Dense Stereo Object Pose Estimation

Master thesis by Jan Emrich
Date of submission: 25.07.2022

1. Review: Prof. Dr. Arjan Kuijper
2. Review: M.Sc. Thomas Pöllabauer
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computer Science
Department
GRIS

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Jan Emrich, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 25.07.2022

J. Emrich

Contents

Abstract

Fast, accurate, and robust estimation of 6D poses of objects remains a challenging problem. Such estimators could enable seamless augmented reality experiences or smooth robotic manipulations that are unattainable today. Recently, methods for direct pose regression from RGB images using dense features have been introduced. These approaches are promising: direct regression enables fast inference and dense features increase accuracy. Only direct regression methods are in the range of real-time requirements for the RGB modality. Nevertheless, accuracy is poor compared to state-of-the-art methods such as refinement-based estimators. In addition, dense RGB methods are limited to estimating single images, as the dense features are inherently predicted in image space. Stereo vision provides an additional view of the object from a different perspective, reducing the potential for pose ambiguity and minimizing occlusion. The distance of an object can be inferred directly from stereo images, whereas this is impossible with mono-vision without internalized knowledge of the size of the object. Therefore, stereo vision offers a great opportunity to improve the accuracy of dense pose estimation. We extend the state-of-the-art in dense 6D object pose estimation to stereo vision and create a dense stereo dataset. By comparing our method to the state-of-the-art in direct regression for mono-images, we show the potential for dense estimators with stereo modality. Achieving more than twice the accuracy of the baseline on a difficult object, we show a path for dense methods to close the gap in accuracy, enabling real-time 6D Object Pose estimation from images.

Zusammenfassung

Die schnelle, genaue und robuste Schätzung von 6D-Posen von Objekten ist nach wie vor ein schwieriges Problem. Solche Schätzer könnten nahtlose Augmented-Reality-Erlebnisse oder geschmeidige Robotermanipulationen ermöglichen, wie sie heute unerreichbar sind. Kürzlich wurden Methoden zur direkten Posenregression aus RGB-Bildern unter Verwendung Dense Features vorgestellt. Diese Ansätze sind vielversprechend: Die direkte Regression ermöglicht schnelle Inferenz und Dense Features erhöhen die Genauigkeit. Unter der RGB-Modalität liegen nur die direkten Regressionsmethoden liegen im Bereich der Echtzeitanforderungen. Dennoch ist die Genauigkeit im Vergleich zu State-of-the-art Methoden wie verfeinerungsbasierten Schätzern gering. Darüber hinaus sind Dense RGB-Methoden auf die Schätzung von Einzelbildern beschränkt, da die Dense Features von Natur aus im Bildraum vorhergesagt werden. Stereovision bietet eine zusätzliche Ansicht des Objekts aus einer anderen Perspektive, wodurch das Potenzial für Posenmehrdeutigkeit reduziert und die Verdeckung minimiert wird. Die Entfernung eines Objekts kann direkt aus Stereobildern abgeleitet werden, während dies bei Mono-Vision ohne verinnerlichtes Wissen über die Größe des Objekts unmöglich ist. Daher bietet Stereo eine große Chance, die Genauigkeit der Dense Posenschätzung zu verbessern. Wir erweitern den State-of-the-art in der Dense 6D-Objektposenschätzung auf Stereo und erstellen einen Dense Stereodatensatz. Durch den Vergleich unserer Methode mit dem State-of-the-art bei der direkten Regression für Monobilder zeigen wir das Potenzial für Dense Schätzer mit Stereomodalität. Indem wir bei einem schwierigen Objekt eine mehr als doppelt so hohe Genauigkeit erreichen, zeigen wir einen Weg auf, wie Dense Methoden die Lücke in der Genauigkeit schließen und eine 6D-Objektposenschätzung aus Bildern in Echtzeit ermöglichen können.

1. Introduction

6D Object Pose Estimation is the problem of finding the 3D Translation and 3D Rotation of an object in a scene. It is an important task for computer vision, as it finds use in fields such as robotics, quality control and augmented reality. In robotics, factory automation and quality control, object pose estimation is a required technology to grasp objects to move them, apply manufacturing processes to them or assure their position as part of an assembly. In augmented reality, object pose estimation is required to merge the real world with overlayed information or virtual worlds. In all of the above fields, low precision and accuracy of a pose estimate make these tasks impossible or result in an unsatisfying user experience. Therefore high precision and accuracy are desired.

Precise Object Pose Estimation remains a hard problem for Computer Vision, especially in settings where real-time estimation is required. In these application cases, one being robotics, it is still widely common to rely on specialized depth cameras based on time-of-flight or projected light sensors. These sensors are fast, but also have shortcomings such as high noise and failure modes for reflective or transparent objects.

RGB-only based methods on the other hand are often not real-time capable, such as the state-of-the-art method CosyPose[32]. They achieve high precision by employing iterative pose rendering and refinement predicting, which in its current form is inherently slow.

Direct regression methods are recently catching up to the state of art in object pose estimation accuracy. These methods are made possible due to powerful dense intermediate representations such as 2D-3D correspondences. In this representation for each pixel in the image plane the related 3D coordinate in the object coordinate system is predicted. From these correspondences, the pose can be iteratively estimated with algorithms such as RANSAC[15]. In recent work this step was replaced by a neural network, enabling end-to-end training and much faster inference. New dense representations[11] were found to improve low accuracy caused by occlusions and pose ambiguities. Predicting the pose directly in a single stage, direct estimators achieve real-time requirements by

avoiding iterative algorithms. Nevertheless, direct methods still have lower accuracy than refinement-based ones.

For dense methods, multiple views could be natively integrated into pose estimators relying on RANSAC, but for the new end-to-end methods, there is no one obvious integration mechanism. As a first step, we want to focus on stereo imaging. Stereo inherently offers another perspective and indirect depth images. Mono object pose estimation also suffers from object size ambiguity as the depth of the same object with different sizes is unknowable. Stereo imaging can also provide improvements where direct regression estimators are lacking, namely pose ambiguity and depth prediction. Keypoint methods[41][51] were adapted or developed for stereo pose estimation. They increase their accuracy by 14 to 25 percentage points over their mono versions on the StereOBJ-1M[40] dataset, showcasing the potential for object pose estimation.

As dense end-to-end methods are made for the mono modality and we know about advantages of the advantages of stereo mentioned above, we ask: Can end-to-end methods relying on dense features be extended to stereo such that pose estimation accuracy is increased? If so, how can we fuse the images to achieve the highest pose estimation accuracy? And: Are there other features, such as disparity, that can further improve stereo pose estimation accuracy?

In the next chapter, we will discuss the history of object pose prediction and current approaches. It also covers the state-of-the-art in dense object pose estimation and describes features of this approach extensively. Then, we will present our Dense Stereo 6D Object Pose Estimation method and reasoning for the various fusion approaches and additional features. In chapter 4 we introduce our stereo YCB dataset used to train and evaluate our method. Finally, we evaluate all proposed approaches and compare our method to the state-of-the-art in end-to-end object pose estimation.

2. Related Work

During the 70s and 80s, the most common approach to 3D object recognition relied on 3D representations of classes of objects[3]. This was a theoretically interesting approach but had limited success with real-world images. At the time a robust method to extract features of 2D images was lacking. The ambiguity that is present in many cases when matching 2D views to 3D models was a problem as well.

In the 2000s methods could employ more robust features with interest points such as SIFT[42] and more powerful machine learning. Machine Learning has the limit of requiring a great amount of high-quality data and methods at the time were constrained in the views they could detect.

Range(Depth) data has a long history for 3D object detection and pose estimation as seen in review[47]. These methods often rely on 3D features and are followed up with Iterative Point Matching (ICP)[65] for pose refinement. These methods were computationally expensive and had difficulty with clutter.

2.1. Post-Kinect

In 2010 Microsoft started selling the RGB-D Kinect sensor in its first version. As this was the first time a depth sensor was mass produced and widely available for low prices, it was of great interest to the computer vision community. The Kinect sensor made it accessible to infer the 6D Pose of objects.

Thus, object pose estimation became a focus of researchers. This great increase in interest generated by the Microsoft Kinect sensor and deep learning resulted in many new methods, datasets and evaluation metrics.

One of the most important of the time was [20], a template-based method. As such, they did not need to rely on a big dataset or long training time but were able to reliably

estimate the pose by exploiting the depth data of the Kinect sensor. Hinterstoisser et al. also introduced the Linemod dataset, which is still widely used.

2.2. Neural Networks and Deep Learning

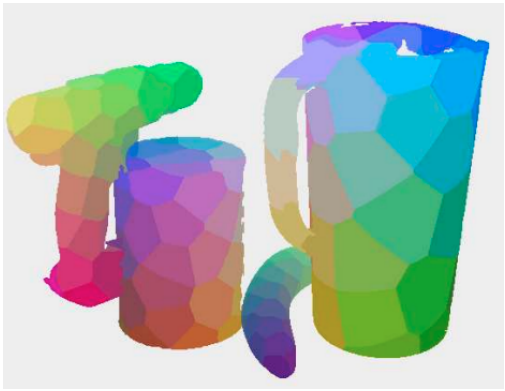
Almost all recent methods make use of deep learning of neural networks. Neural networks as a computational model were first proposed in 1943 by Warren and Pitts[46]. Rosenblatt[52] introduced the first artificial neural network, the Perceptron, in 1958, which has only one layer. The first deep nets were trained by Ivakhnenko et al. in 1965[25][27][26]. Modern backpropagation was first published in 1970 in a master thesis (relating paper[38]). First CNNs were proposed by Fukushima et al. in 1979[16]. Despite these advancements in capability, neural networks were still not widely used, as they were too slow to train and run in inference. With the advent of GPUs this changed in 2004-2006[49][6], but it was still not enough for a breakthrough, especially in computer vision where images require large amounts of computation. That came in 2011 with the first CNNs achieving super-human performance[7][31]. Residual connections enabled the training of even deeper and therefore larger nets[18].

Since learning of large deep neural networks became possible, an array of object pose estimation methods were developed that do not need depth sensors but can infer the 6D Pose of objects based on RGB images only.

2.3. Features and Algorithms for RGB-only Pose Estimation

One subfield of 6D object pose estimation is restricting itself to RGB images only, eliminating the need for special cameras. RGB-only methods have the advantage of RGB camera technology, that is higher resolution, higher framerates and general quality and advancement. Last but not least cameras are mass-produced and therefore cheaply available in high quality. On the other hand, RGB-only methods have disadvantages, such as the need to infer depth from image features. Pose ambiguity and occlusions can also be a problem. To counter these disadvantages several features have been proposed to improve the accuracy of the pose estimate. Such features are:

(a) Dense 2D-3D correspondences for human pose estimation (Picture by [57])



(b) Surface Fragments (Picture by [21])

Figure 2.1.: Dense Representations

Mask In some methods the object mask is predicted, for example in GDR-Net and SO-Pose[59][11]. This can be used to guide a neural network during training, but during inference, the mask is not used in these methods.

Object Center Direction and Object Center Depth are dense features predicted in the PoseCNN method[61]. By interpreting the dense predictions as votes one can obtain the object center and its depth via the Hough Voting scheme described in the next listing.

2D-3D Correspondences were proposed for human pose estimation by Taylor et al.[57] in 2012. They were introduced to object pose estimation in [2]. Figure 2.1a shows an example of 2D-3D correspondences. Every 3D point on the surface of the object is assigned its coordinate values relative to the object coordinate system origin. For each pixel in the image where the object's surface is visible, the corresponding 3D point of the object is predicted. As such it is a dense feature. It has the advantage that it can leverage texture local to the object if available and incorporate global information at the same time. Therefore it works well for textureless as well as texture-rich objects.

Surface Fragments were proposed by Hodaň et al.[21] in 2020. They are also known as regions. Figure 2.1b visualizes surface fragments. Surface fragments improve pose estimation for globally or locally symmetric objects. As the relation of 2D to 3D is not unique for symmetric objects or is unclear if the object is occluded, another feature is required. This is provided by dividing the surface into a controllable number of

regions, that establishes the location of a surface fragment in relation to the global object. Symmetry treatment is simply achieved by classifying the regions. The label for the ground truth region is one-hot encoded and the network is trained to classify the region. If an object has a region that could be matched to 5 regions due to symmetry, it will converge to a probability of 20% for each of the regions. Thus, it implicitly handles and represents symmetry.

Self-Occlusion was proposed in 2021 by Di et al.[11]. Self-Occlusion is similar to 2D-3D

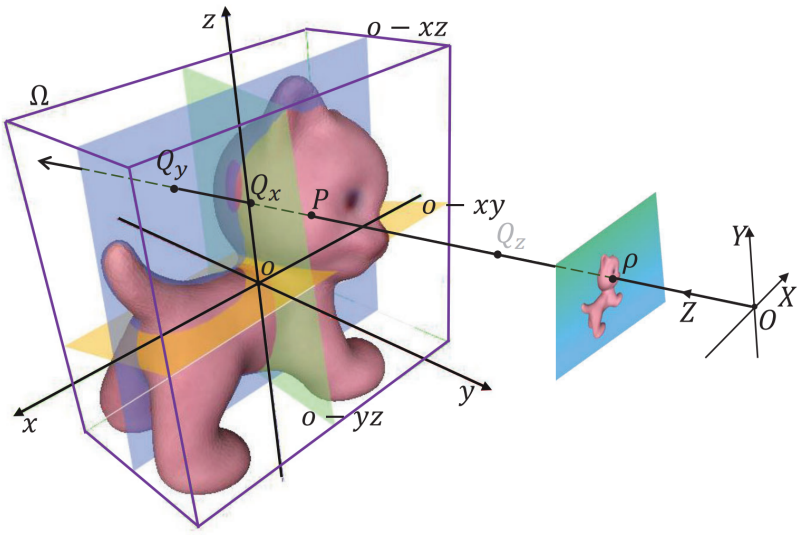


Figure 2.2.: Demonstration of Self-occlusion(Picture by[11])

correspondences in that it relates 2D points on the image plane to 3D points in the object model coordinate system. But unlike 2D-3D correspondences, it is dependent on the view and integrates more information about the shape of the 3D object. As seen in Figure 2.2, for each pixel p in the 2D image projection intersection points Q_x, Q_y and Q_z with the object origin planes $o-yz$, $o-xz$ and $o-xy$ are predicted. The intersection point is defined as the point where the line from the origin O of the camera through the 2D point of a pixel p on the image plane intersects with one of the origin planes. This means for each of the three origin planes, a 2D coordinate relative to the object model origin o is predicted per 2D point of the image plane. This additional geometric information about the 3D model can eliminate ambiguous poses and thus improve pose mismatching, especially for textureless objects.

Once features such as above are predicted the pose can be estimated. For this step, there are different algorithms for the various features. These are:

Hough Voting [23][14] is a method to extract features and detect instances. It has a long history in object pose estimation, for example in one method from 2004[34]. This scheme is used to this day in some methods.

Perspective-n-Point (PnP)[15] is known as the problem where the pose of a camera is estimated given a set of points with known 3D world coordinates and their 2D image projections. As PnP is very sensitive to outliers in the set of point correspondences, a robust algorithm for finding the solution is desired. Often, this is enabled by the iterative algorithm RANSAC[15]. Perspective-n-Point is the second stage for methods with 2D-3D correspondence features or sparse features such as keypoints.

Neural Networks were proposed in 2020 by Yinlin Hu et al. for solving the PnP problem[24]. This has the advantage of shorter inference time and is thus advantageous for applications in robotics and augmented reality. Neural Nets also proved to be easily extendable to other features such as surface fragments and self-occlusion.

2.4. Overview of RGB-only Object Pose Estimation Methods

The pose of an object can be predicted directly or indirectly. Single-Stage approaches predict the pose directly and generally achieve lower accuracy. In contrast, indirect methods first predict an intermediate representation and then apply a second algorithm to obtain the pose. Intermediate representations can be dense like the features in 2.3 or sparse as in keypoint methods. There are also refinement methods to improve initial pose predictions. Recent methods are:

Single Stage

These methods directly predict the 6D Pose from the input image. The rotation is either regressed or viewpoints are classified. SSD-6D[29] was the first to directly predict the pose using a CNN without any engineered features, adapting the SSD[39] paradigm. Recently the transformer detection architecture[5] was also adapted to pose estimation[1]. While this approach is interesting from a deep learning perspective, accuracy without any engineered features or refinement iterations is still lacking.

Dense

Dense methods utilize dense features, in contrast to sparse features such as keypoints. PoseCNN[61] is a dense method. It predicts a dense representation of the object center direction and object center depth. The translation is obtained by a Hough voting scheme from this representation and the rotation regression also integrates the Hough voting results. Pix2Pose[50] is another dense method. It predicts 2D-3D correspondences, which are then fed to a PnP/RANSAC scheme to obtain the pose. The Coordinates-based Disentangled Pose Network (CDPN)[37] predicts dense 2D-3D correspondences which are input to a PnP algorithm using RANSAC[15] to obtain the rotation. The translation on the other hand is regressed directly. EPOS[21] first proposed surface fragments. The method also predicts the 2D-3D correspondences locally for the surface fragments and estimates the pose with the PnP/RANSAC algorithm.

Dense Single Stage

GDR-Net[59] from 2021 was the first single-stage method with dense features and thus the first fully end-to-end trainable network with dense features. This was enabled by introducing a Convolutional/Dense Neural Network to replace the RANSAC algorithm for the PnP problem. Integrating 2D-3D correspondences and surface fragments significantly closed the gap in accuracy toward multi-stage methods. SO-Pose[11] extends GDR-Net with an additional occlusion correspondence feature. This further reduces pose ambiguity, so that it further closed the gap to multi-stage methods. GDR-Net and SO-Pose will be described in further detail in sections 2.6 and 2.7.

Keypoint

Keypoint methods detect a number of characteristic points on the surface of the object. The pose is then derived by solving a Perspective-N-Point problem, usually by running RANSAC[15]. Historically keypoints were detected with SIFT[42]. Today keypoints are predicted with neural networks. One recent method integrating keypoints is HybridPose[55]. Problems in keypoint methods can occur as a single keypoint can be easily occluded.

Dense Keypoint

PVNet[51] combines dense and and keypoint methods. Every Pixel votes for each keypoint. This voting helps with occlusions, as pixels can still vote for hidden keypoints.

Iterative Pose Refinement

Refinement-based methods render the best current guess and predict the difference

to the actual pose in the given input image. This is repeated a given number of times. For RGB-only Pose Estimation DeepIM [36] was the first method to introduce refinement. DeepIM renders the pose given an initial pose predicted by a direct method and predicts the pose with a CNN. As such direct methods and refinement methods complement each other. DeepIM achieved high accuracy on the LineMod dataset [20] for the first time only using an RGB image.

CosyPose [32] combined the DeepIM approach with scene level refinement, improvements such as a better backbone architecture, better rotation representation [66], rotation and translation disentanglement, and good data augmentation. These methods are generally not real-time, because they must generate multiple renderings and forward passes of the pose difference predictor. For every new iteration, it can also be given a new input image, enabling tracking.

One can also render and compare in feature space [28]. This approach allows for fast pose difference prediction by formulating a non-linear least square problem and solving it with Levenberg–Marquardt algorithm [35][44].

2.5. Stereo Object Pose Estimation Methods

As a naive stereo method, RGB-D methods could be used after conversion of the stereo image to an RGB-D image using depth predicted from stereo disparity.

Disparity maps have a long-standing tradition in Computer Vision to obtain depth from stereo images. To predict disparity maps the success of deep-learning methods lead to end-to-end networks [45][30][13][62]. They are using some kind of correlation or distance cost volume as introduced by FlowNet[12].

Keypose[41] introduced a early fusion architecture to Stereo Pose Estimation. They make us of a disparity map as above to predict the depth of the keypoints directly.

The dense PV-Net[51] can be adopted to stereo by predicting keypoints for each image and triangulating the object over all keypoints of both images[41].

2.6. GDR-Net

GDR-Net is the first end-to-end trained network that directly regresses the pose with intermediate dense features. Previous work either did not take advantage of dense representations to improve predictions or relied on a second stage to obtain final results, e.g. RANSAC.

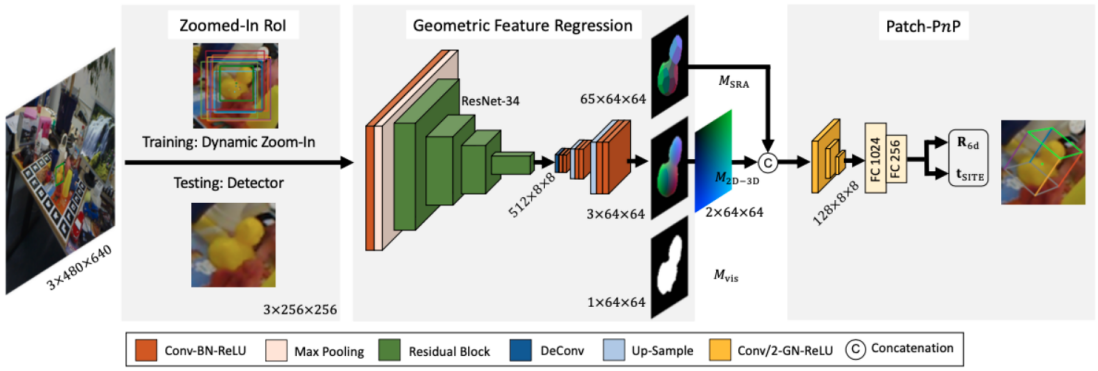


Figure 2.3.: GDRN Architecture (Picture by[59])

The GDR-Net method has three major steps in the pose estimation pipeline as seen in Figure 2.3.

First, a given object detector is utilized to obtain the bounding box and the class of the target object. The input to the next step is the image after zooming in on the bounding box. The image has a resolution of 256 times 256 pixels after this step.

Second, the image is forwarded through the geometric feature regression network. This network has an encoder-decoder architecture. A ResNet[18] encodes the image into a tensor of size 2048 by 8 by 8 if GDR-Net is used in the configuration for YCB-V. The decoder (Figure 2.4) takes this tensor and predicts features in the image space. These features are the mask of the object, 2D-3D correspondences and surface regions. The mask is predicted to guide the network during training but is not used in inference. These features are 64 by 64 in width and height. The 2D-3D correspondences are regressed into three layers, one for each dimension of the object coordinates. The regions are classified into one of 64 via softmax.

Third, 2D-3D Correspondences and surface regions are concatenated and forwarded through the Patch-PnP network. The Patch-PnP network(Figure 2.4) is composed of convolutional layers and two dense heads for regressing translation and rotation of the pose.

GDR-Net has a disentangled 6D pose loss. Meaning the pose loss is separated into parts. The Loss function for the Pose L_{Pose} has three:

$$L_{Pose} = L_R + L_{center} + L_z \tag{2.1}$$

Where L_R is the rotation loss function, L_{center} is the loss function for the scale-invariant object center and L_z is the loss function for the depth estimate.

Representations of rotation that have four or fewer dimensions have discontinuities in euclidian space. Close to the discontinuities significantly large errors can occur when regressing the pose. GDR-Net predicts the rotation as a representation that was introduced[66] to avoid this problem. It is a continous six dimensional representation for the rotation $\mathbf{R}$ in $SO(3)$. This six-dimensional representation $\mathbf{R}_{6d}$ is defined as the first two columns of $\mathbf{R}$:

$$\mathbf{R}_{6d} = [\mathbf{R}_{.1}|\mathbf{R}_{.2}] \tag{2.2}$$

The rotation matrix $\mathbf{R} = [\mathbf{R}_{.1}|\mathbf{R}_{.2}|\mathbf{R}_{.3}]$ is obtained from the six dimensional vector as follows:

$$\begin{aligned} \mathbf{R}_{.1} &= \phi(\mathbf{r}_1) \\ \mathbf{R}_{.2} &= \phi(\mathbf{R}_{.1} \times \mathbf{r}_2) \\ \mathbf{R}_{.3} &= \mathbf{R}_{.3} \times \mathbf{R}_{.1} \end{aligned} \tag{2.3}$$

where $\phi(\cdot)$ is the normalization operation.

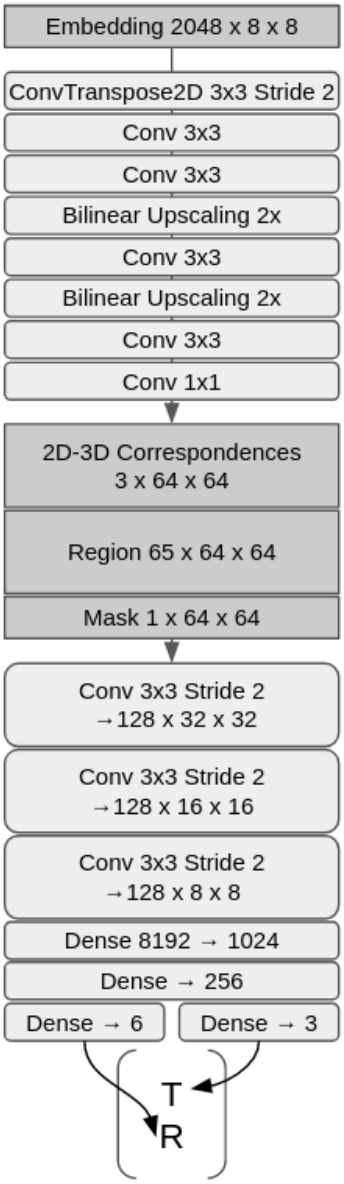


Figure 2.4.: NN Architecture of the GDR-Net Feature Decoder and PnP Net.

The rotation is predicted as an allocentric representation, meaning the rotation is relative to the object coordinate system, as opposed to the camera coordinate system in an egocentric representation. As the allocentric representation is invariant to translations of the object, it was chosen since the method does not predict the rotation directly from the input image, but from the zoomed-in region of interest.

For the same reason, the predicted translation must be represented invariant to scale. Specifically, the region of interest has variations in its size depending on the bounding box given by the detector. Thus the object may appear bigger or smaller and the translation needs to scale with the size of the region of interest. The representation chosen in the GDR-Net method was proposed as Scale-Invariant representation for Translation Estimation (SITE) [37]. SITE is defined as follows: Given size $s_o = \max(w, h)$ and center c_x, c_y of the bounding box that was detected, the ratio $r = s_{\text{zoom}}/s_o$ where s_{zoom} is the zoom size, SITE is:

$$\begin{aligned} \delta_x &= (o_x - c_x)/w \\ \delta_y &= (o_y - c_y)/h \\ \delta_z &= (t_z)/r \end{aligned} \tag{2.4}$$

The translation **t** can then be derived by backpropagation:

$$\mathbf{t} = K^{-1} t_z [o_x, o_y, 1]^T \tag{2.5}$$

With the representations for translation and rotation defined we can specify the parts of the pose loss functions:

$$\begin{aligned} L_R &= \underset{x \in M}{\text{avg}} ||\hat{R}x - Rx||_1 \\ L_{center} &= ||(\hat{\delta}_x - \delta_x, \hat{\delta}_y - \delta_y)||_1 \\ L_z &= ||\hat{\delta}_z - \delta_z||_1 \end{aligned} \tag{2.6}$$

where the hat-like $\hat{\cdot}$ denotes the prediction and the bar $\bar{\cdot}$ the ground truth value. δ_z is the distance, (δ_x, δ_y) the scale-invariant 2D object center and R the rotation of the object. M is the set of vertices of the object model.

The intermediate dense features of GDR-Net have their own loss function L_{Geom}. It is composed of the L_1 and cross-entropy loss (CE):

$$L_{Geom} = ||\bar{M}_{vis} \cdot (\hat{M}_{XYZ} - \bar{M}_{XYZ})||_1 + ||\hat{M}_{vis} - \bar{M}_{vis}||_1 + \lambda CE(\bar{M}_{vis} \cdot \hat{M}_{SRA}, \bar{M}_{SRA}) \tag{2.7}$$

where $\cdot$ is element-wise multiplication and the 2D-3D correspondence map M_{XYZ} and the surface region attention map M_{SRA} have the loss only applied inside the masked visible area as given by the mask M_{vis}. λ is a dataset-dependent parameter that is set to 0.01 for YCB-V.

With the geometric and pose loss we get the whole loss for the GDR-Net method:

$$L_{GDR} = L_{Pose} + L_{Geom} \tag{2.8}$$

Previous work with dense features did not directly regress the pose only based on a neural network. Instead, pose estimation was done by a second stage such as a Hough Layer or the Perspective-n-Point algorithm utilizing RANSAC. This iterative algorithm was replaced in this work by the Patch-PnP neural network. Having the network instead of the RANSAC algorithm results in a predictable short inference time for the pose, which is the main advantage of this method, and is important for applications such as robotics. Furthermore, the network is fully trainable end to end.

Previous single-stage methods did not achieve benchmark accuracies close to refinement-based methods such as DeepIM [36] or CosyPose[32], but GDR-Net significantly improved benchmark results of end-to-end methods [59].

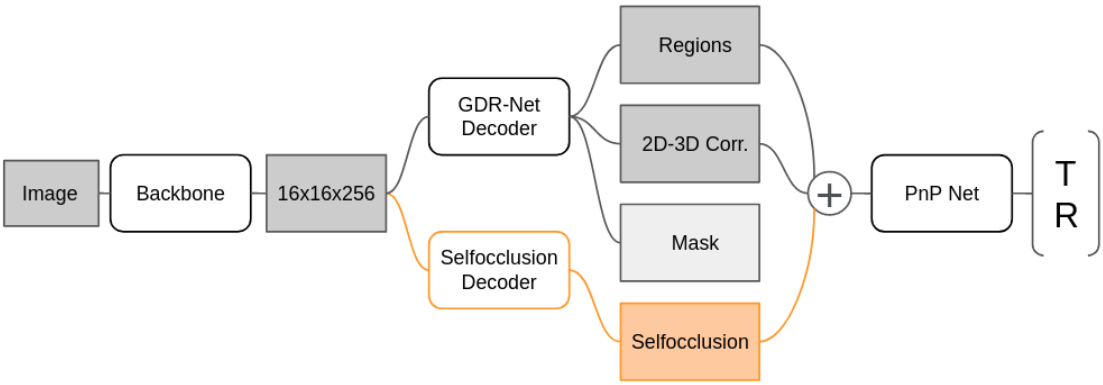


Figure 2.5.: SO-Pose extends GDR-Net with a new decoder and concatenates the dense self-occlusion maps to the

2.7. SO-Pose

SO-Pose[11] extends GDR-Net with an additional intermediate dense feature, self-occlusion, which is described in further detail in section 2.3. In the neural net architecture, a new decoder head(Figure 2.6) was added to the second stage of the network, as shown in Figure 2.5, which predicts these occlusion correspondences. For each origin plane of the object coordinate system, the corresponding other two dimensions are predicted. This gives six layers.

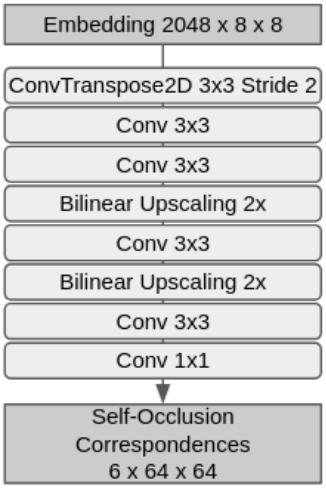


Figure 2.6.: Network Architecture of the Self-occlusion Head

The self-occlusion maps are meant to provide additional information about the 3D object. As this information is independent of the noise in the 2D-3D correspondences, especially in textureless objects, it can eliminate ambiguous poses and thus improve pose mismatching. This self-occlusion information is incorporated in elaborate loss functions, such as the cross-task loss function, which combines 2D-3D correspondences and self-occlusion.

The overall loss function is composed of loss functions for

pose, cross-layer consistency and self-occlusion

$$L = L_{Pose} + L_{CL} + L_{occ} \tag{2.9}$$

where

$$\begin{aligned} L_{Pose} &= L_{GDR} \\ L_{CL} &= \lambda_1 L_{CL-2D} + \lambda_2 L_{CL-3D} \\ L_{occ} &= \lambda_3 L_Q. \end{aligned} \tag{2.10}$$

L_{GDR} is the loss function of GDR-Net described above (section 2.6). This loss function L_{GDR} includes 2D-3D correspondences, surface regions, mask, translation and point matching. The parameters λ_1 and λ_3 are set to 1.0. λ_2 is set to 10. The loss functions L_{CL-2D} and L_{CL-3D} enforce consistency between the occlusion and 2D-3D correspondence layers of SO-Pose. L_{CL-3D} acts in the 3D object space while L_{CL-2D} acts in the 2D image space. For a full in-detail definition of these losses we kindly refer the reader to [11].

SO-Pose further closed the gap in accuracy from end-to-end methods to multi-stage methods in object pose estimation benchmarks.

2.8. Epipolar Geometry

Epipolar Geometry describes the geometry of stereo vision. Under the (perfect mathematical) assumption of the pinhole camera model , there are geometric relations between points in the 3D space and the corresponding points in the image planes as seen in Figure 2.7. These relations lead to constraints between the image points. Let O be the optical centers of the lenses. The places where the image planes are intersected by the baseline are called epipoles e_L and e_R. All points (X_1, X_2, X_3) on the line between a 3D point X and its projection x onto

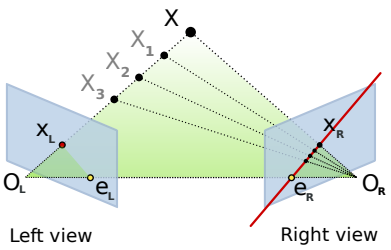


Figure 2.7.: Epipolar Geometry[48]

the image plane lie on one line on the other image plane. This line is called the epipolar line. Since for all possible 3D points X the epipolar plane between X and the optical centers must contain the baseline, all epipolar lines must go through the epipoles.

2.8.1. Coinciding Image Planes

If the image planes of the two cameras are coincident, the epipolar lines are also coincident. Furthermore, the epipolar lines are parallel to the baseline. This makes it possible to align all epipolar lines with the horizontal axes of the images.

The depth of any 3D point X is then given by the distance between the corresponding image points x_L and x_R on the same scanline.

2.9. Stereo Matching

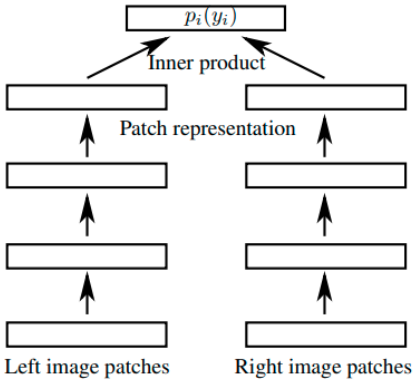


Figure 2.8.: NN architecture with inner product for feature combination (Picture by [43])

Stereo Matching is a task with a long history in Computer Vision. The task ist to estimate the disparity map in horizontally aligned stereo images. The disparity is defined as the distance between two pixels in two respective images, that correspond to the same point in the 3D world. Therefore, disparity directly relates to the depth in the image as described with epipolar geometry above in section 2.8.

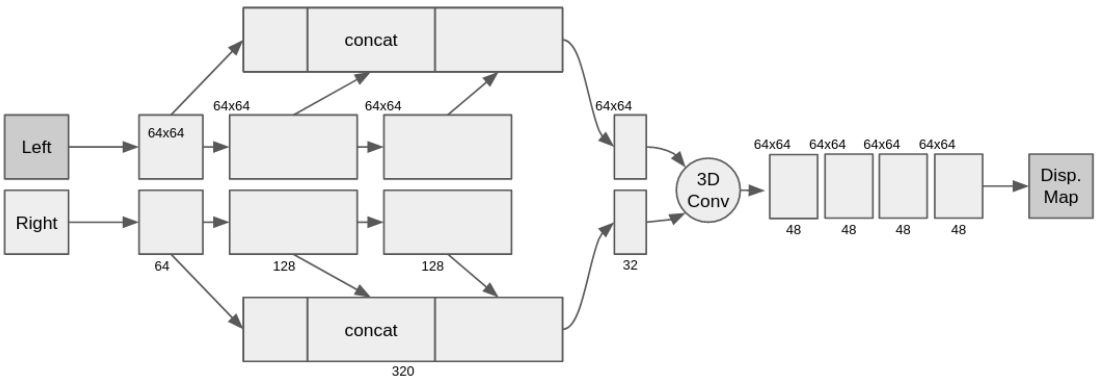


Figure 2.9.: MobileStereoNet Architecture[53]. The numbers to the left top of the box are the width and height of the intermediate tensor. The number in the bottom right corner is the number of channels of the tensor. The three levels of features are concatenated into the longer feature, then convolved and both sides are combined through a 3D convolution. The disparity is then refined through four iterations.

As stereo matching has such a long history, we will limit ourselves here to related work utilizing deep learning.

Zbontar and LeCun[64], as well as Zagoruyko and Komodakis[63] first proposed a Convolutional Neural Network for stereo matching, leveraging ground truth data. They employed a siamese network, combining the two images by concatenating their feature representations and predicting a disparity map via further processing layers.

Luo, Schwing and Urtasun[43] improved upon this model. Instead of concatenating the image features, they computed the inner product as figured in Figure 2.8. This accelerated inference time while improving matching accuracy.

Skipping to 2022, Shamsafar et al. [54] build upon these by introducing 2D MobileNet [53] blocks to stereo matching, reducing computational expense while keeping accuracy high. In addition, they proposed an "Interlacing Cost Volume Construction". Their network architecture is detailed in Figure 2.9.

2.10. 6D Object Pose Estimation Datasets

Since the upcoming of the Kinect sensors many 6D object pose estimation datasets were published. The first major dataset is the Linemod dataset, which became the de-facto standard dataset in 6D object pose estimation. It was originally published as part of the Linemod [20] template-based method for the RGB-D modality, that is depth combined with RGB images. The dataset offers labels for 15 common household objects. The objects differ in size, shape and color. The test contains clutter, but the occlusion is very limited and the test objects are always upright on a flat surface. Thus, this dataset is considered too easy for the state-of-the-art.

Therefore it was extended with a harder test set with significantly heavier occlusions in [2] as a test set now called Linemod-Occlusion.

With PoseCNN[61] a first large-scale dataset was published intended for Deep Learning. The dataset provides 92 videos with 133,827 frames of simple scenes that contain 21 objects of the YCB-Dataset [4].

FAT, a similar, but completely synthetic stereo object pose estimation dataset was released in [58].

Recently another stereo dataset was announced but published too late for consideration in this thesis, the StereoOBJ-1M dataset. It offers real stereo images of objects related to medical tasks.

For the YCB-Video and Linemod datasets, 2D-3D correspondence ground truth labels are available, but not for the stereo datasets. The self-occlusion labels required for SO-Pose training are not available for any of the datasets.

2.11. BOP-Challenge

Since the methods were evaluated on different datasets with different challenges, such as very different levels and variations of lighting, occlusion, the texture of objects and distributions of test poses in space, the reported results were not comparable. To solve these problems Hodaň et al. [22] introduced a rigorous benchmark for rigid object pose estimation for Mono RGB-D images. Their contributions were manifold. Eight datasets were selected and transformed into a common format. Evaluation metrics were defined.

Among the selected datasets are the Linemod datasets and the YCB-Video dataset. An evaluation system was also provided, as well as results and discussion on methods on all datasets. Further, the BOP Toolkit [56] was published to aid researchers on this benchmark.

2.12. Related Work Summary

There is no prior work for end-to-end dense stereo 6D object pose estimation, but there is great state-of-the-art to extend in the mono modality.

As we have seen in this related work chapter, there is great work to build on in dense object pose estimation. GDR-Net laid good groundwork integrating prior work. Combining surface fragments and 2D-3D correspondences with a Perspective-n-Point neural network the method significantly improved accuracy toward the general 6D object pose estimation state-of-the-art. SO-Pose builds on this, adding another dense feature with self-occlusion correspondences. We also have a very long history in computer vision on which to build a disparity feature.

Considering the datasets, prior work for stereo object pose estimation includes the FAT and StereoOBJ-1M datasets. The latter was unfortunately not available during the necessary timeframe for this work and the former is not part of the BOP Challenge, nor is it in the practical BOP Format. As there is no prior work for dense stereo 6D object pose estimation, for neither are 2D-3D correspondences published. Further, both are only available as real data or as synthetic data.

What we can utilize is the great BOP environment with tools for generating synthetic data and labels. Also, tools for generating 2D-3D correspondence and self-occlusion labels are published with GDR-Net and SO-Pose.

3. Dense Stereo 6D Object Pose Estimation

The state-of-the-art in end-to-end 6D object pose estimation focuses on the mono modality and it is not clear how to extend it beyond one view. To make the first step, we extend dense object pose estimation to the stereo modality. As there are many questions to be answered, we explore various fusion approaches of the two views. Additionally, we propose a disparity and deep image feature.

In the next sections, we present various architectures that are tested to find the best way to fuse the images and predict disparity to estimate the object pose. The approaches to evaluate for the fusion of views are Early Fusion, Mid Fusion, Late Fusion and Double Fusion. Early Fusion happens at the embedding stage while Mid and Late Fusion are at the Perspective-n-Point stage. Double Fusion is proposed to combine the advantages of both. These can be combined with additional features.

For now, we give an overview of the method. The estimator is given detections with bounding boxes by the previous stage. These bounding boxes define a region of interest (RoI). Zooming into the RoI we obtain our network inputs. From these images, embeddings are obtained. The next stage in the network is the prediction of dense features such as 2D-3D correspondences, regions and occlusion labels. Here, we add disparity to the dense features. A final network regresses the translation and rotation of the object from these features. Along these steps, both images are merged.

The next section describes common parts of our method.

3.1. Common Configuration

This section describes common parts of the architecture. The training and loss function is the same in all configurations except for additional loss terms and is also detailed here.

One issue that arises in all methods that start from two detections and wish to exploit stereo matching is the question of how to unify the regions of interest.

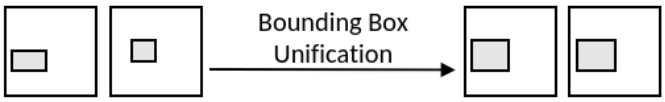


Figure 3.1.: Bounding Box Unification

Bounding Box Unification The bounding boxes of the two images are expanded to include the bounding boxes from both images, resulting in one bigger bounding box used for both images as depicted in Figure 3.1. Unification is required for merging two detections into a single prediction, as the prediction is made relative to the bounding box. Furthermore, the horizontal lines should be aligned for stereo matching with depth estimation.

Backbone The first part of the architecture obtains embeddings from the input images. We use a ResNet-D 50[18]. This model is pre-trained on Imagenet and improved with the ResNet-D trick from [19]. The dimension of the used embedding is 2048x8x8.

Stereo Matching For obtaining general depth information, we employ the disparity estimation network MobileStereoNet[53]. It is comparatively lightweight but achieves good accuracy. We reduced the size of the hidden layers to reduce memory requirements and the computational cost of the network. The architecture can be seen in detail in Figure 2.9.

Feature Heads The feature heads except for the stereo matching features are unchanged from the GDR-Net and SO-Pose baselines. They are described in great detail in section 2.6 and section 2.7.

Loss Function The loss function comprises the loss functions of GDR-Net, SO-Pose and the stereo matching network.

$$L = L_{Pose} + L_{CL} + L_{occ} + L_{Stereo} \tag{3.1}$$

where the top-level loss functions are all equal weight. Parameters λ for the lower level loss functions were unchanged from the SO-Pose configuration for YCB-V.

Optimization For optimization we used the Ranger[60] optimizer with learn rate $1 \cdot 10^{-4}$. The rate warms up with a factor of 0.001 over the first 1000 iterations. In the last

72% of training, the learn rate anneals with the cosine function. Learnrate weights for the SO-Pose cross task loss functions are stepped up from zero after 20% of the training, unchanged from prior work. Weight Decay is not used. Nvidia's "Automatic Mixed Precision for Deep Learning (AMP)" is enabled.

3.2. Late Fusion

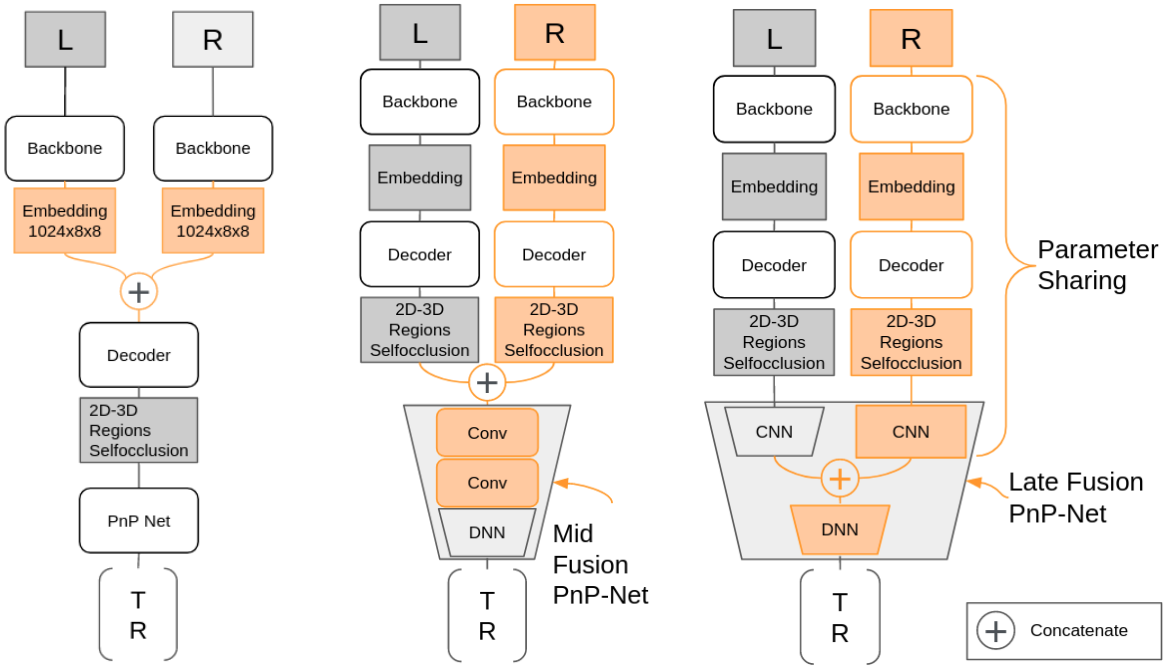
We extend the SO-Pose baseline to stereo with late fusion. In late fusion, the left and right images are merged at the end of the network. Our intuition for late fusion is that as the PnP network performs the pose estimation from 3D-2D correspondences, it can be more robust if it receives two views of the object. On the other hand, late fusion might miss opportunities to improve the features themselves to achieve higher precision.

The net has the overall architecture of a siamese net with separate embeddings and dense features for the left and right images. The architecture is depicted in Figure 3.2c. Here, the fusion happens inside the perspective-n-point network. Image-space features, that is the 2D-3D correspondences, self-occlusion maps and regions for both images are input to the PnP network. After convolving the respective image features separately, both tensors are flattened and concatenated into one tensor. Then the dense layers receive the resulting concatenated stereo features and predict the pose of the object.

3.3. Mid Fusion

While the fusion point for late fusion makes intuitive sense, concatenating the convolved dense features inside the Perspective$n$Point net, it is possible that earlier fusion at the start of the PnP net improves fusion.

To test this thesis, we propose the mid-fusion network. Here, we concatenate the dense image features from both sides just before they enter the PnP net. A visualization of the architecture is shown in Figure 3.2b. As the net now also has to make sense of another view, we double the channel dimension of the three convolutional layers and add two more convolutional layers with the normal channel dimension. In the vanilla PnP net, the convolutional layers have 3x3 kernels with stride two. We keep the kernel for the first three layers but use kernels with stride one for the added two layers.



(a) Embeddings are merged in the early fusion network. The embedding output of the backend is halved in the channel dimension to keep the same channel dimension after merging for the next stage.

(b) In the mid-fusion network, we concatenate the dense features in the channel dimension before they enter the PnP net. Inside the Perspective$n$Point net, the number of channels of the convolutional layers is doubled to help merge the features. Another two convolutional layers with smaller strides are added.

(c) SO-Pose (grey) is extended with a stereo image and its features (orange). The features are fused very late, after the convolutions in the PnP network but before the final dense layers.

Figure 3.2.: Architecture of the Mid Fusion (a), Early Fusion (b) and Double Fusion (c) networks. In orange are the respective changes made to the network.

3.4. Early Fusion

While mid and late fusion both provide two distinct views directly to the PnP net, early fusion aims to improve the dense features the PnP net receives. Our intuition is that by training the dense feature decoders from a merged embedding, comprised of the left and right image, the pose embedded will be of higher accuracy. The decoders may suffer less from pose ambiguity as they can integrate both views, improving the region, 2D-3D and self-occlusion correspondences.

For early fusion, we concatenate the embeddings of the left and right backbone in the channel dimension as depicted in Figure 3.2a. The channel dimension of the left and right embeddings are halved to keep the merged channel dimension the same.

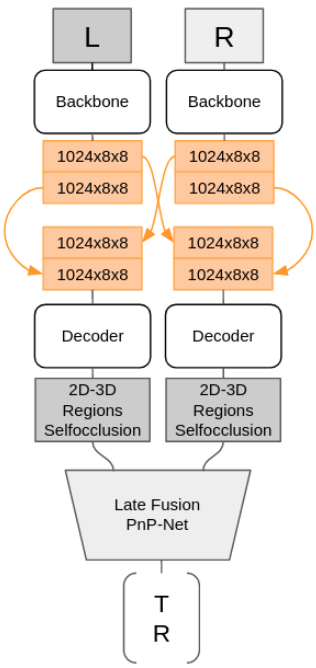
3.5. Double Fusion

As early fusion improves the dense features and late fusion adds a new view to the Perspective$n$Point, we wonder whether the combination can further improve pose estimation. We achieve double fusion by mixing the embeddings and predicting the pose with the Late Fusion PnP net, as shown in Figure 3.3a. The first half of the embeddings are kept on the respective side, while the other half is taken from one side and concatenated to the other.

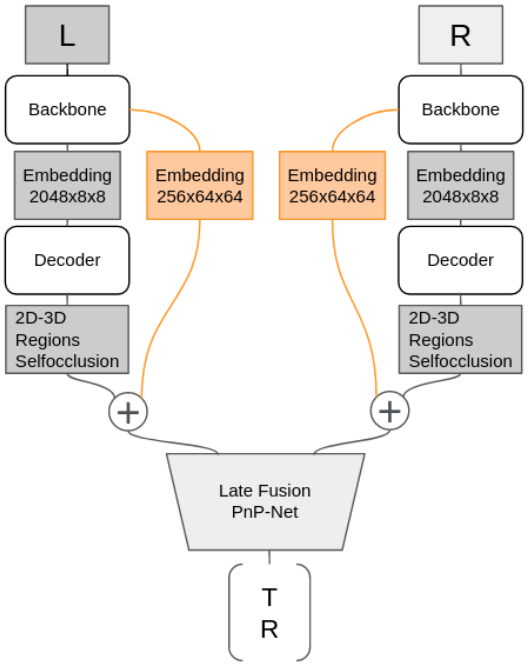
3.6. Deep Image Features

GDR-Net and SO-Pose estimate the 6D Pose only from engineered dense features. The pose prediction or PnP net only receives region, 2D-3D and self-occlusion correspondences, but has no access to the image features themselves. It is clear from prior work that without these engineered features estimation accuracy suffers greatly, at least with the neural networks of today. It stands to reason that adding deep image features may improve accuracy by providing context to the engineered features.

For the deep image feature network, we concatenate the features of the third ResNet block of size 256x64x64 (channels x width x height) directly to the input tensor of the Perspective$n$Point net. The network is depicted in Figure 3.3b. We keep the stereo



(a) In the double fusion network features are merged inside the PnP net and before the decoders. For merging the dense features, we take advantage of the already existing Late Fusion Perspective$n$Point net. Embeddings are divided into local and global part. The first half of the embedding stays local, while the global part is swapped with the other side.



(b) For deep image features, we forward the tensor after the third block of convolutional layers of the ResNet backbone. The features are then concatenated in the channel dimension with the engineered dense image features.

Figure 3.3.: Architecture of the Double Fusion (a) and Deep Image Feature (b) networks. In orange are the respective changes made to the network.

architecture with late fusion to stay on topic for this thesis, but this addition could also be made to the state-of-the-art mono estimators.

3.7. Late Fusion with Disparity

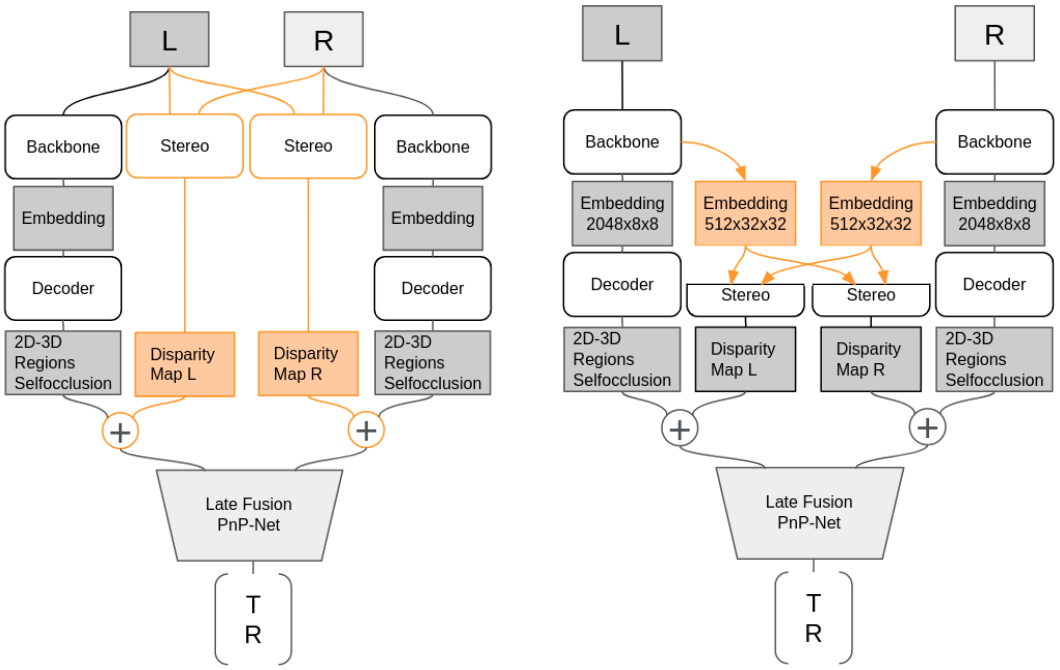
We extend the late fusion network with disparity maps from a stereo matching network. With stereo modality, we not only have two perspectives that can be used for the PnP network as above but can also exploit depth. As described in section 2.8 we can obtain disparity from stereo vision, which directly relates to depth with a stereo camera. Therefore disparity maps are depth maps and directly improve the distance part of the translation prediction. In addition, our intuition is that disparity maps can help the Perspective$n$Point network match the perspective to the 3D points. The PnP network utilizes sets of 3D points, namely 2D-3D correspondences and Self-Occlusion correspondences. Interpreting the disparity maps as additional 3D points from the camera coordinate system instead of the object coordinate system, the network has more points to match.

A disparity map is predicted for the left image as well as the right image as shown in **??**. Two disparity maps are predicted as opposed to only one to keep the symmetry of the architecture and especially the features because the weights of the backbones and other CNNs are shared. As a stereo matching network, the 2D version of the MobileStereoNet[54] is used. We visualised its architecture in Figure 2.9. The width and height of the layers were reduced to keep memory requirements and training requirements, as well as have short inference time. The resulting stereo maps are concatenated to the other dense features in the channel dimension.

3.8. Disparity with Shared Backend

In this network, the first half of the MobileStereoNet, where the features of the images are extracted, is replaced by sharing embeddings from the already existing ResNet backbone. Changes are visualized in Figure 3.4b.

The feature extraction of the MobileStereoNet is sophisticated and employs the pyramid scheme widely used in computer vision. We argue that it is not strictly necessary to have pyramid features in this task. The range of disparities most important in correspond to depths that are not varying much and are mostly in the range of 0.7m to 1.2m. Most



(a) This network simply runs the input images through a MobileStereoNet[53]. The disparity maps are concatenated in the channel dimension before entering the Late Fusion PnP network.

(b) Instead of extracting image features inside the stereo net, we share features of the backbone. This lets us cut the first half of the MobileStereoNet and force the common backend to learn features relevant to depth.

Figure 3.4.: Architecture of the Late Fusion with Disparity (a) and Late Fusion with Disparity from Shared Backend (b) networks. In orange are the respective changes made to the network.

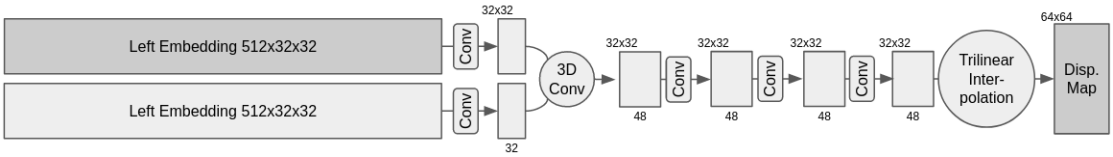


Figure 3.5.: Cut MobileStereoNet. Instead of extracting its features, the disparity estimation shares the ResNet backbone. The feature pyramid is also skipped. Reduced widths and heights of the convolutional layers are shown by numbers in the top left of tensor blocks. The number of features is written at the bottom of each tensor.

importantly, sharing the embeddings reduces the size of the model, enabling faster training and inference. Another advantage is that the common embeddings are trained to contain information necessary for depth estimation, which might be helpful for other downstream tasks such as 2D-3D correspondences.

To further reduce training and inference compute and time, we reduced the width and height of the hidden layers to 32x32 while the previous Late Fusion with Disparity architecture had 64x64. The new architecture for the stereo net is depicted in Figure 3.5.

The resulting disparity maps are concatenated just before the Perspective-n-Point network, as in the Late Fusion with Disparity architecture.

3.9. Early Fusion with Shared Backend Disparity

We aim to combine the advantages of early fusion and the disparity feature. Early fusion should increase the quality of the dense features by merging the embeddings, which eliminates cases of pose ambiguity. Thus early fusion can increase especially the estimation of the rotation. Disparity on the other hand should mainly improve the translation prediction by inferring depth. It can also help the rotation estimate when interpreting the disparity map as additional 3D points from the perspective of the camera. The Perspective$n$Point network already interprets the 3D points in the object coordinate system given by the 2D-3D correspondences and Self-Occlusion correspondences. Adding the disparity loss has the added benefit of forcing the backend to learn features relevant to depth, which may improve the embedding for other tasks as well.

The architecture of this network integrates many features while keeping complexity low. For the second image, we only need to run the backend to obtain the embeddings for disparity estimation and merging of embeddings. The disparity is concatenated to the dense features in the channel dimension.

4. Dense Stereo 6D Object Pose Dataset

As we learned in section 2.10, there is neither a stereo 6D object pose estimation dataset with both real world and rendered data, nor a stereo dataset of any type of data with dense labels. Therefore, we create our own. For one, we render a Physically-Based-Rendering (PBR) dataset for training with high-accuracy labels and diverse distribution of poses, texture effects and scene environments. Further, we record various scenes with two Kinect DK cameras in the real world. Unfortunately, these scenes could not be utilized for testing our method due to time constraints. For compatibility, the dataset adheres to the common BOP format and features the YCB objects YCB[4] as targets.

The goal of the dataset is to offer a great number of realistic stereo training and test samples with YCB[4] objects. YCB objects are part of the YCB-V dataset [61] that is very common in Object Pose Estimation. As such, a great number of related work is done with YCB models. The dataset should be similar in objects, distances to objects and scene setup to the YCB-V dataset, but provide much more variation. The real data was recorded with two depth cameras (Azure Kinect DK). We chose a stereo baseline of 50 mm for the dataset, as it is within the range of typical stereo cameras and is wide enough for the typical distances from the camera to objects in object pose datasets. The dataset offers RGB and depth as modalities, and a wide range of labels, including 2D-3D correspondence and self-occlusion labels.

4.1. Physically-Based-Rendered Dataset

The PBR dataset has a wide distribution of scenes. The dataset is made up of random views of random scenes generated by Physically Based Rendering (PBR). Objects of interest, distraction objects, friction and location of objects are sampled randomly. The location of objects is sampled above the ground. We physically simulated the objects falling to the ground in PyBullet[9]. This ensures a wide range of realistic occlusions in the resulting

scenes. Background texture, lighting direction and intensity, viewpoints, material and reflectivity of objects are sampled randomly before rendering in Blender[8]. Blender integrates PyBullet. This whole process was greatly facilitated by BlenderProc[10], which extends Blender with a modular procedural pipeline for photorealistic object rendering. BlenderProc integrates the BOP Toolkit[56] to write the generated data to a BOP formatted dataset. Furthermore, BlenderProc features BlenderProc4BOP, a suit to generate datasets for the BOP competition, such as YCB-V. This suit was extended for a stereo dataset.

(a) A sample render.

(b) A sample render with ground truth bounding boxes.

Figure 4.1.: Physically-Based-Rendered Dataset Examples

Fifty thousand stereo frames of RGB (Figure 4.1a) and depth images were rendered, including up to 15 target objects per frame. For each scene, 25 frames were obtained from random viewpoints of the scene. Annotations for ground truth object pose were given by BlenderProc. Annotations such as object masks and visibility percentages were computed with the BOP Toolkit.

2D-3D Correspondences Further annotations such as precise 2D-3D correspondences and Occlusion correspondences were computed with modified tools from the GDR-Net and SO-Pose repositories. First, the extent of the object masks was obtained via modified scripts from the GDR-Net repository. Then, precise 2D-3D correspondences were computed with modified scripts from the SO-Pose repository. High precision of 2D-3D correspondences is required for SO-Pose loss function optimization. As these tools were too slow for our purposes, we accelerated them with Numba[33], an LLVM-based Python JIT compiler and parallelized tasks via Python Multiprocessing.

Furthermore, we moved the file formats from previous papers to a compressed file format (.npz) for these labels without loss of precision, reducing memory requirements for the whole dataset by about 8 times.

Occlusion Correspondences SO-Pose occlusion labels were also generated from scripts from the SO-Pose repository. We also accelerated these with Numba and parallelized with Python Multiprocessing and compressed the files.

Ground truth labels where less than 10% of the object surface is visible in one of the stereo images were removed from the dataset.

4.1.1. Detections

We sampled random detections from a distribution similar to the Pix2Pose [50] detections for each target object. To the x and y coordinate of the ground truth bounding box, a clipped random gaussian normal was added. The overall width and height of the box were manipulated with multiplicative noise that was sampled uniformly from 0.9 to 1.1.

Listing 4.1: Pseudocode Random Detection Generation

```
def get_noisy_bbox(bbox):
    x, y, width, height = bbox
    x_noise = clip(normal(loc=0, scale=3), min=-10, max=10)
    y_noise = clip(normal(loc=0, scale=3), min=-10, max=10)
    width_noise = uniform(low=0.GB/high=1.1)
    height_noise = uniform(low=0.9, high=1.1)

    x += x_noise
    y += y_noise
    width *= width_noise
    height *= height_noise
    return (x, y, width, height)
```

4.1.2. Dataset Statistics

The dataset has a train and test split. The train split contains 433.645 labels and the test split contains 48.080 labels. The average visibility of target objects in our dataset is only 62%. The distribution among the objects is listed here.

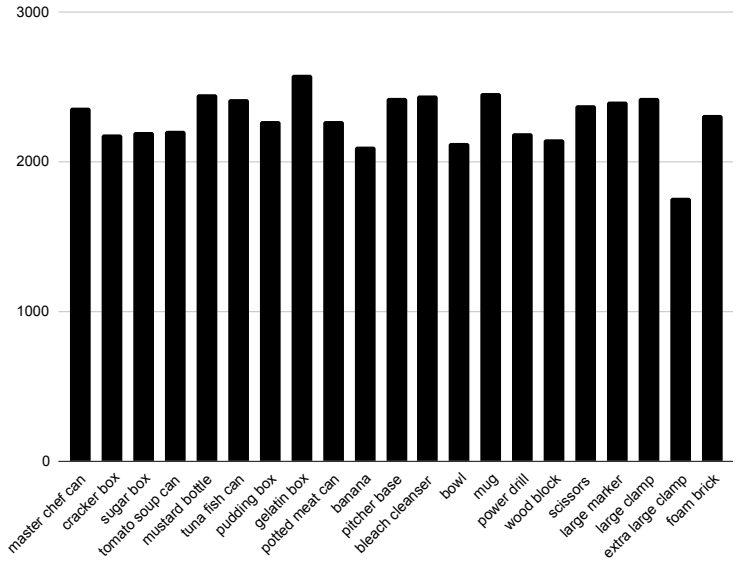


Figure 4.2.: Distribution of instances among object classes.

4.1.3. Baseline Optimization Divergence

We noticed that the SO-Pose baseline diverges during training after a sufficient length of training. As the only difference between our model and theirs in terms of configuration was the visibility threshold, we run an experiment to test this configuration change. The visibility threshold has the effect of withholding the samples from training that are less than 20% visible. Our dataset has all samples excluded that are less visible than 10%, normally this function is disabled in all our experiments. As the SO-Pose baseline, but not the stereo methods are diverging, we set up two training runs for SO-Pose on our dataset that only differ in this option.

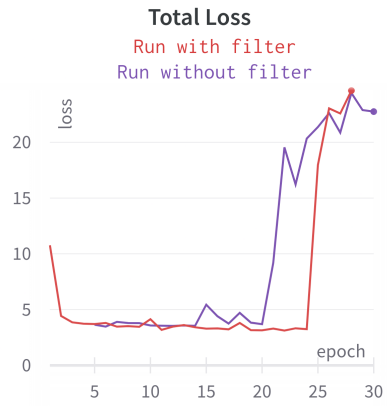


Figure 4.3.: Training Loss Divergence in SO-Pose

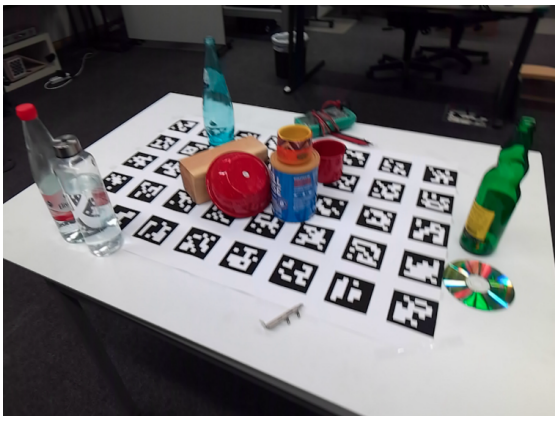
The trajectory of total training loss is shown here in Figure 4.3. We can see divergence after a random but sufficient length of training, long after finishing the first pass over the whole dataset.

As the divergence happens after the first epoch, the issue could arise due to the random bounding box augmentation in GDR-Net and therefore also SO-Pose.

An investigation revealed that this divergence correlates with specific samples in the dataset that sometimes, but not deterministically results in NaN loss values in correspondence losses. After these specific samples were removed from the dataset, we did not notice further divergence. Likely the issue did not arise in training our stereo method, because of the second shifted view and thus additional labels were provided.

4.2. Real Dataset

(a) Kinet DK Stereo Setup



(b) Sample of our Recording

In addition to the synthetic dataset, we also record a real dataset.

Target objects for this dataset are the 21 objects of the YCB object model set[4] that are also used in the YCB-V dataset. Unfortunately, some of the objects are not obtainable anymore, thus they were replaced by newer versions (see table A.1). These objects differ from the ones in the synthetic dataset. We used various objects to distract object detectors and pose estimators. Among them are transparent or metallic objects such as glass water bottles or pans.

Recording of the dataset was done with two Azure Kinect DK cameras. Microsoft provided the recording software Azure Kinect DK Sensor SDK. For obtaining stereo images the cameras were mounted on a fixture (see Figure 4.4a) giving a baseline of 50 mm.

To ensure ground truth generation we kept an ArUco[17] marker board on the table below the objects.

We recorded scenes in various lighting conditions and environments. Locations were inside in four different rooms. One room is big and cluttered and has a big window where we recorded in various lighting conditions, naturally by clouds and by manipulating the blinds. Another room was illuminated by very bright direct sunlight. Two rooms were inside and shielded from sunlight. One of them is artificially lit or left very dim. The final room is a recording lab with consistent artificial light. A sample is provided in Figure 4.4b

5. Evaluation

In this chapter we evaluate the different approaches to extending the method from mono to stereo. We run extensive experiments on the scissor object and also learn a model that can predict poses for all objects. For evalution a multitude of metrics are measured and presented. The experiments are trained and evaluated on our PBR dataset. As the compute required is large, we also train from pre-trained models.

5.0.1. Experiment Setup

We run four experiments, first two preliminary pre-trained experiments, then two final experiments trained from scratch. For each, we have one experiment on the scissor object only and another one on all objects. The experiment on the scissor object only is done as the training compute required for the all-object model is much larger. A training sample of the object is illustrated in Figure 5.1. Still, the scissor object is one of the harder objects in the YCB object set, as it has small surface area with very limited texture. It has only three colors for the main parts while major parts are easily occluded considering our dataset has average visibility of 62%. The small surface area results in a sparse prediction of 2D-3D correspondences and self-occlusion, giving the perspective-n-pose network less to work with.

Pre-trained experiments have all their runs initialized on the same model, irrespective of the architecture. The baseline is also initialized with that same model.

We trained on an array of A100 graphics cards.

Figure 5.1.: Crop of an image with Scissor Object

5.0.2. Baseline

As our method is build upon SO-Pose[11], we benchmark our method against it as a mono image baseline. The authors of SO-Pose made use of various configurations for the different datasets. As our own data is most similar to the YCB-V dataset, we use that configuration provided by the authors. The configuration is left unchanged, except for the parameter controlling the dropping of training parameters based on visibility.

The training parameter controlling the visibility threshold are originally in the SO-Pose paper set to 0.2, meaning all training labels where the visible mask is cut-off or occluded such that less then 20% of object are visible are not used in training. As our synthetic dataset itself has all labels excluded that are less visible than 10% in area and we want to consider training samples where the left label is occluded such that it only has 11% visibility but the right label has 22% visibility, we drop this threshold from all configurations of all methods considered in this work.

5.0.3. Metrics

The following metrics are measured in evaluating the method:

Average Distance (ADD) ADD[20] is a common metric metric for pose estimation, especially in robotics. It measures the accuracy of pose estimation for non-symmetric

objects by summing the distances of vertices. It is computed as follows:

$$ADD(T, \hat{T}) = \frac{1}{V} \sum_{h} ||\hat{T}V^h - TV^h||_2 \tag{5.1}$$

where T is the predicted object pose, $\hat{T}$ is the ground pose, V is the total number of vertices and V^h are the vertices of the 3D object model.

To account for symmetric objects, the closest vertice is considered. Therefore it is computed as follows.

$$ADD\text{-}S(T, \hat{T}) = \frac{1}{V} \sum_{h} \min g||\hat{T}V^h - TV^g||_2 \tag{5.2}$$

If both symmetric and non symmetric objects are evaluated it is common to use a combined metric noted as ADD(-S). Further often results are reported as a percentage of predicted poses that are accurately predicted under a given threshold. As threshold a fraction of the diameter of the object is used. For a threshold of 10% this metric is noted as ADD(-S) 0.1.

n **cm** This metric measures the accuracy of the translation prediction. It gives the fraction of predictions where the translation error is less than n cm.

$n°$ This metric measures the accuracy of the rotation prediction. It gives the fraction of predictions where the rotation error is less than n degrees. This measure always accounts for symmetric objects by considering the smallest error for all possible ground truth poses.

$n°$**,** n **cm** This metric measures wether both rotation and translation are under a given threshold. It combines above metrics.

2D Projection n **px** This metric is measured with the same method as ADD, but instead of comparing the vertices in 3D, they are projected into 2D. With n px we also note the recall threshold.

Ave ° Error Average degree error measures the average error of rotation in degrees.

Ave cm Error Average cm error measures the average error of translation in cm.

Note that only for the last two metrics a lower number is better.

5.1. Pre-trained: Late Fusion on Scissor Object

Method	SO-Pose	Late Fusion	Late Fusion w. Disparity
Batch Size	18	18	18
GPUs	1	1	3
Epochs	630	400	400
Training Time [h]	68	54	106
Compute [A100 hours]	68	54	318

Table 5.1.: Training Parameters for Scissor-Only Experiment

For a preliminary evaluation on a single object, we run this experiment on the scissor object with a pre-trained model. We evaluate the Late Fusion with and without disparity feature against the baseline. The disparity is predicted from its own backend.

As the computation required to converge is very high for these methods, we ensure a good comparison by training for a large number of epochs. This evaluation is trained and evaluated on our synthetic dataset and all training runs in this experiment are initialized with the same pre-trained model.

The training for this experiment is run with the hyperparameters listed in Table 5.1. Batch sizes are purposely kept the same here. As the SO-Pose baseline requires less compute we can it run for more epochs. Results are presented in Table 5.2.

We can see that for most measures Late Disparity has the highest accuracy in this experiment. The commonly reported criteria ADD(-S) 0.1 is 10 percentage point higher for our method than the baseline. But for the measure ADD(-S) 0.02 that requires much more precision, our method has 1 percentage point less accuracy than the mono SO-Pose baseline. For the 2D Projection criteria our method is signficantly worse than the baseline. We also see that our naive stereo extension of SO-Pose is less accurate in all measures than the mono SO-Pose baseline.

Considering that Late Fusion and Late Fusion with Disparity were run for the same number of epochs, the result of adding Disparity is impressive. Similarly, taking into account the 50% more epochs of the baseline, accuracies for Late Fusion are encouraging.

Most importantly, this experiment shows that the order of highest achieving accuracy achieving is mostly dependend on the number of epochs and compute.

Criteria	SO-Pose	Late Fusion	Late Fusion w. Disparity
ADD 0.02	**2.11**	1.01	1.10
ADD 0.05	14.99	10.80	**16.56**
ADD 0.1	34.82	32.60	**42.44**
2° 2 cm	2.49	1.54	**2.98**
5° 5 cm	23.83	19.40	**26.88**
10° 10 cm	52.73	48.58	**55.26**
2°	3.76	2.69	**4.61**
5°	26.17	20.50	**28.42**
10°	54.04	49.26	**55.59**
2 cm	36.98	35.00	**43.98**
5 cm	70.96	69.23	**76.57**
10 cm	88.25	86.03	**90.11**
2D Projection 2px	**4.23**	1.63	2.06
2D Projection 5px	**29.75**	18.29	20.31
2D Projection 10px	**56.44**	46.28	48.82

Table 5.2.: Our method versus baselines trained and evaluated on synthetic data on the scissor object only. Best in bold font.

Another learning is that the compute required for the disparity feature is prohibitively high.

5.2. Pre-trained: Late Fusion on All Objects

In the same vein as the previous experiment, we run this experiment as an preliminary on all objects with a pre-trained model. It is trained and tested on the PBR dataset. Here we target the same number of epochs. Again, we evaluate the baseline, the Late Fusion network and the Late Fusion with Disparity network. The Late Fusion with Disparity network has its own backend.

We train and test on the synthetic dataset and train all estimators from scratch. Only one model is optimized for all objects as opposed to one model per object.

The training for this experiment is run with the hyperparameters listed in Table 5.3. Batch sizes are optimized for memory utilization. Total epochs are set to exactly 25 for our

Training Parameter	SO-Pose	Late Fusion	Late Fusion w. Disparity
Batch Size	100	64	24
GPUs used	1	2	4
Epochs	25	25	20
Training Time [h]	45	47	59
Compute [A100 hours]	45	86	236

Table 5.3.: Training Parameters for Experiment with all objects

method and the baseline. Total epochs were not reached in training because of training job preemption and long training times of the Late Fusion with Disparity network, so we report results for 25, 25 and 20 epochs respectively. Note that this does not give our full method a disadvantage to the baseline in terms of training samples.

From the results listed in Table 5.4 we can see that iterations over the dataset is the most important factor in accuracy, as the results of Late Fusion with Disparity are not comparable. Considering the compute needed (Table 5.3) for this low accuracy we again see that the compute required for disparity of its own backend is prohibitively high.

Comparing our Late Fusion network against the baseline, we see higher accuracy in the important criteria ADD(-S) 0.1 and ADD(-S) 0.05. Prediction of translation is also better than the baseline over all thresholds.

On the other hand, the baseline is ahead in rotation estimation, which carries over to the combined rotation and translation measure. Further, the baseline is slightly ahead in the highest threshold ADD measure.

5.3. Scissor Object

In this section we evaluate the different flavors of architectures to find the best way to exploit stereo. In particular, want to answer wether to fuse test the images early, late or in between. Further, we test the effect of disparity and the deep image feature. As the compute required for training the method in full, we run this experiment on the scissor object only. Training and testing is done on our PBR dataset.

We setup this experiment to train all the architectures described in chapter 3. The exception to this is the architecture that regresses disparity directly from the images, as we found in

Method	SO-Pose	Late Fusion	Late Fusion w. Disparity
ADD(-S) 0.02	**0.93**	0.87	0.01
ADD(-S) 0.05	9.93	**10.53**	0.71
ADD(-S) 0.1	27.89	**31.05**	6.90
2° 2 cm	**1.44**	1.15	0.11
5° 5 cm	**20.01**	16.08	3.99
10° 10 cm	**50.67**	47.11	23.53
2°	**2.84**	2.04	0.57
5°	**22.65**	17.90	6.37
10°	**51.91**	47.92	24.50
2 cm	29.06	**33.96**	15.57
5 cm	66.82	**71.50**	53.86
10 cm	88.74	**91.29**	87.54
2D Projection 2 px	**1.16**	0.60	0.08
2D Projection 5 px	**19.25**	14.31	6.13
2D Projection 10 px	**47.88**	43.70	22.29

Table 5.4.: Table showing recall averaged over objects models on the test set for all criteria. One model was trained for all objects. Due to less training epochs, the late disparity model is not comparable to the other models. Best result in bold font.

section 5.2 it to be too compute costly. Instead we train the method that predicts disparity from the shared backend. The models are initialized randomly and the training is run for exactly 400 epochs for fair comparison of methods.

Training hyperparameters are collected in Table 5.5.

5.3.1. Discussion

Results of this experiment are provided in Table 5.6

Among the different variations of dense stereo estimation, early fusion is by far the most accurate overall. The ADD (-S) measure for early fusion is by far the highest in this experiment. In the 10° criteria, accuracy is almost double of the other tested methods, which also leads to a leading score in the combined rotation and translation measure.

Training Parameter	SO-Pose	Early Fusion	Early Fusion with Disp.	Double Fusion	Mid Fusion	Late Fusion	Late Fusion with Disp.	Late Fusion with Deep Feature
Batch Size	512	256	192	192	256	160	256	192
GPUs used	8	8	8	6	8	5	8	8
Epochs	400	400	400	400	400	400	400	400
Training Time [h]	9	14	15	24	15	22	15	15
Compute [A100 h]	72	112	120	144	120	110	120	120

Table 5.5.: Training hyperparameters for the scissor object experiment. Disp. stands for Disparity

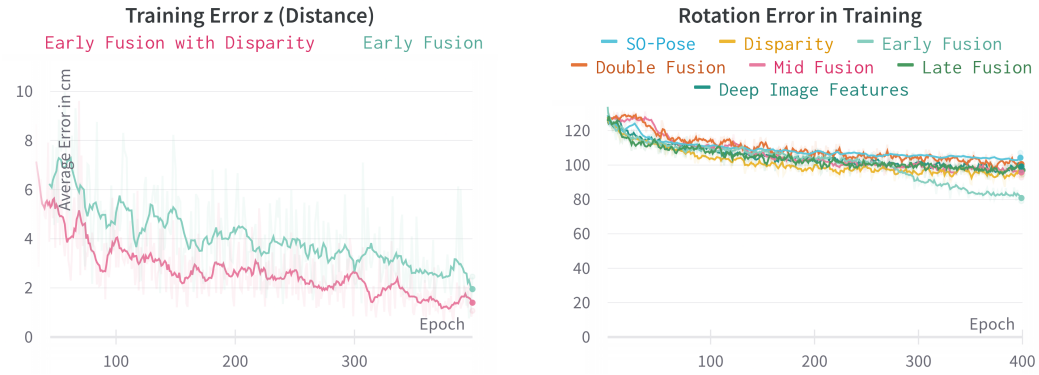
Looking at the rotation error made during training, we can see that the early fusion network is able to signficantly better predict the rotation from epoch 300 on (See Figure 5.2b). This result shows that of the four fusion types, early fusion is the most accurate. It follows that the embedding merged from the stereo view can lead to highly improved dense features. Which in turn enables the Perspective-n-Point network to predict significantly better poses.

The only criteria where the early fusion does not achieve the highest accuracy in our experiment on the scissor object is translation. As depth is inherently the hardest dimension to estimate for RGB-only methods, the disparity network has a slide edge. We also see this during training (Figure 5.2a), where the depth distance dimension of the translation prediction is consistenly better with a disparity feature. Without disparity, depth is the major error source of the translation estimate. This transfers to a slightly higher accuracy in the ADD 0.01 measure, where the Late Fusion network with disparity obtains the best accuracy. Both the Early and Late Fusion networks with Disparity achieve the lowest average translation error. Impressively, the Early Fusion without Disparity almost reaches the same average translation error.

Deep Image Features lead to a slight decrease in accuracies and slight increase in average errors against the Late Fusion baseline. The network does not take advantage of the

Name		Early Fusion	Early Fusion \w Disparity	Double Fusion	Mid Fusion	Late Fusion	Late Fusion \w Disparity	Deep Image Feature
Method	SO-Pose	Our						
1. Stage of Fusion		Early			Mid	Late		
PnP-Net		Mono			Stereo			
Feature		-	Disp	-	-	-	Disp	Deep
ADD .01	0.19	**0.29**	0.29	0.24	0.05	0.19	0.05	0.14
ADD .05	3.57	**7.49**	4.51	3.65	2.88	3.79	3.12	3.74
ADD .1	13.25	**30.72**	17.19	16.80	13.73	15.31	15.89	14.40
2° 2cm	0.09	**0.14**	0.10	0.10	0.05	0.19	0.10	0.05
5° 5cm	1.93	**2.50**	1.44	2.06	1.54	1.82	1.20	1.49
10° 10cm	5.83	**10.51**	6.05	5.57	5.42	6.24	5.33	5.76
2°	**0.19**	**0.19**	0.10	0.14	0.05	**0.19**	0.10	0.05
5°	2.02	**2.54**	1.54	2.06	1.54	1.82	1.25	1.54
10°	6.02	**10.56**	6.10	5.62	5.52	6.24	5.38	5.81
2 cm	37.03	51.27	**54.78**	47.67	42.97	44.98	50.12	44.12
5 cm	70.44	80.84	81.37	77.29	74.94	76.09	**82.00**	75.32
10 cm	88.58	92.27	92.41	90.78	90.25	90.88	**92.51**	90.78
2D 2px	**0.80**	0.62	0.58	0.29	0.19	0.43	0.43	0.29
2D 5px	6.39	**8.74**	4.03	3.98	4.22	4.66	4.46	05.04
2D 10px	16.45	**25.83**	14.31	15.12	14.55	14.74	14.26	14.93
Ave. ° Error	94.99	**57.297**	94.755	91.011	94.871	94.734	94.536	95.029
Ave. cm Error	5	3.7	**3.5**	4.0	4.3	4.1	3.6	4.2

Table 5.6.: Evaluation of our method against the SO-Pose baseline on the scissor object on PBR data. Higher is better except for the bottom two. The best result for each metric is marked in bold. Our early fusion has the best accuracy for the widely used ADD 0.1 metric by a significant margin, almost doubling the accuracy. It also has the lowest average rotation error. Adding a disparity feature to the early fusion network results in the lowest average translation error and highest translation accuracies.



(a) The network with disparity feature consistently has a lower translation error in the depth dimension. The y-axis shows the average error in distance during training in cm.

(b) During training, Early Fusion starts to signficantly better predict the rotation. The y-axis shows the average rotation error on the respective training batches in degrees.

Figure 5.2.: Distance and Rotation Error during training for selected architectures.

additional feature, at least it does not result in improved pose predictions.

Fusing early, but keeping both left and right strains and fusing them late as in Double Fusion, enables higher accuracy. While the result is better than for late fusion, early fusion alone is much better.

To answer the question which type of fusion works best with stereo object pose estimation, we can clearly see that late fusion is preferrable to Late Fusion in all metrics. Late Fusion is on par with the baseline and Early Fusion for 2° criteria, but otherwise not much ahead of Mid Fusion.

5.4. All Objects

In this section we evaluate the most promising architectures of the previous experiment against the baseline.

On the scissor object, we found that the early fusion approach works best, thus we investigate these in a longer running experiment on all objects. We keep testing the

disparity feature and the double fusion approach.

Model	Epochs	A100 hours	ADD(-S) 0.1	10° 10cm
SO-Pose trained by us	60	400	32.86	43.54
SO-Pose trained by us	100	670	44.27	58.05
SO-Pose provided by authors	?	?	52.98	71.37

Table 5.7.: Comparing accuracy of the baseline trained by us for 60 and 100 epochs with the baseline model provided by the original authors. Both were tested on our PBR dataset. We note further training is necessary to achieve full performance.

Training and testing is done on our PBR dataset, and we only learn one model for all objects. The models are initialized randomly and the training is run for exactly 60 epochs for fair comparison of methods. We note that despite employing a large amount of compute, the models still did not fully converge. Training for 100 epochs still does not achieve accuracy of the authors. Table 5.7 illustrates this by comparing the accuracy of our trained baseline with the baseline provided by the original authors.

Training Parameter	SO-Pose	Early Fusion	Early Fusion with Disparity	Double Fusion
Batch Size	256	192	192	192
GPUs used	8	8	6	6
Epochs	60	60	60	60
Training Time [h]	50	53	56	79
Compute [A100 h]	400	424	318	474

Table 5.8.: Training hyperparameters for the final experiment on all objects.

Training hyperparameters are collected in Table 5.8. Note that we permitted the baseline an advantage by having a slightly larger batch size to decrease our training time.

5.4.1. Discussion

Results of this evaluation are shown in Table 5.9. We can see that early fusion and double fusion achieve higher ADD (-S) 0.1 accuracy than the SO-Pose baseline. Unlike the previous experiments double fusion obtains the highest accuracy. This might be due to higher training requirements for the more complex fusing of embeddings, so the advantage is only visible in this longer-running experiment. Double fusion utilizes embeddings of both views for decoding the dense featureas such as 2D-3D correspondences and takes advantage of a stereo PnP-Net for predicting the pose from dense features of two views. This result confirms our hypothesis that a stereo Perspective-n-P Network can improve pose estimation.

Interestingly, the early fusion method with a disparity feature has the best estimation of rotation, which also carries over to the highest accuracies in the combined criteria for rotation and translation. Increased rotation accuracy for disparity supports our thesis that the Perspective-n-Point network can improve its interpretation of the scene with additional depth data. For translation, the disparity estimate might be too coarse to improve upon the already fused embedding in early fusion networks.

Name		Early Fusion	Early Fusion \w Disparity	Double Fusion
Method	SO-Pose	Our		
1. Stage of Fusion		Early		
PnP-Net		Mono		Stereo
Feature		-	Disp	-
ADD .01	1.06	1.15	0.49	**1.37**
ADD .05	12.63	12.96	8.83	**14.45**
ADD .1	32.86	35.77	26.75	**38.46**
2° 2cm	1.16	1.25	**1.45**	1.24
5° 5cm	15.64	15.89	**17.65**	15.06
10° 10cm	43.54	43.91	**44.14**	43.92
2°	2.08	02.07	**2.43**	1.87
5°	17.14	16.74	**19.29**	15.89
10°	44.25	44.13	**45.68**	44.21
2 cm	36.27	41.00	38.33	**43.94**
5 cm	73.04	80.68	74.61	**80.71**
10 cm	91.37	**94.34**	89.11	94.27
2D 2px	**1.27**	0.79	01.04	0.86
2D 5px	18.03	14.84	**18.46**	16.00
2D 10px	**44.15**	41.79	43.93	44.05
Ave. ° Error	30.60	30.33	37.40	**29.53**
Ave. cm Error	4	4	5	4

Table 5.9.: Evaluation of our method against the SO-Pose baseline with one model for all objects on PBR data. Higher is better except for the bottom two. The best result for each metric is marked in bold. Our double fusion has the best accuracy for the widely used ADD 0.1 metric by a significant margin, increasing accuracy by 6 precentage points. Adding a disparity feature to the early fusion network results in the lowest rotation errors, which trranslates to highest combined measures for translation and rotation.

6. Conclusion

In this thesis, we have successfully created a dataset and designed a method for Dense Stereo 6D Object Pose Estimation. The dataset provides all ground truth labels needed for dense object pose estimation. At the same time, it enables stereo modality on the proven YCB objects. By reducing the dataset size from terabyte to gigabyte size, it is possible to publish it with all labels, which was not done before even for mono datasets. The computation of labels for 2D-3D correspondences and self-occlusion correspondences has been accelerated with compile tools. With these tools, we and future work can produce very accurate correspondences, as needed for self-occlusion, quickly in parallel. We have also recorded stereo scenes in strongly varying environments that can be used in future work. For the extension of state-of-the-art end-to-end object pose estimation to stereo, we have proposed and evaluated numerous approaches. Among them are the different choices for the fusion of the views: Early, Mid, Late and the combination of Early and Late Fusion. A disparity and a deep feature were also proposed.

These different possibilities to extend the method from mono to stereo were evaluated in extensive experiments. It has been shown that the stereo modality improves object pose estimation. Of the variants to merge the views, Early Fusion is the best. This means that it is advantageous to already merge the embeddings of the images.

The disparity feature improved pose estimation for some objects, but not others. In some cases, translation or rotation estimation is significantly improved, but on average adding disparity subtracted from Early Fusion accuracy.

In the evaluation of our early fusion method on the difficult scissor object, ADD 0.1 accuracy more than doubled, showcasing the potential of adding additional views to dense 6D object pose estimation.

Thus our work shows a path for end-to-end methods to significantly gain in accuracy, taking on state-of-the-art 6D Object Pose Estimation that yet cannot estimate poses in real-time.

6.1. Future Work

In future work, we want to evaluate our method on the real part of our Dense Stereo 6D Object Pose Estimation dataset. Further, we wish to extend to even more views to increase robustness. The integration of depth could be further improved by predicting depth instead of disparity and fusing the disparity cost volume with the backend embedding. If this alone does not improve the issue that disparity improves pose estimation for some objects while it decreases the average accuracy of others, further investigation should be done in this direction. Keeping the method in the stereo modality, a new geometric loss function that integrates disparity with the existing 2D-3D and self-occlusion correspondences should be evaluated. Such a loss function could force the 2D-3D correspondence for a pixel in the left image to align with the correspondence prediction in the right image. This would be enabled by incorporating the disparity to tie the left and right pixels together.

As training is currently very compute-heavy and slow, improving the PyTorch code of the self-occlusion loss functions would be an important contribution. Further, it should be considered to learn a general pre-trained version of the Perspective-n-Point network. As this network regresses the pose from 3D point features, it should be generalizable to other datasets and models. Learning of the object-specific embeddings and dense features could then be accelerated by starting with the general pre-trained Perspective-n-Point network.

Bibliography

[1] Arash Amini, Arul Selvam Periyasamy, and Sven Behnke. “T6D-Direct: Transformers for Multi-Object 6D Pose Direct Regression”. In: *arXiv preprint arXiv:2109.10948* (2021).

[2] Eric Brachmann et al. “Learning 6d object pose estimation using 3d object coordinates”. In: *European conference on computer vision*. Springer. 2014, pp. 536–551.

[3] Rodney A Brooks, Russell Creiner, and Thomas O Binford. “The ACRONYM model-based vision system”. In: *Proceedings of the 6th international joint conference on Artificial intelligence-Volume 1*. 1979, pp. 105–113.

[4] Berk Calli et al. “The ycb object and model set: Towards common benchmarks for manipulation research”. In: *2015 international conference on advanced robotics (ICAR)*. IEEE. 2015, pp. 510–517.

[5] Nicolas Carion et al. “End-to-end object detection with transformers”. In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[6] Kumar Chellapilla, Sidd Puri, and Patrice Simard. “High performance convolutional neural networks for document processing”. In: *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft. 2006.

[7] Dan Cireşan et al. “A committee of neural networks for traffic sign classification”. In: *The 2011 international joint conference on neural networks*. IEEE. 2011, pp. 1918–1921.

[8] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: `http://www.blender.org`.

[9] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. `http://pybullet.org`. 2016–2021.

[10] Maximilian Denninger et al. “BlenderProc”. In: *arXiv preprint arXiv:1911.01911* (2019).

[11] Yan Di et al. “SO-Pose: Exploiting Self-Occlusion for Direct 6D Pose Estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12396–12405.

[12] Alexey Dosovitskiy et al. “Flownet: Learning optical flow with convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.

[13] Xianzhi Du, Mostafa El-Khamy, and Jungwon Lee. “Amnet: Deep atrous multiscale stereo disparity estimation networks”. In: *arXiv preprint arXiv:1904.09099* (2019).

[14] Richard O Duda and Peter E Hart. “Use of the Hough transformation to detect lines and curves in pictures”. In: *Communications of the ACM* 15.1 (1972), pp. 11–15.

[15] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[16] Kunihiko Fukushima. “Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron”. In: *IEICE Technical Report, A* 62.10 (1979), pp. 658–665.

[17] Sergio Garrido-Jurado et al. “Automatic generation and detection of highly reliable fiducial markers under occlusion”. In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292.

[18] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[19] Tong He et al. “Bag of tricks for image classification with convolutional neural networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 558–567.

[20] Stefan Hinterstoisser et al. “Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes”. In: *Asian conference on computer vision*. Springer. 2012, pp. 548–562.

[21] Tomas Hodan, Daniel Barath, and Jiri Matas. “Epos: Estimating 6d pose of objects with symmetries”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11703–11712.

[22] Tomas Hodan et al. “Bop: Benchmark for 6d object pose estimation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 19–34.

[23] Paul VC Hough. *Method and means for recognizing complex patterns*. US Patent 3,069,654. Dec. 1962.

[24] Yinlin Hu et al. “Single-stage 6d object pose estimation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2930–2939.

[25] Alekse Grigorevich Ivakhnenko and Valentin Grigorevich Lapa. *Cybernetic predicting devices*. Tech. rep. PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING, 1966.

[26] Alexey Grigorevich Ivakhnenko. “Polynomial theory of complex systems”. In: *IEEE transactions on Systems, Man, and Cybernetics* 4 (1971), pp. 364–378.

[27] Alexey Grigorevich Ivakhnenko. “The group method of data handling, a rival of the method of stochastic approximation”. In: *Soviet Automatic Control* 13.3 (1968), pp. 43–55.

[28] Shun Iwase et al. “RePOSE: Real-Time Iterative Rendering and Refinement for 6D Object Pose Estimation”. In: *arXiv preprint arXiv:2104.00633* (2021).

[29] Wadim Kehl et al. “Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1521–1529.

[30] Alex Kendall et al. “End-to-end learning of geometry and context for deep stereo regression”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 66–75.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[32] Yann Labbé et al. “Cosypose: Consistent multi-view multi-object 6d pose estimation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 574–591.

[33] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. “Numba: A llvm-based python jit compiler”. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 2015, pp. 1–6.

[34] Bastian Leibe, Ales Leonardis, and Bernt Schiele. “Combined object categorization and segmentation with an implicit shape model”. In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 2. 5. 2004, p. 7.

[35] Kenneth Levenberg. “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[36] Yi Li et al. “Deepim: Deep iterative matching for 6d pose estimation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 683–698.

[37] Zhigang Li, Gu Wang, and Xiangyang Ji. “Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7678–7687.

[38] Seppo Linnainmaa. “Taylor expansion of the accumulated rounding error”. In: *BIT Numerical Mathematics* 16.2 (1976), pp. 146–160.

[39] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[40] Xingyu Liu, Shun Iwase, and Kris M Kitani. “StereOBJ-1M: Large-scale Stereo Image Dataset for 6D Object Pose Estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10870–10879.

[41] Xingyu Liu et al. “Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11602–11610.

[42] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[43] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. “Efficient deep learning for stereo matching”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5695–5703.

[44] Donald W Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[45] Nikolaus Mayer et al. “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4040–4048.

[46] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[47] Ajmal S Mian, Mohammed Bennamoun, and Robyn A Owens. “Automatic correspondence for 3D modeling: an extensive review”. In: *International Journal of Shape Modeling* 11.02 (2005), pp. 253–291.

[48] Arne Nordmann. *Epipolar Geometry*. File: images/Epipolargeometrie.svg.pdf. 2007. URL: https://commons.wikimedia.org/wiki/File:Epipolar_geometry.svg.

[49] Kyoung-Su Oh and Keechul Jung. "GPU implementation of neural networks". In: *Pattern Recognition* 37.6 (2004), pp. 1311–1314.

[50] Kiru Park, Timothy Patten, and Markus Vincze. "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7668–7677.

[51] Sida Peng et al. "Pvnet: Pixel-wise voting network for 6dof pose estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4561–4570.

[52] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[53] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[54] Faranak Shamsafar et al. "Mobilestereonet: Towards lightweight deep networks for stereo matching". In: *Proceedings of the ieee/cvf winter conference on applications of computer vision*. 2022, pp. 2417–2426.

[55] Chen Song, Jiaru Song, and Qixing Huang. "Hybridpose: 6d object pose estimation under hybrid representations". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 431–440.

[56] M. Sundermeyer T. Hodaň. *BOP Toolkit*. https://github.com/thodan/bop_toolkit. 2020.

[57] Jonathan Taylor et al. "The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 103–110.

[58] Jonathan Tremblay, Thang To, and Stan Birchfield. "Falling things: A synthetic dataset for 3d object detection and pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 2038–2041.

[59] Gu Wang et al. "GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16611–16621.

[60] Less Wright. *Ranger-Deep-Learning-Optimizer*. 2019. URL: https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer.

[61] Yu Xiang et al. "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes". In: *arXiv preprint arXiv:1711.00199* (2017).

[62] Guorun Yang et al. "Segstereo: Exploiting semantic information for disparity estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 636–651.

[63] Sergey Zagoruyko and Nikos Komodakis. "Learning to compare image patches via convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4353–4361.

[64] Jure Zbontar and Yann LeCun. "Computing the stereo matching cost with a convolutional neural network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1592–1599.

[65] Zhengyou Zhang. "Iterative point matching for registration of free-form curves". PhD thesis. Inria, 1992.

[66] Yi Zhou et al. "On the continuity of rotation representations in neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5745–5753.

A. Appendix

Object	YCB Description	Change
1	002_master_chef_can	Texture changed
2	003 _cracker _box	
3	004 _sugar _box	
4	005 _tomato _soup _can	
5	006 _mustard _bottle	
6	007 _tuna _fish _can	
7	008 _pudding _box	
8	009 _gelatin _box	
9	010 _potted _meat _can	
10	011 _banana	
11	019 _pitcher _base	Color changed from blue to transparent for container and to red for lid
12	021 _bleach _cleanser	
13	024 _bowl	
14	025 _mug	
15	035 _power _drill	Slight changes to the model
16	036 _wood _block	
17	037 _scissors	
18	040 _large _marker	
19	051 _large _clamp	
20	052 _extra _large _clamp	
21	061 _foam _brick	

Table A.1.: Stereo Dataset Objects

Method	SO-Pose		Our /wo Disparity		Our	
Criteria	ADD(-S) 0.1	10°, 0.1cm	ADD(-S) 0.1	10°, 0.1cm	ADD(-S) 0.1	10°, 0.1cm
002_master_chef_can			50.99	86.10		
003_cracker_box			51.71	73.49		
004_sugar_box			43.81	69.73		
005_tomato_soup_can			34.25	82.07		
006_mustard_bottle			46.38	70.09		
007_tuna_fish_can			23.91	73.08		
008_pudding_box			35.79	66.36		
009_gelatin_box			26.77	65.16		
010_potted_meat_can			34.09	78.06		
011_banana			21.45	26.06		
019_pitcher_base			51.04	72.41		
021_bleach_cleanser			48.70	64.72		
024_bowl			69.32	80.22		
025_mug			35.64	75.94		
035_power_drill			37.78	55.60		
036_wood_block			78.57	70.30		
037_scissors			14.98	11.19		
040_large_marker			9.57	15.94		
051_large_clamp			51.42	33.30		
052_extra_large_clamp			51.69	31.21		
061_foam_brick			44.88	53.22		
Average over Objects			41.08	59.73		

Table A.2.: Results of the experiment on the synthetic dataset. The table shows accuracy for two of the most important critera for each object.