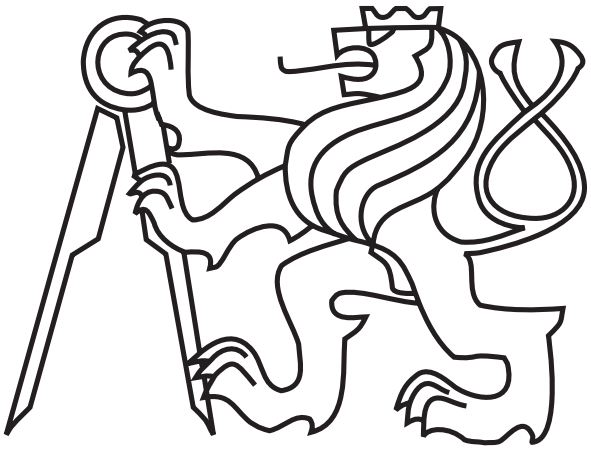
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Master's Thesis



Bc. Pavel Zedník

Detection and Localization of Texture-Less Objects in RGB-D Images

Department of Cybernetics

Thesis supervisor: **Ing. Tomáš Hodaň**

Prague 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Pavel Z e d n í k

Studijní program: Kybernetika a robotika (magisterský)

Obor: Robotika

Název tématu: Detekce a lokalizace netexturovaných objektů z RGB-D snímků

Pokyny pro vypracování:

1. Seznamte se a porovnejte vlastnosti existujících metod pro detekci a lokalizaci objektů z RGB-D snímků (např. [1,2,3,4]) z hlediska jejich vhodnosti pro netexturované objekty.
2. Vybranou metodu implementujte a otestujte její vlastnosti.
3. Na základě testování navrhněte vylepšení a implementujte je.
4. Experimentálně vyhodnoťte vlastnosti vylepšené metody na zvolených datech. Výsledky porovnejte s referenční metodou.

Seznam odborné literatury:

[1] Choi Changhyun – 3D Pose Estimation of Daily Objects Using an RGB-D Camera – 2012
[2] Drost Bertram – 3D Object Detection and Localization Using Multimodal Point Pair Features – 2012
[3] Drost Bertram – Model Globally, Match Locally: Efficient and Robust 3D Object Recognition – 2010
[4] Hinterstoisser Stefan – Multimodal Templates for Real – Time Detection of Texture-Less Objects in Heavily Cluttered Scenes – 2011

Vedoucí diplomové práce: Ing. Tomáš Hodaň

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 20. 1. 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Pavel Z e d n í k

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Diploma Thesis: Detection and Localization of Texture-Less Objects in RGB-D Images

Guidelines:

1. Study and compare properties of methods for detection and localization of objects in RGB-D images (eg. [1,2,3,4]) in terms of suitability for texture-less objects.
2. Implement selected method and test its properties.
3. Based on the tests, propose and implement improvements.
4. Experimentally evaluate the improved method on selected data. Compare results with the reference method**.**

Bibliography/Sources:

[1] Choi Changhyun – 3D Pose Estimation of Daily Objects Using an RGB-D Camera – 2012
[2] Drost Bertram – 3D Object Detection and Localization Using Multimodal Point Pair Features – 2012
[3] Drost Bertram – Model Globally, Match Locally: Efficient and Robust 3D Object Recognition – 2010
[4] Hinterstoisser Stefan – Multimodal Templates for Real – Time Detection of Texture-Less Objects in Heavily Cluttered Scenes – 2011

Diploma Thesis Supervisor: Ing. Tomáš Hodaň

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 20, 2015

Prohlášení autora práce

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

V Praze dne ....

Podpis autora práce

Acknowledgement

I would like to thank my supervisor, Ing. Tomáš Hodaň for his guidance and help throughout this thesis. Many thanks to prof. Ing. Jiří Matas, PhD for his assistance and consultations.

I would also like to thank my parents for their endless support during my studies.

Abstrakt

V této práci jsem implementoval metodu pro detekci a lokalizaci netexturovaných objektů v RGB-D snímcích založenou hlasování pomocí *Point-pair feature* [12], která sestává ze vzdálenosti dvou bodů ve scéně a úhlu jejich normálových vektorů. Navrhl a implementoval jsem několik vylepšení, zejména vylepšený výběr párů ve scéně, vážené hlasování s prořezáváním hypotéz a přepočet skóre. Point-pair feature metoda i její vylepšení jsou vyhodnoceny na Mian a Hinterstoisser datasetech. Výsledky jsou porovnány i s komerční implementací Point-pair feature metody v MVTec HALCON software. Navržený vylepšený výběr párů ve scéně a vážené hlasování s prořezáváním hypotéz umožňují významné zrychlení metody. Přepočet skóre se ukázal jako stěžejní krok, díky kterému je možné dosáhnout značně vyšší úspěšnosti detekce.

Klíčová slova

Detekce netexturovaných objektů, Lokalizace, Point-pair feature, RGB-D snímky

Abstract

In this thesis we implemented a method for detection and localization of texture-less objects in RGB-D images. It is based on the voting scheme which uses the *Point-pair feature* [12] consisting of the distance between two points in the scene and angles of their normal vectors. We proposed and implemented several improvements, notably the restricted selection of pairs of scene points, weighted voting with hypothesis pruning and calculation of a matching score. The Point-pair feature method and the proposed improvements are evaluated on Mian's and Hinterstoisser's datasets. The results are also compared with the commercially available implementation of the Point-pair feature method from the MVTech HALCON software. The proposed restricted selection of pairs of scene points and the weighted voting with hypothesis pruning proved to significantly reduce the time needed for detection. Calculation of the matching score turned out to be crucial for reliable pruning of false positives.

Keywords

Texture-less object detection, Localization, Point-pair feature, RGB-D images

Contents

List of Figures

List of Tables

1 Introduction

In this thesis we focus on detection and pose estimation of texture-less objects using RGB-D images. This is a difficult but important problem with applications especially in robotic perception and grasping where the determination of the full 6 degree of freedom pose is crucial. Until recently, the methods using 2D images had been the main approach for the object detection due to the cheap cameras and fast image acquisition. However, general detection methods using the 2D images are usually not suitable for the texture-less objects. Methods focusing on texture-less object detection have been proposed, but the recognition capability is inherently lower compared to the methods utilizing depth. The depth information has become easily available thanks to the recent introduction of the low cost Kinect-like RGB-D sensors.

First RGB-D sensors are introduced in Section 2. Next in Section 3, we review the current approaches which utilize colour information, depth data or both simultaneously. We focus on the suitability of these methods for detection of texture-less objects.

Section 4 describes in detail the method proposed by Drost et al.[12]. This method utilizes only the depth information and is based on the *Point-pair feature* matching. In Section 5 we then present a several proposed improvements of this method, notably the restricted selection of pairs of scene points and the weighted voting with hypotheses pruning. These improvements significantly reduce the number of false positive hypotheses and thus reduce the detection time.

Our implementation of the method in the MATLAB® software is described in Section 6. The commercially available implementation of the method in the MVTech HALCON software is also presented.

In Section 7, the evaluation of the method and the proposed improvements is described. For the evaluation we chose the datasets published by Mian et al. [27] and Hinterstoisser et al. [20] and compare against the detection results published by the Drost et al [12] and Hinterstoisser et al. [20], and against the detection results from the commercial implementation.

The proposed improvements are integrated into the adjusted detection pipeline which proved to enhance the detection rate and to significantly reduce the time needed for the detection.

2 RGB-D sensors

The RGB-D camera refers to the device which provides the colour (RGB) and depth (D) information for all pixels in the image. In general these devices combine the camera which takes the colour images and the device which measures the depth of the scene. A variety of new RGB-D cameras became available in the recent few years notably the Kinect developed by Microsoft. The low cost, reliability and measurement speed are the key aspects for the usage in the field of robotics.

Figure 1: The Kinect device. Photograph by Evan Amos, distributed under a CC0 1.0 licence.

It consists of the IR projector of the pattern and the IR camera which triangulates the points in the space. The IR projector project the known pattern of the points into the scene. This projected pattern is then captured by the IR camera. From the correspondences between the points in original pattern and the projected one it is possible to compute the depth. The sensor also includes a standard RGB camera. The output of the sensor is the RGB image, IR image and the inverse depth image from which a real depth map can be computed [35].

The depth map is the image which contains the information relating to the distance of the surfaces in the scene from a viewpoint. The example of the depth map from the Hinterstoisser's dataset is shown in Figure 2.

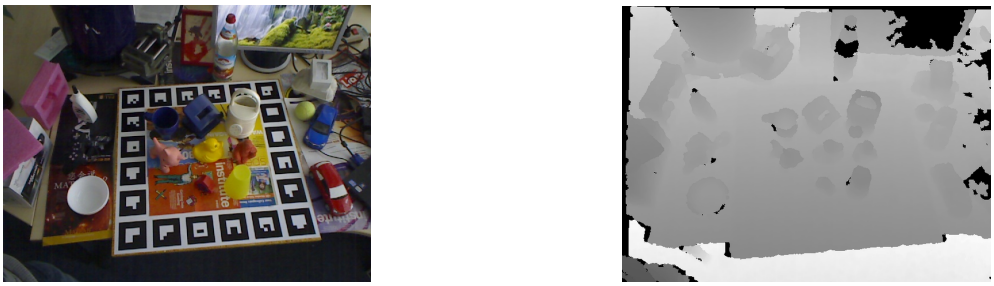


Figure 2: An RGB image and the corresponding depth map. The scene is from the Hinterstoisser's dataset.

3 Related work

In this section we will discuss the current methods for the object detection and pose estimation. Various methods have been proposed. They can be divided into two main categories: methods based on 2D images and methods utilizing the depth data. Our main focus is on methods suitable for detection of texture-less objects.

3.1 2D images

For many years methods using 2D images were the main approach for the object detection. This is especially due to the availability of the cheap cameras and the possibility of the fast image acquisition. Matching the 2D image features with corresponding features in the 3D model is the biggest issue. This is hard especially because of the changes in rotation, scale and illumination in the image. Particular views of the object can also lead to the ambiguity of the true pose.

A several invariant feature descriptors [26, 2] had been used to find correspondences between the image and the 3D model of the object constructed from the stored database of the reference images or the CAD model of the object [15]. However most of these methods assume the presence of the texture therefore they are not suitable for texture-less object detection.

In general the methods utilizing the 2D images can be divided into two groups; the methods based on local features and the methods based on the template matching.

Local feature methods are based on the usage of local invariant features to encode local image structures into the representation which is invariant to the image transformations. One of the most popular feature descriptor is the SIFT [26]. In case of texture-less objects, the most informative local feature is the edge, which is caused by discontinuity in the depth or shape.

The selection of contour fragments was proposed in [31]. Chia et al. [6] presented a contour-based approach where a discriminative selection of lines and ellipses forms a shape structure used for an object detection in images. Damen et al. [10] uses spacial constellations of short straight segments (edgelets) extracted from the edge map.

Template matching methods had the major role in the object detection for many years. The main principle is that the rigid template is scanned over the entire image and some measure is used to find the best match. These methods are typically more suitable for low textured objects than the approaches based on local features. However, increased computational demands due to better robustness and necessity to use a large number of templates to cover all viewpoint lead to the problem that these methods are often not suitable for real-time applications. The similarity measures based on local dominant gradient orientations have been proposed by [19, 29].

To tackle with the lack of the texture the approach here is to rely on edges in the images. When using the boundary of the object a set of edge templates of the object can be computed a priory and then searched to obtain corresponding template. The edge orientation is used in [30, 25], the hierarchical approach is presented in [13]. The method presented by Cai et al. [5] utilizes the edge maps by combining the chamfer matching and the scanning window technique to allow a real-time recognition of the objects in the scenes.

3.2 Depth images

The development of new algorithms for the 3D sensors is driven by increasing cost effectiveness and the availability of commercial 3D sensors. The object detection involves finding correspondences between 3D features in the scene and model. Compared to the 2D images the advantage of the 3D data is relative invariance to the illumination changes. The goal is to find correspondences in presence of the sensor noise, background clutter and occlusion. The features used for finding the correspondences between 3D data of the scene and the model of the object are usually based on surface normals and object boundaries.

The history review of the object detection in range data is presented in [28]. The standard method for the pose estimation is the ICP algorithm [39]. The ICP algorithm minimizes the distance between each point in point cloud and its closest neighbour. However, this method requires a good initial pose estimate. A typical detection pipeline therefore consists of a detection method based on 3D features whose result is then refined by the ICP algorithm.

In [37], the depth and intensity image data are used in Depth encoded Hough voting scheme to detect an object, estimate pose and recover shape information. The Viewpoint Feature Histogram proposed in [34] encodes angular distributions of the surface normals

on the segmented surface. The Clustered Viewpoint Feature Histogram [1] improves the pose estimation and allows to obtain 6-DOF pose of the object. These approaches are able to obtain the object pose. But they rely on a good segmentation from the background therefore they are not well suited for cluttered scenes.

In case of the general pose estimation it is necessary to match the scene with the model directly. Several local invariant features have been proposed. Most notably the spin images [22] where the surface around the reference point is represented in form of the histogram. The histogram of the point feature as a descriptor of the local geometry is used in [33]. The object detection in range images using edge-based descriptor is presented in [36]. An extensive survey of object recognition using the local surface feature methods is presented in [16].

The multimodal template matching method called LINE-MOD which combines the surface normals and the image gradient features is presented in [18]. The method performed well even for the texture-less objects in highly cluttered scenes. However, sensitivity to the occlusion and large amount of data required for the training are the main drawbacks. The framework for automatic modelling, detection and tracking based on the LINE-MOD method is presented in [20]. The pose estimation and the the colour information are used to check the detection hypotheses and therefore to improve the detection results.

The oriented point-pair feature first presented in [38] is used by Drost et al. [12] in Hough like voting scheme to detect objects in 3D point clouds. The method is based on the Point-pair feature which is used in the voting scheme to find transformation from the model to the scene space. This approach is able to successfully work with texture-less objects and recover 6-DOF pose. Despite the efficiency and generality of the method the main issue is detection of objects with a large cluttered background. A similar approach is shown in [7] where the information about colour is utilized by augmenting the original point-pair feature and introducing the color point-pair feature. In [32] the point-pair feature descriptor is used in the RANSAC like sampling scheme.

Due to the large number of types of the range sensors with different properties it is difficult to design an edge detector for depth images. Some have been proposed in [36, 21]. The incorporation of the geometric edge information is shown in [11]. The multimodal point-pair feature combines 3D surface point and the point on the geometric edge to improve the robustness to the occlusion and clutter. The extensive study of various point-pair features

incorporating 3D points with normals as well as the depth edges is presented in [8].

The method proposed by Drost et al. [12] have been selected for our work. The principle and detailed description is presented in the next chapter.

4 Point-pair feature method

For our work we have selected a method proposed by Drost et al. [12]. This method works with the geometric point-pair feature which is invariant to the object colour therefore it can be successfully used for texture-less objects. Moreover for the training phase it requires only a simple 3D mesh model consisting of the oriented points - the 3D surface points with associated normals.

The method assumes that the model and the scene are represented as a finite set of oriented points, $m_i \in M$ denotes the point on the model and $s_i \in S$ points in the scene. The essential part of the method is the point-pair feature describing a pair of two oriented points. The global description of the model consisting of the point-pair features is computed in the off-line phase from all point pairs on the model surface and later used in the on-line phase to obtain a set of possible matches in the scene. A set of reference points in the scene is selected and paired with all other points in the scene. For these pairs the point-pair feature is computed. All features are then matched to the global model and used as voters for a specific object pose.

The best object pose is selected for each reference point. All hypotheses are then clustered and averaged in the clusters. The poses from the clusters with the highest number of votes are returned as the result.

4.1 Point-pair feature

The point-pair feature describes a relative position and orientation of the oriented points. The points m_1 and m_2 with their respective normals n_1 and n_2 define the point-pair feature F:

$$F = \Big(\|d\|, \quad \angle(n_1,\, d) \quad \angle(n_2,\, d) \quad \angle(n_1,\, n_2) \Big) \tag{1}$$

where $d = m_2 - m_1$ and $\angle(x_1,\, x_2)$ denotes the angle between two vectors in the range $\langle 0,\, \pi)$. The point-pair feature is used to build the global model description and also to localize the object in the scene in the on-line phase.

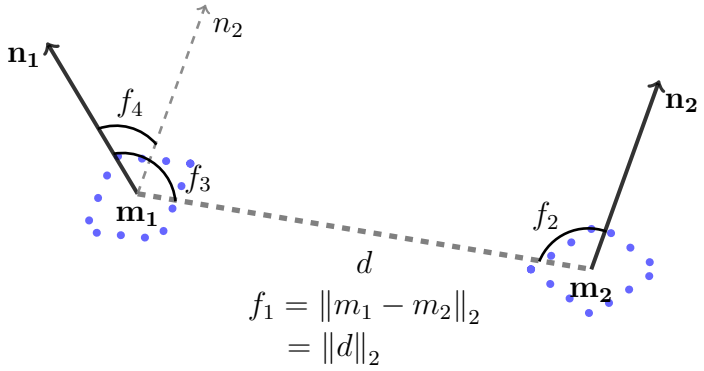


Figure 3: The point-pair feature $F = (f_1, f_2, f_3, f_4)$. The component f_1 is the distance between two points, f_2 and f_3 are angles between the vector d and the normal at the point, and f_4 is the angle between the two normals.

4.2 Global modelling

The global model description is built in the off-line phase using the point-pair feature described above. The model is composed of a set of the point-pair features so that the similar ones are grouped together. First the feature vector F is computed for all point pairs m_i, $m_j \in M$ on the surface of the model. The distances and the angles are sampled in steps specified by d_{dist} and $d_{angle} = 2\pi/n_{angle}$ respectively. Sampled feature vectors with the same quantized distances and angles are then grouped together.

The global model description can be seen as the mapping $L : \mathbb{Z}^4 \to A \subset M^2$. Four dimensional point-pair features are mapped to a set A of all pairs $(m_i, m_j) \in M^2$ that define an equal feature vector. The model descriptor is stored in a hash table with the sampled feature vector F used as a key therefore the pairs (m_i, m_j) with a similar feature $F_m(m_i, m_j)$ to a feature $F_s(s_i, s_j)$ in the scene can be searched in the hash table using the feature F_s as the key. The Figure 4 shows the example of similar features stored in the same slot in the hash table.

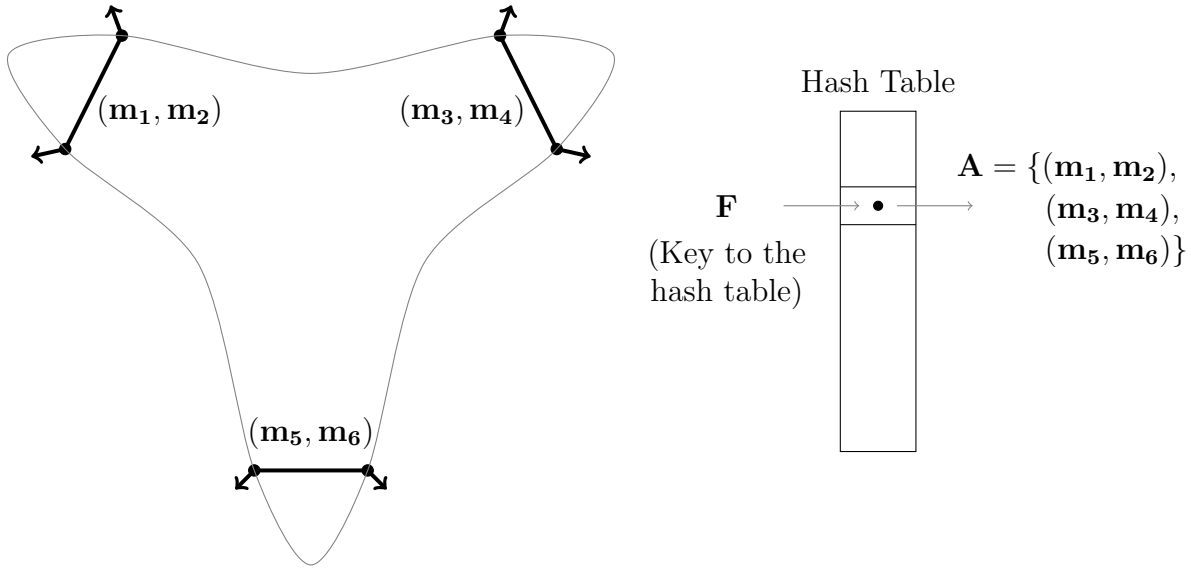


Figure 4: Point pairs forming the similar point-pair features are stored in the same slot in the hash table.

4.3 Local matching

In the on-line phase a reference point s_r selected from the scene is assumed to lie on the detected object. If the assumption is correct there is a corresponding point m_r in the model.

If these two points and their normals are aligned and the object is rotated around the normal of s_r by angle α, we obtain a transformation from the model space to the scene space. The pair (m_r, α) which describes this transformation is called the *local coordinates* of the model with respect to the point s_r. The transformation between the model and the scene coordinates is defined as

$$s_i = T_{s\to g}^{-1} R_x(\alpha) T_{m\to g} m_i. \tag{2}$$

The model point pair $(m_r, m_i) \in M^2$ is aligned with the point pair $(s_r, s_j) \in S^2$ in the scene. The transformation is described in detail in Figure 5.

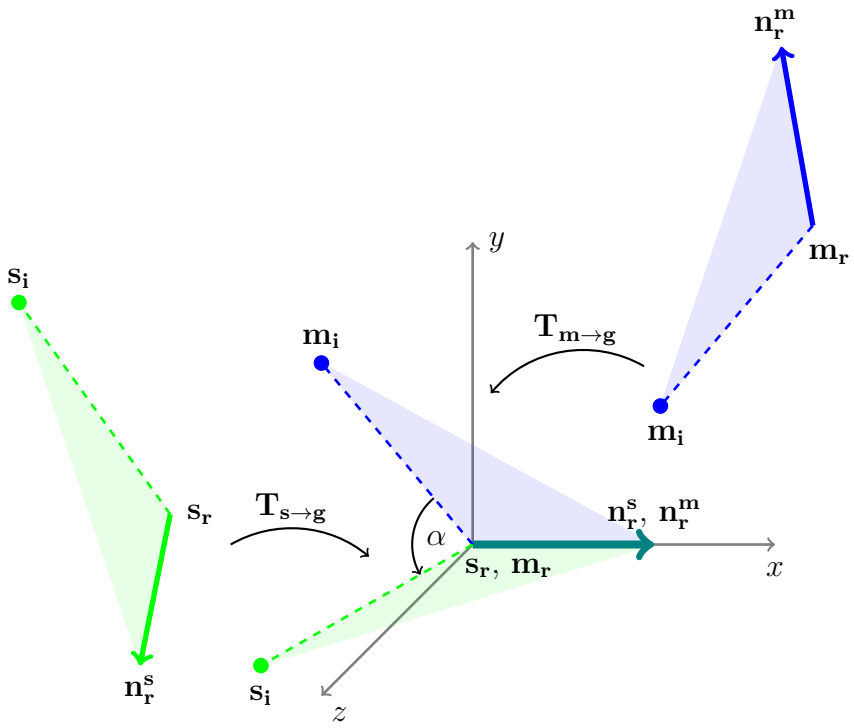


Figure 5: The transformation between the model and scene coordinates. The model reference point m_r is transformed to the origin by the transformation $T_{m\to g}$ and its normal n_r^m is rotated to the x-axis. Similarly, the scene reference point s_r is transformed to the origin by $T_{s\to g}$ and then its normal n_r^s is rotated to the x-axis. Combining the transformations $T_{m\to g}$, $T_{s\to g}$ and the rotation $R(\alpha)$ we obtain the transformation from the model to the scene space.

For the acceleration of the voting process it is possible to decompose the transformation to the two parts. The rotation α is split to two components, the rotation from the model space to the intermediate coordinate system α_m and the rotation from the scene space to the intermediate coordinate system α_s so that $\alpha = \alpha_m - \alpha_s$. Therefore the angles α_m and α_s depend only on the point pair on the model and scene respectively. Thus the rotation can be split to the $R_x(\alpha) = R_x(-\alpha_s)R_x(\alpha_m)$ and then used to obtain

$$\mathbf{t} = R_x(\alpha_s)T_{s\to g}s_i = R_x(\alpha_m)T_{m\to g}\mathbf{m_i} \in \mathbb{R}\mathbf{x} + \mathbb{R}_0^+\mathbf{y}. \tag{3}$$

$\mathbf{t}$ lies on the half-plane defined by x-axis and the non-negative part of y-axis, and is unique for any point pair in the model or scene so the angle α_s can be pre-calculated for

all point pairs on the model in the off-line phase and stored in a global model descriptor. The angle α_s is calculated once for each scene point pair in the on-line phase. The final angle α is a difference of these two angles.

4.4 Voting scheme

The voting scheme is based on the idea that we try to find the best local coordinates for a specific point s_r to maximize the number of point in the scene which lies on the model. This is done in the way that all hypotheses vote in the accumulator. When there are optimal local coordinates found the global pose of the object can be computed.

The accumulator is a two dimensional array where the rows correspond to the points in the model and the columns to the sampled rotation angles α. So the number of the rows N_m is equal to the number of the model points $|M|$ and number of columns N_{angle} is the number of sample steps of the rotation angle α.

During the voting all other scene points $s_i \in S$ are paired with a selected scene reference point s_r. For each point pair the feature vector $F_s(s_r, s_i)$ is computed and then used as the key to the hash table of the global model to obtain possible corresponding model point pairs (m_r, m_i). The rotation angle α is computed using Equation 2. The obtained model reference point m_r and the corresponding computed rotation angle α form a local coordinates (m_r, α) which maps (m_r, m_i) to (s_r, s_i). The local coordinates are used as the index to the accumulator to cast the vote for that certain hypothesis.

When all points s_i are processed the accumulator is searched and a set of the local coordinates with the highest number of votes is returned. Each of the returned local coordinates is used to calculate a transformation from the model to the scene space and therefore to obtain the global coordinates of the object in the scene.

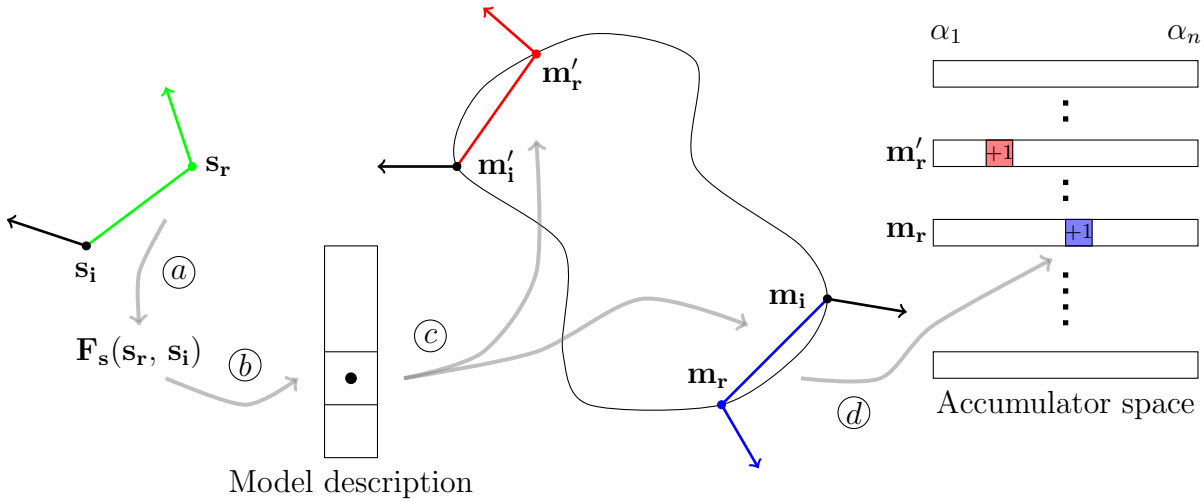


Figure 6: Visualization of the voting scheme. (a) The scene reference points s_r are paired with all other points in the scene s_i, their point-pair feature F_s is calculated, hashed and used as the key to the global model descriptor (b). The corresponding set of model point pairs which have similar distance and orientation is obtained from the global model description (c). (d) The local coordinate α is calculated for each of them and then used to cast the vote (m_r, α) into the accumulator.

4.5 Pose clustering

In the voting scheme it is assumed that the scene reference point s_r lies on the detected object. This is not always true therefore it is necessary to perform voting with several different scene reference points s_r. Each voting results in one possible object pose supported by a certain number of votes. The pose clustering filters out the incorrect poses and increases the accuracy of the estimated pose.

All obtained poses are clustered in the way that poses with similar position and rotation are grouped together. This means that poses whose distance is smaller than some threshold, e.g. 1/10th of the diameter of the object, and whose rotation does not differ more than a certain threshold angle (e.g. $\frac{2\pi}{30}$ rad).

Poses in each cluster are averaged and the score of the cluster is the sum of votes from all poses in the cluster. The clusters with the highest scores contain the most supported poses.

4.6 Hypotheses refinement

When the poses from the clusters with highest number of votes are obtained the author [12] suggests the usage of the pose refinement. For example, the ICP algorithm [39] could further improve the detection rate and the accuracy of the detected poses since the precision of the poses from previous steps is limited by the selected sampling steps.
As well as the author we also decided not to implement this step.

5 Proposed improvements

In this section, we will present the proposed improvement of the algorithm and its implementation. Each improvement is explained and the comparison of partial results is shown. The overall comparison of all improvements is presented in Section 7.

5.1 Restricted selection of pairs of scene points

In the on-line phase of the algorithm the selected reference point is paired with all other points of the scene to create point-pair features. Due to the fact that typical scenes are larger than objects to be detected it is clear that not all the point pairs in the scene can lie on the detected object.

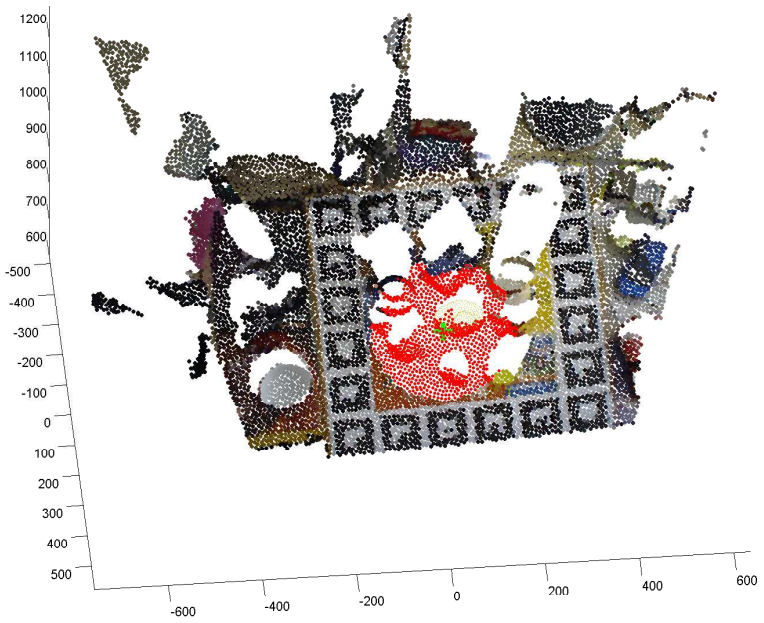


Figure 7: The example of the restricted selection of pairs of scene points. The reference point is indicated by the green cross, the points paired with the reference point are red.

Assuming that the selected reference point lies on the model the points whose distance from the reference point is bigger than the diameter of the object to be detected cannot be part of the object. Therefore it is unnecessary to process them. Note that for the actual implementation the diameter of the axis-aligned bounding box is used. In the case of more

objects to be detected simultaneously with a single detector the diameter of the largest object is used.

Pairing the reference point only with points whose distance is smaller than the diameter of the object bounding box helps to reduce the false votes from the point-pair features which cannot exist on the model, i.e. the case when the incorrect correspondence is obtained from the hash table due to the hash collision.

However, the main advantage is the reduction of the time needed to search the scenes. This is especially true for the large scenes. The example of the reduced search region is shown in Figure 7. The comparison of the original approach and the applied improvement is shown in Figure 8.

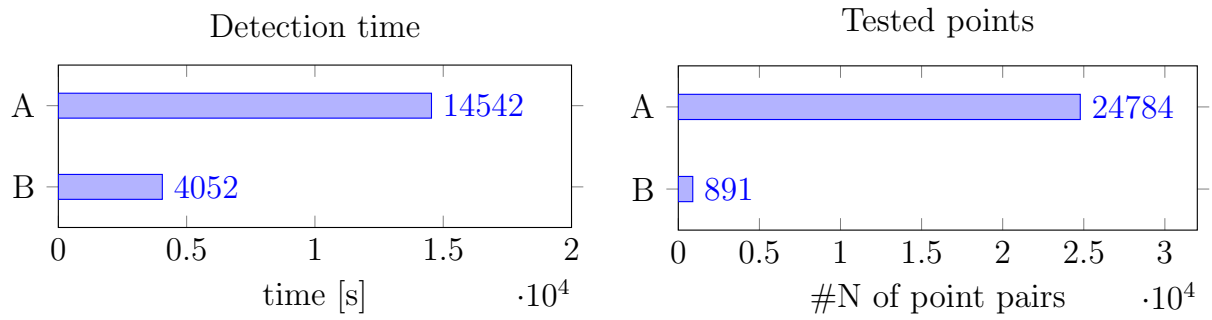


Figure 8: The comparison of the running time and the number of tested point pairs for each reference point. The result is the average of the first 50 scenes from the Hinterstoisser's dataset (duck object sequence). A and B denote the Point-pair feature method and the improvement respectively.

5.2 Weighted voting

The point-pair feature consists of four elements three of which are based on the directions of the surface normals. Therefore to obtain discriminative point-pair features for the successful object pose description, the sufficient variance in the surface normal directions is crucial.

However, the typical scenes in which the objects are detected often consist of the large planar surface (table) and several smaller objects. The dominant surface has all surface normals parallel and therefore generating very similar point-pair features, i.e. the dot product of the respective normals is close to one. The histogram of the angles between the pairs

of the normals in the scenes is shown in Figure 10.

In case the model of the detected object also contains a part with the flat surface a very large number of votes is received for this parts of the surface, which practically overweights the rest of the votes. This leads to many incorrect hypotheses of the objects in the areas of large flat surfaces in the scene. The example is shown in Figure 9.

(a)

(b)

Figure 9: The example of the propagation of the incorrect poses in the scene from the Hinterstoisser's dataset (duck object sequence). The Figure 9a shows the scene with objects placed according to the three best hypotheses. Note that the objects are detected wrongly upside down under the table. The same scene with marked points which voted for the particular poses is shown in Figure 9b. The red circles denote the reference points and the green ones the paired scene points.

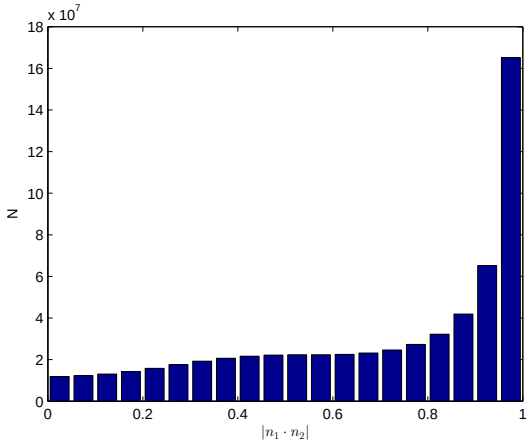


Figure 10: The histogram of the absolute value of the dot product between all pairs of normals. The result is the cumulative histogram of the first twenty scenes corresponding to the four tested objects in the Hinterstoisser's dataset.

To tackle this issue we propose the weighted voting. Instead of each feature having the equal vote in the voting process the vote is computed based on the angle between the normals of the particular point-pair feature. The parallel normal vectors are considered as the least informative ones. On the other hand normals which are perpendicular are considered to be the most descriptive ones.

The vote value is computed as

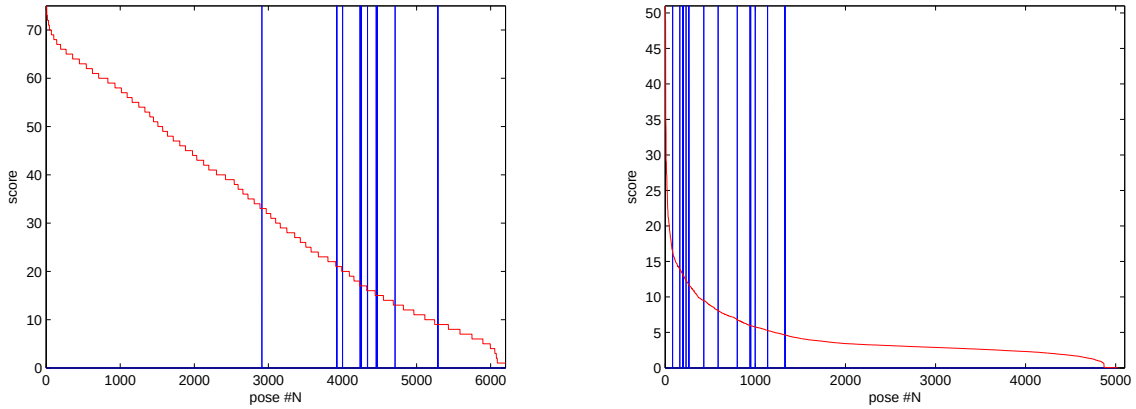
$$\text{vote} = 1 - \lambda \left| n_1 \cdot n_2 \right|$$

where $|n_1 \cdot n_2|$ denotes the absolute value of the dot product of two normal vectors and λ is a weighting parameter in default set to $\lambda = 1$. The value of the vote is defined in range $\langle 0, 1 \rangle$, so that for the parallel vectors it is approaching zero and for the perpendicular ones is close to the 1.

The comparison of the poses computed by the Point-pair feature method and the ones computed by the improved version are presented in Figure 11. In the Point-pair feature method the correct poses were outweighed on the other hand with the improvement applied the correct hypotheses were among the ones with the highest vote value.

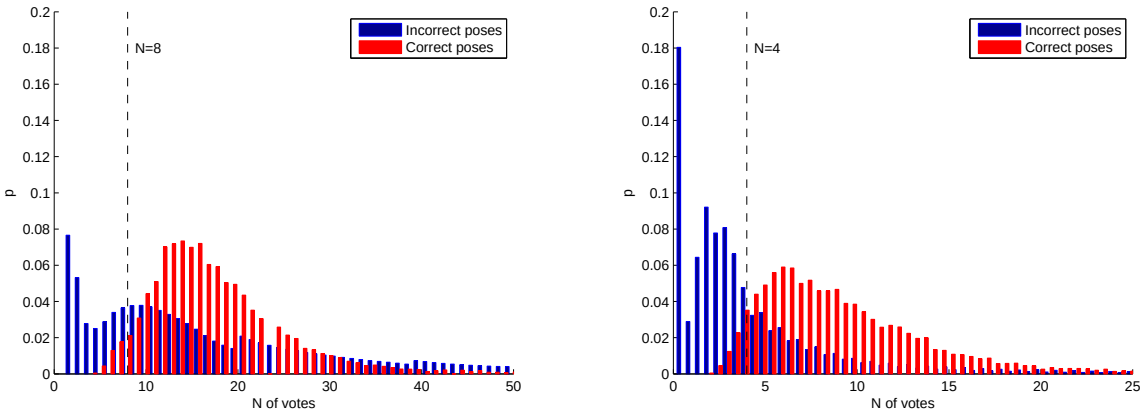
This is shown in a greater detail in Figure 12 which shows the distribution of the

correct and incorrect poses according to the number of votes of the particular pose. The weighted voting allows better separation of the distributions of the correct and incorrect pose hypotheses.



(a) Point-pair feature method. (b) Weighted voting applied.

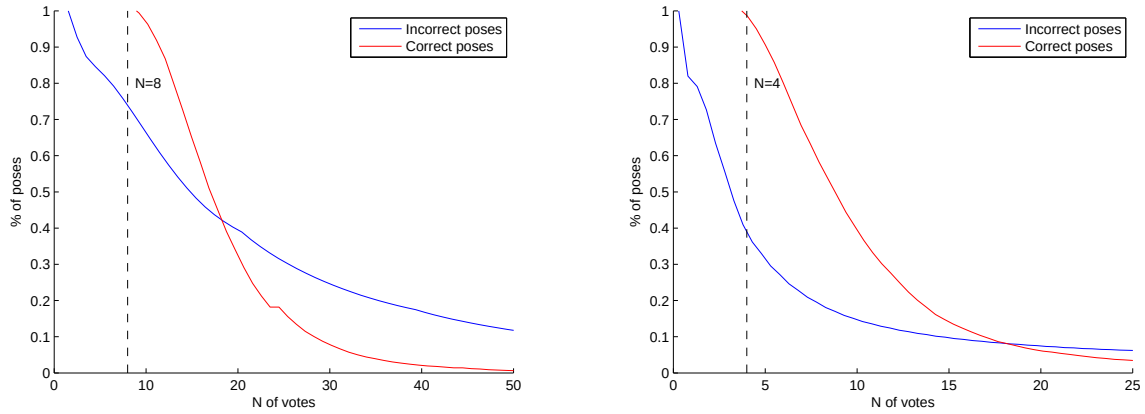
Figure 11: The comparison of the computed poses. The graphs show all poses sorted according to their vote value in the descending order. The blue bars indicate the correct poses (with respect to the ground truth) and the red line is the score of each pose. The results are for the first scene of the duck object sequence in the Hinterstoisser's dataset, however other scenes produce similar results.



(a) Point-pair feature method. (b) Weighted voting applied.

Figure 12: The distribution of the correct and incorrect poses according to the number of votes of the particular pose. The dashed line shows the proposed threshold. The result is the cumulative histogram for four objects and their respective scenes from the Hinterstoisser's dataset.

The relative number of the correct and incorrect pose hypotheses is presented in Figure 13. The proposed threshold between the correct and incorrect hypotheses is $t_h = 8$ for the Point-pair feature method and $t_h = 4$ when the weighted voting is applied. The threshold for the weighted voting is a half of the original one since we assume that the expected weighted vote is 0.5.
The improved separation of the correct and incorrect hypotheses distributions allows better filtering of incorrect poses based on vote value. In case of the Point-pair feature method, removing 28% of the incorrect pose hypotheses removes also 3.5% of the correct ones. However, when the weighted voting is applied, it is possible to remove 63% of incorrect hypotheses while only 4% of the correct ones are lost.
On condition that the correct pose hypotheses account for only about 0.15% of all poses the possibility to remove more than a half of all hypotheses allows to speed up following stages significantly.



(a) Point-pair feature method.

(b) Weighted voting applied.

Figure 13: Dependence of the fraction of correct and incorrect hypotheses on the selected threshold value.

5.3 Reference point selection

One of the proposed improvements is the enhanced selection of the reference points pairs. In the on-line phase of the Point-pair feature method the reference points are selected randomly from the set of all scene points. This approach leads to the sparse distribution of the reference points in the scene. Therefore if the object to be detected is occupying only

a small portion of the scene only a few if any points selected as the reference are actually located at the detected object itself.

To tackle this issue we aim to prefer the selection of the reference points located in the interesting parts of the scene, i.e. the parts which have a sufficient variance in a surface curvature. We suggest to evaluate the surface of the scene and assign the measure describing the local curvature of the surface to each point in the scene. The measure used to evaluate the surface is the shape index.

The shape index S_I was proposed by Koenderink et al. [23]. It represents a local surface topology at point $\mathbf{p}$ so that

$$S_I(\mathbf{p}) = \frac{2}{\pi} \arctan \frac{k_2 + k_1}{k_2 - k_1} \qquad k_1 \geq k_2$$

where k_1 and k_2 are the principal curvatures. The range of the shape index values is $[-1, 1]$, in case of the plane ($k_1 = k_2 = 0$) is not defined but usually assigned to be zero. The Figure 14 shows the illustration of the shape index value for the canonical shapes of the surfaces.

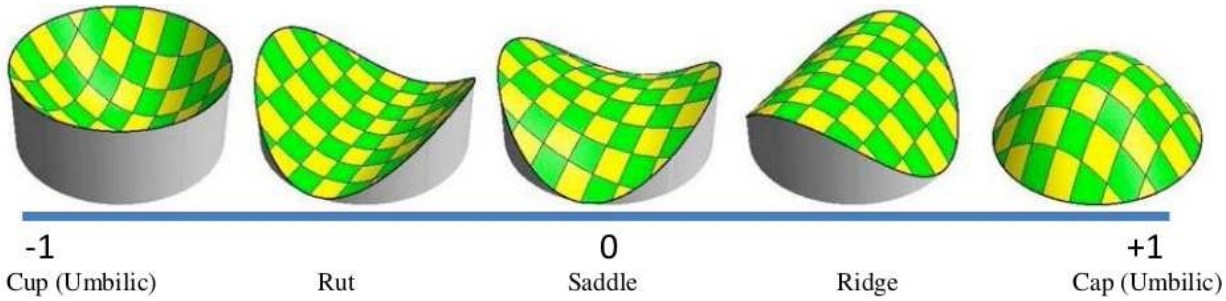


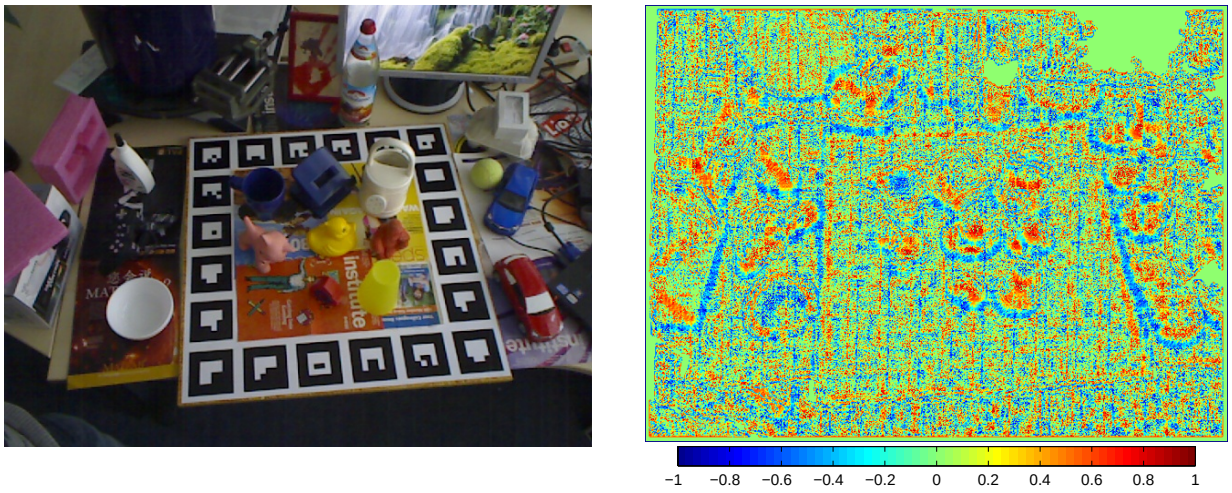
Figure 14: Illustration of the shape index for various surfaces. Image courtesy [23]

The shape index for the point $\mathbf{p}$ can be also expressed in form of its surface normal $\mathbf{n}$ as:

$$S_I(\mathbf{p}) = \frac{2}{\pi} \arctan \frac{\left(\frac{\partial \mathbf{n}}{\partial x}\right)_x + \left(\frac{\partial \mathbf{n}}{\partial y}\right)_y}{\sqrt{\left(\left(\frac{\partial \mathbf{n}}{\partial x}\right)_x - \left(\frac{\partial \mathbf{n}}{\partial y}\right)_y\right)^2 + 4\left(\frac{\partial \mathbf{n}}{\partial x}\right)_y \left(\frac{\partial \mathbf{n}}{\partial y}\right)_x}}$$

The shape index is computed for each point in the depth image so that the derivative of the surface normals is approximated by the difference of the neighbouring normals. We are interested in the surfaces which have a sufficient variance in a surface curvature. Therefore, the absolute value of the shape index would indicate which points in the scene are likely to be selected as the reference point. The shape index value of one represents highly curved and therefore the most interesting parts of the surface. On the other hand the shape index value of zero (i.e. the plane) is considered the least descriptive part of the surface and thus the least likely to get selected as the reference point.

The Figure 15 shows the example of scene from the Hinterstoisser's dataset with points coloured according to the computed shape index. The resulting shape indices suffer of a high noise of the depth map. This could be reduced by a suitable filtering of the shape index map. However due to a tight schedule, these ideas have not been further explored within this thesis and could be a subject of further work.



(a) The original scene.

(b) Computed shape index map.

Figure 15: The example of the computed shape index for the scene from the Hinterstoisser's dataset.

5.4 Matching score

Inspired by the Dense refinement step from the MVTec HALCON implementation of the Point-pair feature method (see Section 6.3) the matching score calculation is applied. When all poses are calculated and clustered the matching score of each pose is calculated

so that it counts the amount of the surface of the model visible in the scene.

The neighbourhood of each point in the sub-sampled model is searched whether it contains a points from the scene. The matching score represents the fraction of the model points which are near some of the scene points. The diameter of the searched neighbourhood is defined as the scene sampling step therefore the minimal distance between points in the scene.

5.5 Detection pipeline

Based on the individual improvements presented in previous sections we propose a three-stage detection pipeline. A simple schematics of the pipeline is depicted in Figure 16.

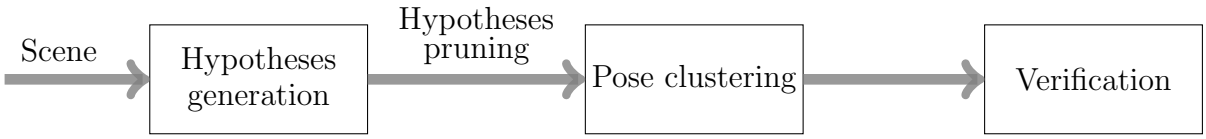


Figure 16: Detection pipeline.

Hypotheses generation The first stage of the pipeline the hypotheses generation is based on the Point-pair feature detection method proposed by Drost et al. [12]. This method is enhanced using the previously introduced improvements at first the restricted selection of pairs of scene points in Section 5.1 and then the weighted voting in Section 5.2.

The result of this step is a set of possible pose hypotheses. As it was also described in Section 5.2, the set of the pose hypotheses from this stage is filtered so that the ones with the vote value smaller than the threshold are removed.

Pose clustering The pose clustering stage clusters the filtered pose hypotheses from the previous step. The poses whose rotation and translation is similar are grouped together into the clusters. The pose hypotheses in each cluster are then averaged and passed to the next stage.

Verification The final stage of the detection pipeline introduces the verification step in which the score of each pose is calculated according to the algorithm presented in Section 5.4 - Matching score. A detailed testing presented in Section 7.2.2 showed that this new matching score is more stable than the original one.

The output of this stage are the pose hypotheses sorted according to the newly calculated matching score.

Pose alignment The next possible step is the pose alignment in which the subset of the best pose hypotheses from the previous stage is refined. This could be done using the ICP algorithm [39]. The influence of the pose alignment was tested in Section 7.2 where the dense pose refinement proved to lead to more accurate pose hypotheses at the cost of the significant increase in the required time for the detection.

6 Implementation

The algorithm has been implemented in the MATLAB® software as the PPF3DDetector object. First the preprocessing of the model and the scene is described then the algorithm pipeline itself.

6.1 Depth data preprocessing

The algorithm requires a point cloud with normals as the input. Therefore it is necessary to pre-process the data which are in a form of the depth image. The point cloud is also sub-sampled at the beginning of the algorithm.

6.1.1 Conversion of depth map to point cloud

The RGB-D images from the depth sensor such as the scenes from the Hinterstoisser's dataset (see Section 7.1.2) are first converted to the form of the 3D point cloud. Each pixel from the depth map is converted so that the coordinates of the corresponding 3D point are

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (i - p_x)\frac{z_{i,j}}{f_x} \\ (j - p_y)\frac{z_{i,j}}{f_y} \\ z_{i,j} \end{bmatrix}$$

where $z_{i,j}$ is the depth corresponding to the row and column indexes i and j respectively. f_x, f_y, p_x and p_y are the camera calibration parameters.

A normal vector for each 3D point is computed from the depth image using the method proposed by the Hinterstoisser et al. [17]. Around each pixel location x, the first order Taylor expansion of the depth function $\mathfrak{D}(x)$ is considered

$$\mathfrak{D}(x + dx) - \mathfrak{D}(x) = dx^T \nabla\mathfrak{D} + \text{h.o.t.}$$

so that the value of $\nabla\mathfrak{D}$ is constrained by the equations yield by each pixel offset dx within patch defined around x. This depth gradient corresponds to a 3D plane going through three points X, X_1 and X_2 from which the normal vector can be estimated as the

normalized cross-product of $X_1 - X$ and $X_2 - X$. We use the implementation provided by the OpenCV library [3].

6.1.2 Point cloud sampling

At the beginning of the detection algorithm the model and the scene are sub-sampled. The sub-sampling is implemented using the algorithm of the *Poisson disc sampling* described by [4]. The sampling process is described in detail in the Algorithm 1. The input is the point cloud `PC` and the input parameter `relativeSampling` which specifies the sampling step relative to the diameter of the object. For example, if the diameter of the object is 10 cm and `relativeSampling=0.05` the sampled points will be approximately 0.5 cm apart (i.e. the parameter `distanceStep=0.5`). The output of the algorithm is the sampled point cloud `sampledPC`.

Algorithm 1 Sample point cloud

Input: `relativeSampling`, $PC = \{(p_1, n_1), ..., (p_N, n_N)\}$
Output: sampledPC

```
distanceStep ← d · relativeSampling
initialize grid G with cell size distanceStep/√3
for i ≤ N_PC do
    p_i ← PC(i)
    in G select cells C in 5 × 5 × 5 neighbourhood of p_i
    select all points p_j from C
    if ‖p_i p_j‖ < distanceStep then
        G ← p_i
    end if
end for
sampledPC ← ∀p ∈ G
```

6.2 Detector

The detector is first initialized and parameters `angleBins` and `relativeSampling` are set. The parameter `relativeAngle` specifies into how many bins the angles are quantized. The default value is 30 meaning that each bin is $\frac{2\pi}{30} = 0.2094\,\text{rad}$.

6.2.1 Training

In the training phase the point cloud of the model PC_m is loaded and sub-sampled. The point-pair features for all possible point pairs on the model are computed. The hash computed for each point-pair feature serves as an index into the specific slot in the hash table in which the indexed of the points which form that particular feature are stored. The precomputed angle α_M is store in the same slot as well. For detailed description, see Algorithm 2.

Algorithm 2 Train model

Input: `relativeSampling`, `angleBins`, $PC_m = \{(p_1^m, n_1^m), ..., (p_N^m, n_N^m)\}$
Output: Hashtable

```
Hash table ← {∅}
PC_m ← samplePoisson(PC_m, relativeSampling)
distanceStep ← d_m · relativeSampling
for i ≤ N_m do
    p_1 ← PC_m(i)
    for j ≤ N_m do
        if i ≠ j then
            p_2 ← PC_m(j)
            f ← computePPF(p_1, p_2)
            hash ← hashPPF(f, angleBins, distanceStep)
            α_m ← computeAlpha(p_1, p_2)
            Hashtable ← node(hash, i, j, α_m)
        end if
    end for
end for
```

6.2.2 Detection

The next part is a detection. The scene PC_s is loaded and sampled. Sampling is controlled with the parameter `sceneSampling` which controls the distance between sampled points and is given relative to the diameter of the model. The sampling step parameter `sceneDistanceStep` is computed as $d_m \cdot$ `sceneSampling` where d_m is the diameters of the model.

By default the scene sampling depends on the diameter of the model. Drost et al. [12] does not discuss the possibility of more objects detected simultaneously. The default ap-

proach is to use an independent detector for each object. Another possibility is to add an object identifier into the hash table slots in the training phase so that global models of multiple objects can be stored in the same hash table and still be distinguishable. This would also require a separate voting accumulator for each object to be detected.

In this case the scene sampling step would be independent of the model diameter and instead of that would be specified relative to the scene diameter so that `sceneDistanceStep` would be calculated as $d_s \cdot$ `sceneSampling`.

Firstly the reference point is selected in the scene, i.e. the scene is tested sequentially and every n-th point is selected as the reference. The fraction of the points selected as the reference is controlled by the parameter `sceneFraction` (typically, 1/5th or 1/10th of the scene points is used). For each selected reference point, the point-pair features are computed by combining the reference point and all other points in the scene. The feature vectors are hashed and their hashes used as the keys to the precomputed hash table. The angle α_s is computed for each point pair in the scene and together with the model reference point i_m obtained from the hash table serves as the index into the voting accumulator.

Once all hash table slots corresponding to all computed features cast the vote the accumulator is searched for maximum m_v. Hypotheses with more votes than `maxCoef` $\cdot\, m_v$ are returned. Parameter `maxCoef` is usually set to 0.95. For each returned hypothesis, the respective pose is computed and stored.

Once all reference points are processed the resulting poses are clustered and poses in each cluster are averaged. The number of votes in each cluster is the sum of the votes from all poses in the cluster. The clusters are sorted according to the number of the votes and the first (the one with most votes) is returned. Whole recognition pipeline is depicted in Algorithm 3.

Algorithm 3 Detection

Input: `sceneSampling`, d_m , $PC_s = \{(p_1^s, n_1^s), ..., (p_N^s, n_N^s)\}$

Output: FinalPoses

```
poseList ← {∅}
sceneDistanceStep ← (d_m · sceneSampling)/d_s
PC_s ← samplePoisson(PC_s, sceneDistanceStep)
for i ≤ N_s do
    accumulator ← {∅}
    p_1 ← PC_s(i)
    for j ≤ N_m do
        if i ≠ j then
            p_2 ← PC_s(j)
            f ← computePPF(p_1, p_2)
            hash ← hashPPF(f, relativeAngle, distanceStep)
            α_s ← computeAlpha(p_1, p_2)
            nodes ← Hashtable(hash)
            while nodes ≠ 0 do
                node ← nodes
                i_m ← node
                α_m ← node
                α ← α_m − α_s
                accumulator{i_m, α} ← accumulator{i_m, α} + 1
            end while
        end if
    end for
    peaks ← max(accumulator, maxCoef)
    while peaks ≠ 0 do
        peak ← peaks
        T_s ← computeRT(p_1)
        T_m ← computeRT(PPF(i_{m,peak}))
        newPose ← computePose(T_s, T_m, α_peak)
        poseList ← appendPose(newPose)
    end while
end for
poseClusters ← clusterPoses(poseList)
FinalPoses ← averageClusters(poseClusters)
```

6.3 Commercial implementation

To compare the results of our implementation the data were tested on the commercially released implementation of the Point-pair feature method. The algorithm is part of the HALCON software tool produced by the MVTech [14].

This implementation features three steps. The first one is the actual detection algorithm, the other two are optional refinements:

Approximate matching searches the scene for the instances of the model. The algorithm is based on the Point-pair feature method proposed by Drost et al. [12].

Sparse pose refinement refines the approximate poses found in the previous step. The pose is optimized so that the distances from the sampled scene points to the plane of the closest model point are minimal. The plane of each model point is defined as the plane perpendicular to its normal.

After the Sparse pose refinement the score of each pose is recomputed so that it is a percentage of the points on the model which have corresponding points in the scene.

Dense pose refinement is the last step which is used to accurately refine the poses from the previous steps. Similarly to the previous step it minimizes the distances between the scene points and the planes of the closest model points, but it uses the original (not sub-sampled) model.

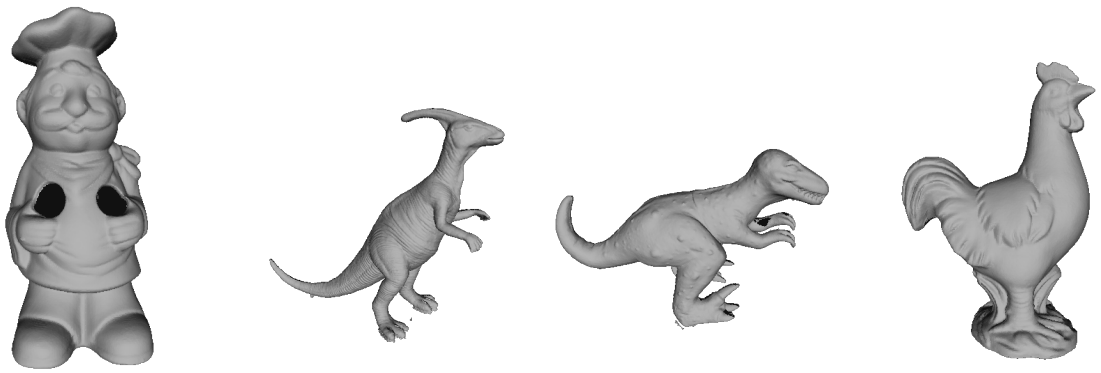
7 Experiments

The datasets used to test and compare our implementation and proposed improvements are shortly introduced. The detection results with the commercial implementation of the Point-pair feature method are presented. Next the detection results of our implementation and proposed improvement are shown and compared.

7.1 Datasets

7.1.1 Mian's dataset

First the Mian's dataset [27] consisting of the fifty scenes scanned with the Minolta Vivid 910 scanner and saved as the 3D point cloud in the PLY file. Scenes feature five partially occluded objects which are placed variously among the scenes. The ground truth and the percentage of the occlusion is available. To allow direct comparison with the original paper only four objects - *Chef*, *Parasaurolophus*, *T-rex* and *Chicken* were used. The detected objects are depicted in Figure 19.



(a) Chef (b) Parasaurolophus (c) Rex (d) Chicken

Figure 17: Objects from the Mian's dataset.

The example of the scene from the Mian's dataset, the sub-sampled version and the detection results are shown in Figure 18.

(a) Original scene. (b) Sub-sampled scene. (c) Scene with detected objects. Correct poses in green, incorrect in red.

Figure 18: Example scene from the Mian's dataset.

7.1.2 Hinterstoisser's dataset

Another dataset used to test and compare the algorithm is provided by Hinterstoisser [20]. It consists of the 15 objects with their 3D meshes and more than 18000 RGB-D images of the real scenes where these objects are placed. Scenes also include many other objects which cause significant background clutter. The examples of scenes from the dataset are shown in Figure 20.

For our testing we selected four objects and their respective RGB-D image sequences - *Duck*, *Lamp*, *Driller* and *Bench vise*. Object models are depicted in Figure 19.

(a) Duck (b) Lamp (c) Driller (d) Bench vise

Figure 19: Objects from the Hinterstoisser's dataset.

Figure 20: Example scenes from the Hinterstoisser's dataset.

7.2 Evaluation of the commercial implementation

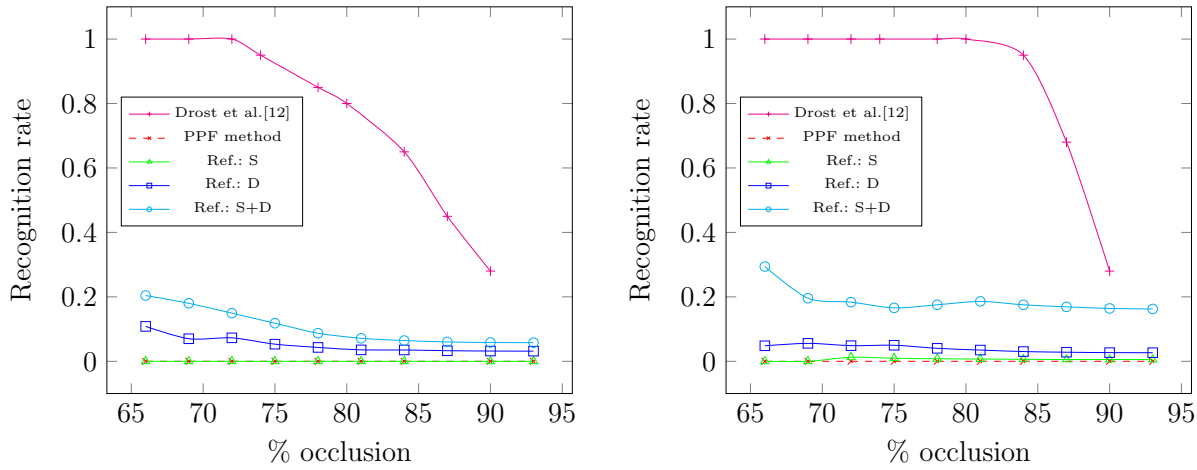
The commercial implementation of the Point-pair feature method presented in Section 6.3 is tested on both datasets.

7.2.1 Results on the Mian's dataset

The MVTec HALCON software was first tested on the Mian's dataset, the settings were chosen so that the parameter `relativeSampling` is set to $\tau = 0.04$ and $\tau = 0.025$ with $\frac{|S|}{10}$ and $\frac{|S|}{5}$ reference point selected respectively - during the detection phase every 1/5th or 1/10th point in the scene is selected as the reference point. This choice allows direct comparison with the results published by the Drost et al. [12].

Both settings were tested on several variants of the detection pipeline consisting of the Point-pair feature method and either activated or deactivated refinement steps. Additionally, the scenes normals were recomputed using the moving least squares method [24] and then tested again. The normal re-computation was suggested in the personal communication with Bertram Drost.

The threshold for the correctly detected object is defined the same as in the original paper [12] therefore $\frac{d_M}{10}$ is the threshold for the translation and $\frac{2\pi}{30}$ rad for the rotation. So the detected pose is considered to be correct if the translation and rotation between the estimated pose and the ground truth pose is smaller than 1/10th of the model diameter and 0.2094 rad respectively.



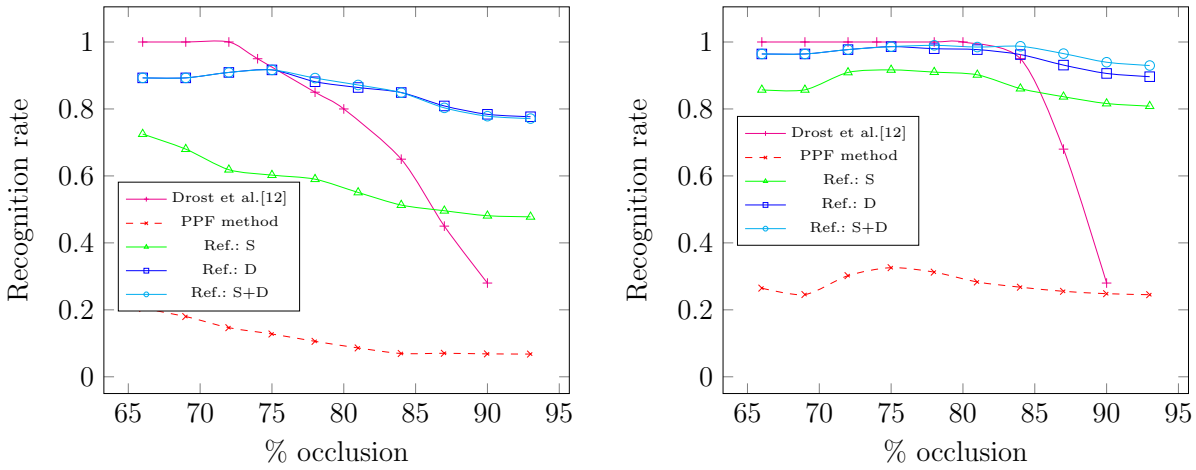
(a) $\tau = 0.04$, $\frac{|S|}{10}$ (b) $\tau = 0.025$, $\frac{|S|}{5}$

Figure 21: The detection results on the Mian's dataset according to the percentage of the occlusion. The detection rate is the average for all 50 scenes and 4 tested objects. Note that *PPF method* stands for the Point-pair feature method, *S* for the applied *Sparse refinement*, *D* for the *Dense refinement* and the *S+D* for the *Sparse and Dense refinement* used together.

The average detection rate for all 50 scenes and four tested objects is depicted in Figure 21. The Point-pair feature method failed to detect any objects in the scenes for both sampling settings. When both refinements were enabled the detection rate for objects with less than 84% occlusion is 8.4% and 17.5% for sampling rate $\tau = 0.04$ and $\tau = 0.025$ respectively. The results reported by Drost et al. [12] with the detection rate 89.2% and 97.0% respectively clearly outperforms even when both refinements are enabled.

The result for the scenes with the recomputed normals is shown in Figure 22. In case of the sampling rate $\tau = 0.04$ the Point-pair feature method was again surpassed by the reported detection rate [12] comparable results are achieved only when the Dense refinement was enabled.

For the sampling rate $\tau = 0.025$ the results with enabled Dense refinement are comparable to the reported detection rate, for the higher occlusion values even outperform.



(a) $\tau = 0.04$, $\frac{|S|}{10}$ and MLS normals.

(b) $\tau = 0.025$, $\frac{|S|}{5}$ and MLS normals.

Figure 22: The detection results on the Mian's dataset according to the percentage of the occlusion for scenes with recomputed normals. The detection rate is the average for all 50 scenes and 4 tested objects.

The Figure 23 shows the average time needed to detect one object instance in one scene for various detection settings. Note the increase of the required detection time when the Dense refinement is enabled.

Comparing the Figures 21 and 22 it is clear that the scenes from the Mian's dataset require a re-computed normal vectors as Bertram Drost suggested. This is likely due to the fact that objects and scenes have wrinkled surfaces with frequency higher than the sampling step. This leads to the unstable normal directions when the sampling is performed. Therefore it is necessary to sample the data and then recompute the normal vectors.

Nevertheless the detection results show that contrary to the results reported by the Drost et. al. [12] the Point-pair feature method (without any refinement applied) is unable to detect objects at a sufficient detection rate comparable to the reported one.

By enabling the Sparse and Dense refinement, the detection rate is significantly increased. Although as mentioned before the Dense refinement step increases the detection time enormously (see Figure 23). The Sparse refinement step appears to be a compromise between the detection rate and the required time.

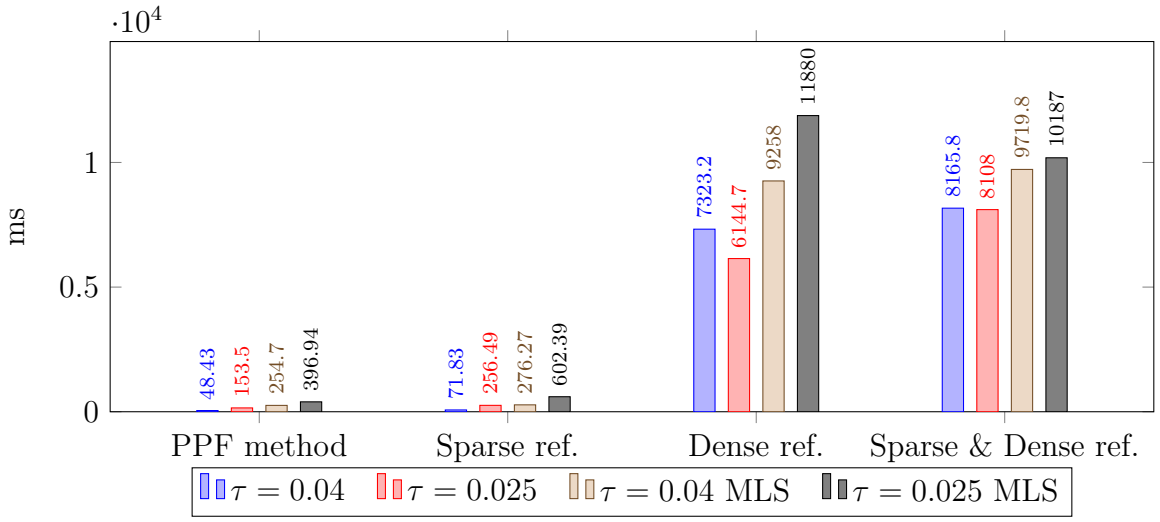


Figure 23: Time comparison of the Point-pair feature method and applied refinements on the Mian's dataset.

7.2.2 Results on the Hinterstoisser's dataset

The testing for the Hinterstoisser's dataset was performed on four objects and their scene image sequences. The result for each object is the average detection rate among their first 1000 scenes. For the comparison Hinterstoisser et al. [20] reported the average detection rate for each object in the dataset when using the Point-pair feature method proposed by Drost et al. [12]. Although no further information including the object and scene sampling rate is provided.

The comparison of detection rate according to the refinement used is depicted in Figure 24. Selected model sampling is $\tau = 0.03$, scene sampling $\tau = 0.05$ and $\frac{|S|}{5}$ of points in the scene is selected as the reference. The basic version of the algorithm performed poorly, in case of the *Duck* and *Bench vise* objects is the detection rate zero for the other objects up to the 35%. However, when the Sparse refinement is used the detection rate increases to the 54% for the *Bench vise* sequence and even 92% for the *Driller* sequence which is fully comparable with the result reported by Hinterstoisser et al. [20].

The Figure 25 illustrates the influence of the model and scene sampling on detection results for different models. The Figure 25a shows the detection rate with respect to the sampling step for the Point-pair feature method, Figure 25b when the Sparse and Dense re-

finement is applied. Note that $\tau_{M,S} = \{0.03; 0.05\}$ stands for model sampling step $\tau_M = 0.03$ and scene sampling $\tau_S = 0.05$.
The comparison of the detection times for different objects, sampling steps and used refinement is shown in Figure 26.

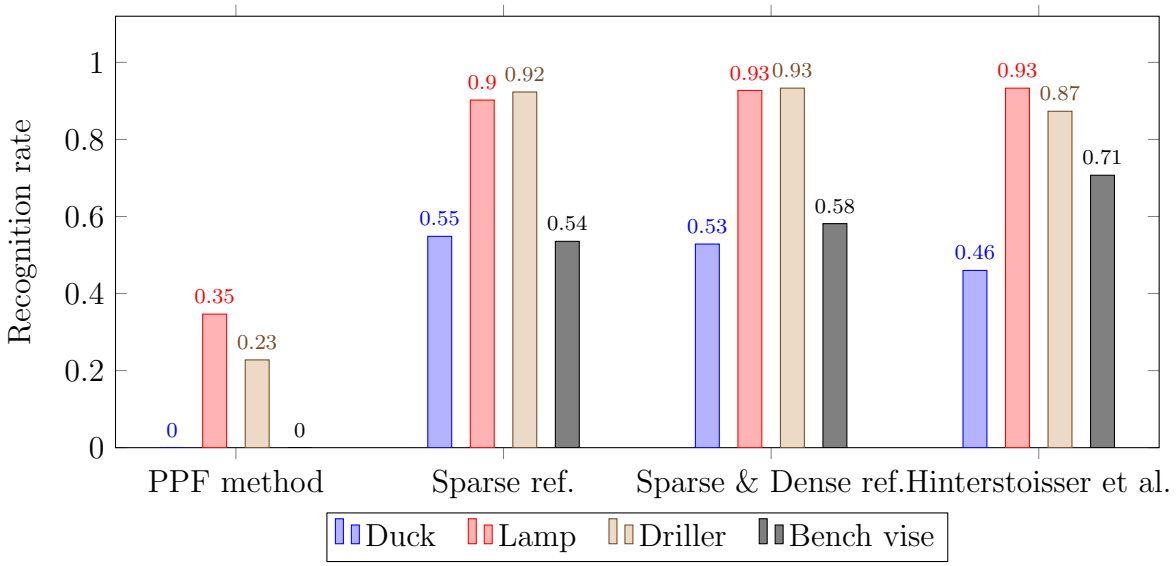
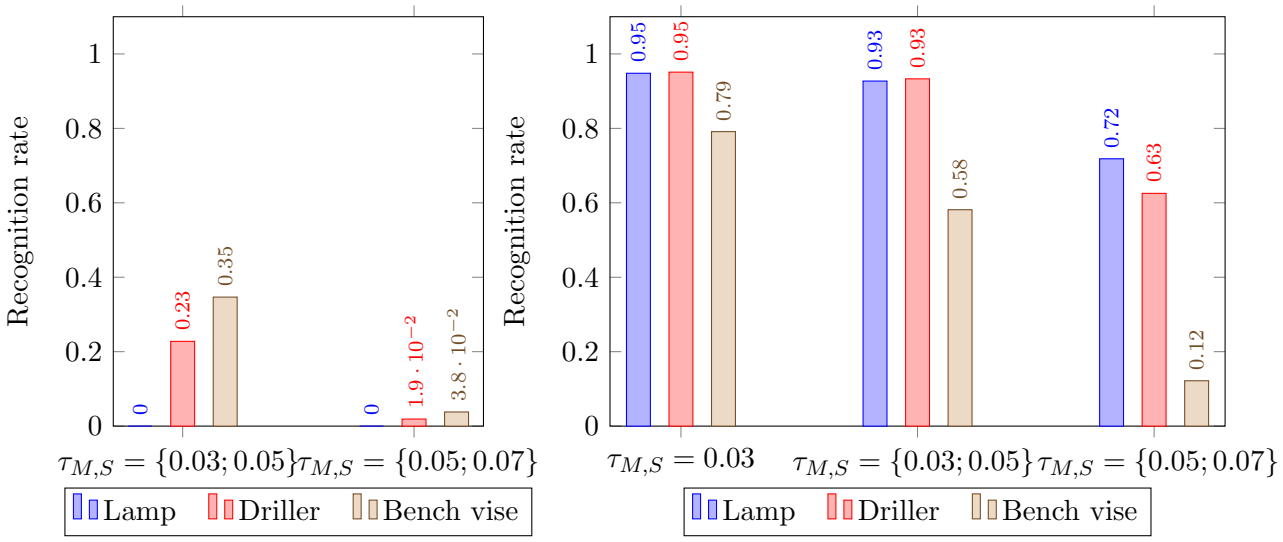


Figure 24: Detection results for the Hinterstoisser's dataset. Model sampling $\tau = 0.03$, scene sampling $\tau = 0.05$ and $\frac{|S|}{5}$. Note that the results for Hinterstoisser et al. are derived from [20].

The detection results are similar to those for the Mian's dataset. The Point-pair feature method (without any refinement applied) again performed poorly for every sampling rate (see Figure 25a). However, the Figure 24 shows that when the Sparse refinement was enabled the detection rate increased significantly and was comparable to the results reported by Hinterstoisser et al. [20].

The difficulty of the scenes in the Hinterstoisser's dataset was already discussed (see Section 5.2) the large flat surfaces with the parallel surface normals cause the generation of many similar point-pair features which leads to the generation of the incorrect pose hypotheses in these regions. The Sparse refinement step in the MVTec HALCON implementation includes the score re-computation which proved to be more stable than the score computed by the Point-pair feature method itself and therefore leading to much better detection rates. The Figure 24 also shows that the influence of the applied Dense refinement

is very small particularly in the light of the increased computation time (see Figure 26).



(a) Point-pair feature method.

(b) Sparse & Dense refinement used.

Figure 25: The comparison of the effect of the sampling step to the detection rate.

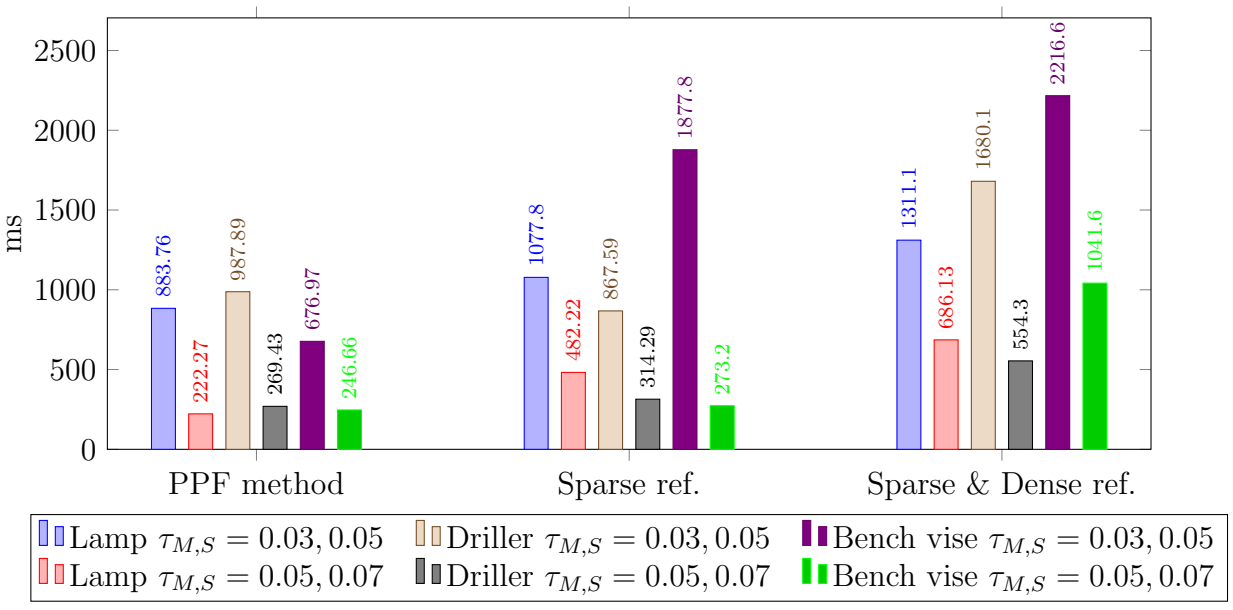


Figure 26: Time comparison of the Point-pair feature method and applied refinements on the Hinterstoisser's dataset.

7.3 Evaluation of our implementation

In this section we will present and discuss the results of our implementation and the proposed improvements. First the Mian's dataset is elaborated then the results on the Hinterstoisser's dataset are presented. Finally we compare the results of our implementation and proposed improvements and discuss the differences to the commercially available implementation.

We test our detection pipeline (see Section 5.5) which includes the proposed improvements namely the restricted selection of pairs of scene points, weighted voting with consequent hypotheses pruning and the matching score calculation.

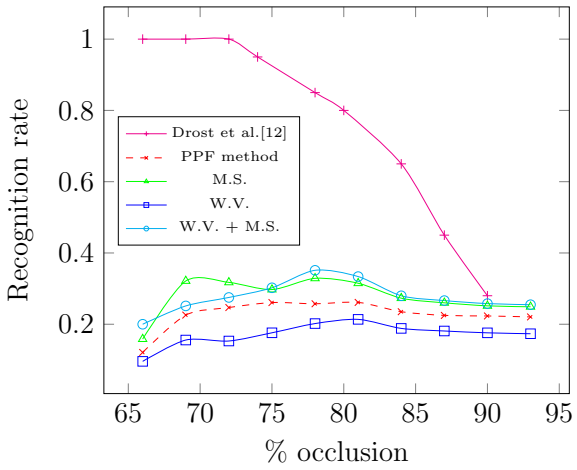
The weighted voting, hypotheses pruning and the matching score calculation are optional steps which can be enabled or disabled. The algorithm is tested in various configurations of these optional steps to show the influence of each individual improvement. When all these improvements are disabled the detection pipeline is equivalent to the Point-pair feature method presented in [12].

7.3.1 Results on the Mian's dataset

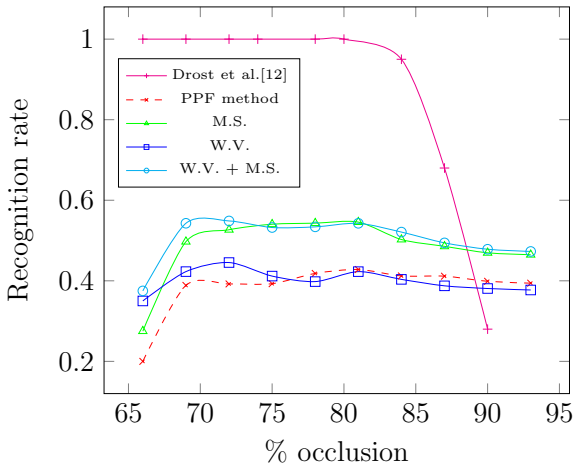
First we will show the result of our implementation and the proposed improvements on the Mian's dataset. Again the setting was chosen to be comparable with the detection rates reported by the Drost et al. [12]. Due to the different implementation of the point cloud sampling the sampling steps are $\tau = 0.025$ and $\tau = 0.016$ which corresponds to the sampling steps $\tau = 0.04$ and $\tau = 0.025$ used in the published results and the commercial implementation. In that way the number of the points in the sub-sampled scene remains constant. Drost et al. [12] reported the average number of point in the sub-sampled scene to be $|S| \approx 1690$ in case of the commercial implementation it is $|S| \approx 1390$ and for our chosen sampling step $|S| \approx 1580$.

The Figure 27 shows the detection results for our detection pipeline and also the corresponding detection rate reported by Drost et al. [12] for both sampling settings. We also tried the approach suggested by Bertram Drost and prior to the detection we sub-sampled the point cloud of the scene and recomputed the normal vectors. This was done manually using the MeshLab software [9] so that we first applied the poison disc sampling and then recomputed normal vectors using the moving least squares method. The detection results

for these pre-computed scenes with various settings of the detection pipeline are shown in Figure 28.

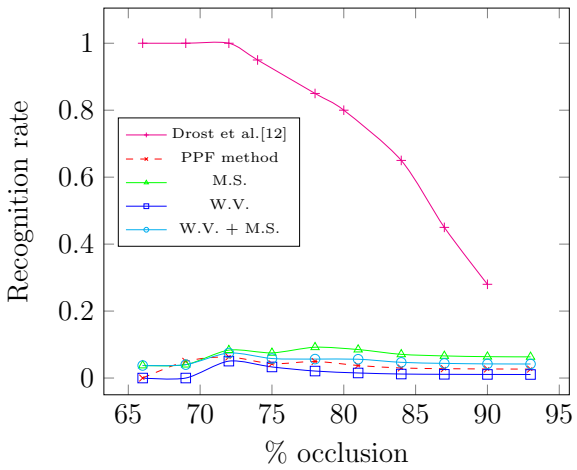


(a) $\tau = 0.025$, $\frac{|S|}{10}$

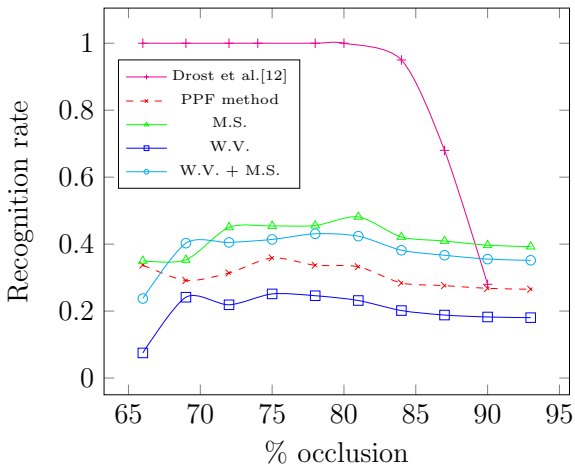


(b) $\tau = 0.016$, $\frac{|S|}{5}$

Figure 27: The detection results on the Mian's dataset according to the percentage of the occlusion. The detection rate is the average for all 50 scenes and 4 tested objects. Note that *PPF method* stands for the Point-pair feature method, *M.S.* for the matching score calculation, *W.V.* for the weighted voting and *W.V. + M.S.* for both improvements applied together. *PPF method* indicated the Point-pair feature method without any optional improvements applied.



(a) $\tau = 0.025$, $\frac{|S|}{10}$ and MLS normals.



(b) $\tau = 0.016$, $\frac{|S|}{5}$ and MLS normals.

Figure 28: The detection results on the Mian's dataset according to the percentage of the occlusion for scenes with recomputed normals. The detection rate is the average for all 50 scenes and 4 tested objects.

In contrast to the results with the commercial implementation the detection rate with recomputed normals is in overall worse than the one with original normals. Giving the results obtained with the commercially available implementation which implements the re-computation of the normal vectors, a better approach to the normal vector re-computation could lead to much superior detection results. The best performance gain is when the matching score calculation is applied. The best detection results are in case of the detection pipeline with weighted voting and matching score calculation enabled. However, the detection rate is still significantly lower than the one reported by Drost et al. [12]. The evaluation of the effect of the sampling step on the detection rate is presented in Figure 29.

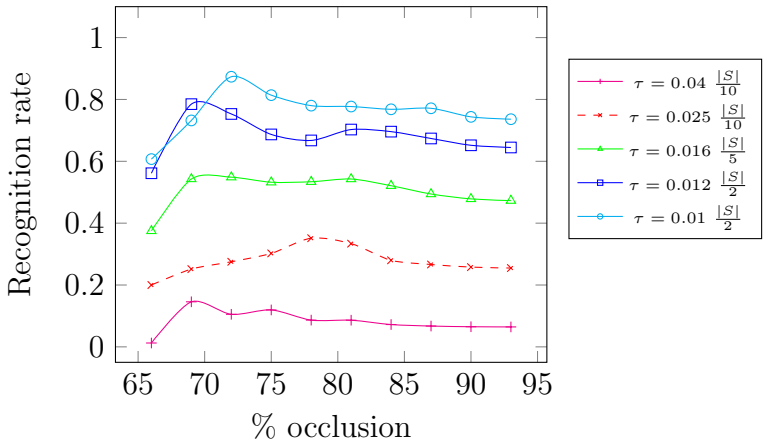
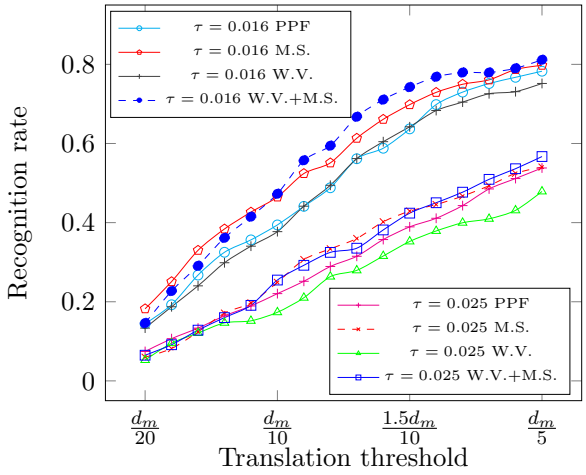


Figure 29: The influence of the sampling step on the detection rate. The weighted voting and matching score calculation improvements are enabled.

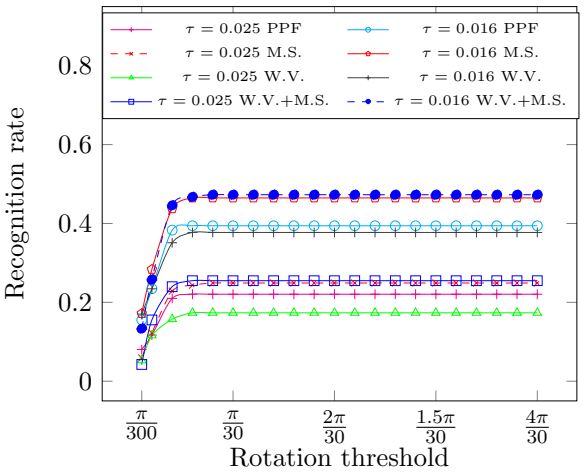
The Figure 30 shows the influence of the change in the threshold for the translation and rotation on detection results. The default settings is the 1/10th of the diameter of the object to be detected for the translation and $\frac{2\pi}{30} \approx 0.2094$ rad for the rotation threshold. We varied the translation and rotation thresholds separately the rotation one in range from 0.5× to 2× of the original value and the translation threshold in range from 0.05× to 2× of the original value. Therefore the translation threshold ranged from 1/20th to 1/5th of the diameter of the object to be detected. The rotation threshold ranged from $\frac{\pi}{300}$ rad to $\frac{4\pi}{30}$ rad.

The influence of the change in the translation threshold on the detection results is shown in Figure 30a, the change in the rotation threshold is depicted in Figure 30b. The detection

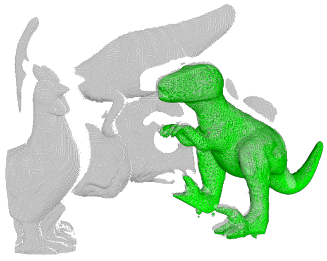
results depend heavily on the setting of the translation threshold. With the standard setting ($d_m/10$ which means 1/10th of the diameter of the object to be detected) the detection rate for $\tau = 0.016$ is 46.5%, however, when the translation threshold is doubled (i.e. $d_m/5$) the detection rate is 78.3%. In contrast, the change in the rotation threshold has almost on effect.



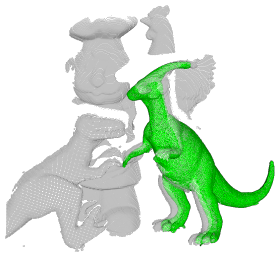
(a) The change of the translation threshold.



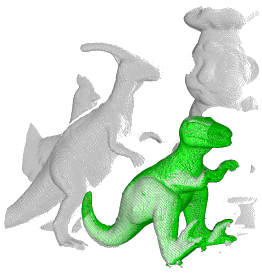
(b) The change of the rotation threshold.



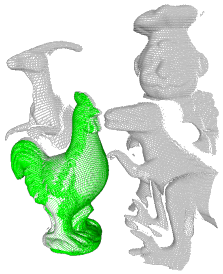
(c) Rex, scene 39
$th_{min} = \frac{d_m}{20}$.



(d) Para, scene 1
$th_{min} = \frac{d_m}{10}$.



(e) Rex, scene 6
$th_{min} = \frac{1.5d_m}{30}$.



(f) Chicken, scene 9
$th_{min} = \frac{d_m}{5}$.

Figure 30: The influence of the translation and rotation threshold on the detection rate.

The Figures 30c to 30f show the examples of the detected object and the respective minimal translation threshold required to accept these objects as correctly detected. For example, the Figure 30e shows the detection of the object *Rex* in the scene 6. This object is considered to be correctly detected only if the translation threshold is at least $1.5d_m/10$.

Due to the strictly defined thresholds by Drost et al., many objects which are clearly detected correctly are considered as incorrect during the evaluation. This corresponds to the results obtained with the commercially available implementation when the standard

algorithm also performed poorly, however, the detection rate increased significantly when the Dense refinement is enabled (see Figure 22). We do not implement the final refinement such as the ICP for our implementation. However taking into account the results on the commercial implementation and our study of influence of the translation threshold this could significantly increase the detection rate on the Mian's dataset.

7.3.2 Results on the Hinterstoisser's dataset

Similarly to the evaluation of the commercial implementation for the testing on the Hintersoisser dataset four objects and their respective image sequences are selected: *Duck*, *Lamp*, *Driller* and *Bench vise*. For each object the first 200 scenes were used. The Figure 31 shows the detection results for the various setting of the detection pipeline for each object. The sampling step $\tau = 0.03$ for the model and $\tau = 0.05$ is chosen as a trade-off between the precision and the required detection time. With the exception of the *Driller* object the detection rate of the Point-pair feature method (without any improvements applied) is very low, for the *Duck* and *Bench vise* objects even zero. The detection rate significantly increases if the matching score calculation is enabled. The detection pipeline with only matching score calculation is also the one with best performance slightly outperforming the combination of the weighted voting and the matching score calculation.

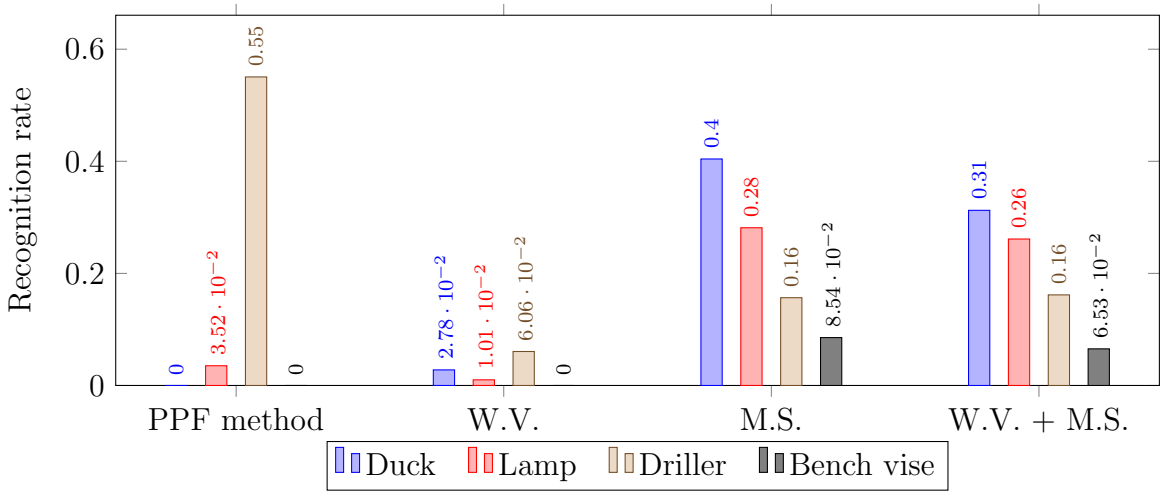


Figure 31: Detection results on the Hinterstoisser's dataset. Model sampling $\tau = 0.03$, scene sampling $\tau = 0.05$ and $\frac{|S|}{5}$. Note that W.V. stands for weighted voting and M.S. for the matching score calculation.

Hinterstoisser et al [20] reported the results for this dataset using the implementation provided by Drost. No further information including the used sampling step or possible application of any refinement is provided. The reported detection rate for these four objects ranged from 46 percent to 93 percent. The detection rate for our implementation is lower, however, this could by caused by the selection of a low sampling step and possibly by the lack of the final refinement step. The comparison of the various sampling steps and their influence on the detection rate is depicted in Figure 32.

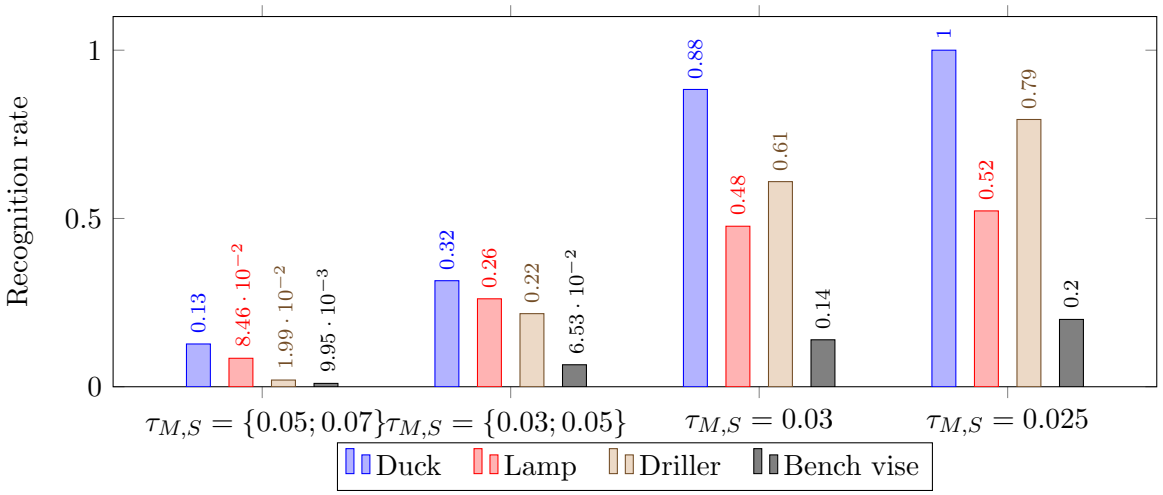
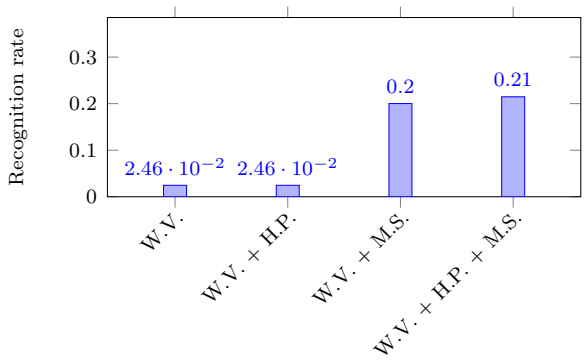


Figure 32: The influence of the sampling step on the detection rate on Hinterstoisser's dataset. The weighted voting, hypotheses pruning and matching score calculation improvements are enabled.

The influence of the hypotheses pruning is presented in Figure 33. It shows two detection pipelines both with the hypotheses pruning either disabled or enabled. The basic part of the algorithm is always the same. Therefore the time consumption is the same. However, when the hypotheses pruning is enabled the time required to perform the pose clustering and eventually also the matching score calculation drops significantly. The pose clustering is 10 times faster and the matching score calculation more than 4 times while the filtering part took only about 0.12 seconds.
The time required for the whole detection pipeline drops from 813 seconds to 338 second (from 2540 to 769 in case of the detection pipeline with the matching score calculation). But at the same time the detection rate remains the same.



(a) The influence of the hypotheses pruning on detection results.

Improvements applied	PPF method	Hypotheses pruning	Clustering	Matching score calculation	Total
W.V.	291.04	-	522.39	-	813.44
W.V. + H.P.	291.62	0.12	46.25	-	338.00
W.V. + M.S.	297.78	-	528.46	1714.00	2540.20
W.V. + H.P. + M.S.	295.64	0.13	47.20	425.54	768.51

(b) Time in [s] required in each step of the algorithm.

Figure 33: The influence of the hypotheses pruning on detection results and the required detection time.

7.4 Comparison

We presented the results of two different implementations; the commercially available one with its optional refinement steps and our implementation in MATLAB® including the proposed improvements. The results for the Point-pair feature method are significantly lower than those reported by the author [12] using both evaluated implementations. The same applies for the reported results on the Hinterstoisser's dataset where the implementation provided by Drost et al. [12] was used.

It seems that none of these results presents the detection rate of the Point-pair feature method as described in Drost et al. [12] and without any post processing. To obtain comparable results, it is necessary to apply an extra steps. In case of the Mian's dataset the matching score calculation step is able to improve the results slightly, however, the final refinement using ICP improves the detection rate significantly.

For the Hinterstoisser's dataset the matching score calculation is the crucial improvement which significantly increases the detection rate. This corresponds with the case of the commercial implementation where it is necessary to use a similar approach the Sparse refinement step to obtain results comparable with the reported ones. The example of the detection results on the Hinterstoisser's dataset is presented in Figure 34.

Nevertheless the other proposed improvements such as the restricted selection of pairs of scene points and the weighted voting with hypotheses pruning offer substantial reduction

of the detection time with none or very small decrease in the detection rate.

Figure 34: Examples of the detected objects in Hinterstoisser's dataset. Coloured point clouds of each scene with detected object in green colour are shown in the left column. The points in the scene which voted for that particular object pose are shown in the right column. The red circles denote the reference points and the green ones the paired scene points.

8 Conclusion

In this thesis, we have proposed several improvements of the Point-pair feature method of Drost et al. [12]. The existing methods for detection and localization of texture-less objects in RGB-D images were investigated. We have chosen the method based on the *point-pair feature* which is calculated only from depth information and therefore suitable for texture-less object detection. As a model of each object to be detected the method requires only a single 3D point cloud with associated normal vectors.

The Point-pair feature method was implemented in MATLAB® software. The commercially available implementation of the Point-pair feature method from the MVTec HALCON software [14] was also introduced.

The improvements are integrated in the proposed detection pipeline. First the restricted selection of pairs of scene points allows to reduce dramatically the number of tested point pairs in the large scenes and consequently the time needed for detection. In case of the Hinterstoisser's dataset [20] the average number of processed point pairs in one scene drops to 3.6% and the detection time is less than 30% of the original one.

Using the proposed weighted voting scheme and the subsequent hypotheses pruning, it is possible to filter out a large number of incorrect poses with low votes and therefore to further speed-up the later phases of the detection pipeline. On the Hinterstoisser's dataset [20], 63% of incorrect hypotheses are removed on average. The time required for the whole detection pipeline drops down to less than 50% when the filtering is applied.

Overall time required for the whole detection pipeline drops down to 15% when the restricted selection of pairs of scene points together with the weighted voting scheme and the subsequent hypotheses pruning is applied. However, the whole detection pipeline (implemented in MATLAB®) still took about 12 minutes for one scene in Hinterstoiser's dataset. We believe that a C++ implementation would significantly decrease the detection time. The commercial implementation of the baseline method, which is in C++, required only about one second for one scene.

Calculation of the matching score turned out to be crucial for reliable pruning of false positives. The matching score is more discriminative than the count of votes used in the method of Drost et al. [12] and was shown to be necessary in order to achieve an acceptable performance on the Hinterstoisser's dataset [20].

We couldn't reproduce the results of the method as published by Drost et al. [12] and Hinterstoisser et al. [20]. We suppose that the reported results were not obtained using the baseline Point-pair feature method as described in [12], but with further refinements which are available in the MVTec HALCON software [14]. The refinement steps were shown to be necessary also to obtain detection results on the Mian's dataset [27] comparable to the ones published in [12].

8.1 Future work

The further enhancements by using the dense refinement, such as the ICP algorithm would increase the detection rate, however, the experiments have shown that this would lead to a noticeable increase of the required detection time.

A greater attention could be paid to the proposed selection of the reference points in the scene. Selecting the reference points in interesting regions could increase the detection rate while keeping the number of reference points and consequently the time consumption low. The proposed method utilizing the shape index seems promising, however, better computation of the shape index which would be less susceptible to the noise in the image should be devised.

Another extension could be to utilize the colour of the object and to use it to prune the hypotheses based on the dominant object colour.

References

[1] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. CAD-model recognition and 6DOF pose estimation using 3D cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 585–592. IEEE, Nov 2011.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] G. Bradski. *Dr. Dobb's Journal of Software Tools*.

[4] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH*, volume 2007, page 5, 2007.

[5] Hongping Cai, Tomáš Werner, and Jiří Matas. Fast Detection of Multiple Textureless 3-D Objects. In *Computer Vision Systems*, volume 7963, pages 103–112. Springer Berlin Heidelberg, 2013.

[6] AY Chia, Susanto Rahardja, Deepu Rajan, and Karhang Leung. Object recognition by discriminative combinations of line segments and ellipses. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2225–2232. IEEE, June 2010.

[7] Changhyun Choi and Henrik I Christensen. 3D pose estimation of daily objects using an RGB-D camera. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3342–3349. IEEE, Oct 2012.

[8] Changhyun Choi, Yuichi Taguchi, Oncel Tuzel, Ming-Yu Liu, and Srikumar Ramalingam. Voting-based pose estimation for robotic assembly using a 3D sensor. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1724–1731. IEEE, May 2012.

[9] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference*, pages 129–136. The Eurographics Association, 2008.

[10] Dima Damen, Pished Bunnun, Andrew Calway, and Walterio W Mayol-Cuevas. Real-time learning and detection of 3d texture-less objects: A scalable approach. In *BMVC*, pages 1–12, 2012.

[11] Bertram Drost and Slobodan Ilic. 3D Object Detection and Localization Using Multimodal Point Pair Features. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 9–16. IEEE, Oct 2012.

[12] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005. IEEE, June 2010.

[13] Dariu M Gavrila. A Bayesian, Exemplar-Based Approach to Hierarchical Shape Matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(8):1408–1421, Aug 2007.

[14] MVTec Software GmbH. *HALCON Software, Version 12.0*. München, Germany, 2014.

[15] Iryna Gordon and David G Lowe. What and where: 3D object recognition with accurate pose. In *Toward Category-Level Object Recognition*, pages 67–82. Springer Berlin Heidelberg, 2006.

[16] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(11):2270–2287, Nov 2014.

[17] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient Response Maps for Real-Time Detection of Textureless Objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):876–888, May 2012.

[18] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of

texture-less objects in heavily cluttered scenes. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 858–865. IEEE, Nov 2011.

[19] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. Dominant orientation templates for real-time detection of texture-less objects. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2257–2264. IEEE, June 2010.

[20] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Computer Vision – ACCV 2012*, volume 7724, pages 548–562. Springer Berlin Heidelberg, 2013.

[21] Xiaoyi Jiang and Horst Bunke. Edge Detection in Range Images Based on Scan Line Approximation. *Computer Vision and Image Understanding*, 73(2):183–199, 1999.

[22] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, May 1999.

[23] Jan J Koenderink and Andrea J van Doorn. Surface shape and curvature scales. *Image and Vision Computing*, 10(8):557–564, 1992.

[24] Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981.

[25] Ming-Yu Liu, Oncel Tuzel, Ashok Veeraraghavan, and Rama Chellappa. Fast directional chamfer matching. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1696–1703. IEEE, June 2010.

[26] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[27] Ajmal Mian, Mohammed Bennamoun, and R Owens. On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010.

[28] Ajmal S Mian, Mohammed Bennamoun, and Robyn A Owens. Automatic correspondence for 3D modeling: an extensive review. *International Journal of Shape Modeling*, 11(02):253–291, 2005.

[29] Marius Muja, Radu Bogdan Rusu, Gary Bradski, and David G Lowe. REIN - A fast, robust, scalable REcognition INfrastructure. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2939–2946. IEEE, May 2011.

[30] Clark F Olson and Daniel P Huttenlocher. Automatic target recognition by matching oriented edge pixels. *Image Processing, IEEE Transactions on*, 6(1):103–113, Jan 1997.

[31] Andreas Opelt, Axel Pinz, and Andrew Zisserman. A Boundary-Fragment-Model for Object Detection. In *Computer Vision – ECCV 2006*, pages 575–588. Springer, 2006.

[32] Chavdar Papazov and Darius Burschka. An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes. In *Computer Vision – ACCV 2010*, pages 135–148. Springer Berlin Heidelberg, 2011.

[33] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3212–3217. IEEE, May 2009.

[34] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D recognition and pose using the Viewpoint Feature Histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, Oct 2010.

[35] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3D with Kinect. In *Consumer Depth Cameras for Computer Vision*, Advances in Computer Vision and Pattern Recognition, pages 3–25. Springer London, 2013.

[36] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard. Point feature extraction on 3D range scans taking into account object boundaries. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2601–2608. IEEE, May 2011.

[37] Min Sun, Gary Bradski, Bing-Xin Xu, and Silvio Savarese. Depth-Encoded Hough Voting for Joint Object Detection and Shape Recovery. In *Computer Vision – ECCV 2010*, volume 6315, pages 658–671. Springer Berlin Heidelberg, 2010.

[38] Simon Winkelbach, Sven Molkenstruck, and Friedrich M Wahl. Low-Cost Laser Range Scanner and Fast Surface Registration Approach. In *Pattern Recognition*, pages 718–728. Springer Berlin Heidelberg, 2006.

[39] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

Appendix

DVD Content

In table 1 the names of all directories on DVD are listed with description.

Directory name	**Description**
computeNormals	source codes for the Hinterstoisser's dataset preparation
datasets	datasets used for testing
↪ hintersotisser	data for Hinterstoisser's dataset
↪ mian	data for Mian's dataset
↪ occlusion	occlusion data for Mian's dataset
evaluateHalcon	Matlab scripts for evaluation of the HALCON results
halconResults	results from the HALCON software
ppfDetector	source code of the implemented detector
↪ example	directory with example of the detector usage
thesis	sources of this thesis in the LaTeX
↪ fig	directory with used images
↪ src	directory with chapters
readme.txt	description of the content of the DVD
thesis.pdf	thesis in pdf format

Table 1: DVD Content