3D Pose Estimation of known textured objects using Monocular Images for COMP9517 2016s2

Daniel Svensson
(Contributions: Front-End Development, PnP, 2nn-Matching)
School of Electrical and Computer Engineering
University Of New South Wales
Sydney, Australia

Pece Karadzovski
(Contributions: ORB-feature extraction, RANSAC, Testing)
School of Electrical and Computer Engineering
University Of New South Wales
Sydney, Australia

***Abstract*—Pose estimation is the elementary problem in a number of application, particularly where the position and orientation of an object is required. The aim of this project is to estimate the six degrees of freedom that describe the 3D pose of an object given a data stream from a single monocular camera. Fundamental in this process is a model of an object of interest from which feature matching between this object model and the data stream can be used to firstly recognise pixels as belonging to an object. Secondly, an iterative algorithm is used to estimate the pose. This paper will show that given good feature detection and matching methods and iterative algorithm the pose of a known object can be accurately and reliably estimated.**

1. Introduction

Pose estimation is an elemental problem in augmented reality, photogrammetry, and robotics, where the position and orientation of a known 3D object is defined from a 2D image data stream. The pose of an object is described using six degrees of freedom, referring to origin location and rotation about each principle axes. This project builds a real time application to estimate the six degrees of freedom that define the relative position and orientation between the camera and the 3D textured object using data stream from a single monocular perspective camera.

2. Literature Review

Approaches to pose estimation can be classified under several categories: template-based approaches; feature-based approaches; and dense based approaches. Template-based approaches employ edge-based distance metric and refine pose estimates using `ICP` to achieve accurate 6D pose. Sparse feature-based approaches perform well with textured objects. The general methodology is to extract points of interest and describe them with local descriptors before matching the descriptors against a database. In dense approaches every pixel produces some prediction about the desired output. But the general methodology for textured objects is generically similar, with the key to success for most systems being the use of a sparse representation of local features through methods like `SIFT` to describe the object in an image and then sparse feature matching and geometric verification of matched features through iterative processes to find the pose of the object.

2.1. State of the Art Methods

Popular state of the art methods in the literature fall under the template-based approach. Templates are built or learned from 3D models or a set of images of and object, before the six degrees-of-freedom pose and the pose is usually refined using an iterative process. State of the art methods [2], [3] uses analysis-by-synthesis as their underlying approach. Machine learning is used to obtain pixel-wise dense predictions, which is used to predict which object a pixel belongs to and where it is located on the surface of this object. In [3] decision forests are used to predict 3D object coordinates and object instance probabilities, an energy function based on forest output and `RANSAC` for optimisation, and in [2] a convolutional neural network inside a probabilistic context is used to learn to compare rendered and observed images as shown in figure 1.

In a forward synthesis, images are generated from a model from geometric interpretations of the world, and the interpretation that best agrees with the measured visual evidence, by way of minimising an energy function, is selected. In [10] a convolutional neural network is trained to compute descriptors, as a set of template views which are stored as the object model with the 3D pose of the view, and a scalable `k`-nearest neighbour search method in the descriptor space is implemented so as to handle a large number of objects in various poses. Current approaches rely on one classifier per object, so do not scale well to multi-object recognition and pose estimation. This project only considers the pose estimation for a single object. In [12] computational costs are transferred from matching to training. During training,

Figure 1. Results from [2] using convolutional neural networks for rubustness

at least one image of the target object is used to synthesise a large number of views of individual keypoints. The full pose is recovered from the matches using the POSIT algorithm in tandem with RANSAC, and successfully recovers the pose of a textured box in almost any position using six different views for training. In [11] an object detection and pose estimation algorithm was developed for the purposes of warehouse automation, which would involve pick-and-place tasks for products that are located on shelving units. It identifies accurate pose estimation as being crucial for success in this task. Another state of the art method [4] relies on greyscale images, noting that depth cameras fail on metallic objects, and states that pose prediction degrades in presence of large occlusion. It uses a part-based framework, and also assumes it is given a 3D model of the object. Other methods rely on part-based models for category recognition and pose estimation, where objects themselves are not recognised, but subsets are recognised as parts of the object. This aids in overcoming the challenges of occlusion and such that an accurate pose can be calculated. In the LINEMOD approach a 3D mesh object model is received, from which various viewpoints and features are sampled. The features are filtered to a robust set and stored as a template for the object and the given viewpoint. This is repeated until there is sufficient coverage from different viewpoints. The detection phase consists of a template matching algorithm followed by several post-processing steps to refine the pose estimate. LINEMOD uses surface normal in the template matching algorithm and limits RGB gradient features to the objects silhouette. Many papers, including state-of-the-art methods compare their results to LINEMOD and its dataset as a baseline, indicating that LINEMOD is a popular benchmark method and dataset to use.

2.2. More Conventional Methods

Information for pose estimation is usually given in the form of a set of point correspondences, each composed of a 3D reference point expressed in object coordinates and its 2D projection expressed in image coordinates. In [5] detections are initially passed through a colour check. Pixels that lie on the object projection are counted if they are the expected colour. False positives are eliminated by comparing them to a threshold. An Iterative Closest Point algorithm is used to align the 3D model surface with the depth map, with the initial translation estimated from the depth values covered by the initial model projection. The 3D points are subsampled from the depth map to match the object to the model, and if the expected depth sum is greater than a certain threshold then the object is found with the final pose. Global 3D object detection methods use statistical classification techniques to compare an input image of interest and decide whether or not it appears in the input image. Local approaches use simple 2D features such as corners and edges making them resistant to partial occlusion and cluttered backgrounds. These methods can still detect objects as long as enough feature points are found and matched. Local approaches have been shown to work well on highly textured objects, handle partial occlusion and tolerate errors in the correspondences. In this project a 2D feature detector will be used to find local features and descriptors since textured objects are the focus for this reason. One approach to get the pose is the Direct Linear Transform (DLT) algorithm. It estimates the projection matrix by solving a linear equations systems with unknown camera parameters. But this method should use an iterative optimisation to refine the non-linear re-projection error. Perspective-n-Point (PnP), is a variant of DLT and is commonly used to estimate pose. It assumes the camera intrinsic parameters are known. One PnP approach uses two algorithms: POS (Pose from Orthography and Scaling) and POSIT (POS with Iterations). It assumes that four or more non-coplanar feature points of the object can be detected and matches, and that the relative geometry of the object is known. The POS algorithm approximates the perspective projection and produces a scaled orthographic projection (SOP), which is an approximation to the perspective projection in which it is assumed that the depths of all points does not vary much from one another such that one depth can be considered representative. The POSIT algorithm uses the result found by POS and uses this to iteratively find a better projection matrix. It shifts the feature points iteratively to obtain a new orthographic projection of these shifted points. The iterative process continues until it converges to an accurate pose. Another PnP method [6] formulates the pose estimation problem by minimising an object-space collinearity error. From this objective function its algorithm operates on successively improving an estimate of the rotation and then an associated translation. This method has been described as being the most representative in terms of speed and accuracy, but relies on a good initial guess to converge to the correct solution. Other pose estimation methods exist that are non-iterative, including Efficient PnP (EPnP). These methods were designed because iterative solutions can get trapped in local minimum, where the aim is to get the global minimum. N3M [9] works well for partial occlusion. It uses an offline learning stage to consider both photometric and geometric properties simultaneously during the detection and pose estimation. N3M consists of a set of

Figure 2. Results from [7] using kalman filtering for rubustness

Figure 3. The dataset LINEMOD

4 or 5 close feature points with distinctive photometric and geometric properties. In [6], [7] the tracking by detection methodology, in the form of Kalman Filter, is used to recover the pose considering natural feature points and a previously registered 3D textured model, the result of [7] is shown in figure 2.

2.3. Dataset

LINEMOD was designed explicitly in the context of object detection and pose estimation, with the focus on 6D pose estimation. LINEMOD contains training and test data for object recognition and pose estimation of 15 objects with accurate ground truth annotated images, shown in clutter from a number of viewpoints. It consists of over $18,000$ images of the 15 objects including an ape, benchvise, bowl, can, cat, cup, driller, duck, glue, holepuncher, iron, lamp, phone, cam, and eggbox as shown in figure 3.

A 3D mesh for each object is also provided, and also has RGB image sequences and depth maps recorded with a Kinect sensor. Many papers have constructed their own additional dataset to better equip the community in evaluation and improving robotic perception solutions. The dataset in [11] includes over $10,000$ depth and RGB registered images with complete hand annotated 6DOF poses, as ground truth, for 24 APC objects, as well as 3D YAML mesh models. It compares its dataset against LINEMOD to highlight the need for additional varying conditions to accommodate clutter, noise and illumination. Desk3D was introduced in [1] to demonstrate the performance of instance recognition algorithms in clutter and under occlusion and also to improve

Figure 4. LIU contains a mousetrap object with sufficient texture and a rectangular shape

upon the LINEMOD dataset. Each scene point-cloud is obtained by integrating a few frames of depth data from a Kinect sensor to aid in better extraction of edge features. Desk3D contains multiple desktop scenarios where each test frame is obtained by integrating a few frames from a Kinect sensor. The Desk3D dataset contains 6 object instances face, toy Ferrari, Kettle, toy Mini, Phone, and Statue, and consists of four scenarios to test varying degrees of clutter, occlusion and similar looking feature descriptors. [3] also has a dataset of $10,000$ images of 20 objects captured under three different lighting conditions and labelled with accurate 6D pose and is publically available. Due to the significance of LINEMOD in the literature as a baseline comparison and because of its relevance to pose estimation it has been decided that the LINEMOD dataset should be used for evaluation, although it must be noted that other dataset exists and some have been briefly discussed. LIU pose estimation data set [13] contains 16 objects in a variety of rotations and translation in both the presence and absence of clutter. Considering the problems the authors had with evaluation of the Desk3D and LINEMOD data sets, in that the simpler mesh objects are texture-less or texture-poor or have too complicated a mesh to model accurately it was decided to undertake testing and evaluation on the LIU dataset, which is older but contains an object that is both simple enough to build a mesh and has enough texture to build viable descriptors. This object is a mousetrap as shown in figure 4. [13] compares performance of pose estimation using fourteen types of local descriptors, and has shown that SIFT on average perform better.

3. Challenges

According to leading papers in this field the most challenging problems in pose estimation include: *Background Clutter* refers to the presence of excessive useless visual stimulus, such as additional objects in a scene. Background clutter is known to cause difficulty for pose estimation algorithms. The presence of even a small number of additional objects with vaguely similar colour or other visual features can cause simple approaches to fail causing many false positive identifications. For this reason robustness in the

algorithm needs to be incorporated for object recognition and pose estimation. *Noise* refers to irregular fluctuations in the data stream and can be present to many reasons. Camera sensors bring noise, so we need a robust method to deal with it. A robust method to eliminate outliers from feature selection and matching is implemented in the form of `RANSAC`. *Texture-less objects* are objects that have no texture, including transparent objects. Template-based techniques have been demonstrated to be superior for texture-less and texture-poor rigid objects. For the purposes of this project we assume that we will not have to deal with texture-less objects. *Occlusion* is a well-known challenge in many computer vision applications and refers to partial or complete obstruction of an object. Start of the art approaches use machine learning methods like random forests to train and predict which pixels in an image belong to an object. Template-based techniques do not do well in clutter and occlusion. Sparse feature-based techniques are more robust with respect to occlusion. *Changing Illumination* refers to changes in the intensity of an image. Many successful methods to find objects in an image use colour as a prominent feature. But colour methods are susceptible to changing illumination, and should not be the main attribute for object selection.

4. Specification

This project will be evaluated using a benchmark dataset which considers clutter and occlusion, as described in the Dataset section. This dataset contains a model for each of its test objects and the ground truth for comparison. In [8] the `POS`/`POSIT` algorithm performance is evaluated by computing the orientation and position error, where orientation error is the rotation angle in degrees. State of the art methods [2] evaluate the performance of its algorithm by calculating the percentage of correctly predicted poses for a given sequence. It calculates the average distance between the 3D model vertices under the estimated pose and under the ground truth pose and assigns it as correct when average distance is below 10% of the object diameter. In [8] pose estimation is considered successful is the error of rotation is less than 5degrees and error of estimated translation is less than 5cm on each axis. The number of correctly recovered poses is counted and an indicator of the performance. As per other papers, the performance of this algorithm will be evaluated by computing the average orientation and position error, and the percentage of correctly predicted poses compared to the ground truth dataset. Accuracy is defined as the percentage of test frames for which the error is below a certain threshold, otherwise cited as the pose estimation of a frame is correctly predicted. This threshold for correct pose prediction will be defined to be 5degrees rotation and no greater than 5cm on each axis. Another relaxed accuracy will be measured and is defined as a up to 10degrees rotation and up to 5cm translation error.

The evaluation criteria is the percent of correctly defined poses for the number of frames. This is defined as:

$$\textit{Accuracy = frames with correct poses / total frames} \quad (1)$$

The accuracy has been calculated using the following formulas:

$$CT = \begin{cases} 1, \text{if } \forall i \ T_i - T_i' < t \\ 0, \text{otherwise} \end{cases} \quad (2)$$

Where CT is a correct translation, T_i is the subset of the predicted pose vector relating to translation, T_i is the subset of the ground truth pose vector relating to translation as provided in the dataset, and t is the defined threshold.

$$CR = \begin{cases} 1, \text{if } \forall i \ R_i - R_i' < \theta \\ 0, \text{otherwise} \end{cases} \quad (3)$$

Where CR is defined as a correct rotation, R_i is the subset of the predicted pose vector relating to rotation, R_i is the subset of the ground truth pose vector relating to rotation as provided in the dataset, and θ is the rotation threshold. Then finally a frame with correct pose is defined by:

$$\text{correct pose} = \begin{cases} 1, \text{if } CT \cdot CR = 1 \\ 0, otherwise \end{cases} \quad (4)$$

This final equation states that if all dimensions are within their threshold then the pose is considered to be classified correctly. Each frame is calculated this way, the accuracy formula is applied.

5. Methodology

Project execution can be separated into four main subsections: model registration, model detection, pose projection, testing and evaluation. An overview of this process is given in figure 5.

5.1. Model Registration

This section involves generating a model and mesh of 3D textured object and/or reading an existing 3D textured object model and object mesh. A model is crucial in recognising a specific object. Model registration should be done offline and prior to detection stage, especially if the model registration process is elaborate as in state of the art methods. The model is composed of a set of 2D feature descriptors about the object and associated 3D coordinates with respect to the object reference frame. In this project model are incorporated in two ways: for the evaluation of the algorithm in which the authors of this paper attempt to use existing models from existing benchmark datasets like `Desk3D` and `LINEMOD` and a hand generated model of a simple textured rectangular prism for purposes of live demonstration. For the former, existing models and 3D meshes are provided with the dataset for the purposes of training a model. In many cases a ply mesh model or a `YAML` file accompanies the dataset for each of the objects in that dataset. This requires

importing the models into the designed program and verifying that the model is corrected loaded, and trained. In the latter, a full model of a rectangular prism has to be generated by hand. A mesh of the object is created segmenting each face into triangles, as is common in most mesh models of an object. An image, or set of images from various angles, is captured with known intrinsic camera parameters is also required. Together the mesh and the images can be used to extract meaningful 2D feature descriptors and associated 3D coordinates with respect to the object reference frame for the object. This set of meaningful descriptors is the model for which the algorithm can undertake object recognition and pose estimation. In this project, the algorithm to generate a model from a mesh and input image was implemented from available open source methods as per [7].

5.2. Model Detection

This category implements the core for object detection from a data stream. Once object vertices are defined we have a sufficient set of 2D-3D correspondences to apply `PnP` to estimate camera pose. Input from a camera or a video is processed to extract and descriptors of the scene so that it can match these descriptors to the model/mesh that was built in model registration. Scale-invariant and illumination-invariant features and descriptors are required so as to tackle some of the known challenges. Lowes `SIFT` descriptor has been shown to be one of the most efficient descriptor methods for object recognition. It has also been suggested that alternate methods such as SURF and ORB can also be used to improve computation performance, for this reason `ORB` is used in this project since we have a timing constraint. It should be noted that a change in feature generation here means the same change in the model generation in the model registration phase. Feature matching is done after feature detection and can be done using brute force or `FLANN`, before match filtering is undertaken. In the first instance the two nearest neighbours are generated for each valid descriptor and the Nearest Neighbour Ratio is applied with a value of 0.8 to remove features that are not distinctive enough. Robustness is introduced so that noise and outliers are excluded, in the form of `RANSAC`. The pose can be estimated from these matching features solving the set of points as a linear equation, or applying one of the `PnP` methods discussed in literature review. In this project a `PnP` method is used to iteratively update a pose estimate until a probable pose is generated by way of convergence.

5.3. Pose Projection

Once the pose is estimated, a mesh and the local coordinate system are projected onto the video stream to show and visually verify the object pose in the image.

5.4. Testing and Evaluation

The final category involves testing the program to ensure it operates correctly and evaluating the performance of the program to the dataset/s described in the dataset section of this report using methods described in the evaluation section. The approach in [10] is compared to `LINEMOD` and `HOG` on the `LINEMOD` dataset. In this paper the `LINEMOD` is not very useful since it doesn't contain any densely textured rectangular objects. In [13] the `LIU` dataset is introduced which fits this paper better since it contains some rectangular textured object. Amongst other objects it contains a mousetrap model which will be evaluated against in this paper. The performance against the `LIU` dataset compared to the results in [13] will provide a baseline comparison for the implemented algorithms.

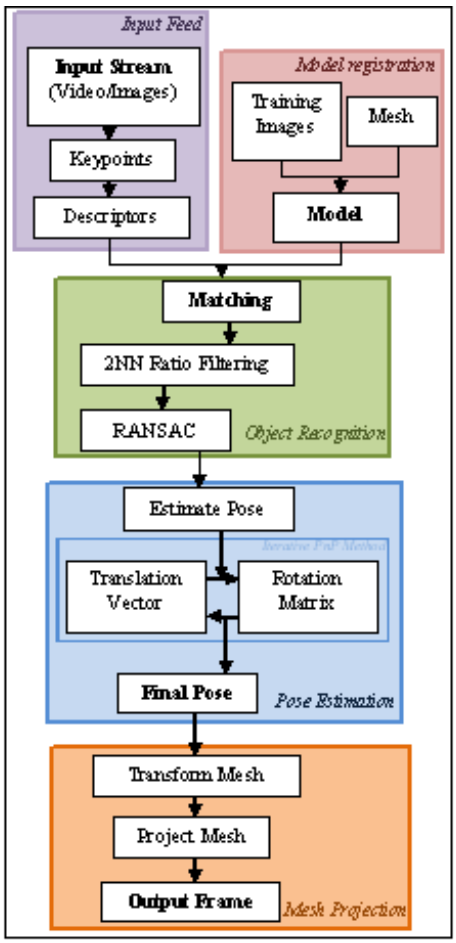

Figure 5. Pose estimation pipeline as implemented in this project.

6. Algorithms

In the literature there is a variety of algorithms used for object recognition and pose estimation. This section discusses the algorithms used in this project and justifies their use over other algorithms.

6.1. Model Registration

Model Registration is the process of converting a known object into a mesh and descriptors. The methodology described this as reading a mesh and image or set of images and extracting viable descriptors. The built model consists of a set of 2D points, 3D point correspondences and associated descriptors. In this project we generate the mesh by creating a PLY file with key vertices. Object faces are represented as triangular segments connecting the vertices. This PLY file is then provided along with at least one image to an open source algorithm to generate descriptors for the model. Training the model with more than one image from a vastly different pose proved to generally increase the performance of pose estimation. The supplied PLY file and user interaction is used to identify the key vertices in the images, before extracting the descriptors for each face. Naturally using this algorithm limits the complexity of the model.

6.2. Object Recognition

The object recognition section of the algorithm consists of three subsections extracting the features from the input frames and matching descriptors to the model, identifying distinctive descriptors using `2NN` ratio testing and removing outliers using `RANSAC` this is shown in algorithm 1.

Algorithm 1 object descriptor matching algorithm to identify an object from the model in the current frame

INPUT: the current frame f and mode M
OUTPUT: A set of descriptors d that identify the object.

```
keypoint ← frame.getKeyPoints()
descriptor method ORB.create();
descriptors ← ORB.descriptors(keypoints, f);
matches ← 2NNmatch(descriptors, M);
goodmatches ← [];
for all match in matches do
    if match[0] / match[1] < threshold then
        goodmatches.push_back(match[0]);
    end if
end for
d ← goodmatches;
```

Firstly keypoints are extracted from the frame and descriptors are built using the `ORB` descriptor method. `ORB` is used as a fast alternative to `SIFT` and is justified given that the ambition of this project is to implement pose estimation in real time. Additionally [13] has demonstrated that `SIFT` based methods produce relatively better results as compared to other methods. Next descriptors in the frame are matched to the two nearest neighbours in the model, before `2NN` ratio test using a threshold of 0.8 is applied to remove points that are not distinctive enough. `RANSAC` is then applied when generating the pose estimate by using a `PnPRansac` method, as implemented in `OpenCV C++` environment, to remove outliers prior to pose estimation. The `2NN` ratio and `RANSAC` provide robustness to the algorithm, allowing for objects to be recognised and their pose estimated in clutter.

6.3. Pose Estimation

The algorithm below describes the iterative approach taken to pose estimation. It takes an estimate of a pose and iteratively improves the translation vector and rotation matrix using the 2D-3D correspondences and descriptors in the model. The iterative process continues until the solution converges to the truth or a count threshold has been achieved. The algorithm returns the predicted pose as shown in algorithm 2.

Algorithm 2 Iterative PnP algorithm to estimate pose

INPUT: set of Descriptors d, pose estimate D
OUTPUT: estimated pose D'.

```
Threshold = 500;
Tolerance = 0.1;
Count = 0;
Difference = +∞;
while Count < Threshold || Difference > Tolerance
do
    update Translation Vector, T;
    update Rotation Matrix, R;
    D' ← {T_x, T_Y, T_z, R_x, R_y, R_z};
end while
Return D';
```

Of the many `PnP` algorithms that exist, the iterative algorithm is the most logical in terms of conception and one of the simpler algorithms to implement. These factors align with our implementation time frame and the fact that [12] states that it produced reasonable results justifies its use in our application.

7. Results

The algorithm implemented is evaluated against a dataset to reveal its validity and correctness, using the procedure as described in section 6 to estimate the pose and then process the results as defined in section 4 to determine whether a frame is firstly correct and then the accuracy on the dataset. In section 2.3 three prominent datasets where described `Desk3D`, `LINEMOD` and `LIU`. This section will briefly describe the implementation issues with testing against the former two datasets before discussing the results obtained with the latter.

7.1. Evaluation Issues

Since `Desk3D` is the most recent dataset it was decided to look at this dataset to begin with. But since few papers cited this as a reference, as compared to `LINEMOD` it is justified to use the `LINEMOD` dataset so as to provide a comparative result for the method implemented in this paper. Two objects of the `LINEMOD` dataset were considered for

evaluation the egg carton and drill. An approximate mesh and model were built for both the egg carton and drill, and the process of evaluation was applied firstly to the egg carton. The result showed an accuracy of zero percent. A brief analysis into the results showed that the problem lies in the texture of the model. Although the object is not texture-less it is texture-poor with led to few to no descriptors being identified in each frame. This means that the pose could never be estimated correctly, and also identifying this particular object as irrelevant to our method. The drill, on the other hand, has definable texture but the model was too intricate to represent it using several prisms. In this case, the model of the drill failed to register during the model registration phase, and it was impossible to attain any valid result using this object. The remaining objects in the LINEMOD dataset have these mentioned problems either the object shape is to intricate to build a mesh and model or the object is simple enough to build a mesh, but the object is texture-poor. Here lies the implementation issues with the algorithm in this paper the object of interest needs to be shaped simply enough to build a basic mesh and model and textured enough for key features and descriptors to be identified.

7.2. LIU mousetrap

The LIU dataset is an older dataset, compared to LINEMOD but has sufficient content to produce a valid evaluation. Of the objects in the dataset, only one holds both the properties that is defined in the previous section the mousetrap. The mousetrap has in excess of 600 frames taken from various angles and poses in the absence of clutter and in clutter contributing to in excess of 1200 frames for evaluation. The evaluation methodology is applied as described in the beginning of this section, and the results of the algorithm implemented in this paper as well as the results of the paper that generated the dataset will be discussed in the following section.

7.3. Accuracy Results

The results of applying the algorithm to the LIU [13] dataset is shown in table 1. However, our results only shows estimation of the mousetrap model whereas the results from [13] shows the results of all objects in the dataset. The results for applying the algorithm to LINMOD is omitted since the results were close to zero as explained in section 7.1. The accuracy and the relaxed accuracy is also defined in section 4. A picture of the pose estimation in progress is shown in figure 6.

7.4. Discussion

The results obtained appear to be consistent with the average accuracy for the LIU paper [13] but not as great, partly because the algorithm implemented attempts to speed up the process so that it is possible to use it in real time.

Figure 6. Pose estimation of the mousetrap object from the LIU dataset

TABLE 1. TABLE OF RESULTS RUNNING PROPOSED ALGORITHM ON LIU DATASET

	Without clutter	In clutter	Avg
Accuracy	53.8%	46.4%	50.1%
Relaxed Accuracy	70.8%	56.6%	63.7%
[13] with SIFT	65%	50%	57%

The predominate reason for the better performance of the LIU method is that it trains its model using more initial poses then our method, which uses at most three to four pose perspectives, due to time constraints. Testing revealed that the pose was more accurately estimated in the neighbourhood that the model is trained. More input for training would ideally increase the number of correctly identified poses, and can be identified as a possible extension to this project. Relaxing the thresholds for the model also increases the accuracy of this method. Relaxation of the threshold allows an additional tolerance for the poses that can be considered on the boundary of correctly classified, allowing the algorithm to capture any pose that is in the general vicinity of the actual object pose. Relaxing the threshold further yields little improvement as beyond this boundary region the number of descriptors used for matching is not sufficient to estimate a correct pose.

8. Conclusion

The authors of this paper set out to estimate the pose of a textured object using monocular images, which was accomplished using local 2D descriptors and an iterative PnP algorithm. Challenges were encountered during testing and evaluation in that building a model requires very specific criteria an accurate mesh and a prominent texture. Although the method does not produce results as superior as the state-of-the-art methods the results were comparable to other local descriptor methods. The algorithm was tested against an object from the LIU dataset, a mousetrap, in over 1200 difference poses in the absence of clutter and in clutter, managing to attain an average accuracy of 50%, which increases to 64% when the tolerance thresholds are relaxed.

8.1. Further Extensions

The amount of time to research, develop and implement this project was limited. Should there have been more time allotted to this project the authors would have liked to have implemented additional features. It was identified that state-of-the art methods use some form of machine learning to create the model for the object. Convolutional neural networks and random forests are two of the methods identified in the literature. This is identified as a possible extension, which will have the effect of improving the model. An improved model in this capacity would have the ability to perform better object recognition and in turn better pose estimation, even being able to recognise and estimate the pose for texture-poor objects which proved problematic in our project. Another possible extension in the implementation of a tracking algorithm. As the algorithm was demonstrated using a video stream a tracking algorithm, like Particle Filter or Kalman Filter will be able to reject bad poses and identify the region the object is from the previous frame. This improves both the object recognition and the initial pose that is guessed to begin the pose estimation iteration. Should time have permit the authors would have liked to implement both or either of these suggested improvements.

References

[1] V. Badrinarayanan and R. Cipolla, *Robust Instance Recognition in Presence of Occlusion and Clutter*. In ECCV, 2014.

[2] A. Krull, E. Brachmann, F. Michel, M. Ying-Yang, S. Gumhold and C. Rother, *Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images*. In ICCV, 2015.

[3] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother *Learning 6d object pose estimation using 3d object coordinates*. In ECCV, pp 536-551, 2014.

[4] A. Crivellaro, M. Rad, Y. Verdie, K. Moo Yi, P. Fua and V. Lepetit, *A novel representation of parts for accurate 3d object detection and tracking in monocular images*. In ICCV, 2015.

[5] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige and N. Navab, *Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes*. In ACCV, 2013.

[6] C.P. Lu, G.D. Hager and E. Mjolsness, *Fast and globally convergent pose estimation from video images*. 22(6), 2000.

[7] E.R. Pi, *Implementation of a 3D pose estimation algorithm*. Universitat Politecnica de Catalunya Masters Thesis, 2015.

[8] D. DeMenthon and L.S. Davis, *Model-based object pose in 25 lines of code*. 15 (1-2), 1995.

[9] S. Hinterstoisser, S. Benhimane and N. Navab, *N3m: Natural 3d markers for real-time object detection and pose estimation*. In ICCV, 2007.

[10] P. Wohlhart and V. Lepetit, *Learning descriptors for object recognition and 3d pose estimation*. In CVPR, 2015.

[11] C. Rennie, R. Shome, K. E. Bekris and A. F. D. Souza, *A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place*. in IEEE International Conference on Robotics and Automation, 2016.

[12] V. Lepetit, J. Pilet and P. Fua, *Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation*. in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol 2, 2009.

[13] F. Viksten. P. Forssn. B. Johansson. and A. Moe, *Comparison of local image descriptors for full 6-degree-of-freedom pose estimation*. In ICRA, 2009.