Feature Segmentation for Object Recognition Using Robot Manipulation

Oleg O. Sushkov and Claude Sammut
School of Computer Science and Engineering
University of New South Wales, Australia
osushkov@cse.unsw.edu.au, claude@cse.unsw.edu.au

Abstract

Object recognition and localisation is vital for many robotics applications, such as household service robotics. However, it is not always possible to provide information about an object *a priori*. Instead we aim to have the robot autonomously acquire information about an object for the purposes of recognition and localisation. We present a system combining robot-initiated object motion, stereo vision, and long term feature tracking to effectively extract object features from scene images. This allows a robot to learn to recognise previously unseen objects in the presence of clutter, noise and background motion.

1 Introduction

This paper presents a method for a humanoid robot to actively learn to recognise a previously unseen object instance. Our method uses motion to separate object features from the background in order to build an object feature database. The robot picks up an object and performs a series of motions, while stereo vision and local image features, specifically Scale Invariant Feature Transform (SIFT) [Lowe, 2004], are used to gather trajectory data for each feature over multiple frames. Our contribution is demonstrating that by tracking stable image features over many frames we can effectively separate object and background features in a dynamic, highly cluttered environment (example scene in Figure 1). These features can then be used for building an all-aspect object model [Cyr and Kimia, 2004], allowing recognition, localisation, and effective manipulation of the object by the robot.

Most object recognition methods are split into two parts: learning the appearance of an object, and later using the learned data to recognise the object in a scene. Training data are typically segmented views of the object where it is clear which image regions belong to the object. These data are typically provided *a priori*. However, in many scenarios (eg: household service robot) this is not desirable, as it is impossible to predict which objects may be encountered. Instead we would like the robot to learn to recognise new objects autonomously. This can be done by generating training data autonomously by segmenting object features from the background. The challenges include separating background features from object features, segmenting robot arm and object features, being robust to changes in arm appearance and background motion. This paper will address the issue of finding object features in a scene.

Figure 1: Robot view of the arm gripping an object in a cluttered environment.

The remainder of this paper will present a review of related work, a description of our algorithm, an evaluation of the effectiveness of our approach, and finally future work.

2 Related Work

Object recognition is crucial for robotics applications such as household service robotics [Kemp *et al.*, 2007], with some recognition methods using local image descriptors [Mikolajczyk and Schmid, 2005] as SIFT [Lowe,

2004] for robust detection and and localisation in cluttered scenes [Collet *et al.*, 2009]. SIFT is a scale and rotation invariant interest point detector which generates highly discriminatory feature vectors for image interest points. However these require accurately segmented views of the objects to use as training examples. In the case of an autonomous robot, working in a dynamic environment, it may not be feasible to provide training examples for all objects it may encounter.

One solution is to have the robot generate the required training examples itself. Specifically, the robot must generate segmented views of an unfamiliar object, separating the object from the background. There are several static image segmentation algorithms which may be used for this purpose [Beveridge *et al.*, 1989; Shi and Malik, 2000]. However, these may not be effective in complex cluttered environments due to ambiguity between object and background image regions. An alternative is to take an active approach, where the robot can resolve object-background ambiguities by performing a probing action. Background subtraction [Stauffer and Grimson, 1999] can be used for segmenting object views by modelling the background and then placing the object in the camera view with the robot manipulator [Ude *et al.*, 2008]. Optical flow [Horn and Schunk, 1981] can be used to track scene motion as the robot manipulator nudges the target object [Fitzpatrick, 2003; Kenney *et al.*, 2009]. The resulting motion provides a cue for the segmentation of the object image region.

The disadvantage of these approaches is their inability to robustly handle background motion, making them unsuitable for a dynamic environment. The challenges to be addressed include:

1. separating object features from a cluttered background,
2. separating object and robot gripper features,
3. robustness to background motion, and
4. robustness to changes in the visual appearance of the robot arm.

We solve these by tracking the trajectory of each feature over a time period long enough to allow effective segmentation of object and background features.

3 Feature Segmentation Algorithm

3.1 Overview

Our algorithm allows a robot to autonomously find object image features in a dynamic unstructured environment; these can later be used for object recognition and localisation. We do this by using the robot arm to move the object through a scene, recording this motion with a stereo camera, extracting long-term feature trajectories from the resulting video stream, and finally using this data to extract the object features. We track the motion in 3D world space rather than in 2D image space as this provides more information to determine whether a feature belongs to the background or the object. The use of SIFT features, which can be reliably correlated from frame to frame, allows a feature to be tracked through multiple frames. This provides more information for segmentation as compared to instantaneous motion in the case of optical flow methods.

Figure 2: Robot and workspace setup. The robot has a stereo camera on a pan-tilt unit and a 6-DOF arm with a gripper attached.

The rest of this section describes the different stages of our algorithm. These are:

1. extracting and correlating SIFT features from the stereo video stream (described in sub-section B),
2. tracking the trajectory of each feature during the motion (described in sub-section C),
3. periodically take snapshots of object features when sufficient trajectory data are available (described in sub-section C),
4. separate arm and object features (described in sub-section D),
5. filter out background motion by ignoring features whose motion differs from the trajectory of the arm, (described in sub-section E),
6. finally we improve the separation of arm and object features by comparing the feature snapshots from different objects and remove common features (described in sub-section F).

The platform we used for implementing and testing our algorithm consists of a Bumblebee2 stereo camera and a

6-DOF arm with gripper attached, a workspace is accessible in front of the robot (see Figure 2).

We do not give details on grasping [Bicchi and Kumar, 2000] an object with the robot gripper. We use objects that are easy to grip by aligning on the long axis of the object. Additionally, we assume that a previously unknown object can be located in a scene in order to perform the initial grasp. This can be done by detecting some contrasting features such as vertical displacement on a flat table top.

3.2 Stereo Feature Generation

Our algorithm tracks *stereo features*, which are pairs of matching SIFT features from the left and right images of the stereo camera and a corresponding 3D world position. A SIFT feature is characterised by a highly discriminatory 128 dimensional description vector and an image gradient orientation at the feature point.

Algorithm 1 Finding stereo features

$leftFtrs \leftarrow SIFTFeatures(leftImage)$
$rightFtrs \leftarrow SIFTFeatures(rightImage)$
$stereoFtrs \leftarrow [\,]$

for all $lFtr$ **in** $leftFtrs$ **do**
  $epipolarFtrs \leftarrow epipolarFeatures(rightFtrs, lFtr)$
  $rFtr \leftarrow findNearest(epipolarFtrs, lFtr)$
  $oriDist \leftarrow oriDistance(lFtr, rFtr)$
  if $oriDist \leq oriThreshold$ **then**
    $fvDist \leftarrow ftrVecDistance(lFtr, rFtr)$
    if $fvDist \leq ftrVecThreshold$ **then**
      $worldPos \leftarrow findPos(lFtr, rFtr)$
      $newStereoFtr \leftarrow \{lFtr, rFtr, worldPos\}$
      $stereoFtrs \leftarrow newStereoFtr : stereoFtrs$
    end if
  end if
end for

Algorithm 1 describes the procedure for finding the stereo features in a frame. We extract the SIFT features for both the left and right images and match them by comparing their image positions, description vectors and orientation. For each feature in the left image, we search the right image for a feature on the same epipolar line with the closest description vector by Euclidean distance. If this pair of left and right image features have their orientations and description vectors within a threshold distance they are said to match and form a new *stereo feature*.

The performance of this algorithm is dependant on the choice of orientation threshold ($oriThreshold$) and the feature vector threshold ($ftrVecThreshold$) values. We determined suitable values by manually finding corresponding SIFT feature pairs in a large number of stereo images and recording the distance between their description vectors and orientation difference. We found the mean orientation difference to be 3.7° with a standard deviation of 3.1°; we set the orientation threshold to be $\overline{x} + 3\sigma$(which is equal to 13°). For the description vector distance threshold we consider the percentage of corresponding stereo feature pairs with description vector distance less than $x : \{0 \leq x \leq 700\}$ (we found that no feature desciption vector distance exceeded 700), as compared to random non-corresponding feature pairs. Figure 3 shows this relationship in detail. We set the feature vector threshold to 300 as we found that 90% of corresponding stereo feature pairs have description vectors with a distance less than this, while the average distance between non-matching feature pairs is 534 with a standard deviation of 65.

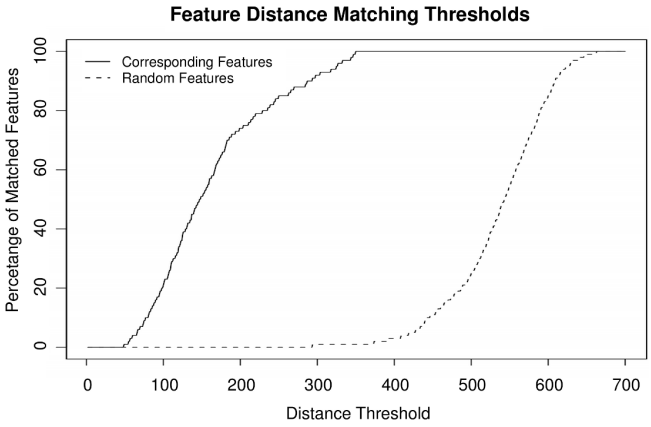

Figure 3: This shows the percentage of corresponding and non-corresponding SIFT stereo feature pairs with description vectors within a given distance threshold.

Evalutating these threshold parameters we found that the rate of incorrect feature correspondence was 1.3%. The above values should be applicable to a variety of scenes and environments since SIFT features are not dependent on the overall scene composition and are robust to lighting changes. However, cameras of varying quality, frame rate and baseline length may require different parameters.

3.3 Feature Tracking and Snapshotting

The purpose of the feature tracking module is to track each *stereo feature* across multiple frames, dealing with intermittent feature visibility and motion of the features in 3D space. This is done by maintaining a set of feature trajectories, an ordered list of $(Feature, DetectTime)$ tuples. $DetectTime$ refers to when the given $feature$ was detected. The ultimate aim of keeping track of the feature trajectories is to determine which features are part of the background and which are part of the object held by the gripper.

Algorithm 2 Matching a stereo feature to a trajectory

Input: $allTrjctrs$
Input: $nFtr$
$nearTrjctry \leftarrow \phi$
$bestFvDist \leftarrow \infty$

for all $trjctry$ **in** $allTrjctrs$ **do**
  $(tFtr, tDetectTime) \leftarrow head(trjctry)$
  $posDist \leftarrow worldPosDist(nFtr, tFtr)$
  $oriDist \leftarrow oriDistance(nFtr, tFtr)$
  $fvDist \leftarrow ftrVecDistance(nFtr, tFtr)$
  if $fvDist \leq min(ftrVecThreshold, bestFvDist)$ **then**
    if $posDist \leq distThreshold(tDetectTime)$ **then**
      if $oriDist \leq oriThreshold$ **then**
        $nearTrjctry \leftarrow trjctry$
        $bestFvDist \leftarrow fvDist$
      end if
    end if
  end if
end for

if $nearTrjctry \neq \phi$ **then**
  $nearTrjctry \leftarrow (nFtr, cTime) : nearTrjctry$
else
  $allTrjctrs \leftarrow [(nFtr, cTime)] : allTrjctrs$
end if

We maintain a set of visible feature trajectories; each new *stereo feature* is inserted into an existing matching trajectory, or if none match, a new trajectory is created. Algorithm 2 describes this process. Matching a new feature to an existing trajectory relies on spatial locality and descriptor locality. When inserting a feature, we consider it's description vector, orientation and 3D world coordinates. Successive features in a trajectory should have spatial locality in 3D world space and similar feature description vectors and orientation, as they should be centred on the same point on an object. When inserting a new *stereo feature* we try to find a trajectory whose most recent feature is within a threshold distance in world space (*distThreshold*), whose orientation differs less than a threshold amount (*oriThreshold*), and whose description vector is both within a threshold distance (*ftrVecThreshold*) and is the closest such feature on the epipolar line.

The *distThreshold* depends on how long ago the latest feature of the trajectory was detected. Since no point on the held object should move faster than the arm, we set the distance threshold to be:

$$ArmSpeed \times (CurTime - DetectTime) \qquad (1)$$

ArmSpeed is the maximum arm movement speed, *CurTime* is the current time, and *DetectTime* is time when the particular trajectory was last updated with a new feature.

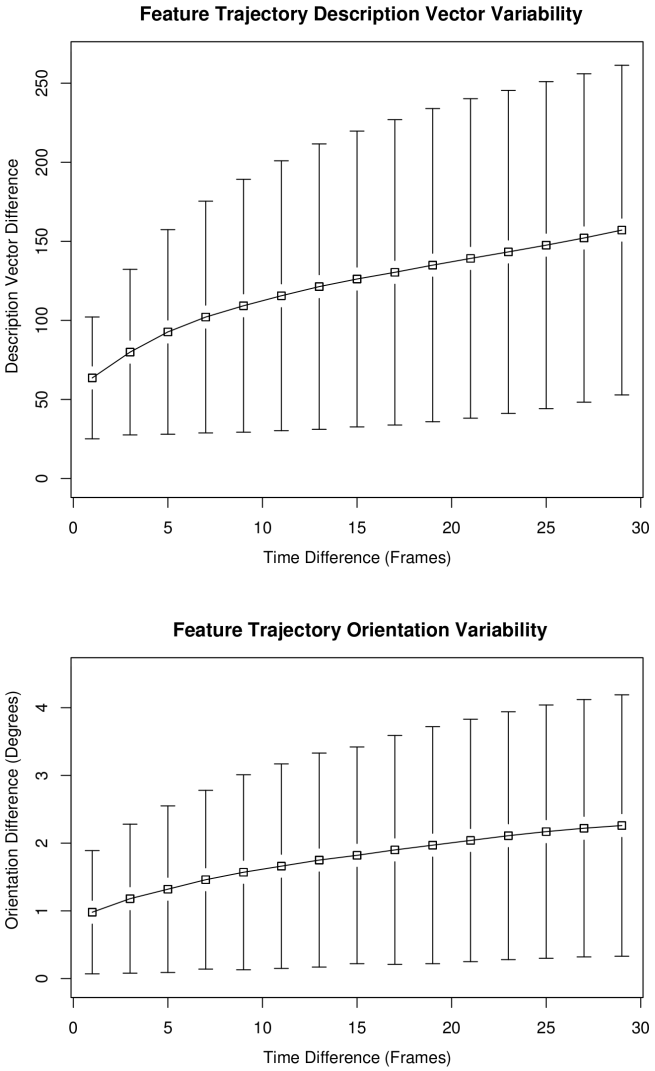

Figure 4: These graphs show the increasing description vector and orientation difference between the most recent *stereo feature* of a trajectory and older features of the same trajectory.

We determined appropriate values for the other thresholds by examining the variability of the description vector and orientation of features during linear movement (movement such that the object remains in the same orientation in the image). We plot the average description vector distance and orientation difference between a feature and its corresponding match in each of the previous 30 frames; this is shown in Figure 4. We can see that after 3 frames the average feature has its description vector shift by approximately 100 units with a standard deviation of 50, and the orientation changes

by 1.3° with a standard deviation of 1.2°. We set our thresholds for matching a feature to a trajectory to be $\overline{x}+3\sigma$ for both the feature descriptor and orientation difference. This corresponds to a value of 250 for the feature vector threshold (*ftrFecThreshold*) and 5° for the orientation threshold (*oriThreshold*). These values should be applicable to a variety of environments as SIFT keypoints have been shown to be robust to addition of noise and illumination changes [Lowe, 2004].

The purpose of tracking feature trajectories is to use motion to separate foreground object features from the background, these features then form a feature database used for object recognition. While the object is being moved by the robot, we periodically take snapshots of the leading features of trajectories which have a displacement over a certain threshold (see Figure 5). The intent is to use the snapshots of object features to build a database of object features for later recognition and localisation. The frequency of feature snapshots is dependent on several factors: the camera frame rate, speed with which the robot moves the object and the desired level of detail for the learned object feature database. We perform a snapshot every *five* frames, inserting into the object feature database the latest features of trajectories which have moved in 3D world space at least $ArmSpeed \times (TimeSinceLastSnapshot - 1)$ since the last snapshot.

3.4 Separating Arm and Object Features

The result of the previous step is a set of snapshots of arm and object features, separated from the background using motion tracking. The next step is to separate the object features from the arm features in each snapshot. We do this by using a known database of arm SIFT features. We generate this database using a training stage where the robot performs the previous set of steps without holding an object. In this way, the robot generates a database of arm SIFT features, as the only features which would be captured belong to the arm. An alternative method would be to use a human labeled set of training images of the arm, with the disadvantage of being time consuming for the human operator and lacking in robot autonomy.

Once a robot arm feature database is generated, we can use it to filter out arm features from the feature snapshots, leaving behind object features. We use Lowe's [2004] method for SIFT feature to database matching. SIFT description vectors are not equally discriminatory and a simple Euclidean distance for feature matching is not sufficient for comparing against a database of features generated under potentially different lighting conditions and orientations. The solution is to search the arm feature database for the nearest and second-nearest features to the snapshot feature. If the Euclidean distance of the snapshot feature to the nearest database arm feature is less than 80% of the distance to the second-nearest feature then the snapshot feature matches the arm feature. We remove from each snapshot all features which match an arm feature, the remaining features are labeled as object features. The justification for this approach is that the density of features in the neighbourhood of a feature indicates how discriminatory that feature's description vector is. The 80% value was determined by Lowe [2004] experimentally to give the optimal trade-off between false positives and false negatives.

Figure 5: This image shows in green the trajectories of *stereo features*, the ones on the robot arm and the held object have long trails because they are moving through the scene. The red markers indicate *stereo features* which form a snapshot due to having moved over a threshold amount. Note that this is before arm features are separated from object features.

3.5 Filtering Background Motion

So far we have assumed a largely static background, where the only significant motion in the scene is due to the robot arm moving with a held object. In a dynamic environment significant background motion must be filtered out in order for the trajectory snapshot method to work.

Background motion can be filtered in several ways. First, distant features outside the robot's workspace can be removed by examining their 3D world position. A further refinement is to use arm kinematics to determine the gripper position in each frame and only consider features near this point. In our case, however, the accuracy of the kinematics while the arm was in motion was poor due to lag and camera-arm synchronisation difficulties, and not sufficient to filter out background motion in close proximity to the gripper.

Our solution to this problem is to use the fact that the gripper and the features of the held object follow similar trajectories. If the robot gripper features are known, we can compare the trajectory paths of the arm features to

all others and remove trajectories with dissimilar paths, thus filtering out background motion. Prior to generating a trajectory snapshot the arm feature trajectories are found by comparing the features comprising each trajectory against the database of arm features. The remaining trajectories are compared to the arm trajectories by comparing the 3D world path of a trajectory. To compare two trajectories, a list of feature positions is extracted from each and normalised to the origin as follows:

$$p_n, p_{n+1}, ..., p_m \rightarrow (p_n - p_m), (p_{n+1} - p_m), ..., (p_m - p_m) \quad (2)$$

where p_n is the position of a trajectory feature at frame n. To calculate the difference between two trajectories, take the average distance between positions on matching frames across the two normalised position lists:

$$\frac{1}{n} \sum |p_a - q_a| \quad (3)$$

where n is the number of matching frames, p_a is the normalised position of the first trajectory at frame number a, and q_a is the normalised position of the second trajectory at the same frame number. We remove any feature trajectories which are not within a threshold distance of an arm feature trajectory as determined by the above distance measure. We determined the appropriate threshold by considering the variability of trajectories of a rigid object. We found that the average distance, using the above measure (3) of feature trajectories, on a rigid object is $0.21cm$ with a standard deviation of 0.25. We set the threshold to be $\bar{x} + 3\sigma$ which is 0.96.

It should be possible to extend our approach of separating background motion from object motion to utilise accurate arm kinematics when available. We compare feature trajectories to gripper feature trajectories to find background motion. However, if accurate gripper kinematics are available we could compare feature trajectories to the gripper position trajectory derived directly from the kinematics. We leave this for future work.

It is important to note that this method of background motion filtering can only be used if the motion performed by the robot is a translation, rather than a rotation. In the case of a rotation, the path of each feature depends on its distance from the axis of rotation, which is not suitable for this approach. However, SIFT features cannot be reliably tracked through more than a few degress of rotation (out of the camera plane). As a result, to gather long term feature trajectory data, we are restricted to translation motion anyway. In the case of learning features of multiple aspects of the object, we would need to perform a translation motion for each separate aspect.

3.6 Refining the Segmentation

Up to this point we have assumed that we can reliably separate the robot arm and object features using a known database of arm features (sub-section D). However, due to various types of noise, lighting changes, or actual changes in arm appearance (for example dirt or other wear), some arm features may fail to match the stored database and instead be mislabelled as object features. This would have a detrimental effect on the accuracy of object recognition using the generated database of features.

We avoid this problem by performing some post-processing of the database features. We compare snapshots of features of different objects to one another and remove any common features. This is because similar features appearing in snapshots of different objects are likely to be misclassified arm or background features (assuming that the different objects are visually unique from one another).

When a set of feature snapshots has been generated for a number of objects, we consider each object in turn and all of the feature snapshots of that object are compared against the snapshots of the remaining objects. Call the set of snapshots belonging to the current object A, and the set of snapshots belonging to the remaining objects B. Every snapshot $a \in A$ is compared against a number of randomly sampled snapshots $b \in B$, searching for matching features; the number of snapshots $b \in B$ that are considered varies depending on the desired accuracy and time constraints.

When comparing snapshots, we search for features that match. The matching criterion is similar to that used for matching arm features: feature f_a from snapshot a matches feature f_b from snapshot b if f_b is the nearest neighbour to f_a of all features occurring in b using the SIFT descriptor distance metric, and this distance is less than 80% of the distance to the second nearest neighbour. All features f_a with a matching feature in b are marked as arm features. If some of the objects are expected to have a similar appearance, and thus share similar SIFT features, this criteria can be extended to only mark features f_a which match multiple snapshots from different objects.

4 Evaluation

We tested the effectiveness of our algorithm by examining the effect of each separate refinement on the feature segmentation quality during training, as well as the feature recognition performance of the database that was generated. To test the object feature segmentation the robot picked up an object and moved it following a square shaped path of side length approximately 50cm with its hand. During this movement, 18 feature snapshots were taken, with the number of arm and object

Figure 6: Some of the objects used to test the robot learning of object recognition.

	Feature Detection
True Arm Features	328
True Object Features	378
False Arm Features	8
False Object Features	40

Table 1: The average number of correctly and incorrectly identified object and arm features during the learning phase.

features detected being recoded for each snapshot. We tested with ten different objects, simple shapes such as a soft drink cans and cubes of edge length 7cm with an image on the side (Figure 6).

The baseline result is summarised in Table 1. This shows how successful our method is in terms of separating object features from arm and background features during training. We can see that the number of background and arm features misclassified as object features is low when compared to the number of correctly classified object features.

To test the effectiveness of background motion filtering we introduced background movement into the scene in the form of a textured object (the same form and size as the held object) moving in a circle (radius 10cm) in the middle of the workspace. We then compared the performance of our algorithm with background motion filtering disabled and enabled by counting the number of correct and incorrect arm and object features found during the course of the arm movement (see Table 2). The effect of background motion filtering was a ten-fold reduction in false object features, the filtering largely removing feature trajectories of the object moving in the background. However there was a slight reduction in the number of correctly detected object features.

To test the feature correlation, filtering the appearance of the arm is altered slightly as compared to the robot training stage by attaching a small textured marker (see Figure 7). The features from this marker would not match against the database of arm features generated during training, and simulates the scenario of the arm getting dirty or worn. When correlating snapshot features for each object, all nine other objects are

Motion Filtering	*Disabled*	*Enabled*
True Arm Features	289	289
True Object Features	312	282
False Arm Features	3	6
False Object Features	366	38

Table 2: Motion filtering reduces false positive object features in the presence of background motion.

Figure 7: The marker attached to the robot gripper is circled in red. Used to test correlation filtering effectiveness.

compared against, with 18 snapshots per object. Table 3 shows that as a result of correlation filtering the number of false object features falls by 81%. This is due to the removal of the texture marker features from the object features set.

Finally we tested the object recognition accuracy of the generated object feature database. The robot autonomously learns to recognise each object using our algorithm, adding the object features from each snapshot to a database. A baseline is created by a human manually segmenting six image snapshots of each object; the object features from these are added to a comparison feature database. To compare the performance of the two databases we take nine image snapshots of each of the ten objects located amongst clutter, with the robot arm

Correlation	*Disabled*	*Enabled*
True Arm Features	253	581
True Object Features	446	494
False Arm Features	1	6
False Object Features	380	72

Table 3: Feature cross correlation reduces false positive object features when the arm appearance changes.

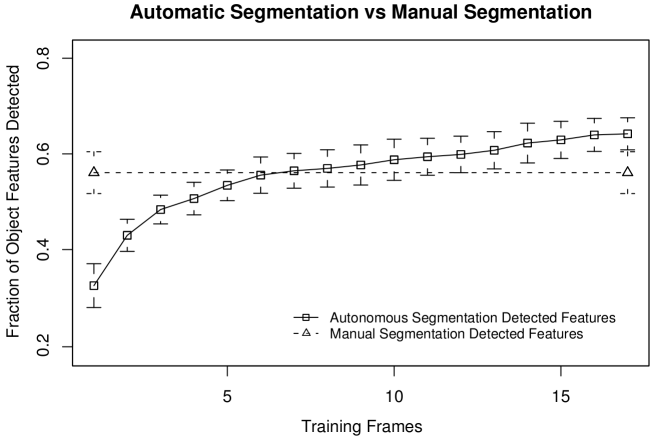

Figure 8: Performance of object recognition using automatic segmentation vs manual segmentation. The error bars represent the 95% confidence interval.

visible, and with the objects partially occluded. Each of these images has a manually generated ground truth feature classification, determining which of the features in the image are object features and which are background or arm features. Both the human generated feature database and the autonomously generated database are tested on this set of images by recording the number of correctly and incorrectly classified object features as a fraction of all object features in each image. For feature matching we use Lowe's [2004] matching method. For the automatically generated database we test the effectiveness of the database after each feature snapshot is added, to determine improvement of the classification as more snapshots are added (for a maximum of 18 snapshots per object). These results are presented in Figure 8.

The autonomously generated database detected 65% of object features after learning the full series of snapshot features; the human generated database detected 56% of object features. Neither database incorrectly matched any background features as object features.

5 Discussion

We have presented an effective algorithm for active learning of object recognition on a robot platform using motion based autonomous feature segmentation. Stable image features localised in 3D world space using stereo vision, combined with motion, provide sufficient data to perform effective object feature segmentation from background features in the presence of background motion and clutter.

However, this algorithm has some shortcomings. The motion the robot performs while holding the object is chosen to be a translation, allowing one particular aspect of the object's appearance to be learned. It is still possible to learn to recognise an object from multiple view points by performing a separate translation motion for each individual view point. However, this is not ideal. Ideally the robot should rotate the object in place so as to view the object from all sides, allowing multiple aspects to be learned. However, we found that tracking features becomes a problem under rotation (pitch or yaw relative to the camera), only being able to track features through approximately 15° of rotation. This reduces the trajectory information per feature as compared to translation motion, making the separation of foreground object and arm features from background features less reliable. Furthermore background motion filtering based on arm feature trajectories becomes ineffective. As a result, to learn the features of an object from multiple viewpoints, we fall back to performing multiple translation motions each with a slightly different rotation of the robot wrist rather than a single rotation motion.

6 Conclusion and Future Work

There are many possible applications and extensions of our work. One improvement would be to accurately determine the robot gripper pose in every frame based on the arm kinematics, and use it to improve and simplify both the background motion filtering and the feature tracking. We could compare each feature trajectory against the arm trajectory derived from the kinematics, which may be faster and simpler than our current approach. By considering the gripper motion, feature trajectory tracking could be improved, narrowing down the search area when inserting new features into existing trajectories.

Our algorithm may be used by a robot as a basis to learn to recognise, localise, and manipulate new objects autonomously. One possibility is to use the described approach to extract object features for different aspects and to stitch these together into a single coherent 3D feature point-cloud. This would then allow the object's 3D shape to be determined by fitting a model to the points, as well as allowing accurate 3D positioning of the object within a scene. Together this would lead to the robot being able to manipulate and use a previously unknown object.

It may also be worth exploring using different types of local image features such as SURF [Bay *et al.*, 2006] or PCA-SIFT [Ke and Sukthankar, 2004] rather than SIFT, as these may provide improved performance in some circumstances.

References

[Lowe, 2004] D. Lowe, "Distinctive Image Features from Scale-Invariant Key-points", International Journal of Computer Vision, 2(60):91-110, 2004.

[Cyr and Kimia, 2004] C. Cyr, B. Kimia. A Similarity-Based Aspect-Graph Approach to 3D Object Recognition. *International Journal of Computer Vision*, 57:5-22, 2004.

[Kemp *et al.*, 2007] C. Kemp, A. Edsinger, E. Torres-Jara. Challenges for robot manipulation in human environments. *IEEE Robotics & Automation Magazine*, 14:20-29, 2007.

[Mikolajczyk and Schmid, 2005] K. Mikolajczyk, C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1615-1630, 2005.

[Collet *et al.*, 2009] A. Collet, D. Berenson, S. Srinivasa, D. Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. *IEEE International Conference on Robotics and Automation*, pages 48-55, 2009.

[Beveridge *et al.*, 1989] R. Beveridge, J. Griffith, R. Kohler, A. Hanson, E. Riseman. Segmenting images using localized histograms and region merging. *International Journal of Computer Vision*, 2(3):311-347, 1989.

[Shi and Malik, 2000] J. Shi, J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888-905, 2000.

[Stauffer and Grimson, 1999] C. Stauffer, W. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition*, 2:252, 1999.

[Ude *et al.*, 2008] A. Ude, D. Omrcen, and G. Cheng. Making object learning and recognition an active process. *International Journal of Humanoid Robotics*, 5(2):267-286, 2008.

[Horn and Schunck, 1981] B. Horn, B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185-203, 1981.

[Fitzpatrick, 2003] P. Fitzpatrick. First contact: an active vision approach to segmentation. *Intelligent Robots and Systems Proceedings*, 3:2161-2166, 2003.

[Kenney *et al.*, 2009] J. Kenney, T. Buckley, O. Brock. Interactive segmentation for manipulation in unstructured environments. *IEEE International Conference on Robotics and Automation*, pages 1377-1382, 2009.

[Bicchi and Kumar, 2000] A. Bicchi, V. Kumar. Robotic grasping and contact: a review. *IEEE International Conference on Robotics and Automation*, 1:348-353, 2000.

[Bay *et al.*, 2006] H. Bay, T. Tuytelaars, L. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision - ECCV 2006*, pages 404-417, 2006.

[Ke and Sukthankar, 2004] Y. Ke, R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:506-513, 2004.