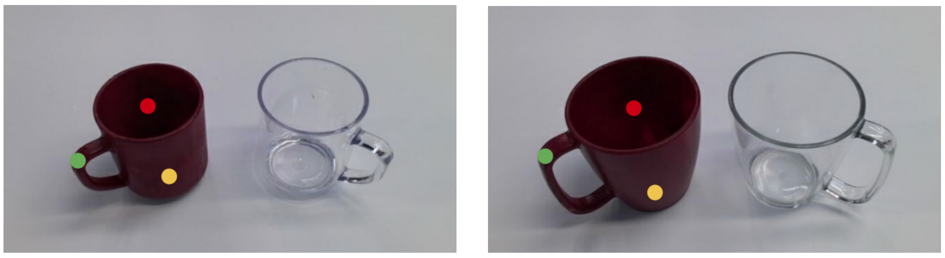

Figure 4.5. mug_2 and mug_3 objects, with the keypoints used by Keypose, next to an opaque counterpart

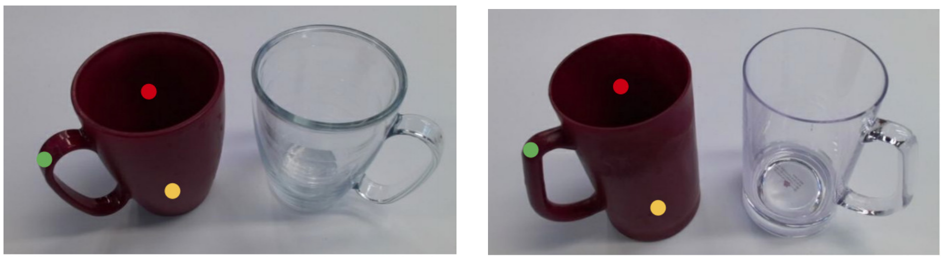

Figure 4.6. mug_4 and mug_5 objects, with the keypoints used by Keypose, next to an opaque counterpart

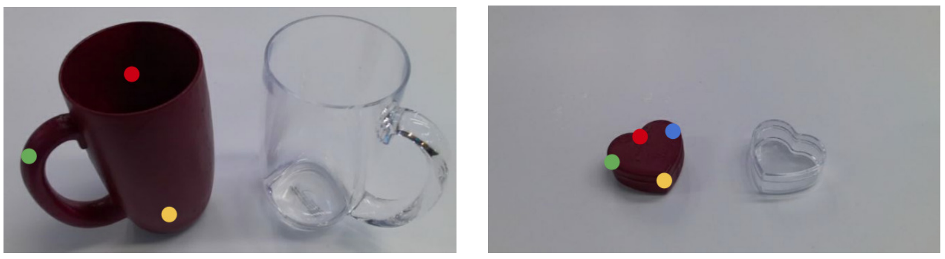

Figure 4.7. mug_6 and heart_0 objects, with the keypoints used by Keypose, next to an opaque counterpart

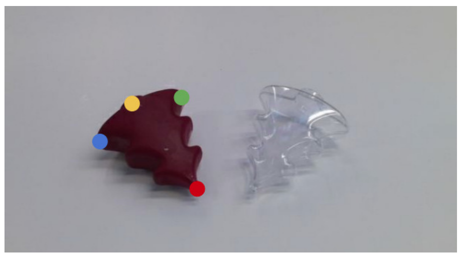

Figure 4.8. tree_0 object, with the keypoints used by Keypose, next to an opaque counterpart

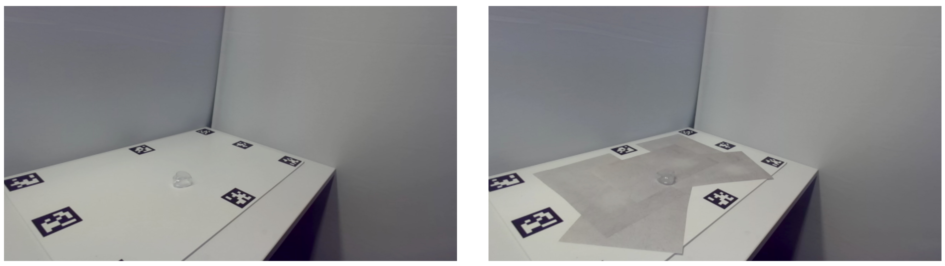

Figure 4.9. heart_0 object over textures 0 and 1

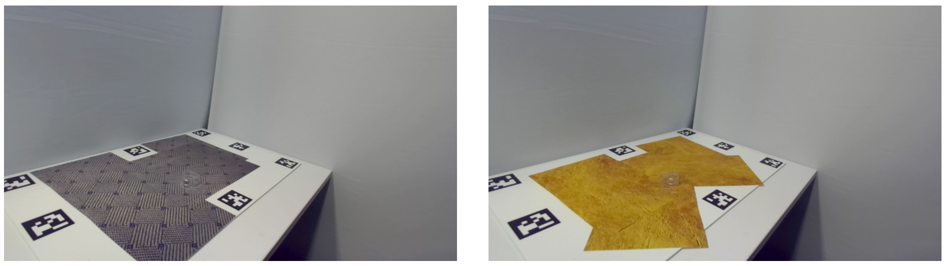

Figure 4.10. heart_0 object over textures 2 and 3

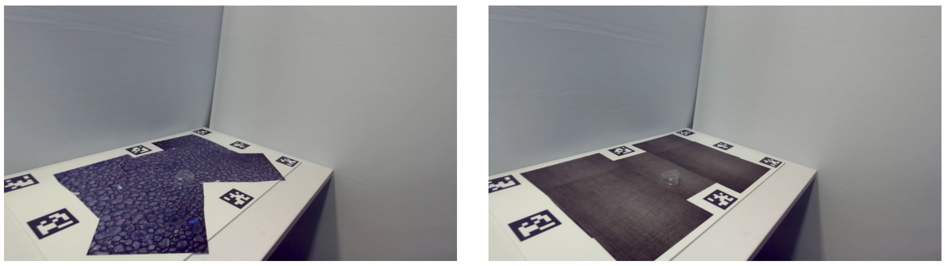

Figure 4.11. heart_0 object over textures 4 and 5

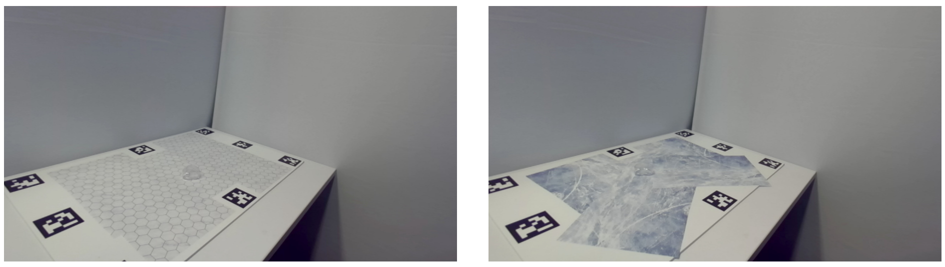

Figure 4.12. heart_0 object over textures 6 and 7

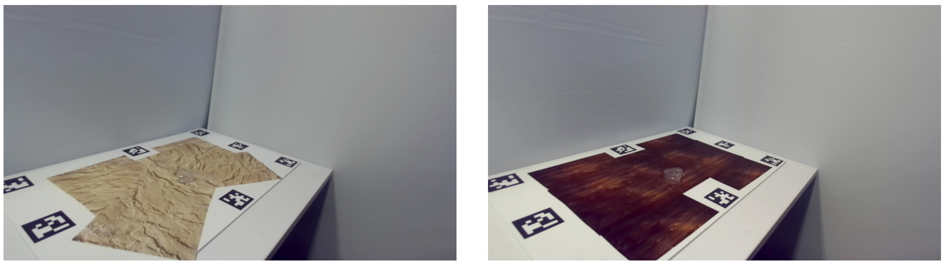
Figure 4.13. heart_0 object over textures 8 and 9

4.2 Metrics definition

The object detection and pose estimation tasks are evaluated on the following metrics:

- Mask AP 70: percentage of samples whose mask labels are correct for at least 70% of the pixels;
- 2D projections metric: percentage of samples for which the average distance between predicted and ground truth 2D keypoints is at most 5 pixels;
- ADD(-S): percentage of samples for which the average distance between predicted and ground truth 3D keypoints is less than 10% of the object's diameter; if the object has a symmetry axis, then distances are computed between each predicted keypoint and its closest ground truth keypoint; in the TOD dataset, the ball, cups and bottles are considered symmetrical objects;
- $<$ 2cm metric: like ADD-(S) but the threshold is static at 2cm instead of 10% of the object's diameter;
- 5 cm 5 degrees metric: percentage of samples whose predicted transformation matrix differs from the ground truth one by a maximum of 5 degrees in rotation and 5 centimeters in translation;
- MAE (mean absolute error): average absolute distance between predicted and ground truth 3D keypoints for all samples;
- AUC (area under curve): computes the area under a 2D curve where the x values are a set of thresholds from 0 to 10 cm and the y values are the ADD(-S) metric computed with the corresponding threshold.

4.3 Networks training

Each model presented in the previous chapter was trained on 10 different sets, each one obtained by selecting samples coming from 9 textures out of 10 (as detailed in table 4.1). The remaining texture is then used as the test set, while the validation set is obtained by randomly picking 15% of the training samples. This results in a completely disjointed train-validation-test split. Training and validation set are composed of similar samples, while the test set contains instances that significantly differ from the training ones, due to the different background texture.

All architectures and models were trained for 150 epochs and the best network parameters were chosen on the basis of validation ADD(-S) and Mask AP 70 metrics.

The monocular version of PVNet is trained and evaluated on left images only.

Data augmentation: All networks have been trained on real full resolution samples only, augmented with color jittering, random blurring, normalization and random scaling between 80% and 120% of the original size. Data in input to the "parallel" network (which requires a single RGB image) is also subjected to random rotation about the optical axis up to a maximum of 30 degrees.

For all architectures, each training sample has a 50% chance of being subjected to random background augmentation. This procedure has demonstrated to be particularly effective in regularizing models, since training and test sample differ mainly for the background used. The background images used in this step are sampled randomly from a dataset coming from an unrelated domain.

Model	Trained on	Tested on
0	Textures 1,2,3,4,5,6,7,8,9	Texture 0
1	Textures 0,2,3,4,5,6,7,8,9	Texture 1
2	Textures 0,1,3,4,5,6,7,8,9	Texture 2
3	Textures 0,1,2,4,5,6,7,8,9	Texture 3
4	Textures 0,1,2,3,5,6,7,8,9	texture 4
5	Textures 0,1,2,3,4,6,7,8,9	Texture 5
6	Textures 0,1,2,3,4,5,7,8,9	Texture 6
7	Textures 0,1,2,3,4,5,6,8,9	Texture 7
8	Textures 0,1,2,3,4,5,6,7,9	Texture 8
9	Textures 0,1,2,3,4,5,6,7,8	Texture 9

Table 4.1. Train-test split

4.4 Results

In this section, fully detailed result tables are provided, allowing the reader to check the performance of each model for each test set of each object. The results are then combined and reported in a more concise table in section 4.4.5, for easy comparison with the baseline models.

In the following tables, the models used for the evaluation have the following shortened names:

- DF-O: Dense fusion trained on RGB images plus depth maps acquired substituting the transparent objects with opaque counterparts in the framed scene;
- DF-T: Dense fusion trained on RGB images plus depth maps acquired leaving the original transparent objects in the framed scene;
- KP: KeyPose;
- Mono: the original monocular PVNet implementation;
- Pr-18: the "parallel" approach for adapting PVNet to stereo images (uses ResNet18 as its backbone);
- St-18: the "stereo" approach for adapting PVNet to stereo images (uses ResNet18 as its backbone);
- St-34: the "stereo" approach for adapting PVNet to stereo images (uses ResNet34 as its backbone).

Furthermore, the following abbreviations are employed for metrics names:

- P2D: 2D projection metric;
- 5c5d: 5 cm 5 degrees metric;
- APL: mask AP70 for the left image in the stereo pair;

- APR: mask AP70 for the right image in the stereo pair;

- AP: mask AP70 (used in the case of monocular PVNet).

The results are tabulated so that each column represents a different metric, while each row contains the evaluation made on a different test set (equivalent to a different background texture), from 0 to 9. The last row contains the average, for each metric, of all 10 previous rows. The name of the method being evaluated can be found in the table's caption (and is the name of the section containing the table), while the name of the object to be detected is in the first cell of each table.

All metrics are reported as a percentage, ranging from 0 to 1. The only exception is the MAE metric which is measured in meters.

Failed test samples: In some particular cases, which will be analyzed after the results, it may happen that a model cannot correctly detect the object in some input image pairs. Such cases are recognizable by a MAE metric greater than one meter, which is about ten times greater than the average diameter of the objects in the TOD dataset. Since the MAE metric does not have an upper limit, counting such samples in the test metrics computation would lead to results which are greatly skewed by a small number of outliers. Instead, test sets from which image pairs have been excluded, will be marked by an asterisk (*) and the number of failed detections will be reported under the corresponding table.

4.4.1 Stereo34 network

ball_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.965	0.783	0.943	0.011	0.667	0.898	0.994	0.994
T1	0.943	0.770	0.918	0.017	0.421	0.888	0.978	0.978
T2	0.978	0.747	0.959	0.007	0.491	0.905	0.997	0.981
T3	0.972	0.828	0.972	0.012	0.577	0.905	0.994	0.991
T4	0.928	0.831	0.966	0.007	0.531	0.908	0.988	0.991
T5	0.968	0.855	0.962	0.007	0.416	0.914	0.997	1.000
T6	0.975	0.828	0.963	0.006	0.531	0.917	1.000	1.000
T7	0.959	0.643	0.956	0.009	0.589	0.893	1.000	1.000
T8	0.978	0.669	0.959	0.008	0.591	0.897	0.994	1.000
T9	0.988	0.797	0.959	0.007	0.528	0.909	1.000	1.000
Average	0.966	0.775	0.956	0.009	0.534	0.903	0.994	0.993

Table 4.2. Stereo34 network results on ball_0 object (Test set)

heart_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.938	0.581	0.909	0.013	0.319	0.867	0.997	1.000
T1	0.866	0.791	0.881	0.012	0.209	0.873	0.997	0.997
T2	0.819	0.472	0.775	0.015	0.147	0.831	1.000	0.994
T3	0.925	0.688	0.896	0.013	0.192	0.860	1.000	0.992
T4	0.628	0.209	0.428	0.054	0.103	0.630	0.838	0.694
T5	0.622	0.444	0.653	0.020	0.369	0.789	1.000	1.000
T6	0.850	0.331	0.669	0.018	0.250	0.800	1.000	0.953
T7	0.963	0.662	0.775	0.012	0.438	0.864	1.000	1.000
T8	0.429	0.175	0.450	0.086	0.000	0.632	0.738	0.492
T9	0.700	0.578	0.778	0.014	0.188	0.842	0.994	0.988
Average	0.774	0.493	0.721	0.026	0.221	0.799	0.956	0.911

Table 4.3. Stereo34 network results on heart_0 object (Test set)

tree_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.633	0.783	0.821	0.015	0.542	0.838	0.996	1.000
T1	0.796	0.629	0.762	0.015	0.279	0.838	0.975	0.971
T2	0.347	0.301	0.410	0.048	0.130	0.640	0.845	0.611
T3	0.713	0.775	0.887	0.012	0.409	0.860	0.997	0.991
T4*	0.262	0.185	0.275	0.103	0.120	0.516	0.489	0.343
T5	0.597	0.700	0.781	0.018	0.253	0.815	0.981	0.969
T6	0.138	0.192	0.296	0.052	0.021	0.637	0.821	0.637
T7	0.400	0.312	0.362	0.056	0.075	0.614	0.700	0.450
T8*	0.159	0.142	0.189	0.193	0.077	0.367	0.348	0.176
T9	0.863	0.896	0.942	0.010	0.483	0.885	0.988	0.988
Average	0.491	0.491	0.573	0.052	0.239	0.701	0.814	0.714

Table 4.4. Stereo34 network results on tree_0 object (Test set)

Failed test samples: Texture 4 had 7/240 failed samples (2.9%), while texture 8 had 6/240 (2.5%).

bottle_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.871	0.818	0.884	0.011	0.313	0.876	1.000	1.000
T1	0.919	0.812	0.903	0.009	0.297	0.888	1.000	1.000
T2	0.938	0.503	0.747	0.017	0.247	0.816	1.000	0.953
T3	0.831	0.721	0.846	0.016	0.323	0.848	0.966	0.959
T4*	0.634	0.215	0.426	0.065	0.098	0.603	0.804	0.517
T5	0.875	0.803	0.903	0.010	0.300	0.881	1.000	1.000
T6	0.903	0.728	0.906	0.011	0.325	0.874	0.997	1.000
T7*	0.851	0.413	0.579	0.030	0.187	0.726	0.932	0.787
T8	0.616	0.272	0.497	0.047	0.125	0.680	0.825	0.684
T9	0.881	0.697	0.900	0.012	0.269	0.870	1.000	1.000
Average	0.832	0.598	0.759	0.023	0.248	0.806	0.952	0.890

Table 4.5. Stereo34 network results on bottle_0 object (Test set)

Failed test samples: Texture 4 had 3/320 failed samples (0.9%), while texture 7 had 5/240 (2.1%).

bottle_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.997	0.997	0.997	0.006	0.988	0.925	0.981	0.947
T1	0.981	0.991	0.991	0.006	0.906	0.921	0.978	0.953
T2	0.981	0.897	0.916	0.009	0.825	0.889	1.000	0.906
T3	0.978	0.956	0.969	0.007	0.912	0.908	0.984	0.878
T4	0.856	0.768	0.812	0.020	0.649	0.832	0.928	0.737
T5	0.984	0.988	0.991	0.006	0.931	0.916	0.994	0.959
T6	0.953	0.963	0.984	0.010	0.897	0.886	0.991	0.941
T7	0.938	0.971	0.983	0.007	0.858	0.913	0.983	0.958
T8	0.956	0.863	0.891	0.012	0.688	0.867	0.956	0.831
T9	0.916	0.988	0.994	0.007	0.881	0.911	0.966	0.903
Average	0.954	0.938	0.953	0.009	0.854	0.897	0.976	0.901

Table 4.6. Stereo34 network results on bottle_1 object (Test set)

bottle_2	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.902	0.830	0.840	0.012	0.585	0.866	0.917	0.412
T1	0.960	0.870	0.875	0.012	0.695	0.868	0.953	0.405
T2	0.858	0.752	0.772	0.017	0.665	0.824	0.922	0.362
T3	0.940	0.910	0.922	0.011	0.762	0.878	0.922	0.360
T4	0.781	0.799	0.812	0.017	0.661	0.827	0.887	0.329
T5	0.953	0.909	0.925	0.011	0.872	0.875	0.922	0.391
T6	0.790	0.787	0.790	0.019	0.672	0.818	0.905	0.465
T7	0.870	0.800	0.810	0.015	0.615	0.837	0.922	0.312
T8	0.599	0.484	0.491	0.027	0.546	0.733	0.887	0.316
T9	0.845	0.805	0.815	0.014	0.540	0.847	0.927	0.405
Average	0.850	0.795	0.805	0.015	0.661	0.837	0.917	0.376

Table 4.7. Stereo34 network results on bottle_2 object (Test set)

cup_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.562	0.416	0.594	0.029	0.097	0.733	0.988	0.884
T1	0.778	0.512	0.647	0.022	0.153	0.771	1.000	0.897
T2	0.584	0.362	0.500	0.035	0.119	0.720	0.950	0.725
T3	0.713	0.550	0.692	0.020	0.121	0.792	1.000	0.812
T4	0.331	0.191	0.274	0.117	0.051	0.448	0.592	0.268
T5	0.753	0.475	0.641	0.025	0.178	0.767	0.994	0.859
T6	0.713	0.550	0.700	0.019	0.175	0.802	0.988	0.887
T7*	0.266	0.119	0.176	0.156	0.045	0.390	0.494	0.163
T8*	0.303	0.143	0.245	0.100	0.070	0.470	0.624	0.293
T9	0.741	0.522	0.678	0.020	0.150	0.799	0.997	0.916
Average	0.574	0.384	0.515	0.054	0.116	0.669	0.863	0.671

Table 4.8. Stereo34 network results on cup_0 object (Test set)

Failed test samples: Texture 7 had 8/320 (2.5%), while texture 8 had 6/320 (1.9%)

cup_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.925	0.928	0.947	0.010	0.344	0.886	0.991	0.928
T1	0.903	0.869	0.909	0.012	0.316	0.866	0.988	0.875
T2	0.828	0.681	0.747	0.017	0.241	0.814	0.975	0.791
T3	0.881	0.784	0.825	0.015	0.341	0.838	0.975	0.713
T4	0.420	0.458	0.525	0.051	0.063	0.669	0.803	0.433
T5	0.809	0.613	0.666	0.019	0.219	0.792	0.981	0.700
T6	0.844	0.684	0.791	0.017	0.322	0.822	0.975	0.859
T7	0.834	0.731	0.812	0.015	0.269	0.839	0.975	0.791
T8	0.503	0.403	0.459	0.073	0.153	0.638	0.806	0.491
T9	0.784	0.637	0.706	0.019	0.228	0.794	0.978	0.812
Average	0.773	0.679	0.739	0.025	0.249	0.796	0.945	0.739

Table 4.9. Stereo34 network results on cup_1 object (Test set)

mug_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.292	0.367	0.546	0.020	0.075	0.783	1.000	0.996
T1	0.156	0.166	0.291	0.033	0.009	0.678	1.000	0.997
T2	0.356	0.269	0.372	0.041	0.169	0.654	0.997	0.978
T3	0.419	0.566	0.703	0.020	0.222	0.795	1.000	0.984
T4	0.331	0.259	0.362	0.038	0.078	0.646	0.981	0.825
T5	0.464	0.608	0.746	0.019	0.097	0.798	1.000	0.981
T6	0.125	0.325	0.463	0.025	0.000	0.743	1.000	0.950
T7	0.019	0.038	0.069	0.060	0.006	0.510	0.881	0.868
T8	0.380	0.297	0.405	0.053	0.044	0.613	0.930	0.892
T9	0.800	0.863	0.919	0.014	0.359	0.858	1.000	0.997
Average	0.334	0.376	0.488	0.032	0.106	0.708	0.979	0.947

Table 4.10. Stereo34 network results on mug_0 object (Test set)

mug_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.416	0.391	0.516	0.026	0.212	0.729	0.997	0.872
T1	0.203	0.266	0.334	0.032	0.091	0.672	0.981	0.741
T2	0.105	0.105	0.206	0.072	0.029	0.490	0.752	0.311
T3	0.444	0.375	0.519	0.031	0.163	0.693	0.994	0.794
T4*	0.078	0.000	0.000	0.421	0.007	0.059	0.157	0.046
T5	0.519	0.374	0.516	0.034	0.201	0.701	0.956	0.714
T6	0.207	0.080	0.139	0.077	0.072	0.490	0.823	0.456
T7	0.125	0.225	0.325	0.043	0.000	0.583	0.875	0.375
T8*	0.022	0.022	0.057	0.304	0.009	0.181	0.209	0.052
T9	0.230	0.186	0.286	0.066	0.047	0.553	0.840	0.469
Average	0.235	0.202	0.290	0.111	0.083	0.515	0.758	0.483

Table 4.11. Stereo34 network results on mug_1 object (Test set)

Failed test samples: Texture 4 had 7/160 (4.4%), while texture 8 had 10/240 (4.1%).

mug_2	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.909	0.781	0.847	0.017	0.725	0.828	0.997	0.766
T1	0.891	0.713	0.762	0.016	0.684	0.832	0.997	0.703
T2	0.484	0.319	0.438	0.031	0.272	0.711	0.984	0.419
T3	0.853	0.753	0.841	0.015	0.556	0.841	1.000	0.619
T4	0.274	0.186	0.236	0.067	0.101	0.466	0.906	0.321
T5	0.678	0.622	0.725	0.017	0.328	0.813	0.994	0.738
T6	0.756	0.628	0.731	0.025	0.419	0.774	0.994	0.731
T7	0.494	0.412	0.516	0.030	0.178	0.740	0.991	0.591
T8	0.617	0.433	0.529	0.033	0.312	0.696	0.963	0.562
T9	0.844	0.456	0.559	0.024	0.491	0.750	1.000	0.584
Average	0.680	0.530	0.618	0.027	0.407	0.745	0.982	0.603

Table 4.12. Stereo34 network results on mug_2 object (Test set)

mug_3	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.875	0.863	0.875	0.017	0.662	0.833	0.994	0.372
T1	0.869	0.734	0.759	0.026	0.613	0.771	0.997	0.509
T2	0.512	0.425	0.456	0.031	0.309	0.699	0.984	0.375
T3	0.828	0.850	0.863	0.031	0.762	0.778	0.984	0.247
T4	0.192	0.229	0.250	0.065	0.050	0.515	0.554	0.175
T5	0.738	0.694	0.709	0.025	0.416	0.784	0.997	0.225
T6	0.931	0.928	0.934	0.014	0.778	0.870	0.991	0.447
T7	0.831	0.753	0.794	0.020	0.506	0.823	0.997	0.400
T8	0.338	0.438	0.469	0.031	0.253	0.705	0.953	0.166
T9	0.809	0.547	0.606	0.025	0.441	0.764	0.969	0.231
Average	0.692	0.646	0.672	0.028	0.479	0.754	0.942	0.315

Table 4.13. Stereo34 network results on mug_3 object (Test set)

mug_4	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.884	0.925	0.919	0.013	0.619	0.866	0.966	0.122
T1	0.829	0.900	0.892	0.013	0.575	0.856	0.963	0.087
T2	0.619	0.550	0.544	0.027	0.200	0.732	0.969	0.134
T3	0.688	0.872	0.841	0.018	0.397	0.834	0.981	0.059
T4	0.588	0.696	0.675	0.022	0.512	0.774	0.975	0.037
T5	0.753	0.734	0.703	0.021	0.491	0.798	0.963	0.166
T6	0.750	0.859	0.856	0.017	0.484	0.834	0.966	0.131
T7	0.637	0.756	0.741	0.018	0.481	0.806	0.963	0.069
T8	0.429	0.614	0.592	0.026	0.141	0.748	0.959	0.075
T9	0.487	0.650	0.637	0.024	0.344	0.754	0.981	0.087
Average	0.666	0.756	0.740	0.020	0.424	0.800	0.968	0.097

Table 4.14. Stereo34 network results on mug_4 object (Test set)

mug_5	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.755	0.824	0.809	0.027	0.721	0.777	0.937	0.113
T1	0.854	0.821	0.817	0.024	0.758	0.763	0.971	0.125
T2	0.812	0.842	0.829	0.017	0.683	0.835	0.917	0.062
T3	0.705	0.850	0.837	0.021	0.665	0.795	0.925	0.066
T4	0.580	0.592	0.589	0.034	0.552	0.720	0.909	0.066
T5	0.812	0.778	0.762	0.028	0.703	0.764	0.944	0.053
T6	0.844	0.847	0.828	0.018	0.637	0.826	0.959	0.044
T7	0.622	0.684	0.666	0.035	0.350	0.696	0.959	0.041
T8	0.562	0.641	0.625	0.029	0.369	0.740	0.922	0.075
T9	0.834	0.744	0.728	0.023	0.747	0.777	0.969	0.081
Average	0.738	0.762	0.749	0.026	0.619	0.769	0.941	0.073

Table 4.15. Stereo34 network results on mug_5 object (Test set)

mug_6	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.783	0.775	0.733	0.019	0.475	0.818	0.958	0.046
T1	0.797	0.759	0.728	0.021	0.725	0.790	0.969	0.056
T2	0.891	0.934	0.909	0.012	0.706	0.875	0.941	0.084
T3	0.821	0.793	0.737	0.020	0.646	0.806	0.934	0.016
T4	0.609	0.694	0.659	0.037	0.278	0.729	0.878	0.041
T5	0.769	0.869	0.853	0.019	0.537	0.816	0.956	0.066
T6	0.716	0.859	0.831	0.020	0.431	0.831	0.978	0.081
T7	0.845	0.795	0.770	0.021	0.632	0.800	0.958	0.063
T8	0.358	0.592	0.546	0.028	0.333	0.718	0.925	0.013
T9	0.803	0.784	0.750	0.020	0.528	0.816	0.938	0.044
Average	0.739	0.785	0.752	0.022	0.529	0.800	0.944	0.051

Table 4.16. Stereo34 network results on mug_6 object (Test set)

4.4.2 Stereo18 network

In the following tables it is possible to see that in many cases the MaskAP70 metric for the right images is very low. This does not mean that the network made disastrous predictions on the right images, but that the prediction variance on the right images was proportionally much higher than the variance on the left images and so the system decided to discard them to avoid interference from data which was too noisy. Results are not reported for all objects since this method had a failed detection rate above 10% of the total number of test samples in some datasets and because, as can be seen from the tables below, in many cases it just reverts to monocular PVNet.

heart_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.922	0.841	0.928	0.009	0.303	0.892	1.000	1.000
T1	0.881	0.778	0.900	0.059	0.237	0.881	0.997	0.997
T2	0.750	0.438	0.775	0.017	0.075	0.831	0.978	0.953
T3	0.600	0.429	0.608	0.023	0.075	0.774	0.996	0.975
T4	0.619	0.153	0.347	0.123	0.109	0.591	0.869	0.506
T5	0.550	0.544	0.606	0.018	0.391	0.805	1.000	1.000
T6	0.834	0.537	0.881	0.013	0.184	0.854	1.000	1.000
T7	0.938	0.787	0.912	0.010	0.550	0.877	0.975	0.988
T8	0.588	0.217	0.433	0.059	0.000	0.659	0.938	0.675
T9	0.622	0.412	0.625	0.020	0.134	0.782	0.972	0.975
Average	0.730	0.514	0.702	0.035	0.206	0.795	0.972	0.907

Table 4.17. Stereo18 network results on heart_0 object (Test set)

tree_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.633	0.867	0.925	0.012	0.562	0.866	0.996	0.988
T1	0.750	0.762	0.917	0.015	0.171	0.843	0.992	0.963
T2	0.662	0.529	0.683	0.027	0.263	0.747	0.908	0.850
T3	0.625	0.675	0.853	0.013	0.269	0.849	1.000	0.975
T4	0.534	0.504	0.592	0.043	0.277	0.716	0.706	0.681
T5	0.656	0.694	0.806	0.017	0.322	0.818	0.988	0.972
T6	0.318	0.251	0.397	0.049	0.213	0.634	0.665	0.724
T7	0.215	0.165	0.253	0.084	0.013	0.506	0.519	0.291
T8	0.143	0.160	0.225	0.148	0.048	0.417	0.403	0.277
T9	0.838	0.775	0.900	0.014	0.629	0.852	0.975	0.979
Average	0.537	0.538	0.655	0.042	0.277	0.725	0.815	0.770

Table 4.18. Stereo18 network results on tree_0 object (Test set)

Failed test samples: Texture 8 had 8/240 (3.3%).

mug_3	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.897	0.834	0.847	0.017	0.759	0.839	0.991	0.366
T1	0.887	0.828	0.844	0.022	0.569	0.805	0.994	0.506
T2	0.528	0.362	0.409	0.036	0.284	0.648	0.988	0.362
T3	0.803	0.850	0.866	0.020	0.622	0.818	0.991	0.138
T4	0.219	0.211	0.232	0.091	0.059	0.451	0.591	0.190
T5	0.637	0.619	0.631	0.030	0.416	0.753	0.991	0.219
T6	0.891	0.912	0.931	0.013	0.738	0.872	0.997	0.391
T7	0.778	0.728	0.747	0.020	0.453	0.806	0.994	0.316
T8	0.336	0.368	0.409	0.042	0.217	0.637	0.987	0.088
T9	0.725	0.569	0.622	0.023	0.366	0.776	0.963	0.212
Average	0.670	0.628	0.654	0.031	0.448	0.740	0.948	0.279

Table 4.19. Stereo18 network results on mug_3 object (Test set)

mug_4	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.838	0.922	0.909	0.012	0.672	0.867	0.966	0.119
T1	0.775	0.850	0.846	0.014	0.558	0.854	0.975	0.133
T2	0.591	0.597	0.588	0.028	0.191	0.726	0.975	0.106
T3	0.600	0.656	0.637	0.023	0.331	0.772	0.972	0.056
T4	0.438	0.454	0.417	0.050	0.354	0.624	0.912	0.021
T5	0.787	0.900	0.881	0.014	0.481	0.854	0.963	0.134
T6	0.637	0.762	0.756	0.020	0.369	0.804	0.959	0.113
T7	0.656	0.659	0.647	0.023	0.559	0.772	0.988	0.050
T8	0.514	0.665	0.655	0.031	0.135	0.736	0.972	0.047
T9	0.459	0.512	0.481	0.028	0.281	0.721	0.972	0.066
Average	0.630	0.698	0.682	0.024	0.393	0.773	0.965	0.084

Table 4.20. Stereo18 network results on mug_4 object (Test set)

mug_5	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.812	0.859	0.850	0.022	0.743	0.810	0.959	0.100
T1	0.846	0.875	0.875	0.020	0.783	0.808	0.963	0.108
T2	0.800	0.787	0.779	0.020	0.708	0.815	0.958	0.021
T3	0.724	0.793	0.787	0.024	0.586	0.781	0.928	0.050
T4	0.589	0.661	0.655	0.028	0.564	0.745	0.937	0.050
T5	0.853	0.800	0.787	0.023	0.775	0.791	0.928	0.075
T6	0.794	0.881	0.878	0.016	0.637	0.841	0.963	0.047
T7	0.697	0.713	0.709	0.030	0.362	0.740	0.972	0.041
T8	0.628	0.691	0.678	0.024	0.516	0.765	0.944	0.034
T9	0.831	0.800	0.794	0.017	0.741	0.826	0.959	0.075
Average	0.757	0.786	0.779	0.023	0.642	0.792	0.951	0.060

Table 4.21. Stereo18 network results on mug_5 object (Test set)

mug_6	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL	APR
T0	0.821	0.858	0.808	0.016	0.517	0.831	0.938	0.075
T1	0.784	0.863	0.838	0.017	0.706	0.830	0.906	0.066
T2	0.831	0.869	0.841	0.015	0.606	0.851	0.972	0.094
T3	0.846	0.815	0.784	0.018	0.661	0.816	0.972	0.031
T4	0.637	0.681	0.644	0.029	0.375	0.753	0.903	0.072
T5	0.822	0.894	0.881	0.016	0.603	0.839	0.956	0.041
T6	0.781	0.787	0.759	0.018	0.469	0.818	0.972	0.078
T7	0.774	0.791	0.778	0.021	0.611	0.791	0.967	0.063
T8	0.379	0.625	0.596	0.028	0.204	0.739	0.863	0.004
T9	0.838	0.847	0.812	0.017	0.613	0.824	0.959	0.041
Average	0.751	0.803	0.774	0.020	0.537	0.809	0.941	0.056

Table 4.22. Stereo18 network results on mug_6 object (Test set)

4.4.3 Parallel network

ball_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.956	0.412	0.909	0.015	0.167	0.859	0.994
T1	0.972	0.476	0.909	0.016	0.197	0.868	0.991
T2	0.984	0.559	0.928	0.009	0.125	0.887	1.000
T3	0.962	0.542	0.953	0.012	0.191	0.881	0.994
T4	0.922	0.647	0.956	0.009	0.181	0.892	0.991
T5	0.956	0.384	0.934	0.011	0.209	0.867	1.000
T6	0.919	0.469	0.881	0.011	0.178	0.873	1.000
T7	0.950	0.470	0.937	0.011	0.176	0.877	1.000
T8	0.966	0.466	0.872	0.012	0.181	0.862	0.997
T9	0.941	0.450	0.884	0.011	0.131	0.870	0.997
Average	0.953	0.488	0.916	0.012	0.174	0.874	0.996

Table 4.23. Parallel network results on ball_0 object (Test set)

heart_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.953	0.666	0.950	0.010	0.319	0.882	1.000
T1	0.894	0.647	0.906	0.011	0.225	0.870	0.997
T2	0.781	0.541	0.891	0.013	0.106	0.854	0.988
T3	0.683	0.258	0.713	0.019	0.150	0.795	1.000
T4	0.759	0.491	0.838	0.014	0.041	0.844	0.997
T5	0.666	0.269	0.688	0.019	0.228	0.796	1.000
T6	0.863	0.347	0.806	0.015	0.116	0.834	1.000
T7	0.950	0.713	0.950	0.010	0.613	0.881	1.000
T8	0.746	0.438	0.792	0.032	0.037	0.810	0.938
T9	0.444	0.284	0.562	0.023	0.097	0.753	1.000
Average	0.774	0.465	0.809	0.017	0.193	0.832	0.992

Table 4.24. Parallel network results on heart_0 object (Test set)

tree_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.633	0.792	0.925	0.015	0.504	0.842	0.996
T1	0.742	0.758	0.921	0.012	0.279	0.861	0.983
T2	0.754	0.825	0.938	0.010	0.292	0.878	0.988
T3	0.525	0.878	0.972	0.010	0.159	0.880	0.981
T4	0.975	0.912	0.996	0.008	0.504	0.898	0.996
T5	0.659	0.762	0.812	0.018	0.369	0.806	0.994
T6	0.467	0.379	0.562	0.029	0.279	0.714	0.750
T7	0.762	0.738	0.863	0.024	0.250	0.848	0.950
T8	0.490	0.695	0.833	0.014	0.142	0.840	0.874
T9	0.792	0.825	0.933	0.011	0.325	0.868	0.988
Average	0.680	0.756	0.875	0.015	0.310	0.844	0.950

Table 4.25. Parallel network results on tree_0 object (Test set)

bottle_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.812	0.592	0.834	0.016	0.094	0.825	1.000
T1	0.866	0.562	0.838	0.016	0.156	0.825	1.000
T2	0.872	0.512	0.828	0.017	0.138	0.828	1.000
T3	0.644	0.441	0.787	0.018	0.094	0.819	0.759
T4	0.722	0.331	0.722	0.020	0.081	0.798	0.928
T5	0.838	0.662	0.825	0.016	0.094	0.839	1.000
T6	0.822	0.528	0.825	0.017	0.109	0.822	0.997
T7	0.679	0.425	0.696	0.026	0.075	0.768	0.804
T8	0.629	0.434	0.664	0.026	0.079	0.770	0.828
T9	0.822	0.562	0.819	0.015	0.091	0.838	0.997
Average	0.770	0.505	0.784	0.019	0.101	0.813	0.931

Table 4.26. Parallel network results on bottle_0 object (Test set)

bottle_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.984	0.916	0.963	0.010	0.297	0.879	0.991
T1	0.963	0.872	0.934	0.010	0.334	0.880	0.975
T2	0.963	0.872	0.938	0.011	0.475	0.867	0.991
T3	0.953	0.825	0.938	0.012	0.381	0.861	0.994
T4	0.938	0.822	0.891	0.012	0.434	0.859	0.991
T5	0.950	0.869	0.925	0.010	0.419	0.880	0.994
T6	0.978	0.659	0.778	0.015	0.416	0.835	0.997
T7	0.950	0.804	0.846	0.013	0.358	0.850	0.992
T8	0.959	0.831	0.903	0.011	0.497	0.873	0.984
T9	0.853	0.828	0.922	0.011	0.356	0.869	0.994
Average	0.949	0.830	0.904	0.012	0.397	0.865	0.990

Table 4.27. Parallel network results on bottle_1 object (Test set)

bottle_2	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.887	0.877	0.892	0.013	0.385	0.854	0.938
T1	0.927	0.785	0.820	0.014	0.445	0.845	0.943
T2	0.838	0.825	0.840	0.016	0.458	0.833	0.958
T3	0.635	0.695	0.713	0.017	0.400	0.813	0.902
T4	0.705	0.807	0.820	0.015	0.380	0.837	0.858
T5	0.938	0.875	0.900	0.012	0.591	0.861	0.938
T6	0.723	0.770	0.797	0.019	0.422	0.818	0.917
T7	0.792	0.812	0.830	0.016	0.235	0.825	0.915
T8	0.527	0.557	0.580	0.026	0.278	0.757	0.848
T9	0.700	0.672	0.703	0.020	0.362	0.797	0.940
Average	0.767	0.768	0.790	0.017	0.396	0.824	0.915

Table 4.28. Parallel network results on bottle_2 object (Test set)

cup_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.559	0.456	0.653	0.023	0.072	0.765	0.991
T1	0.716	0.519	0.678	0.022	0.078	0.775	0.997
T2	0.625	0.428	0.619	0.023	0.059	0.761	0.991
T3*	0.051	0.141	0.269	0.094	0.043	0.557	0.237
T4	0.352	0.333	0.459	0.057	0.031	0.622	0.706
T5	0.694	0.569	0.747	0.018	0.072	0.807	0.997
T6	0.692	0.417	0.546	0.025	0.092	0.746	0.996
T7*	0.260	0.234	0.330	0.088	0.003	0.533	0.606
T8	0.263	0.272	0.408	0.068	0.032	0.596	0.659
T9	0.497	0.503	0.647	0.024	0.075	0.779	0.850
Average	0.471	0.387	0.536	0.044	0.056	0.694	0.803

Table 4.29. Parallel network results on cup_0 object (Test set)

Failed test samples: 6/240 (2.5%) for texture 3 and 8/320 (2.5%) for texture 7.

cup_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.734	0.697	0.728	0.019	0.144	0.796	0.975
T1	0.838	0.681	0.762	0.017	0.153	0.819	0.972
T2	0.662	0.641	0.688	0.023	0.109	0.767	0.975
T3	0.634	0.647	0.713	0.022	0.084	0.779	0.953
T4	0.317	0.433	0.492	0.058	0.071	0.666	0.625
T5	0.806	0.731	0.791	0.015	0.159	0.837	0.981
T6	0.725	0.691	0.744	0.024	0.113	0.777	0.963
T7	0.744	0.650	0.728	0.020	0.103	0.789	0.991
T8	0.558	0.561	0.611	0.033	0.069	0.732	0.812
T9	0.672	0.653	0.725	0.024	0.125	0.759	0.953
Average	0.669	0.639	0.698	0.025	0.113	0.772	0.920

Table 4.30. Parallel network results on cup_1 object (Test set)

mug_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.204	0.483	0.650	0.020	0.071	0.787	0.996
T1	0.472	0.531	0.697	0.019	0.188	0.796	0.997
T2	0.512	0.472	0.581	0.027	0.212	0.730	0.997
T3	0.084	0.287	0.431	0.031	0.009	0.712	0.672
T4	0.306	0.331	0.466	0.029	0.037	0.706	0.994
T5	0.803	0.768	0.868	0.015	0.345	0.843	0.994
T6	0.438	0.412	0.725	0.017	0.025	0.813	1.000
T7	0.006	0.087	0.144	0.044	0.000	0.559	0.988
T8	0.219	0.269	0.350	0.037	0.113	0.639	0.969
T9	0.850	0.562	0.884	0.016	0.409	0.827	1.000
Average	0.389	0.420	0.580	0.025	0.141	0.741	0.961

Table 4.31. Parallel network results on mug_0 object (Test set)

mug_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.278	0.353	0.484	0.027	0.075	0.726	0.991
T1	0.228	0.269	0.409	0.036	0.062	0.658	0.956
T2	0.467	0.404	0.529	0.030	0.129	0.700	0.958
T3	0.463	0.481	0.631	0.026	0.044	0.744	0.963
T4	0.344	0.344	0.475	0.032	0.013	0.704	0.694
T5	0.533	0.502	0.658	0.023	0.100	0.759	0.994
T6	0.269	0.298	0.395	0.052	0.151	0.622	0.771
T7	0.175	0.338	0.412	0.041	0.000	0.661	0.975
T8	0.093	0.131	0.186	0.126	0.013	0.381	0.338
T9	0.484	0.478	0.644	0.024	0.216	0.751	0.950
Average	0.333	0.360	0.482	0.042	0.080	0.671	0.859

Table 4.32. Parallel network results on mug_1 object (Test set)

mug_2	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.844	0.759	0.894	0.015	0.662	0.833	0.988
T1	0.866	0.688	0.778	0.018	0.556	0.812	0.991
T2	0.541	0.569	0.647	0.025	0.369	0.741	0.991
T3	0.922	0.863	0.931	0.013	0.644	0.855	0.997
T4	0.388	0.472	0.609	0.025	0.163	0.751	0.944
T5	0.719	0.609	0.756	0.017	0.475	0.821	0.997
T6	0.706	0.581	0.659	0.026	0.362	0.749	0.991
T7	0.641	0.522	0.641	0.025	0.272	0.758	0.994
T8	0.792	0.742	0.850	0.017	0.250	0.826	1.000
T9	0.881	0.625	0.772	0.018	0.753	0.814	1.000
Average	0.730	0.643	0.754	0.020	0.451	0.796	0.989

Table 4.33. Parallel network results on mug_2 object (Test set)

mug_3	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.709	0.794	0.853	0.020	0.319	0.811	0.994
T1	0.822	0.716	0.759	0.032	0.503	0.729	0.994
T2	0.450	0.375	0.431	0.037	0.175	0.650	0.966
T3	0.331	0.662	0.722	0.022	0.259	0.781	0.778
T4	0.308	0.425	0.479	0.039	0.054	0.673	0.900
T5	0.725	0.731	0.812	0.019	0.469	0.815	0.997
T6	0.938	0.887	0.919	0.014	0.844	0.853	0.991
T7	0.684	0.713	0.756	0.017	0.350	0.821	0.988
T8	0.438	0.588	0.616	0.029	0.281	0.722	0.978
T9	0.697	0.697	0.753	0.019	0.512	0.803	0.984
Average	0.610	0.659	0.710	0.025	0.377	0.766	0.957

Table 4.34. Parallel network results on mug_3 object (Test set)

mug_4	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.644	0.916	0.906	0.015	0.441	0.837	0.975
T1	0.775	0.875	0.863	0.016	0.675	0.829	0.971
T2	0.541	0.666	0.647	0.025	0.163	0.745	0.972
T3	0.309	0.744	0.719	0.020	0.138	0.786	0.950
T4	0.454	0.817	0.796	0.022	0.283	0.794	0.975
T5	0.616	0.816	0.803	0.018	0.372	0.814	0.984
T6	0.428	0.731	0.713	0.024	0.219	0.753	0.953
T7	0.578	0.666	0.631	0.026	0.566	0.749	0.978
T8	0.489	0.702	0.693	0.026	0.138	0.754	0.978
T9	0.319	0.550	0.528	0.026	0.212	0.733	0.978
Average	0.515	0.748	0.730	0.022	0.321	0.780	0.971

Table 4.35. Parallel network results on mug_4 object (Test set)

mug_5	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.705	0.815	0.809	0.022	0.536	0.794	0.956
T1	0.879	0.829	0.825	0.022	0.833	0.781	0.979
T2	0.671	0.875	0.871	0.021	0.492	0.812	0.954
T3	0.734	0.884	0.862	0.017	0.599	0.815	0.953
T4	0.592	0.787	0.777	0.020	0.574	0.806	0.918
T5	0.869	0.894	0.891	0.020	0.816	0.802	0.978
T6	0.834	0.900	0.894	0.016	0.487	0.841	0.959
T7	0.738	0.762	0.744	0.027	0.459	0.757	0.963
T8	0.800	0.887	0.872	0.016	0.622	0.833	0.963
T9	0.819	0.794	0.775	0.021	0.637	0.786	0.978
Average	0.764	0.843	0.832	0.020	0.606	0.803	0.960

Table 4.36. Parallel network results on mug_5 object (Test set)

mug_6	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	AP
T0	0.779	0.887	0.838	0.014	0.408	0.849	0.958
T1	0.850	0.947	0.925	0.012	0.787	0.864	0.938
T2	0.672	0.834	0.787	0.016	0.519	0.828	0.981
T3	0.141	0.292	0.260	0.070	0.179	0.439	0.392
T4	0.603	0.925	0.900	0.014	0.500	0.847	0.938
T5	0.863	0.931	0.916	0.014	0.675	0.849	0.953
T6	0.747	0.909	0.884	0.013	0.472	0.856	0.950
T7	0.803	0.858	0.816	0.017	0.531	0.820	0.954
T8	0.375	0.800	0.779	0.017	0.113	0.823	0.921
T9	0.775	0.881	0.841	0.016	0.556	0.829	0.963
Average	0.661	0.826	0.795	0.020	0.474	0.800	0.895

Table 4.37. Parallel network results on mug_6 object (Test set)

4.4.4 Mono network (original PVNet)

ball_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.925	0.789	0.984	0.013	0.311	0.912	0.994
T1	0.940	0.696	0.956	0.018	0.216	0.899	0.991
T2	0.959	0.616	0.897	0.010	0.228	0.881	1.000
T3	0.918	0.505	0.915	0.072	0.241	0.851	0.994
T4	0.894	0.578	0.931	0.010	0.306	0.885	0.991
T5	0.950	0.653	0.931	0.014	0.225	0.891	1.000
T6	0.900	0.237	0.588	0.083	0.219	0.757	1.000
T7	0.953	0.627	0.897	0.009	0.197	0.886	1.000
T8	0.953	0.391	0.766	0.043	0.303	0.824	0.997
T9	0.928	0.697	0.959	0.007	0.247	0.905	0.997
Average	0.932	0.579	0.882	0.028	0.249	0.869	0.996

Table 4.38. Mono network results on ball_0 object (Test set)

heart_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.994	0.434	0.694	0.213	0.325	0.742	1.000
T1	0.947	0.438	0.706	0.100	0.256	0.762	0.997
T2	0.822	0.150	0.325	0.271	0.078	0.552	0.988
T3	0.738	0.192	0.408	0.197	0.208	0.641	1.000
T4	0.725	0.100	0.256	0.295	0.031	0.441	0.997
T5	0.613	0.316	0.466	0.242	0.256	0.607	1.000
T6	0.900	0.206	0.475	0.115	0.119	0.686	1.000
T7	1.000	0.312	0.775	0.014	0.675	0.842	1.000
T8	0.642	0.104	0.242	0.504	0.004	0.439	0.938
T9	0.466	0.031	0.094	0.433	0.081	0.354	1.000
Average	0.784	0.228	0.444	0.238	0.203	0.607	0.992

Table 4.39. Mono network results on heart_0 object (Test set)

tree_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.738	0.629	0.729	0.031	0.592	0.796	0.996
T1	0.796	0.379	0.487	0.091	0.229	0.721	0.983
T2	0.829	0.521	0.604	0.141	0.254	0.690	0.988
T3	0.547	0.006	0.019	0.557	0.044	0.345	0.981
T4	0.967	0.562	0.721	0.020	0.392	0.820	0.996
T5	0.725	0.481	0.550	0.151	0.353	0.679	0.994
T6	0.504	0.125	0.217	0.320	0.204	0.423	0.750
T7	0.850	0.400	0.525	0.095	0.325	0.707	0.950
T8	0.649	0.314	0.410	0.262	0.084	0.593	0.874
T9	0.858	0.463	0.588	0.038	0.237	0.767	0.988
Average	0.746	0.388	0.485	0.171	0.271	0.654	0.950

Table 4.40. Mono network results on tree_0 object (Test set)

bottle_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.404	0.122	0.191	0.636	0.038	0.305	1.000
T1	0.503	0.138	0.278	0.277	0.078	0.442	1.000
T2	0.575	0.222	0.388	0.248	0.100	0.502	1.000
T3	0.266	0.025	0.059	0.681	0.009	0.122	0.759
T4	0.403	0.144	0.322	0.309	0.059	0.435	0.928
T5	0.463	0.087	0.237	0.363	0.050	0.412	1.000
T6	0.500	0.075	0.194	0.350	0.059	0.368	0.997
T7	0.362	0.083	0.138	0.603	0.037	0.273	0.804
T8	0.303	0.141	0.234	0.377	0.041	0.392	0.828
T9	0.447	0.122	0.209	0.371	0.056	0.413	0.997
Average	0.423	0.116	0.225	0.421	0.053	0.366	0.931

Table 4.41. Mono network results on bottle_0 object (Test set)

bottle_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.909	1.000	1.000	0.005	0.581	0.931	0.991
T1	0.928	1.000	1.000	0.004	0.525	0.938	0.975
T2	0.875	0.953	0.966	0.007	0.494	0.911	0.991
T3	0.784	0.950	0.963	0.029	0.394	0.902	0.994
T4	0.816	0.972	0.988	0.011	0.416	0.912	0.991
T5	0.947	0.984	0.997	0.006	0.606	0.919	0.994
T6	0.947	1.000	1.000	0.005	0.609	0.936	0.997
T7	0.963	0.996	0.996	0.005	0.600	0.933	0.992
T8	0.831	0.941	0.963	0.011	0.306	0.903	0.984
T9	0.719	0.994	0.997	0.006	0.444	0.923	0.994
Average	0.872	0.979	0.987	0.009	0.497	0.921	0.990

Table 4.42. Mono network results on bottle_1 object (Test set)

bottle_2	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.660	0.750	0.767	0.076	0.355	0.799	0.938
T1	0.870	0.875	0.880	0.089	0.540	0.849	0.943
T2	0.787	0.907	0.912	0.066	0.512	0.863	0.958
T3	0.642	0.767	0.777	0.028	0.310	0.841	0.902
T4	0.632	0.657	0.660	0.136	0.305	0.739	0.858
T5	0.925	0.966	0.972	0.008	0.700	0.905	0.938
T6	0.620	0.825	0.833	0.061	0.335	0.823	0.917
T7	0.552	0.642	0.650	0.079	0.253	0.751	0.915
T8	0.425	0.660	0.670	0.253	0.275	0.701	0.848
T9	0.595	0.650	0.665	0.074	0.367	0.762	0.940
Average	0.671	0.770	0.779	0.087	0.395	0.803	0.915

Table 4.43. Mono network results on bottle_2 object (Test set)

cup_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.197	0.400	0.503	0.343	0.047	0.582	0.991
T1	0.259	0.450	0.569	0.191	0.072	0.633	0.997
T2	0.178	0.347	0.434	0.399	0.019	0.513	0.991
T3	0.008	0.029	0.075	1.236	0.021	0.092	0.237
T4	0.138	0.312	0.356	0.384	0.013	0.443	0.706
T5	0.247	0.403	0.491	0.346	0.047	0.541	0.997
T6	0.246	0.233	0.304	0.337	0.079	0.464	0.996
T7	0.050	0.209	0.241	0.373	0.009	0.377	0.606
T8	0.091	0.191	0.247	0.505	0.019	0.359	0.659
T9	0.206	0.222	0.294	0.590	0.066	0.377	0.850
Average	0.162	0.280	0.351	0.470	0.039	0.438	0.803

Table 4.44. Mono network results on cup_0 object (Test set)

cup_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.450	0.516	0.550	0.106	0.103	0.672	0.975
T1	0.531	0.613	0.688	0.090	0.175	0.741	0.972
T2	0.347	0.384	0.434	0.205	0.109	0.575	0.975
T3	0.259	0.509	0.550	0.205	0.075	0.610	0.953
T4	0.138	0.362	0.404	0.326	0.042	0.503	0.625
T5	0.450	0.619	0.662	0.132	0.138	0.681	0.981
T6	0.356	0.519	0.553	0.114	0.081	0.650	0.963
T7	0.394	0.588	0.628	0.142	0.122	0.675	0.991
T8	0.250	0.438	0.466	0.340	0.084	0.500	0.812
T9	0.338	0.456	0.519	0.132	0.084	0.607	0.953
Average	0.351	0.500	0.545	0.179	0.101	0.621	0.920

Table 4.45. Mono network results on cup_1 object (Test set)

mug_0	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.192	0.129	0.179	0.538	0.037	0.409	0.996
T1	0.441	0.234	0.316	0.137	0.178	0.535	1.000
T2	0.438	0.156	0.231	0.467	0.184	0.396	0.997
T3	0.100	0.066	0.116	0.537	0.019	0.229	0.697
T4	0.309	0.216	0.319	0.487	0.047	0.426	0.991
T5	0.868	0.621	0.727	0.033	0.364	0.819	1.000
T6	0.425	0.338	0.425	0.037	0.037	0.645	1.000
T7	0.087	0.019	0.025	0.356	0.000	0.141	0.981
T8	0.163	0.062	0.087	0.627	0.062	0.222	0.856
T9	0.891	0.484	0.650	0.018	0.459	0.804	1.000
Average	0.391	0.232	0.308	0.324	0.139	0.463	0.952

Table 4.46. Mono network results on mug_0 object (Test set)

mug_1	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.225	0.194	0.275	0.350	0.072	0.501	0.991
T1	0.253	0.044	0.081	0.568	0.019	0.228	0.956
T2	0.512	0.158	0.212	0.451	0.138	0.438	0.958
T3	0.388	0.069	0.094	0.357	0.056	0.371	0.963
T4	0.300	0.181	0.269	0.358	0.000	0.488	0.694
T5	0.517	0.219	0.292	0.275	0.094	0.554	0.994
T6	0.229	0.100	0.133	0.430	0.062	0.339	0.771
T7	0.100	0.150	0.250	0.125	0.000	0.461	0.975
T8	0.087	0.087	0.121	0.671	0.004	0.207	0.338
T9	0.416	0.191	0.278	0.442	0.131	0.508	0.950
Average	0.303	0.139	0.201	0.403	0.058	0.409	0.859

Table 4.47. Mono network results on mug_1 object (Test set)

mug_2	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.884	0.797	0.859	0.021	0.744	0.871	0.988
T1	0.872	0.775	0.841	0.025	0.597	0.842	0.991
T2	0.528	0.312	0.378	0.174	0.309	0.596	0.991
T3	0.938	0.528	0.616	0.044	0.603	0.789	0.997
T4	0.306	0.169	0.216	0.416	0.103	0.420	0.944
T5	0.744	0.591	0.672	0.095	0.434	0.766	0.997
T6	0.681	0.322	0.369	0.209	0.328	0.557	0.991
T7	0.491	0.400	0.463	0.242	0.200	0.590	0.994
T8	0.779	0.458	0.517	0.105	0.275	0.690	1.000
T9	0.878	0.503	0.631	0.088	0.656	0.759	1.000
Average	0.710	0.486	0.556	0.142	0.425	0.688	0.989

Table 4.48. Mono network results on mug_2 object (Test set)

mug_3	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.681	0.394	0.447	0.153	0.328	0.680	0.994
T1	0.881	0.625	0.669	0.111	0.547	0.776	0.994
T2	0.309	0.256	0.281	0.276	0.159	0.449	0.966
T3	0.216	0.081	0.091	0.636	0.128	0.292	0.778
T4	0.229	0.188	0.212	0.392	0.029	0.411	0.900
T5	0.756	0.491	0.522	0.065	0.444	0.735	0.997
T6	0.909	0.503	0.544	0.091	0.719	0.724	0.991
T7	0.491	0.412	0.434	0.221	0.306	0.565	0.988
T8	0.397	0.312	0.344	0.218	0.247	0.557	0.978
T9	0.709	0.388	0.428	0.148	0.422	0.667	0.984
Average	0.558	0.365	0.397	0.231	0.333	0.585	0.957

Table 4.49. Mono network results on mug_3 object (Test set)

mug_4	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.700	0.606	0.578	0.115	0.553	0.732	0.975
T1	0.704	0.646	0.642	0.092	0.588	0.755	0.971
T2	0.400	0.275	0.272	0.306	0.147	0.516	0.972
T3	0.247	0.259	0.250	0.673	0.069	0.386	0.950
T4	0.546	0.512	0.508	0.150	0.258	0.674	0.975
T5	0.475	0.544	0.516	0.104	0.303	0.671	0.984
T6	0.356	0.441	0.428	0.262	0.178	0.564	0.953
T7	0.591	0.428	0.403	0.253	0.512	0.570	0.978
T8	0.382	0.461	0.455	0.226	0.160	0.582	0.978
T9	0.297	0.537	0.525	0.114	0.253	0.651	0.978
Average	0.470	0.471	0.458	0.230	0.302	0.610	0.971

Table 4.50. Mono network results on mug_4 object (Test set)

mug_5	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.793	0.687	0.680	0.035	0.461	0.790	0.956
T1	0.979	0.838	0.829	0.013	0.904	0.853	0.979
T2	0.650	0.596	0.579	0.052	0.467	0.750	0.954
T3	0.734	0.451	0.439	0.175	0.539	0.646	0.953
T4	0.530	0.536	0.527	0.214	0.467	0.647	0.918
T5	0.944	0.778	0.769	0.030	0.847	0.835	0.978
T6	0.675	0.588	0.569	0.061	0.450	0.739	0.959
T7	0.694	0.578	0.569	0.246	0.403	0.675	0.963
T8	0.762	0.622	0.606	0.035	0.556	0.774	0.963
T9	0.869	0.794	0.772	0.036	0.659	0.818	0.978
Average	0.763	0.647	0.634	0.090	0.575	0.753	0.960

Table 4.51. Mono network results on mug_5 object (Test set)

mug_6	P2D	ADD(-S)	<2cm	MAE	5c5d	AUC	APL
T0	0.717	0.679	0.658	0.145	0.487	0.749	0.958
T1	0.822	0.637	0.613	0.110	0.644	0.741	0.938
T2	0.553	0.531	0.516	0.503	0.322	0.558	0.981
T3	0.110	0.107	0.103	0.637	0.094	0.182	0.392
T4	0.456	0.284	0.259	0.481	0.325	0.450	0.938
T5	0.825	0.625	0.600	0.105	0.688	0.763	0.953
T6	0.675	0.312	0.263	0.326	0.263	0.496	0.950
T7	0.686	0.690	0.674	0.096	0.439	0.727	0.954
T8	0.275	0.438	0.400	0.172	0.113	0.636	0.921
T9	0.725	0.709	0.666	0.228	0.553	0.721	0.963
Average	0.584	0.501	0.475	0.280	0.393	0.602	0.895

Table 4.52. Mono network results on mug_6 object (Test set)

4.4.5 Summary and comparison

In the following tables, the models used for the evaluation have the following shortened names:

- DF-O: Dense fusion trained on RGB images plus depth maps acquired substituting the transparent objects with opaque counterparts in the framed scene;
- DF-T: Dense fusion trained on RGB images plus depth maps acquired leaving the original transparent objects in the framed scene;
- KP: KeyPose;
- Mono: the original monocular PVNet implementation;
- Pr-18: the "parallel" approach for adapting PVNet to stereo images (uses ResNet18 as its backbone);

- St-18: the "stereo" approach for adapting PVNet to stereo images (uses ResNet18 as its backbone);

- St-34: the "stereo" approach for adapting PVNet to stereo images (uses ResNet34 as its backbone).

The only metrics being considered are AUC, <2cm and MAE, since they are the only ones reported in original KeyPose paper [13]. Performances on the KeyPose and DenseFusion models are taken directly from the previously mentioned paper. The two tables are meant to be seen as if they were concatenated horizontally, with each column representing one of the 15 objects in the TOD dataset. Rows report the name of the metric being computed and are combined in groups of three to distinguish between the different models being evaluated.

All metrics are reported as percentages ranging from 0 to 100, with the exception of the MAE metric which is expressed in **millimeters**.

meth.	metr.	ball	bott_0	bott_1	bott_2	cup_0	cup_1	mug_0	mug_1
DF-O	AUC	90.0	88.6	69.1	56.0	84.0	80.7	67.8	66.3
DF-O	<2cm	94.4	97.8	9.1	28.4	79.1	65.3	12.5	10.3
DF-O	MAE	9.9	11.3	57.6	77.8	16.0	37.5	32.2	33.7
DF-T	AUC	84.7	81.6	72.3	47.5	59.4	77.8	54.5	51.3
DF-T	<2cm	78.8	67.5	18.1	9.1	5.6	54.4	4.6	0.3
DF-T	MAE	15.3	18.4	27.6	65.6	40.5	22.1	45.5	48.7
KP	AUC	**96.1**	**95.4**	**94.9**	**90.7**	**93.1**	**92.0**	**91.0**	**78.1**
KP	<2cm	**100**	**99.8**	**99.7**	**91.4**	**97.8**	**95.3**	**94.6**	**63.6**
KP	MAE	**3.8**	**4.6**	**5.1**	**9.3**	**6.8**	**7.1**	**8.9**	**21.9**
Mono	AUC	86.9	36.6	92.1	80.3	43.8	62.1	46.0	40.9
Mono	<2CM	88.2	22.5	98.7	77.9	35.1	54.5	30.7	20.1
Mono	MAE	28.1	421	8.8	86.9	470	179	324	403
Pr-18	AUC	87.4	81.3	86.5	82.4	69.4	77.2	74.1	67.1
Pr-18	<2CM	91.6	78.4	90.4	79.0	53.6	69.8	58.0	48.2
Pr-18	MAE	11.7	18.8	11.7	16.8	44.3	25.5	25.5	41.7
St-18	AUC	-	-	-	-	-	-	-	-
St-18	<2CM	-	-	-	-	-	-	-	-
St-18	MAE	-	-	-	-	-	-	-	-
St-34	AUC	90.3	80.6	89.7	83.7	66.9	79.6	70.8	51.5
St-34	<2CM	95.6	75.9	95.3	80.5	51.5	73.9	48.8	29.0
St-34	MAE	9.1	22.8	8.9	15.3	54.2	24.7	32.5	111

Table 4.53. Results summary - part one

meth.	metr.	mug_2	mug_3	mug_4	mug_5	mug_6	heart	tree	mean
DF-O	AUC	71.4	70.0	69.0	76.8	51.2	61.7	75.5	71.9
DF-O	<2cm	28.1	20.3	4.7	41.9	3.1	17.2	50.9	37.5
DF-O	MAE	28.6	30.0	31.0	23.2	75.2	38.3	24.5	35.1
DF-T	AUC	60.4	67.3	48.1	70.6	64.9	61.2	55.6	63.8
DF-T	<2cm	12.2	8.1	0.0	20.0	4.7	0.0	0.0	18.9
DF-T	MAE	39.5	32.7	54.9	29.4	35.9	38.8	44.4	37.2
KP	AUC	**89.7**	**88.6**	**87.8**	**91.0**	**90.3**	**84.3**	**87.1**	**90.0**
KP	<2cm	**90.1**	**87.2**	**87.1**	**93.1**	**92.2**	77.2	82.5	**90.1**
KP	MAE	**10.1**	**11.3**	**12.1**	**9.0**	**9.7**	**15.6**	**12.8**	**9.9**
Mono	AUC	68.8	75.3	61.0	75.3	60.2	60.7	65.4	63.7
Mono	<2CM	55.6	63.4	45.8	63.4	47.5	44.4	48.5	53.1
Mono	MAE	142	231	230	90.0	280	23.8	171	205
Pr-18	AUC	79.6	76.6	78.0	80.3	80.0	83.2	84.4	79.2
Pr-18	<2CM	75.4	71.0	73.0	83.2	79.5	**80.9**	**87.5**	74.6
Pr-18	MAE	20.0	25.0	21.8	21.0	20.0	16.7	15.3	22.4
St-18	AUC	-	74.0	77.3	79.2	80.9	79.5	72.5	-
St-18	<2CM	-	65.4	68.2	77.9	77.4	70.2	65.5	-
St-18	MAE	-	31.3	24.4	22.6	19.6	35.1	42.4	-
St-34	AUC	74.5	75.4	80.0	76.9	80.0	79.9	70.1	76.7
St-34	<2cm	61.8	67.2	74.0	74.9	75.2	72.1	57.3	68.9
St-34	MAE	27.5	28.5	20.0	25.6	21.6	25.6	52.2	31.9

Table 4.54. Results summary - part two

4.4.6 Results discussion

Keeping the performance of the original monocular PVNet approach as a baseline, it is possible to see from table 4.54 that it outperforms the depth-based Densefusion network on AUC and <2cm metrics but not on MAE, due to the presence of outliers in the detections skewing the mean (since the <2cm metric is 53.1% in average, it means that roughly half the samples have a MAE below 20 millimeters, while the average MAE is 205 millimeters). This was expected since obtaining good depth readings on transparent objects is hard, and so DenseFusion was not able to gain much information from the depth maps. When the depth maps were acquired using a opaque substitute for the object, DenseFusion has a better AUC than PVNet but still a significantly lower <2cm metric, showing again that PVNet is generally more precise but has higher (in magnitude) errors in the presence of a noisy detection.

Aggregating information from both images in the stereo pair significantly increases PVNet's performance, both in the "parallel" and "stereo" approaches. The "parallel" model obtained better results than the "stereo" approach, getting close to the ones of the original method proposed for the dataset (Keypose) in most objects. In particular, table 4.54 shows that the average MAE for all objects goes down from 205 millimeters for the monocular case to 22.4 millimeters in the "parallel" case, just above the 2cm threshold for the corresponding metric. This demonstrates that, even in the presence of noisy detection in one or both images, enforcing camera transformation constraints with an optimization procedure can correct both detections, reducing the error by an order of magnitude.

4.4.7 Possible improvements

Possible improvements for both systems are:

- exploring different sets of keypoints rather than furthest point sampling, for example the interpolated bounding box corners used in [3];
- separating the object detection and keypoint vote regression in two stages, performing segmentation first and then performing regression on the patch highlighted by the detection step. In particular it would be interesting to implement the heatmap object detection approach used in Keypose;
- exploring different backbone networks instead of ResNet, like EfficientNet [56];
- integrating a differentiable RANSAC procedure to optimize directly a loss defined on the 2D keypoints.
- possibly integrating a differentiable PnP module to turn PVNet into an holistic method able to optimize directly the object's pose.

4.4.8 Successful test samples

Here are displayed some ideal network outputs for some test samples. In the following images, green represents ground truth bounding boxes, blue represent monocular PVNet predicted bounding boxes (computed on the left images), and magenta represent bounding boxes predicted by the parallel or stereo models.

When comparing parallel and monocular predictions, if one looks at the left image only it may seem that both networks made very similar predictions. In reality, this reflects well the problem with monocular pose estimation: the model makes a prediction that fits the left image, but that does not necessarily mean that it feats the real word, and that becomes evident when the same pose is transformed in the right image frame. Instead, the parallel network can refine the (possibly very wrong) predictions on left and right images in order to force the result to fit both.

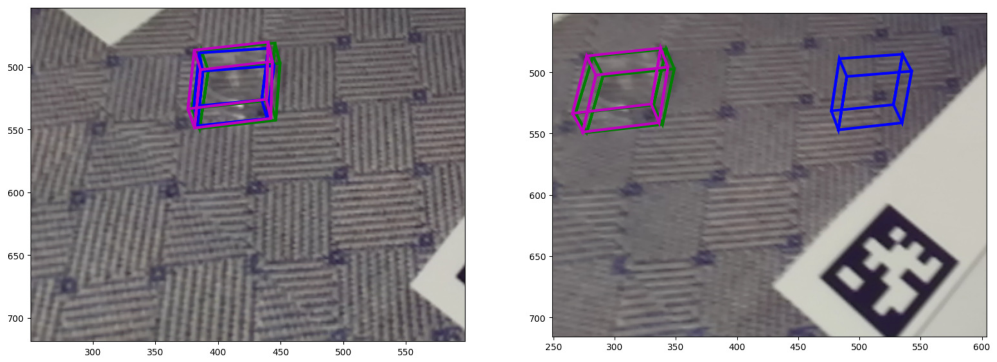

Figure 4.14. heart_0 success case for the parallel network on texture 2; on the right image it is possible to see that the monocular network prediction is not robust to viewpoint changes. The MAE metric for the monocular estimation is 1.74 meters, which is lowered to 9.5 millimeters when applying the parallel method

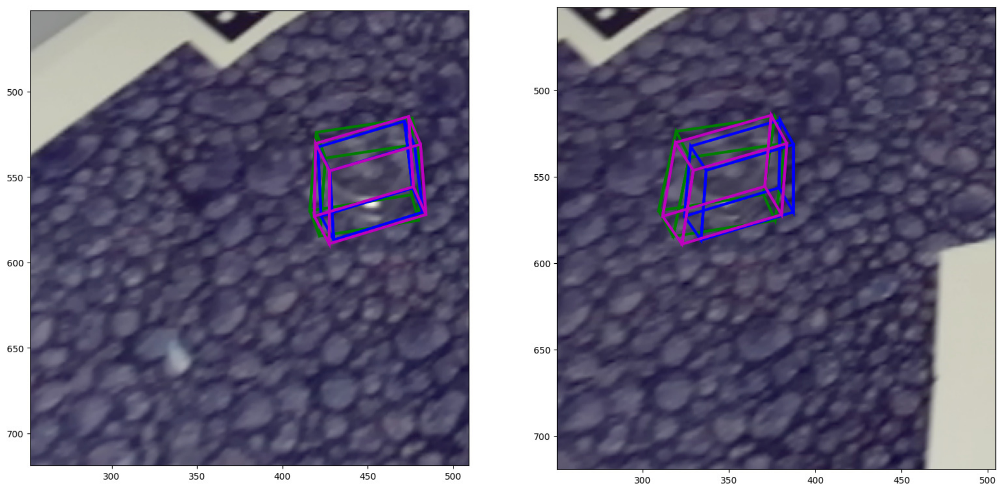

Figure 4.15. heart_0 success case for the parallel network on texture 4; in this case, the monocular network made a better prediction and so the drift in the right image is less apparent. The MAE metric for the monocular estimation is 7.1 centimeters, which is lowered to 4.9 millimeters when applying the parallel method

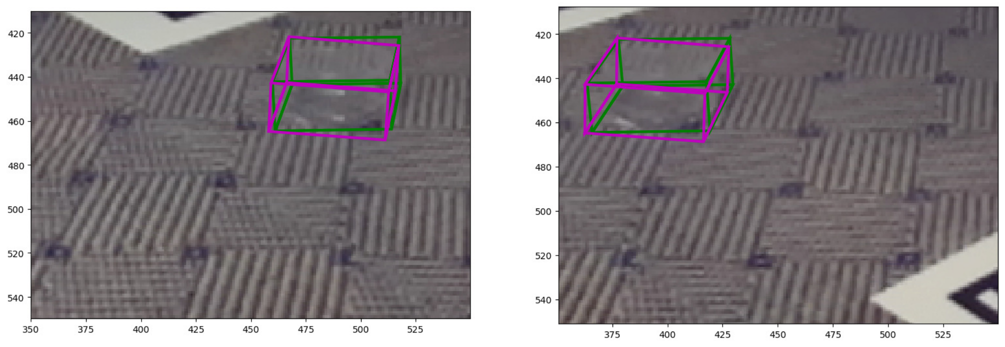

Figure 4.16. heart_0 success case for the Stereo34 network on texture 2; it is possible to notice a slight tilt in the predicted bounding box but the MAE metric shows an error of only 4.2 millimeters

4.4.9 Failed test samples

The TOD dataset contains many challenging samples, mainly when textures 4 and 8 are involved. Below, some failure cases are reported: in the images the green crosses represent the ground truth location for the 2D keypoints, while the blue crosses are the model's prediction; the area highlighted in yellow represents the object's segmentation.

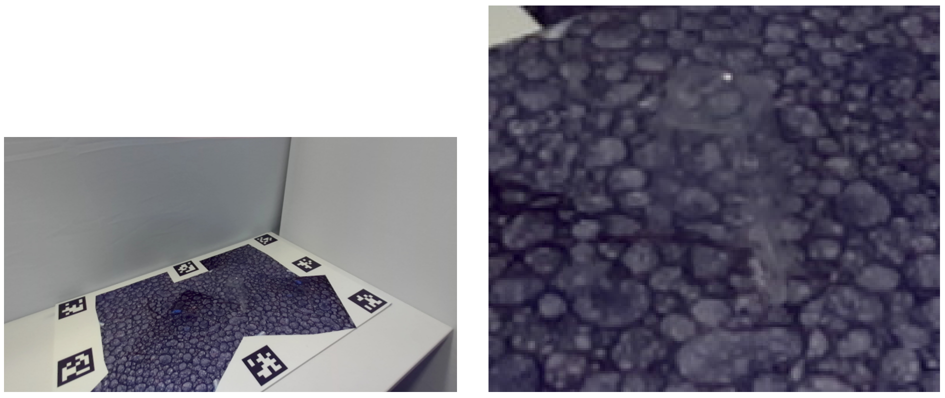

Figure 4.17. Original left camera image for mug_1 object over texture 4. On the right a zoom in on the transparent object is provided. The mug is at the center of the texture

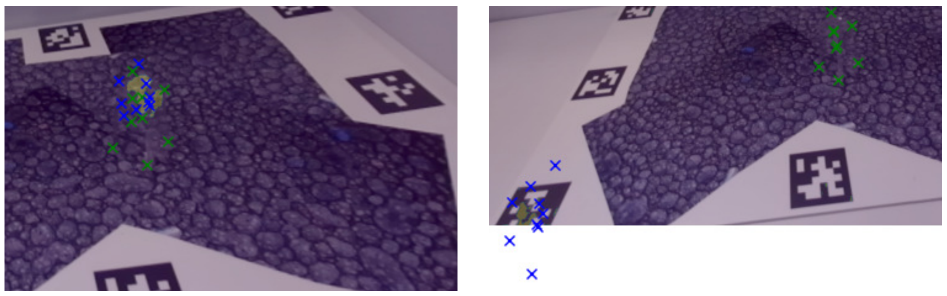

Figure 4.18. Stereo34 network prediction for mug_1's failure case on texture 4: left image's prediction is on the object but with a wrong scale since it only detected a small portion of the object, while the right image's prediction is completely wrong, detecting the object on the corner of the image

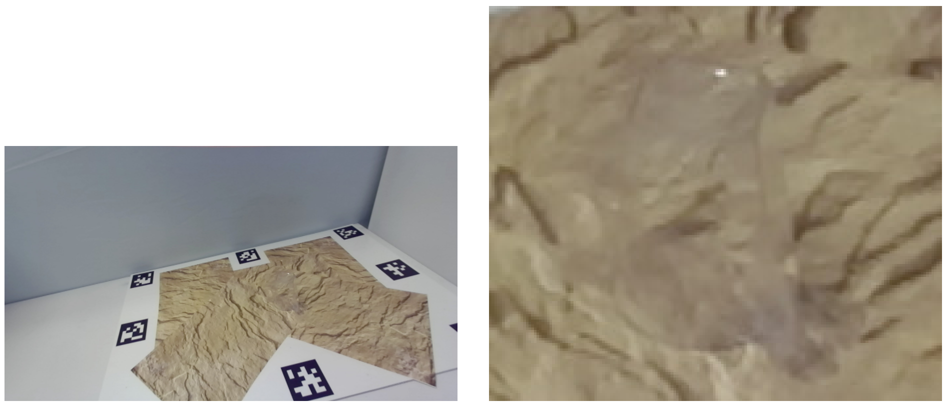

Figure 4.19. Original left camera image for mug_1 object over texture 8. On the right a zoom in on the transparent object is provided. The mug is at the center of the texture

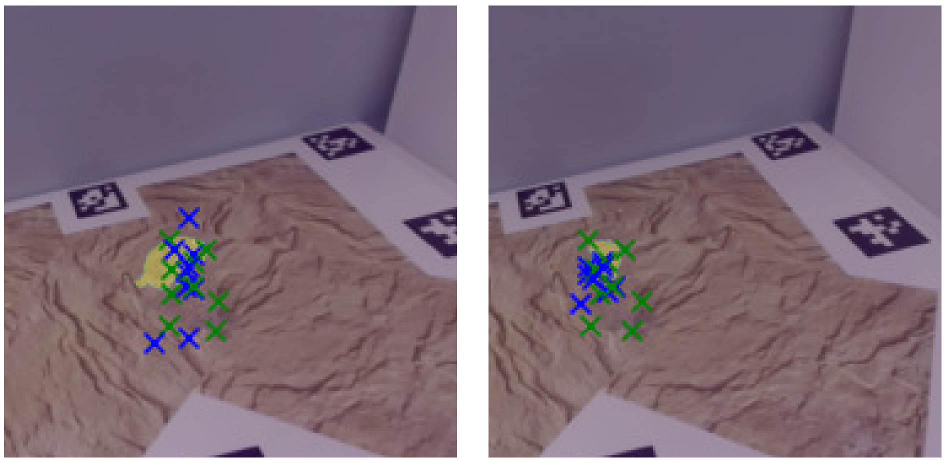

Figure 4.20. Stereo34 network prediction for mug_1's failure case on texture 8: both cameras detect only a small part of the object, resulting in partially correct but incompatible results

Chapter 5

Conclusions

In this work two methods were developed to adapt the existing PVNet [1] monocular pose estimator to the multi-view scenario: one processing each view in parallel with PVNet and then performing post-processing to aggregate and optimize the results, and one processing directly the aggregated views as a single input. Both models were then provided with a custom post-processing procedure, optimizing reprojection errors jointly on all camera views, and enforcing the bias that every image contains one and only one object to detect.

Both models were evaluated on the challenging Transparent Object Dataset [13], and showed significant improvements with respect to the monocular network baseline and also outperformed a depth-based pose estimation model, which struggles to get a good depth reading on transparent objects. Both systems did not reach the performance of the original network developed for the specific dataset on some particularly challenging objects but have the potential to grow and reach better results.

For what concerns the possible improvements , the “parallel” approach performs better as is and can be easily extended to a system with more than two cameras by adding more parallel executions of the same monocular PVNet architecture and modifying the joint reprojection optimization; however, it is hard to add constraints that associate images *during* the network inference step, since it is independent for each camera in the system, so the only improvements that can be made are on the post-processing and aggregation of the individual results. The “stereo” approach performs slightly worse and adding more views directly in the model adds complexity to the network itself, so it is less scalable in that sense; however, since all images are processed concurrently by the same network, it is possible to create direct connections between the views, for example by estimating disparities or leveraging a loss connecting votes from all views with geometrical constraints. For this reason, this last model is the one with the most growth potential for what concerns *stereo* pose estimation.

To conclude, the proposed approaches extending PVNet to the multi-view domain have proven to be effective even in the challenging case of transparent object pose estimation and the results have shown that the complexity of processing more than one view pays off in terms of overall accuracy.

Bibliography

[1] S. Peng, Y. Liu, Q. Huang, H. Bao, and X. Zhou, “Pvnet: Pixel-wise voting network for 6dof pose estimation,” 2018.

[2] C. Song, J. Song, and Q. Huang, “Hybridpose: 6d object pose estimation under hybrid representations,” 2020.

[3] A. Amini, A. S. Periyasamy, and S. Behnke, “Yolopose: Transformer-based multi-object 6d pose estimation using keypoint regression,” 2022.

[4] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *CoRR*, vol. abs/1711.00199, 2017.

[5] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, “SSD-6D: making rgb-based 3d detection and 6d pose estimation great again,” *CoRR*, vol. abs/1711.10006, 2017.

[6] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, “6d object detection and next-best-view prediction in the crowd,” 12 2015.

[7] B. Drost, M. Ulrich, N. Navab, and S. Ilic, “Model globally, match locally: Efficient and robust 3d object recognition,” pp. 998–1005, 07 2010.

[8] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, “Going further with point pair features,” *CoRR*, vol. abs/1711.04061, 2017.

[9] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, “Densefusion: 6d object pose estimation by iterative dense fusion,” *CoRR*, vol. abs/1901.04780, 2019.

[10] M. Rad and V. Lepetit, “Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth,” 2017.

[11] X. Liu, S. Iwase, and K. M. Kitani, “Stereobj-1m: Large-scale stereo image dataset for 6d object pose estimation,” *CoRR*, vol. abs/2109.10115, 2021.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[13] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige, “Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects,” *CoRR*, vol. abs/1912.02805, 2019.

[14] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, pp. 381–395, jun 1981.

[15] X. Gao, X. Hou, J. Tang, and H.-F. Cheng, “Complete solution classification for the perspective-three-point problem,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 930–943, 2003.

[16] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnp: An accurate o(n) solution to the pnp problem,” *International Journal of Computer Vision*, vol. 81, 02 2009.

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

[18] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms.* USA: Cambridge University Press, 2014.

[19] H. E. Robbins, “A stochastic approximation method,” *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 2007.

[20] Y. Lecun, *Generalization and network design strategies.* Elsevier, 1989.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[22] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, “Dive into deep learning,” *arXiv preprint arXiv:2106.11342*, 2021.

[23] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” *CoRR*, vol. abs/1505.04366, 2015.

[24] X.-J. Mao, C. Shen, and Y.-B. Yang, “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections,” 2016.

[25] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *CoRR*, vol. abs/1505.00387, 2015.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017.

[27] Y. Liu, Y. Zhang, Y. Wang, F. Hou, J. Yuan, J. Tian, Y. Zhang, Z. Shi, J. Fan, and Z. He, “A survey of visual transformers,” *CoRR*, vol. abs/2111.06091, 2021.

[28] C. Gu and X. Ren, “Discriminative mixture-of-templates for viewpoint classification,” in *ECCV (5)*, pp. 408–421, 2010.

[29] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, “Gradient response maps for real-time detection of textureless objects,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[30] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, “Comparing images using the hausdorff distance,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, pp. 850–863, sep 1993.

[31] B. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision (ijcai),” vol. 81, 04 1981.

[32] V. Lepetit and P. Fua, “Monocular model-based 3d tracking of rigid objects: A survey,” *Foundations and Trends in Computer Graphics and Vision*, vol. 1, 01 2005.

[33] M. La Cascia, S. Sclaroff, and V. Athitsos, “Fast, reliable head tracking under varying illumination: an approach based on registration of texture-mapped 3d models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 322–336, 2000.

[34] F. Jurie and M. Dhome, “A simple and efficient template matching algorithm,” in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 544–549 vol.2, 2001.

[35] F. Jurie and M. Dhome, “Hyperplane approximation for template matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 996–1000, 2002.

[36] C. Harris, “Tracking with rigid models,” 1993.

[37] D. G. Lowe, “Robust model-based motion tracking through the integration of search and estimation,” *Int. J. Comput. Vision*, vol. 8, pp. 113–122, aug 1992.

[38] D. B. Gennery, “Visual tracking of known three-dimensional objects,” *Int. J. Comput. Vision*, vol. 7, pp. 243–270, apr 1992.

[39] D. Roller, K. Daniilidis, and H.-H. Nagel, “Model-based object tracking in monocular image sequences of road traffic scenes.,” *International Journal of Computer Vision*, vol. 10, no. 3, pp. 257–281, 1993.

[40] C. Schmid and R. Mohr, “Local grayvalue invariants for image retrieval,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–535, 1997.

[41] C. G. Harris and M. J. Stephens, “A combined corner and edge detector,” in *Alvey Vision Conference*, 1988.

[42] J. J. Koenderink and A. J. van Doom, “Representation of local geometry in the visual system,” *Biol. Cybern.*, vol. 55, pp. 367–375, mar 1987.

[43] D. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[44] P. Wohlhart and V. Lepetit, “Learning descriptors for object recognition and 3d pose estimation,” *CoRR*, vol. abs/1502.05908, 2015.

[45] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Košecká, and A. C. Berg, “Fast single shot detection and pose estimation,” in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 676–684, 2016.

[46] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, “SSD: single shot multibox detector,” *CoRR*, vol. abs/1512.02325, 2015.

[47] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv 1409.1556*, 09 2014.

[48] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, “Fast supervised hashing with decision trees for high-dimensional data,” *CoRR*, vol. abs/1404.1561, 2014.

[49] S. Li, Z. Yan, H. Li, and K.-T. Cheng, “Exploring intermediate representation for monocular vehicle pose estimation,” 2020.

[50] P. Kamousi, S. Lazard, A. Maheshwari, and S. Wuhrer, “Analysis of farthest point sampling for approximating geodesics in a graph,” *CoRR*, vol. abs/1311.4665, 2013.

[51] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, “Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes,” vol. 7724, 10 2012.

[52] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, “Learning 6d object pose estimation using 3d object coordinates,” in *ECCV*, 2014.

[53] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, “Densefusion: 6d object pose estimation by iterative dense fusion,” *CoRR*, vol. abs/1901.04780, 2019.

[54] S. Agarwal, K. Mierle, and T. C. S. Team, “Ceres Solver,” 2022.

[55] J. Berge, “J.c. gower and g.b. dijksterhuis. procrustes problems. new york: Oxford university press,” *Psychometrika*, vol. 70, 12 2005.

[56] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” 2019.

[57] P. J. Besl and R. C. Jain, “Three-dimensional object recognition,” *ACM Comput. Surv.*, vol. 17, pp. 75–145, mar 1985.

[58] S. Tulsiani and J. Malik, “Viewpoints and keypoints,” *CoRR*, vol. abs/1411.6067, 2014.

List of Tables

List of Figures