Recognition and Motion Tracking of 3D Objects
TTK4550

Matias Hagen Myrestrand

December 2020

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Supervisor 1: Jan Tommy Gravdahl, Department of Engineering Cybernetics, NTNU
Supervisor 2: Marialena Vagia, Mathematics and Cybernetics, SINTEF
Supervisor 3: Klaus Ening, Mathematics and Cybernetics, SINTEF

NTNU – Trondheim
Norwegian University of
Science and Technology

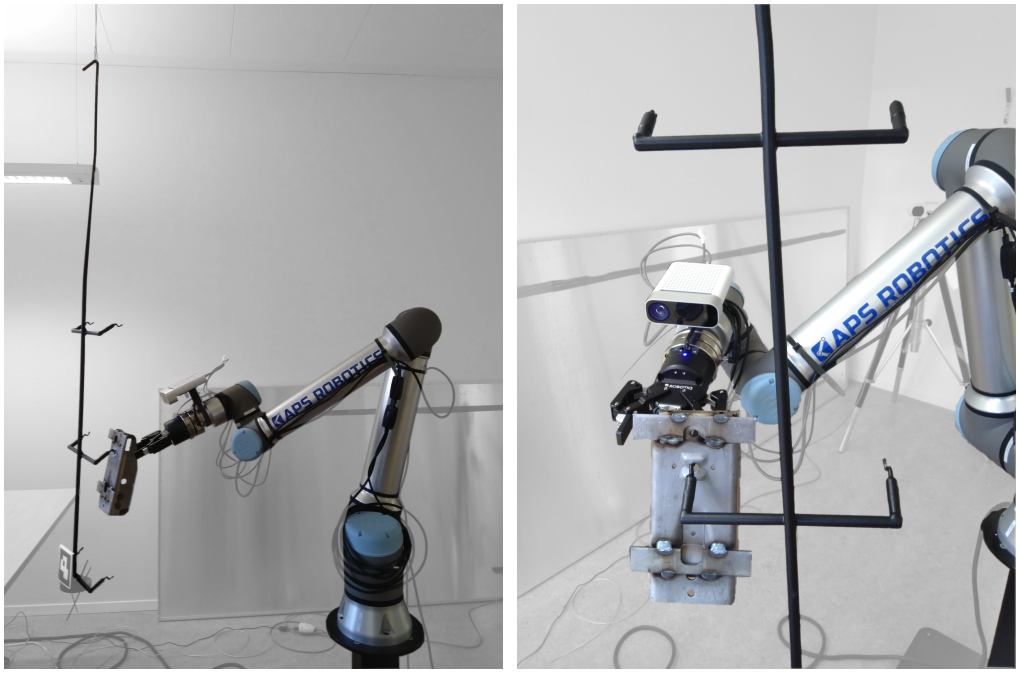
Contents

1 Introduction

Object recognition and 6-DoF pose estimation make up some of the most prominent fields in computer vision and robotics today. With the arrival of more complex use cases of robotic technology such as autonomous driving, or even medical robots performing surgery procedures [14], the systems ability to accurately perceive their surroundings is essential. Industrial assembly robots are examples of autonomous systems which require information about their environment in order to interact with it. RGB cameras, IR depth cameras and LiDAR are all examples of optical sensors that can provide the raw data needed to obtain this information. By utilizing the techniques of object recognition and pose estimation, these systems are able to discover and localize target objects or potential obstacles. Industrial- and inspecting robots, modern visual surveillance [29], augmented reality [48] and intelligent transportation [6] are just some of the applications relying on the techniques of both object recognition and motion tracking.

This thesis will mainly focus on the industrial application of these techniques. As we know, robots are widely used in manufacturing performing task such as pick and place, assembly, packaging and painting. All of these tasks require an accurate identification and pose estimate of the object to be handled. Therefore, robotic vision has great importance in automated industrial processes that otherwise would call for human intervention. Reduced costs, increased production, improved consistency and safety are some of the benefits from vision guided robotic systems [26, 37, 13]. However, recognition and motion tracking still pose some notable challenges. For instance, varying lighting conditions with shadows and glare, occlusion of objects and motion blur can all complicate the tasks of object detection and 6-DoF pose estimation [28]. Furthermore, when moving objects are added to the equation, an additional demand for real-time speed must be taken into account. This demand illustrates a crucial challenge in pose estimation as a trade-off between accuracy and computational cost is inevitable.

The problem to be solved in this thesis addresses to a chair manufacturing setting and involves a UR10 robotic manipulator responsible for loading and unloading objects off a hanger. The hanger, which is suspended from a roof mounted conveyor, is swinging freely. This manufacturing setting is recreated in a lab setup as illustrated in figure 1. The camera, an *Azure Kinect DK*, is mounted on the UR10 and provides both color and depth (RGB-D) vision. In order to enable the use of the robotic manipulator it is important to recognize and track the 3D object that we need the arm to interact with. Hence, a flexible solution for 3D tracking of different objects is needed. Figure 2 illustrates the chair parts that will be used for the object tracking. These objects have not undergone any paint job at this point in the assembly process, and will consequently have a varying surface texture.
The nature of this problem will be the basis for the literature study in section 2, where the viability of relevant existing recognition and 6-DoF estimation techniques will be discussed before proposing a feasible solution. Thereafter, the theories behind the selected methods are shown in section 3 before discussing the implementation in section 4 and presenting the results in section 5. Finally, a discussion and conclusion will be given in section 6 and 7 respectively.



(a) (b)

Figure 1: Lab setup for problem description **a)** Side view of the robotic manipulator and hanger. **b)** Closeup of robotic manipulator loading object onto hanger.

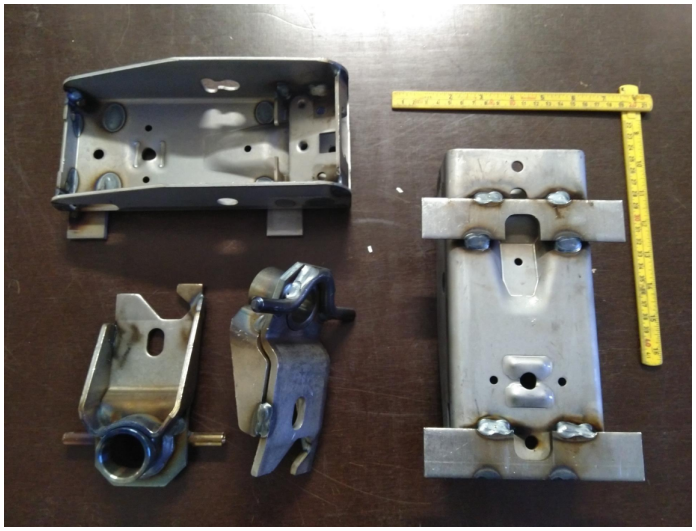


Figure 2: Objects to track.

2 Literature study

In this section we do a brief study of existing 6-DoF estimation methods, while also discussing the suitability of these methods in relation to the problem given in section 1. The study starts off by taking a look at feature based methods in 2.1, before discussing template matching in 2.2, learning-based methods in 2.3, and point cloud based methods in 2.4. Finally, a feasible framework will be proposed in 2.5.

2.1 Feature Based Methods

Feature based methods make up a variety of techniques utilizing local image descriptors to match keypoints between the scene and textured target objects. Using either a monocular RGB camera, or a multi-view stereo vision setup, the 2D key points in image coordinates are back-projected to 3D before retrieving the 6-DoF pose of the object based on these point-to-point correspondences. SIFT [23] and SURF [3] are both examples of feature detection algorithms which describe and detect local features in images. In short, these apply gradient magnitudes and orientations relative to the keypoint's orientation, making the descriptor invariant to rotation. The transition from n 2D-3D point correspondences to a 6-DoF pose is defined as a Perspective-n-point (PnP) problem. Gordon and Lowe [12] presented a 3D object pose estimation framework back in 2006, using SIFT correspondences between the scene and a 3D model of the object, and thereafter solving the following PnP problem. In order to prevent false matches the RANSAC algorithm [10] is also implemented, which removes outliers from potential 2D-3D correspondences. For further pose refinement the Levenberg-Marquardt algorithm [22] can also be applied, as was done in [12], minimizing the geometric reprojection error. In total this framework can provide a speedy 3D object detection and pose estimate for textured objects when provided corresponding 3D object coordinates for each matching 2D scene point.
As a starting point of this project a similar framework was implemented using ORB [34], an efficient alternative to SIFT or SURF, and *solvePnPRansac* which combines RANSAC and a PnP-solver to give an estimated pose in 6-DoF. Both of these are provided by the *Open Source Computer Vision Library* (OpenCV). Initial tests were carried out using a planar flyer as tracking object, giving a simple translation from image to 3D object coordinates. An illustration of the pose estimation result is shown in figure 3.

This testing was for research purposes only, and will not be discussed in the remaining thesis. This solution did however illustrate the importance of explicit features in order to find point correspondences. As for SIFT and SURF, ORB can also be vulnerable to illumination changes such as shadows and glare. Consequently, the flyer pose estimation solution was only able to track its pose correctly in a span of ± 30 degrees in *roll*, *pitch* and *yaw*.
As stated in the problem description in section 1, the target objects will have a varying surface texture. Naturally, this does not make the ideal conditions for a local feature descriptor as each sample object will give rise to different keyponts. Self induced shadows and a general lack of explicit features would further degrate the performance of a feature based framework if applied to this problem.

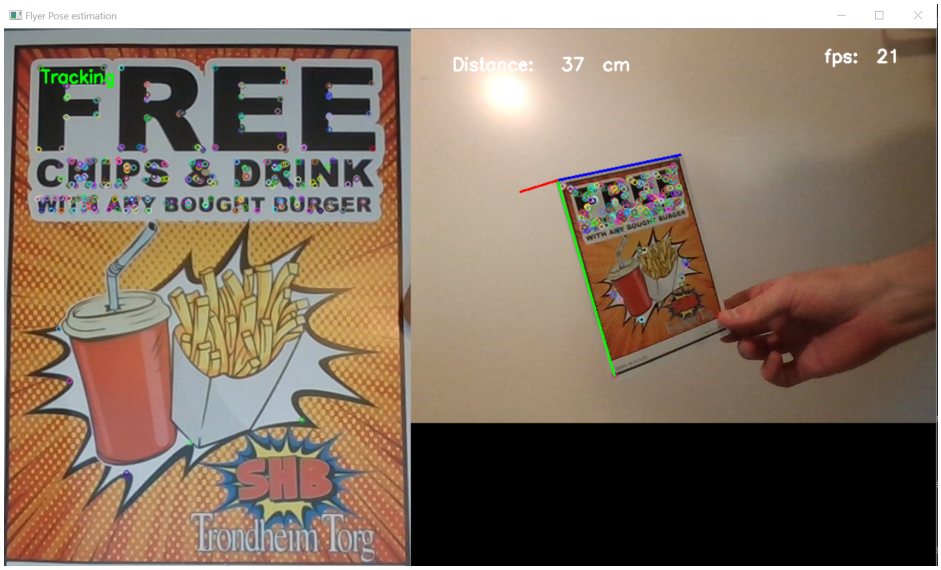


Figure 3: 6-DoF pose estimation of a textured flyer. Each colored circle represent an ORB keypoint available for matching.

2.2 Template Based Methods

Template matching is another common approach for object detection and pose estimation. In contrast to the feature based methods that use local feature descriptors, this approach uses object descriptors. An object descriptor encode the entire observed object based on a given modality, such as color gradients. The similarity score between a scene image and a set of templates will hence decide if an object is present or not. During training, these template images are obtained through sampling from different viewpoints. This way the object's pose can also be determined based on the pose of the template match with the highest similarity score. However, in order to achieve high resolution pose estimates with this approach, a huge set of template images would be required. A trade-off between pose resolution, memory consumption and search speed is inevitable. As a result, pose refinement algorithms like *Iterative Closest Point* (ICP) [4] have been used to improve the initial pose estimate. This method operates on point clouds and can only be applied when depth images are available. More details on ICP will be given in the subsection on point cloud based methods in 2.4. For pose estimation setups using RGB images only, the minimization of photometric energy functions have been used to refine the initial pose estimates, as was done in [40].

2.2.1 LineMOD

LineMOD [18] is a well known method in template based pose estimation which uses multiple modalities. The framework combines both RGB images and dense depth maps, also known as RGB-D images, to give complimentary information on an object. As

demonstrated in [18], the combination of a color gradient descriptor and a surface normal descriptor makes a robust template representation. An illustration of these modalities are shown in figure 4. As the figure shows, the color gradients are mainly located on the

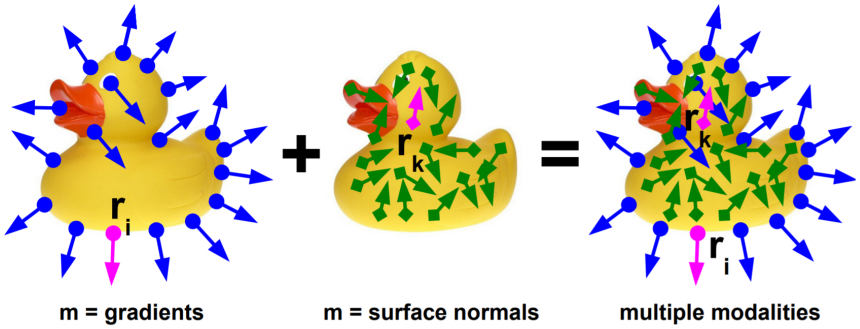


Figure 4: A rubber duck with with different modalities, *m*. (Source: [18])

contours, while the surface normals are located on the body of the object. Unlike the local feature descriptors discussed in 2.1, this template representation approach will also be able to detect texture-less 3D objects. The contours are naturally also more robust to illumination changes and noise compared to local feature keypoint found on the body. For our problem, described in section 1, a LineMOD based template representation seems to be a viable option for obtaining a decent initial pose estimate. As discussed, this solution would however still need a pose refinement step. Given that this method already makes use of depth images, the ICP-required cloud points would be easily accessible. Hence ICP could be implemented and return the final 6-DoF pose for all detected objects.

Despite claiming real-time performance for multiple object detection and pose estimation in [18], the efficiency of LineMOD has been discussed in various papers proposing improvements on the pipeline. The exhaustive nearest neighbour search used for finding the most similar template match is definetely not ideal. Shao et al. [36] recognize this, but still proclaim this method to achieve real-time speed for single object pose estimation. They also discuss *Approximate Nearest Neighbour* (ANN) techniques such as *hashing-based* and *tree-based* matching, while proposing a modified *fuzzy decision forest* framework for improved matching efficiency. Although both hashing-based and tree-based methods have sub-linear complexity for searching, these still have some drawbacks. For one, the design of an efficient hash function is often not trivial [36]. Shao et al. also points out the efficiency suffering related to *the course of dimensionality* due to backtracking for the tree structure. Regardless, [21, 19] and [36, 33] all show improved efficiency on the LineMOD ACCV12 dataset [18] by applying hashing-based and tree-based methods respectively.

2.2.2 Region Based Methods

Region based methods, using RGB images only, constitute a different subcategory of template matching. Simplified, these apply descriptors such as *gradient response maps*

and *local color histograms* to find the template pose and its projected silhouette that best fit the camera image. Just as for LineMOD, region based methods can be used for detection of texture-less 3D objects. LINE-2D [16] is a popular and generic region based method that applies gradient response maps for detection, just as LineMOD. In fact, LINE-2D is what remains if the surface normals are omitted from the LineMOD method. Tjaden et al. present a different approach in [40], using *temporally consistent local color histograms* along the contours of the objects. An illustration of this technique is shown in figure 5.

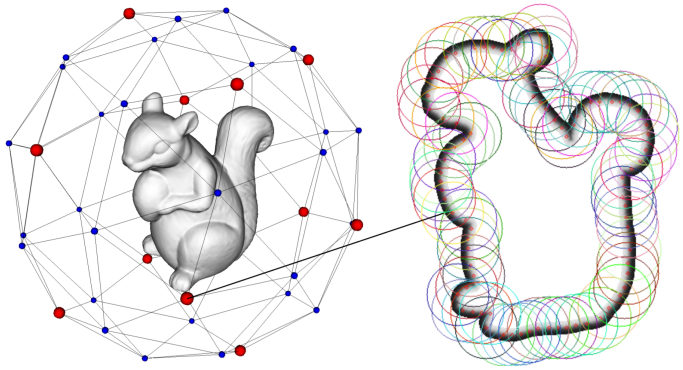


Figure 5: Projected contour of 3D object from a given template view. The local color histogram regions are illustrated by colored circles. (Source: [40])

A drawback of this method is that it requires background knowledge, and must hence be trained in the intended scene to outperform the more generic LINE-2D approach. In addition, Gaussian based methods like [5] still perform better on the ACCV dataset [40] compared to their method with scene knowledge. Stoiber et al. [39] also present a novel sparse Gaussian approach to region based 6-DoF object tracking that was displayed at the *Asian Conference on Computer Vision* (ACCV) in late November 2020. Although [5] and [39] both have demonstrated some interesting results, the lack of implementation details in [5] and documentation in [39] (at the time of writing) makes it difficult to experiment or give a well-founded assessment of these methods in relation to our problem description. If considering region based methods as a whole however, the lack of complementary depth descriptors can arguably make them more prone to false matches and drifting. For instance, the pose refinement step, which normally utilizes something similar to a photometric loss function, will naturally be less robust in terms of ambiguity compared to the ICP algorithm which operates on 3D cloud points. To put it strongly, the shape of the object will have no importance as long as the contours match the ones of a template. For object with less distinct silhouettes than those from [40, 5] and figure 5, it is reasonable to expect a somewhat impaired result in terms of detection and pose estimation. The intended objects to track from figure 2 would also lose a lot of their characteristics if only considering their silhouettes. Based on this brief assessment, LineMOD might be a more suitable method for our problem than the region based approaches.

2.2.3 Template Generating

The task of generating a set of templates naturally applies to all 6-DoF object pose estimation frameworks based on template matching. As mentioned, a large number of template samples from different viewpoints are needed in order to recognize an object and give a decent initial pose estimate. In [15] a total of 12960 templates are used per object. These are rendered from 216 viewpoints uniformly distributed on a synthetic sphere around the target object. For each viewpoint the camera is rotated around the optical axis from $-60°$ to $+60°$ with a step of $10°$. Finally this is repeated for 5 different spheres with varying radii with a step of $0.1m$. A similar setup is used in [36], except they only use the upper hemisphere for sampling as shown in figure 6. The general approach to generate these sets of templates is to synthetically render a 3D mesh model of the target object. These models can for instance be obtained by scanning the object. This way we can obtain flexible template based frameworks for 6-DoF object pose estimation, as a mesh model is the only requirement for tracking a new object.

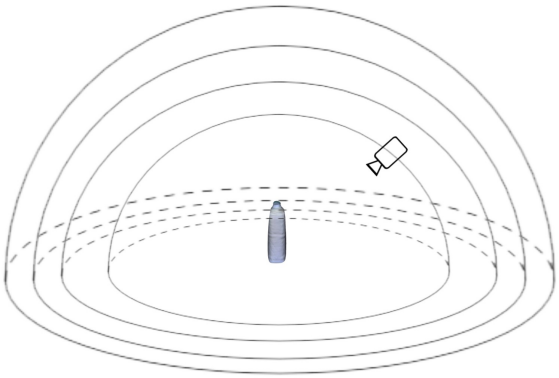


Figure 6: A synthetic rendering sample of a bottle. This setup contains four different hemispheres with varying radii. (Source: [36])

2.3 Learning Based Methods

Learning based methods make up a different approach to 6-DoF pose estimation which generalize better to variations in viewpoint and slight shape deformations. [33] from 2.2, which extended LineMOD by introducing an efficient tree-based search for template matching, is an example of such a method, as the templates are learned in a discriminative fashion. In general, learning based methods often evoke less false positives than nearest neighbour approaches such as the exhaustive LineMOD search from [18]. However, as stated in [36], their efficiency often depends on the quality of negative training samples. If trained for one specific scene, the performance may not be transferable to others. This should also be considered when being presented with results from a learning based methods. As with the temporally consistent local color histogram approach from [40], these have often been trained on that particular dataset to achieve the best performance for that scene.

Some of the latest and most prominent related works include PointNet [30], which directly uses point clouds for object classification and segmentation, and [41] which propose a method for human pose estimation called DeepPose. Both are based on *Deep Neural Network* (DNN) architecture. Gao et al. also present a method for 6-DoF object pose estimation in [11] based on both PointNet and ICP for pose refinement. As for most template and learning based methods, the trade-off between efficiency and accuracy may hurt the performance of real-time systems with moving objects. For instance, [11] demonstrate an average processing time of $0.41s$ for a single object image when running on a Nvidea Titan X GPU. This naturally would not be sufficient for a real-time system unless we were dealing with stationary objects. DOPE [42] and PoseCNN [45], both using DNN architecture, also fail to meet the real-time requirements for dynamic tracking. However, [42] present a novel synthetic data generation procedure which enables a more flexible training setup with pre-labeled data. This way, the cumbersome process of image assembling and labeling is avoided.

To summarize, learning based methods make up a variety of approaches. While typically generalizing better to variations such as shape deformations, their performance often rely on scene specific training and the quality of negative training samples. Furthermore, the popular DNN based architectures introduce high computational complexity, making these methods unfit for the dynamic tracking problem described in section 1. On the other side, approaches such as the tree-based LineMOD extension [36] can actually result in increased efficiency.

2.4 Point Cloud Based Methods

The entering of low-cost 3D cameras in the market has resulted in increased focus on approaches that operate directly on 3D point clouds. The 3D object classifier PointNet [30] from 2.3, and the LineMOD surface normal descriptor from 2.2.1 both utilize 3D point data, making these invariant to object texture and illumination changes. Methods employing depth data exclusively are mainly used for pose refinement or template matching. The previously mentioned ICP algorithm [4] is an example of the former.

2.4.1 ICP

ICP, or Iterative Closest Point, differs from most object pose estimation methods as it does not detect the object. Instead, ICP uses an iterative scheme to align two cloud points. This geometric optimization will hence find the translation and rotation that minimize the distances between corresponding object points in 3D. However, a decent initial guess or estimate of the objects pose is required. The algorithm is sensitive to both the initial pose and sensor noise, which in turn can result in convergence to local optima. In order to reduce sensor noise Ruotao He et al. [15] use a moving least squares algorithm [2] for smoothing scene points before applying ICP. For template based 3D object detection and pose estimation frameworks, such as [15, 18, 36, 21], ICP is often applied as the template matching alone won't give a sufficiently accurate object pose estimate.

2.4.2 Oriented Point Pairs

A different category of point cloud based approaches is presented by Drost et al. in [8]. Using oriented point pair features, they create global model descriptions of the objects, which are later matched using a voting scheme. The features describe relative position and orientation of two point normals, as illustrated in figure 7.

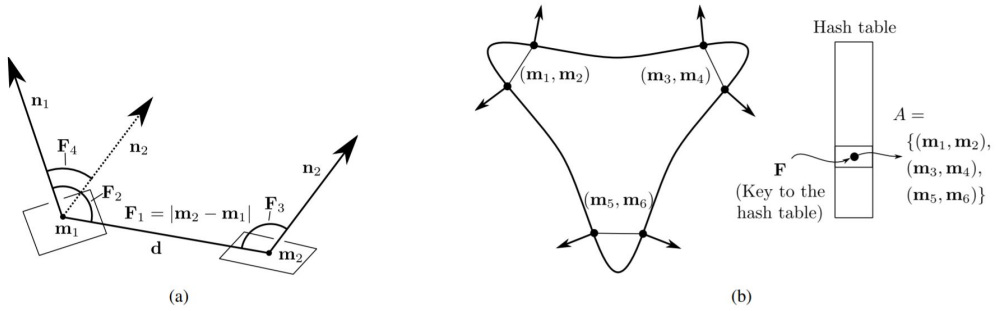


Figure 7: **a)** Point pair feature $\mathbf{F}$ of two oriented points. $\mathbf{F_1}$ is set to the distance between the points, $\mathbf{F_2}$ and $\mathbf{F_3}$ equals the angle between the normals and the vector defined by the two points. Finally, $\mathbf{F_4}$ is set to the angle between the two normals. **b)** The global model description. Point pairs with similar vector $\mathbf{F}$ are stored in the same slot in the hash table. (Source: [8])

By analysing point pair features from the object scene, this method can detect the target objects while simultaneously output probable poses in 6-DoF. For increased stability Drost et al. also use pose clustering which removes isolated poses with low scores. Similar techniques are applied in template based methods such as [15]. Unlike this LineMOD based approach however, [8] is not refined by methods such as ICP. Nevertheless, when requesting high precision, this framework will pay the price in terms of high processing time. For our intended real-time application this naturally won't be desirable. On the other hand, it offer a somewhat flexible solution, as a 3D model is the only thing required for tracking a new object.

2.5 Proposed Method

When considering what method should be applied to the problem described in 1, a template based approach like LineMOD seems to be a reasonable choice. This method has been proven to work on texture-less objects while being robust to illumination changes. Works like [36] and [21] also demonstrate that it can be made more efficient by proposing new template matching strategies. The synthetic template generating approach from section 2.2.3 is also practical, and provides a flexible solution for multiple object tracking. As previously mentioned, the region based methods [5] and [39] also show some interesting results, but the lack of implementation details and documentation respectively makes it difficult to experiment or give a well-founded assessment of these methods in relation to our problem. Hence, a LineMOD based framework with template clustering and subse-

quent ICP pose refinement will make up the final proposed pipeline. A rough sketch of this pipeline is shown in figure 8. LineMOD will accordingly serve as the detector, while template pose clustering and evaluation provide the inital pose estimates before the ICP pose refinement.

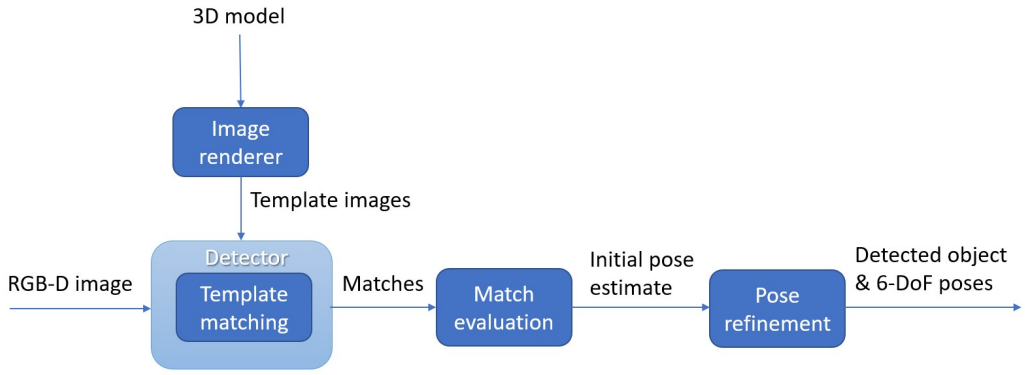


Figure 8: Rough sketch of the proposed 6-DoF pose estimation pipeline. The template generating and loading to detector, i.e. training, will be done offline.

3 Theory

In this section a more detailed description of the proposed framework methods will be provided. First, the LineMOD [18] descriptors and similarity measures are presented in 3.1, before additional information will be given on clustering and the ICP algorithm in 3.2 and 3.3 respectively.

3.1 LineMOD

The following subsection is based on [18] and [17] by Hinterstoisser et al. Both papers cover the LineMOD image representation and template matching strategy.

3.1.1 Similarity Measure

In order to find potential matching objects in an input image, a similarity measure is required. The generalized LineMOD variant can be formalized as:

$$\varepsilon(I, T, c) = \sum_{(r,m)\in P} \max_{t\in R(c+r)} f_m(O_m(r), I_m(t)), \tag{1}$$

where I is the input image and T is a given template. This template is defined as $T = (\{O_m\}_{m\in M}, P)$, where O_m is the template image of modality m, and P is a list of pairs (r, m), where r is the location of a discriminant feature of modality m. The

comparing between a scene image and a template is done through a sliding window approach, where c is the location in I to be evaluated. By summarizing the similarity scores over the discriminant features in P, through a similarity function f_m, a total similarity score is provided. A template is matched if the score is higher than an applied threshold. Furthermore, the separate feature scores corresponding to $(r, m) \in P$ are set to equal the maximum similarity score in a neighbourhood $R(c + r)$ of size N with $r + c$ as the midpoint. This way the similarity measure in equation 1 archives robustness to small translations and deformations.

3.1.2 Modalities

As previously mentioned, LineMOD combines the modalities of both color gradients and surface normals. For the case of **color gradients**, these are obtained by in inspecting each of the three color channels (R,G,B) separately. This naturally increases robustness as the different channels provide a greater option of gradients than what would be the case in grayscale images. Figure 9 also illustrates the difference in contour visibility using both methods. In addition, this method considers only the orientation of the gradients and not their norms, which increases robustness to contrast changes. For each image location the gradient orientation of the channel whose magnitude is largest will be selected. This can be illustrated by equation 2 and 3 where $I_G(x)$ is the gradient orientation at location x in the input image.

$$I_G(x) = \mathbf{ori}(\hat{C}(x)) \tag{2}$$

$$\hat{C}(x) = \arg \max_{C \in \{R,G,B\}} \left\| \frac{\partial C}{\partial x} \right\| \tag{3}$$

As the normalized gradient map only considers the gradients orientation, and not their direction, the orientation space of the map is divided into n_o equal spacings as shown in figure 9. This will prevent the detection from being affected if the background changes from bright to dark. The similarity measure for the gradient orientation can accordingly be stated as:

$$\varepsilon_G(I, T, c) = \sum_{r \in P} \max_{t \in R(c+r)} |cos(\mathbf{ori}(O, r) - \mathbf{ori}(I, t))|. \tag{4}$$

In addition, to make the quantization of orientations more robust to noise, each location will be assigned the quantized orientation which occurs most often in a 3 x 3 neighborhood.

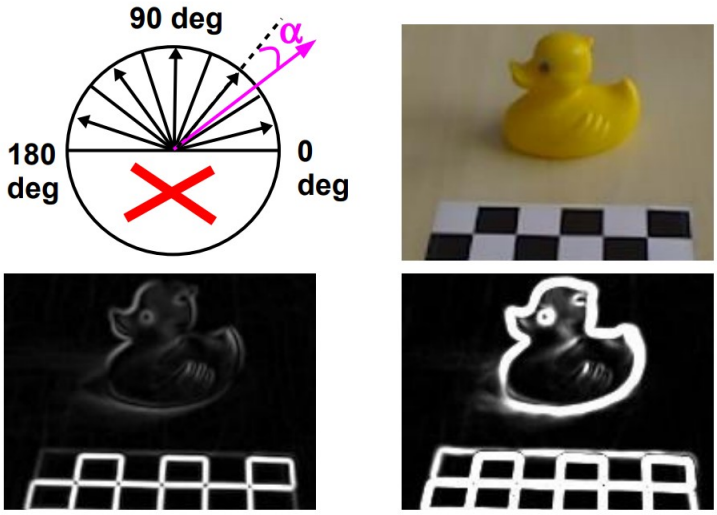


Figure 9: **Upper Left:** Quantization of the gradient orientations: The pink orientation is closest to the second bin. **Upper Right:** A toy duck with a calibration pattern. **Lower Left:** The gradient image computed on a grayscale image. **Lower Right:** The gradient image computed using maximum magnitude from the separate color channels. (Source: [17])

The second modality, **surface normals**, are computed from a dense depth field provided by the 3D camera. The method apply the first order Taylor expansion of the depth function $D(x)$:

$$D(x + dx) - D(x) = dx^T \nabla D + h.o.t. \tag{5}$$

For each pixel location x, an optimal depth gradient estimate $\hat{\nabla D}$ can be found given some pixel offset vectors dx. This gradient can accordingly be expressed as a 3D plane going through three points X_1, X_2 and X_3:

$$X_1 = \vec{v}(x)D(x) \tag{6}$$

$$X_2 = \vec{v}(x + [1, 0]^T)(D(x) + [1, 0]\hat{\nabla D}) \tag{7}$$

$$X_3 = \vec{v}(x + [0, 1]^T)(D(x) + [0, 1]\hat{\nabla D}), \tag{8}$$

where $\vec{v}(x)$ is the vector along the line of sight pointing towards the 3D point given by pixel x. This vector can be seen as a projective element provided by the internal parameters of the depth sensor. Finally, the surface normal at the 3D point can be estimated as

the normalized cross-product of $X_2 - X_1$ and $X_3 - X_1$. The similarity function for these surface normals is defined as the dot product of the normalized surface normals. Hence, the similarity measure for the depth image can be expressed as:

$$\varepsilon_D(I, T, c) = \sum_{r \in P} \max_{t \in R(c+r)} O_D(r)^T I_D(t), \tag{9}$$

where $O_D(r)$ is the normalized surface normal at location r from the reference image, and $I_D(t)$ is the normalized surface normal at location t in the input image. As for the color gradients, the surface normals are also quantized into n_0 bins. These are spread out in a right circular cone as shown in figure 10. In order to reduce the quantization noise on the surfaces, the pixels, or 3D points with substantial depth differences will be ignored. This primarily increases robustness for areas with depth discontinuity. In addition, to further increase the robustness to noise, each location in the normalized surface normal map will be assigned the quantized orientation which occurs most often in a 5 x 5 neighborhood.

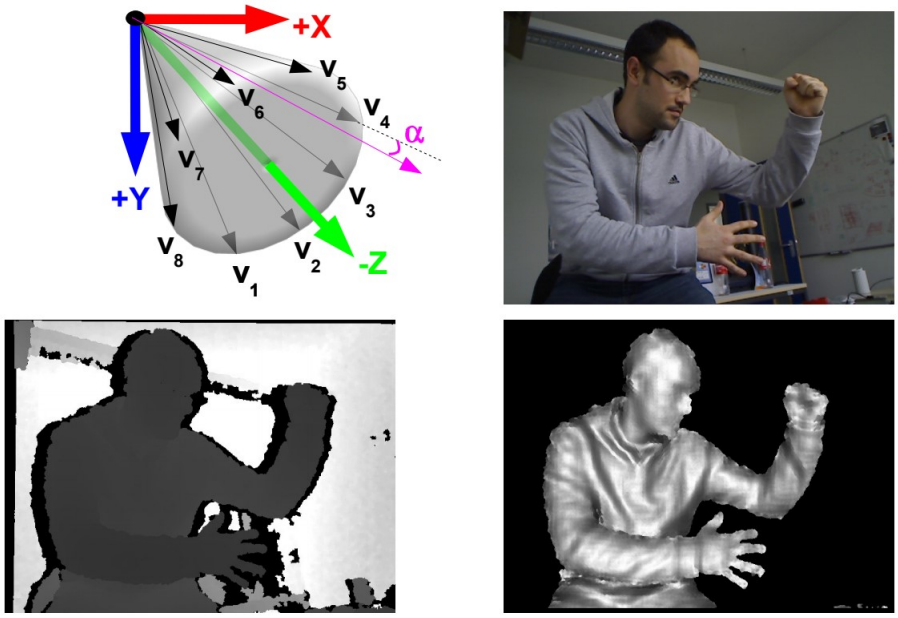


Figure 10: **Upper Left:** Quantization of the surface normals: The pink orientation is closest to the precomputed normal v_4. **Upper Right:** A person standing in an office. **Lower Left:** The corresponding depth image. **Lower Right:** Computed surface normals. The background was removed for visibility reasons. (Source: [17])

3.1.3 Precomputed Response Maps

For efficient computation of similarity scores, the method introduces a binary representation of *spread orientations*, and a lookup table for fast computation of the similarity measures found in precomputed response maps. The spreading of orientations and its simple representation prevents us from having to evaluate the *max* operator in equation 1. For each image location, a binary string indicates the presence of a quantized orientation by setting the corresponding bit to 1. Similar representation is used for the surface normal modality. However, for simplicity, only the color gradient representation and response map computation will be illustrated in this subsection. This representation, and the process of orientation spreading is shown in figure 11. The encoding of this spreading is performed by OR'ing the concerning binary strings resulting in the more robust gradient representaion, denoted by $\jmath_m$.

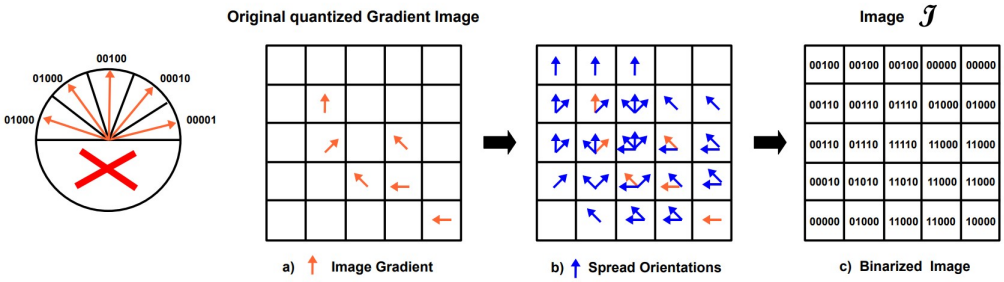


Figure 11: Spreading the gradient orientations. **Left:** The n_0 gradient orientations and their binary code. **a)** The gradient orientations in the input image, shown in orange. **b)** The gradient orientations are spread to a neighbourhood of size T, as shown in blue. **c)** The binary representation of the spread orientations. For this figure $T = 3$ and $n_0 = 5$. In practice, the method uses $T = 8$ and $n_0 = 8$. (Source: [17])

When assessing the similarity for each discriminant feature i of modality m in P, a precomputed lookup table τ_i is utilized, where the integer value of $\jmath_m$ is used as an index to the corresponding similarity score:

$$\tau_{i,m}[\jmath_m] = \max_{l \in \jmath_m} |f_m(i,l)|. \tag{10}$$

For the case of color gradients, i is the index of the quantized gradient orientation of the template feature from P, while l is the individual gradient orientations in a location as shown in figure 11 b). Accordingly, the similarity score between each discriminant gradient feature i, and corresponding spread input image orientations J can be stated as:

$$\tau_i[\jmath] = \max_{l \in \jmath} |cos(\mathbf{ori}(i) - \mathbf{ori}(l))|. \tag{11}$$

As $\jmath$ in principle represent all gradient orientations present in a neighbourhood, the method achieves robustness to small translations and deformations without having to iterate through $t \in R(c + r)$, as done in equation 1 and 4. By defining J as an image of

$\mathcal{J}$-pixels, the values of the response map S_i can be precomputed as:

$$S_i(c+r) = \tau_i[J(c+r)]. \tag{12}$$

Finally, the similarity measure from equation 4 becomes:

$$\varepsilon_G(I,T,c) = \sum_{r \in P} S_{ori(O,r)}(c+r), \tag{13}$$

where the different response map variants are chosen as specified by the orientation i of the current reference image feature in location r. The process of precomputing the response maps is illustrated in figure 12.

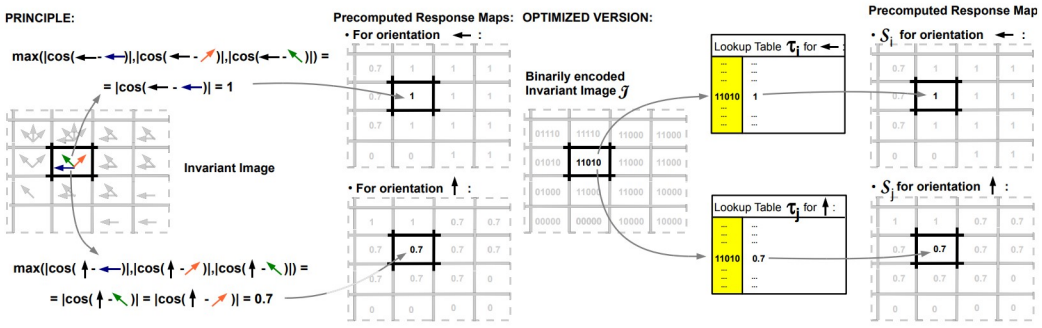


Figure 12: Precomputation of the response maps S_i. **Left:** There is one response map for each quantized orientation. These store the maximal similarity between this quantized orientation, and the corresponding combinations of orientations i the input image. **Right:** This process is done efficiently by using the binary representation in J as index to lookup tables of maximum similarity. (Source: [17])

3.2 Clustering

Clustering or cluster analysis is the task of classifying objects into distinct groups or classes based on their available data [24]. Objects with similar attributes will hence be clustered together, while more dissimilar objects will be placed in different classes. This multi-objective optimization problem can be solved by numerous different clustering methods, such as *connectivity-based clustering*, also known as *hierarchical clustering*, *distribution-based clustering* and *centroid-based clustering* [25]. *k-means clustering* is an example of the latter and is illustrated in figure 13. Given a fixed number of clusters k, this method minimizes the total Euclidean distance between all objects and their nearest cluster center by iteratively changing the centroid positions. The objective function can be expressed as:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} ||x_{i,j} - c_j||^2, \tag{14}$$

where k is the number of clusters, n_j is the number of objects in cluster j and $x_{i,j}$ is object of index i in cluster j. The center position of cluster j is given by c_j. As the optimization problem itself is NP-hard [43] the iterative approach will search for approximate solutions. Lloyd's algorithm [35], also known as *k-means algorithm*, does this by repeatedly assigning each object to its nearest centroid c_j before computing the new centroid positions as the mean of these objects. The expression for finding the closest cluster center for a given object x_i is shown in equation 15, while the computation of new centroid positions is shown in equation 16.

$$arg\min_{j} ||x_i - c_j||^2 \tag{15}$$

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_{i,j} \tag{16}$$

The algorithm stops when the object distribution converges, i.e. no objects are assigned to new clusters.

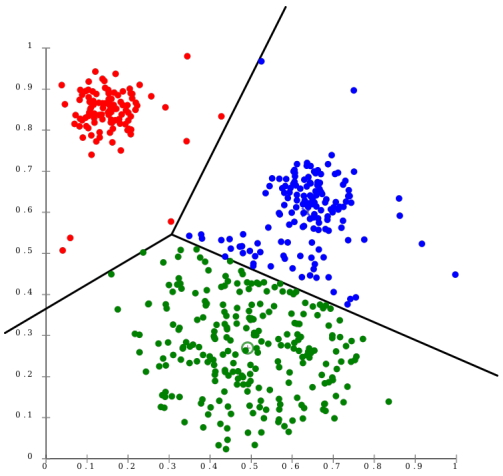


Figure 13: Two dimensional k-means clustering for $k = 3$. The different colors illustrate to which cluster the objects have been assigned. (Source: [7])

3.3 ICP

Iterative Closest Point (ICP) [4] is an algorithm used to minimize the difference between two clouds of points. As discussed in section 2.4.1, ICP is often applied when given an approximate initial pose estimate from a template based framework. Through an iterative scheme, the geometric distances are minimized, resulting in a translation and rotation which aligns the two point clouds as illustrated in figure 14.

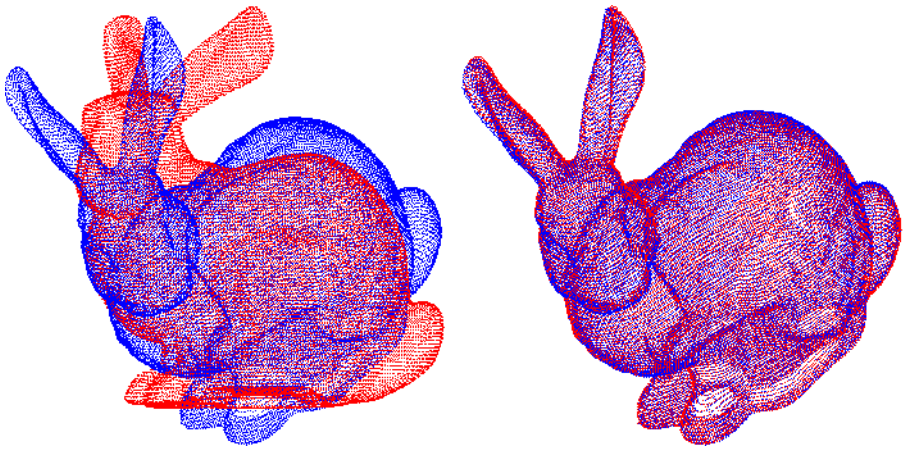


Figure 14: Alingning of point clouds using ICP. **Left:** Two clouds of points (blue and red) given as input for the ICP algorithm. **Right:** The red point cloud has been translated and rotated in order to minimize geometric difference to the blue point cloud. (Source: [27])

In contrast to *Kabsch algorithm* [20] and other solutions, ICP needs no correspondences between the two sets of points. Instead, for each iteration, every point in the source point cloud will be matched with the closest point in the reference point cloud. These matches will then be the basis for the subsequent geometric difference minimization, providing the transformation applied in the next iteration. Zhang [47] also proposes a modified *k-d tree* algorithm for efficient computation of the closest point matches. A statistical method furthermore takes care of outliers and variations in the presence and absence of corresponding object points.

4 Solution and Implementation

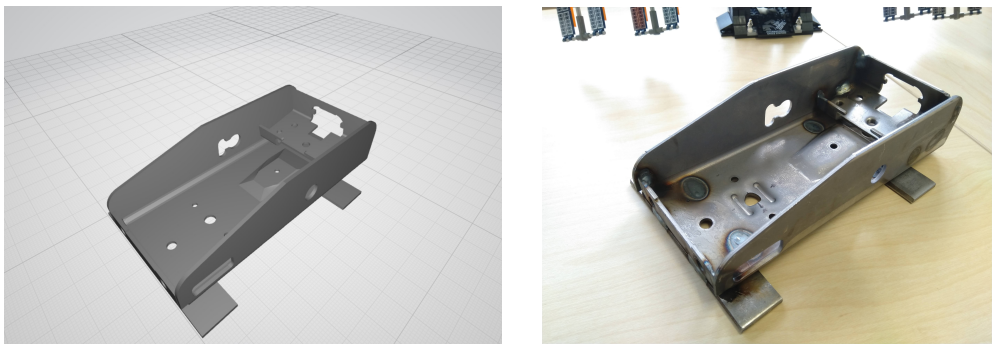
This section will focus on the implementation of the proposed pipeline as described in section 2.5. First, the offline template generating is explained in section 4.1, before the detector implementation is addressed in section 4.2. Both solutions were written in C++. For reasons to be discussed in section 6, the "match evaluation" and "pose refinement" modules were not implemented.

4.1 Template Generating

In order to obtain the required set of template images for the detector, a synthetic image renderer was utilized. Similar to the approach described in section 2.2.3, a 3D mesh model of the target object is given as input. The template images are then acquired by visiting a number of uniformly distributed viewpoints on synthetic spheres of different radii, while also rotating around the optical axis. The source code for this image rendering was provided by the *Object Recognition Kitchen* (ORK) [31]. For 3D model operations and scene creation, *Open Asset Import library* (Assimp) and *Simple DirectMedia Layer* (SDL) functionality is employed. In addition, the *Open Graphics Library* (OpenGL) yields the more elemental graphics operations.

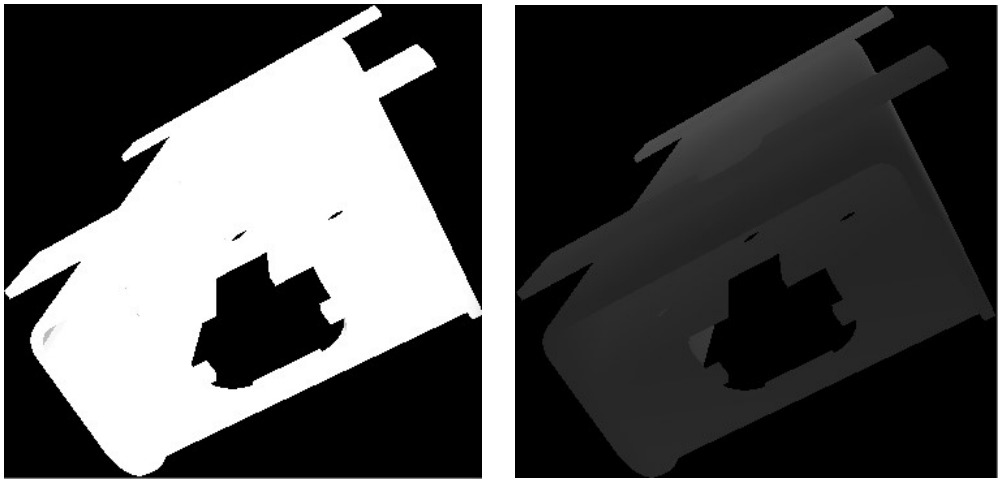
The implementation of this image renderer proved to be a rather demanding process. Although the source code was easily accessible, integrating the different libraries was quite challenging. For instance, not having the correct extension or version of different software caused a lot of time-consuming troubleshooting. This was particularly the case when trying to make use of some additional required OpenGL functions. Various OpenGL extensions, such as *GLFW*, *FreeGLUT* and *GLEXT* were all tested before finally making it work by directly adding the source code from *OpenGL Extension Wrangler Library* (GLEW) into the project. Otherwise, the binary 64 bit versions of SDL 2.0 (SDL2) and Assimp 4.1.0 were both installed before linking these to the project by adding their respective *Dynamic-link library* (DLL) files to the repository. The same was done for FreeImage 3.18.0, which provides image reading and converting functionality for the *model* module in the 3D renderer implementation.

Seeing that a 3D mesh model is required for this template generating approach, models of the objects from figure 2 would be necessary for performing the intended object detection. A 3D model of the large chair part was therefore created using the free and open-source software *FreeCAD* [32]. An illustration of both the 3D model and the actual object is shown in figure 15. By using this model, and the image renderer from ORK, template images such as the ones from figure 16 were produced.



(a) (b)

Figure 15: 3D mesh model and real image of the object to track. **a)** 3D mesh model created in *FreeCAD* by project supervisor Klaus Ening. **b)** Illustrative image of object.



(a) (b)

Figure 16: Sample of produced template images. **a)** Template color image. **b)** Template depth image. This image was edited in order to increase visibility.

As mentioned in section 2.2.1 on LineMOD, the color gradients for texture-less objects will mainly be located on the contours of the object. Consequently, the 3D models require no data concerning the object texture. The binary image representation, as illustrated in figure 16a, will furthermore make the detection more robust to the varying surface texture of target objects in the pre-paint stage as addressed in the problem description in section 1.

4.2 Detector

Given a set of template images, the work of producing efficient modality descriptors and performing template matching still remains. As suggested by the proposed detection and 6-DoF pose estimation pipeline in section 2.5, this was done using a LineMOD detector. This detector implementation was provided by OpenCV, or more precise, their additional repository for unreleased modules called *opencv_contrib* [1]. Despite having to do some extra installs, and linking these to the repository, the process of applying this detector was not too challenging. It did however require a rebuild of the original OpenCV repository in order to add the extra module, *rgbd*. This build was executed using CMake GUI.

The offline template loading, i.e. the response map computation described in section 3.1.3, is performed for one template image pair at a time, as the 3D model is being rendered. When the rendering is done and all templates have been added to the detector, all information is written to an *Extensible Markup Language* (XML) file. In addition, an XML file is created containing rendering parameters like the synthetic camera incintrics and the rotation matrices associated with the id of the corresponding template instance. This approach was inspired by [44]. After loading a given detector and the related parameters from the rendering, the approximate rotation of a target object can be found

through the id of the matched template. In addition, the approximate translation can be estimated from the image coordinates of the match, along with the camera depth image and the camera incintrics. First, the 3D points for the matching object is found using the camera depth image and the camera incintrics. This functionality is provided by a function called *depthTo3d* from the *rgbd* module, and is primarily applied for the possibility of executing a subsequent step of ICP. The 3D points do however give an approximate camera-to-object translation by selecting the coordinates belonging to the center of the matching frame. Examples of such matching frames can be found in figure 17 and 18 in section 5, illustrated by green rectangles. The final step of finding the initial translation estimate is to add the distance from the object surface to the object coordinate center for the given template match. This distance is saved among the rendering parameters and is added to the to the z-axis (camera depth) of the translation. Accordingly, an initial pose estimate can be provided for all potential matches attaining a similarity score above a given threshold.

For handling of the RGB-D sensor data, *Azure Kinect Sensor Software Development Kit* [9] was utilized. In order to reduce computational time, the lowest resolution (1280 x 720) was selected. This toolkit also provided functionality for aligning the depth image into the format of the color camera image, which is required for the lineMOD detector approach.

5 Results

In this section some results from the object detection and initial pose estimation is presented. Referring to the pipeline sketch in figure 8, these will correspond to the the preliminary matches from the detector. Although a complete pipeline will not be implemented before the master thesis, these matches should still contain the information needed to make well-founded assumptions on the presence of target objects, and furthermore give a decent initial pose estimate. For practical reasons, the detection was only tested for the large object illustrated in figure 15b.

As there are many different parameters to consider for the detector, it is difficult to decide an optimal configuration which gives the best results. By changing the sampling step T, illustrated in figure 11, the image resolution, and the number of templates and their geometric distribution, both the detection rate, the initial pose precision and the computation time for matching are affected. However, the number of templates seems to mostly affect the computational time, while not influencing the detection rate. The pose estimation precision also appear to be rather unaffected by moderate changes in the number of templates. For the presented results, a set of approximately 6000 - 8000 templates were used, uniformly distributed on six spheres with radii between 0.4 m and 0.9 m. Using a combination of two different sampling steps, with $T_1 = 8$ and $T_2 = 10$ seems to give the best results. This two-leveled pyramid approach does however increase the computational time for template generating and matching, as an extra set of precomputed response maps must be created and evaluated. By cropping the RGB-D camera images from 1280 x 720 pixels to 640 x 480 pixels, the matching time is reduced by approximately 60 % at the expense of reducing the field of vision. Results from both formats are presented in this

section. For the template generating, the template resolution should remain the same as the original camera resolution, as a change of resolution would affect the response map computation. The input image cropping evade this problem, as the pixel-to-point distribution remain unchanged. Using this camera resolution, and the mesh model from figure 15a, the set of templates is created in around 40 - 50 minutes. Thereafter, the detector can be loaded in approximately half a minute given the implementation in section 4.2. The matching on the other hand can be done in approximately 0.5 s or 1.1 s for cropped and non-cropped input images respectively. All computational times are based on the performance of a personal desktop with 1.80 GHz Intel Core i5 CPU and 8 GB RAM.

The results from the object detection and initial pose estimation are presented in figures 17 - 23. The green rectangles illustrate the matching frame, while the yellow dots illustrate the modality feature locations corresponding to the color gradient orientations. The surface normal features, which are distributed within the contours of the potential matches are not demonstrated in these images. In figure 19 and 21 the poses of the best matches from figure 17 and 20 are also illustrated by plotting their respective local coordinate axes onto the color image. All matching results are obtained using both the color images and their attached depth image. As a consequence of low visibility however, the depth images are not presented in this section.

When studying the results, it is clear that the implemented detector is capable of detecting the target object in several of the presented images. In figure 17, 18 and 20 particularly, there are multiple matches concentrated around the real object. Overlapping true matches like these are especially important when dealing with the presence of false matches. While some of these false matches can be easily thrown away by cross-checking the matching frame size and the depth value at this location, and thus detect mismatches in object size, a cluster of matches can often be necessary in order to exclude other singular false matches.
The process of picking out the true matches become a lot more difficult when dealing with only a few true positive detections, which is the case for the detection in figure 22 with just one positive match on the target object. Needless to say, when given no true positives, as in figure 23, the tracking is lost and there is no way to give an estimated 6-DoF pose of the object. Consequently, one could argue that the current implementation of the object recognition and motion tracking system has some flaws in relation to the recognition of the target object.
In addition to the detection rate, the precision of the initial pose estimates are also key for the system performance. As illustrated in figure 20, even the best matches can still provide poses which deviate from the true object pose. In this case however, a subsequent ICP step could give a much better final pose estimate. At first glance, the initial pose estimate in figure 19 looks better, but as the object in reality is flipped the other way around, a subsequent ICP step could converge to a local minimum [46], still being upside down. This could also be the case for the singular positive match on the object in figure 22, which is clearly upside down relative to the real object.
This match also demonstrate a notable tendency of the detector to favor templates with a uniform flat surface. The detector seems to struggle more when the camera is not aligned along the coordinate axes of the object. In addition, plain surfaces are the source of most

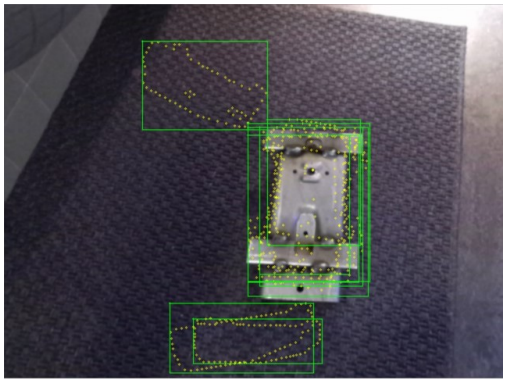


Figure 17: Positive template matches

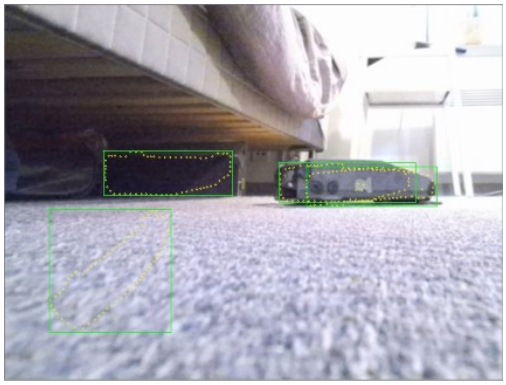


Figure 18: Positive template matches

false matches, as they are being matched with flat surface templates.

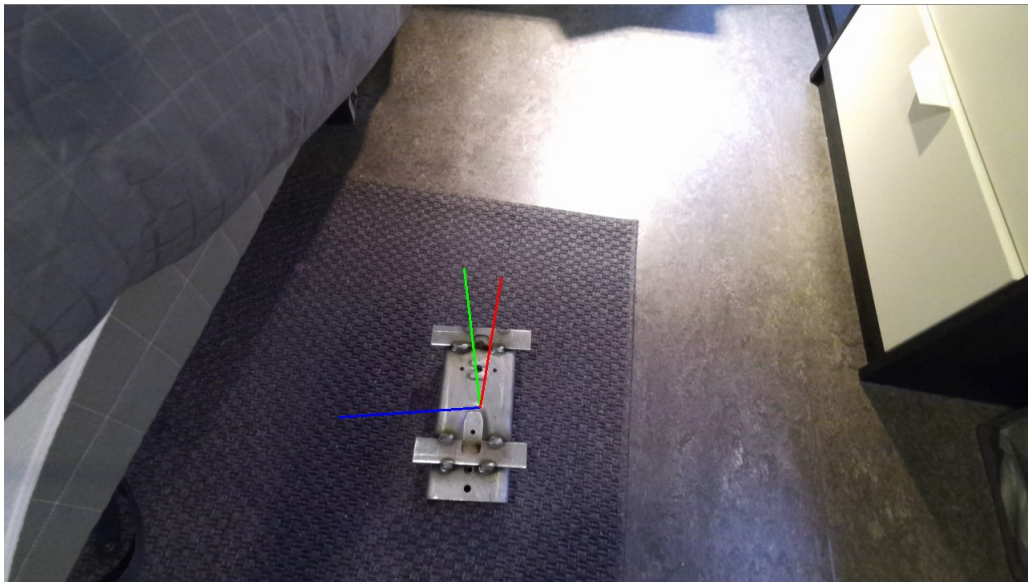


Figure 19: Camera color image with local object coordinate axes corresponding to the best match from figure 17

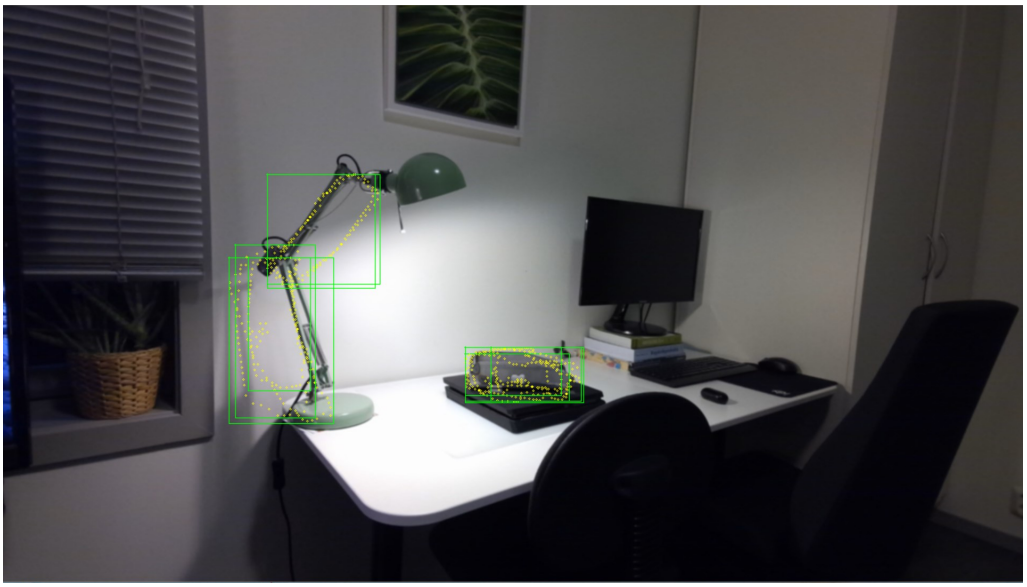


Figure 20: Positive template matches

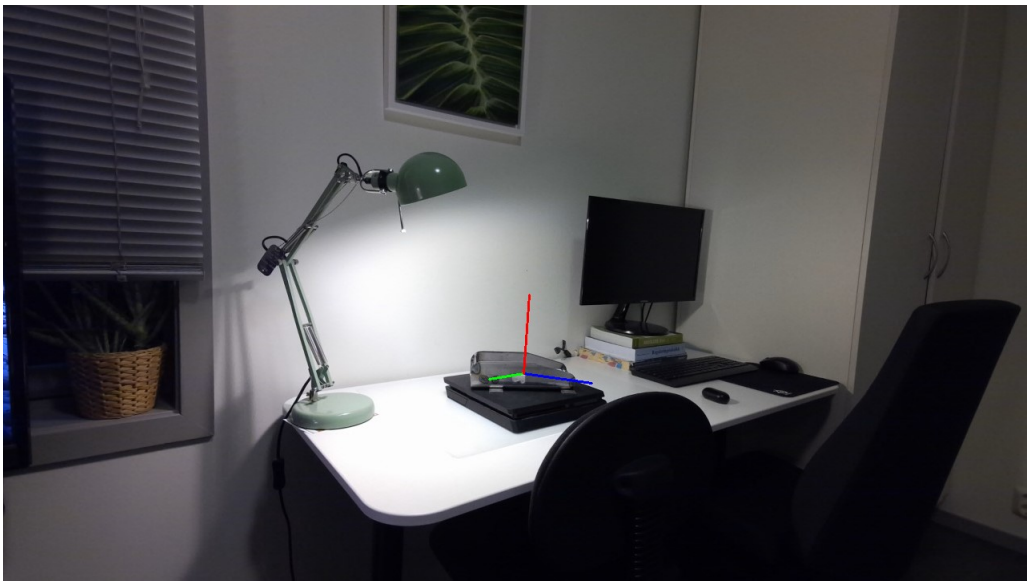


Figure 21: Camera color image with local object coordinate axes corresponding to the best match from figure 20

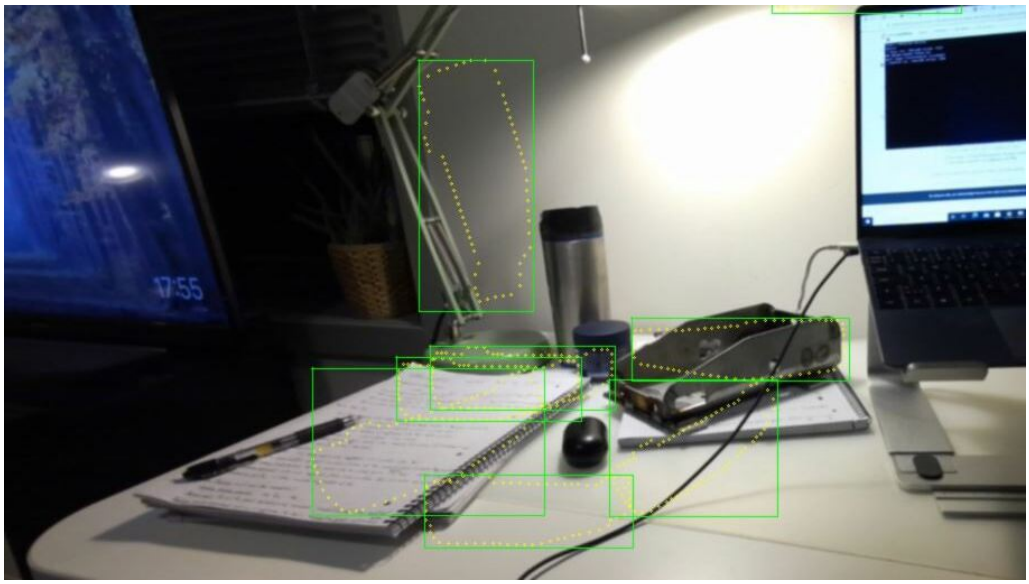


Figure 22: Positive template matches

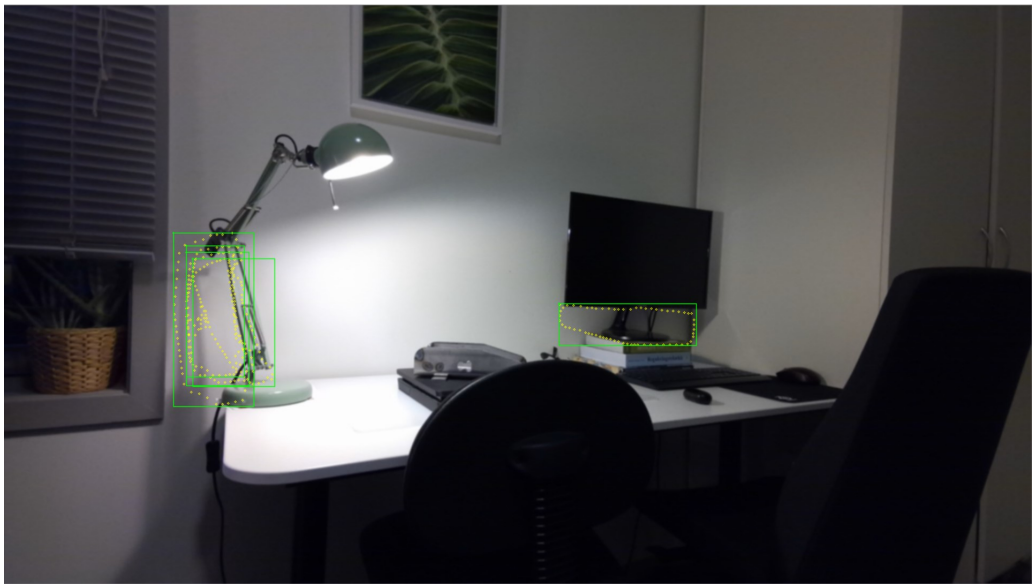


Figure 23: Positive template matches

6 Discussion and Future Work

Overall, the results from the implemented solution clearly shows some potential. The detector usually finds at least one approximate match for the target object, whilst not obtaining too many false matches. For future work however, i.e. during the master thesis, it would be preferable to attain a higher share of matches on the object. It would also be preferable for these matches to present some more precise pose estimates. While small deviations can be corrected by some subsequent pose refinement, this will most likely not be the case when the real object is flipped a different way around, as was pointed out in section 5. These improvements would also permit the utilization of the "match evaluation" and "pose refinement" modules from the proposed pipeline in 2.5, which naturally require a decent selection of potential true matches in order to deliver a proper final estimate of object poses.

The shape of the target object could be a contributing factor to the insufficient recognition rate for the detector. As expressed in section 5, the detector struggles more when the object is not perpendicuarly aligned along the camera's line of view. The current implementation clearly promotes flat uniform surfaces, while object poses not fulfilling this criterion are overturned by false matches, primarily from plain surfaces in the scene. It arguably seems like the discontinuity of the multiple flat surfaces on the target object provides a more challenging task of surface normal matching. The LineMOD source code from [1] indicate that when computing normals, the contribution of a pixel will be ignored if the depth difference with the central pixel is above 50 mm. This threshold should be suitable for the presented object. Nevertheless, the object characteristics appear to complicate the object detection. As an initial poof of concept, the 3D renderer and LineMOD detector was tested using a 3D mesh model of a soda can and some RGB-D images from a kitchen-scene dataset including similar shaped soda cans. In general, this test displayed some very good result, clearly pointing out the target object from the scene. Although the sensor data quality in theory could have an influence, the uniformly curved surface seemed easier to distinguish from the surroundings scene. However, there are some alterations which may improve the detector performance. For instance, more testing of the sampling step T, or additional filtering of the sensor depth data might result in a better detector configuration and more true object matches. Better lighting conditions while testing could also improve the detection rate, by making the object contours easier to detect from the background.

The ability of real time performance is also an important aspect of all object recognition and motion tracking systems. For the problem described in section 1 particularly, fast detection of objects is crucial as we are dealing with a free swinging hanger for loading and unloading. Based on a detection time of 0.5 s for cropped camera images, and furthermore additional computational time for subsequent clustering evaluation and ICP, the computational costs will arguably exceed what is considered real time capable. However, as discussed in section 5, there are ways to reduce the computational time of the presented object detector. For instance, reducing the camera resolution, or the number of templates, can both make the detection faster at the expense of possibly also reducing the detector precision. As discussed in 2.2.1, hashing-based [21, 19] or tree-based [36, 33]

methods could also be applied. By searching for approximate solutions, these methods achieve sub-linear complexity for the template matching. [18, 17] also state that using a GPU can speed up the mathcing process. A different approach could be to combine the implemented detector with a simpler motion tracker, only requesting a refined pose estimate from the full pipeline (figure 8) for every n-th RGB-D image recieved from the sensor. In other words, there are ways to make the implemented pipeline more capable of real time performance.

Apart from the detection rate, the initial pose estimate precision and the computational properties, which have already been discussed, the implemented framework also provides some very practical template generating functionality. In order to track a new object, only a 3D model of the object is required. For common daily-life objects, these can easily be found online. For more unique objects, like industrial parts, the models can either be provided by the manufacturer, or be created using a 3D scanner. Anyhow, this flexible solution enables efficient generation of templates, which furthermore can be loaded to a detector in less than a minute.

Having all of this in mind, it is tempting to see if the proposed modifications can improve the detection rate of the current implementation. Even though the LineMOD detection approach has appeared to be rather challenging for the target object, the proposed pipeline still seems like a viable option with a decent potential. If there is no progression however, it might be necessary to look at different detection methods. If that is the case, Stoiber et al. from section 2.2.2 recently published their paper [38] on their sparse gaussian approach to region-based object tracking, which seems promising. As their source code is available on github [39], it would be interesting to see if this approach could work for this problem.

7 Conclusion

Based on the provided object tracking problem, which includes a free-swinging hanger for loading and unloading, and a brief literature study of existing methods, a feasible template based framework for object detection and 6-DoF pose estimation is proposed. Using a synthetic template rendering approach, new objects can easily be added to the detector. The LineMOD matching strategy then combines the color gradient and depth normal orientations to find the template RGB-D image and belonging pose which best fit the sensor data. Despite the presence of true matches in most of the presented results, the shape of the tested object seems to make the detection rather challenging. Increased detection rate and more precise initial pose estimates are preferable if additional steps of pose cluster evaluation and ICP should be implemented. Consequently, it would be desirable if modifications to the detector could improve the matching results. If not, it might be necessary to look for different detection strategies.

References

[1] Alexander Alekhin, Vadim Pisarevsky, Pavel Rojtberg, and Rostislav Vasilikhin. *linemod*. Open Source Computer Vision Library (OpenCV), Github repository: opencv/opencv_contrib/modules/rgbd, 2019.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. *"Computing and Rendering Point Set Surfaces"*. In the IEEE Transactions on visualization and computer graphics, volume 9: pp. 3–15, 2003.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *"SURF: Speeded Up Robust Features"*. In the proceedings of European Conference on Computer Vision (ECCV), pp 404-417, 2006.

[4] P. J. Besl and N. D. McKay. *"Method for registration of 3-d shapes"*. International Society for Optics and Photonics, Robotics-DL tentative, pp. 586–606, 1992.

[5] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. *"Uncertainty-driven 6d pose estimation of objects and scenes from a single RGB image"*. In the proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3364 - 3372, 2016.

[6] Oliver Brock, Jeff Trinkle, and Fabio Ramos. *"Model Based Vehicle Tracking for Autonomous Driving in Urban Environments"*. MIT Press, Robotics: Science and Systems IV, pp. 175 - 182, 2009.

[7] Chire. *KMeans-Gaussian-data.svg*. Wikimedia Commons, 2011.

[8] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. *"Model Globally, Match Locally: Efficient and Robust 3D Object Recognition"*. In the proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), pp. 998– 1005, 2010.

[9] Tetyana Sych et al. *Azure Kinect Sensor SDK*. Microsoft, 2019.

[10] M.A. Fischler and R.C. Bolles. *"Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography"*. Communications of the Association for Computing Machinery (ACM), volume 24: pp. 381–395, 1981.

[11] Ge Gao, Mikko Lauri, Yulong Wang, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. *"6D Object Pose Regression via Supervised Learning on Point Clouds"*. 2020.

[12] Iryna Gordon and David G. Lowe. *"What and Where: 3D Object Recognition with Accurate Pose"*. Toward category-level object recognition, pp. 67–82, 2006.

[13] Gina Gould. *"How Workplace Robotics Improve Safety and Health"*. Enablon, 2019.

[14] Carlton Gyles. *"Robots in medicine"*. Canadian Veterinary Medical Association, 2019.

[15] Ruotao He, Juan Rojas, and Yisheng Guan. *"A 3D Object Detection and Pose Estimation Pipeline Using RGB-D Images"*. In the proceedings of European Conference on Computer Vision (ECCV), 2017.

[16] S. Hinterstoisser, C. Cagniart, S. Ilic, P. F. Sturm, N. Navab, P. Fua, and V. Lepetit. *"Gradient response maps for real-time detection of textureless objects"*. In the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), volume 34: pp. 876–888, 2012.

[17] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. *"Gradient Response Maps for Real-Time Detection of Texture-Less Objects"*. In the IEEE Transactions on pattern analysis and machine intelligence, 2011.

[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 the proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 858-865, 2011.

[19] Tomas Hodan, Xenophon Zabulis, Manolis Lourakis, Stepan Obdrzalek, and Jiri Matas. *"Detection and Fine 3D Pose Estimation of Texture-less Objects in RGB-D Images"*. In the proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4421 - 4428, 2015.

[20] Wolfgang Kabsch. *"A solution for the best rotation to relate two sets of vectors"*. Acta Crystallographica, volume 32: pp. 922, 1976.

[21] W. Kehl, F. Tombari, N. Navab, S. Ilic, , and V. Lepetit. *"Hashmod: A Hashing Method for Scalable 3D Object Detection"*. Proceedings of the British Machine Vision Conference (BMVC), 2015.

[22] Manolis I. A. Lourakis. *"A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar"*. Institute of Computer Science, Foundation for Research and Technology, 2005.

[23] David G. Lowe. *"Distinctive image features from scale-invariant keypoints"*. International Journal of Computer Vision, volume 60: pp.91-110, 2004.

[24] David MacKay. *"An Example Inference Task: Clustering"*. Cambridge University Press, pp. 284–292, 2003.

[25] David MacKay. *"Clustering methods"*. Data mining and knowledge discovery handbook, Springer US, pp. 321-352, 2005.

[26] The International Federation of Robotics. *"The Impact of Robots on Productivity, Employment and Jobs"*. 2017.

[27] Maks Ovsjanikov. *"PS 1 - Rigid shape registration"*. Informatics laboratory of l'École polytechnique (LIX), Ovsjanikov notes.

[28] Nicolas Pinto, David D. Cox, and James J. DiCarlo. *"Why is Real-World Visual Object Recognition Hard?"*. PLoS Computational Biology, volume 4: pp. 151-156, 2008.

[29] Neelam V. Puri and P. R. Devale. *"Human Tracking and Pose Estimation in Video Surveillance System"*. In the proceedings of International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR), volume 3: pp. 257-274, 2013.

[30] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. *"PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation"*. In the proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77 - 85, 2017.

[31] Vincent Rabaud, Carlos M. Mateo, Scott K. Logan, Jimmy Da Silva, Natalia Lyubova, and Lincoln Lorenz. *ork_renderer*. Willow Garage - Object Recognition Kitchen, Github repository: wg-perception/ork_renderer, 2017.

[32] Jürgen Riegel, Werner Mayer, and Yorik van Havre. *FreeCAD*. 2015.

[33] R. Rios-Cabrera and T. Tuytelaars. *"Discriminatively trained templates for 3d object detection: A real time scalable approach"*. In the proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2048 - 2055, 2013.

[34] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. *"ORB: an efficient alternative to SIFT or SURF"*. In the proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2564 - 2571, 2011.

[35] Lloid S. *"Least squares quantization in PCM"*. In the IEEE Transactions on Information Theory, volume 28; pp. 129–137, 1982.

[36] Mang Shao, Danhang Tang, and Tae-Kyun Kim. *"Real-time Background-aware 3D Textureless Object Pose Estimation"*. 2019.

[37] Amit Shukla and Hamad Karki. *"Application of robotics in offshore oil and gas industry — A review Part II"*. Robotics and Autonomous Systems, volume 75: pp. 508-524, 2016.

[38] Manuel Stoiber, Martin Pfanne, Klaus H. Strobl, Rudolph Triebel, and Alin Albu-Schaeffer. *A Sparse Gaussian Approach to Region-Based 6DoF Object Tracking*. In the proceedings of the Asian Conference on Computer Vision (ACCV), 2020.

[39] Manuel Stoiber, Martin Pfanne, Klaus H. Strobl, Rudolph Triebel, and Alin Albu-Schäffer. *"A Sparse Gaussian Approach to Region-Based 6DoF Object Tracking"*. German Aerospace Center (DLR) - Institute of Robotics and Mechatronics (RM), Github repository: DLR-RM/RBGT, 2020.

[40] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. *"Real-Time Monocular Pose Estimation of 3D Objects using Temporally Consistent Local Color Histograms"*. In the proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 124-132, 2017.

[41] Alexander Toshev and Christian Szegedy. *"DeepPose: Human Pose Estimation via Deep Neural Networks"*. In the proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1653 - 1660, 2014.

[42] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. *"Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects"*. In the proceedings of the Conference on Robot Learning (CoRL), 2018.

[43] Andrea Vattani. *"The hardness of k-means clustering in the plane"*. University of California, San Diego, Vattani papers, 2010.

[44] Hongmin Wu. *"linemod_pose_estimation"*. Biomimetics Robotics lab at Guangdong University of Technology, Github repository: birlrobotics, 2018.

[45] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. *"PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes"*. In the proceedings of the conference: Robotics: Science and Systems, 2018.

[46] Jiaolong Yang, Hongdong Li, Dylan Campbell, , and Yunde Jia. *"Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration"*. In the IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016.

[47] Zhengyou Zhang. *"Iterative point matching for registration of free-form curves and surfaces"*. International Journal of Computer Vision, volume 13: pp. 119–152, 1994.

[48] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. *"Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR"*. IEEE/ACM International Symposium on Mixed and Augmented Reality, volume 7: pp. 193 - 202, 2008.