A motion segmentation technique for mobile robots using probabilistic models

Anchal Dorfling[1*] and *CE* van Daalen[2]

[1,2] Department of Electrical and Electronic Engineering, Stellenbosch University, Stellenbosch, South Africa

Abstract. A dynamic environment can be challenging for a robot to navigate; it should avoid collisions with objects while determining its position in its environment (localisation). Thus, it is necessary for a mobile robot to take measurements of its environment, such as features from camera images, to determine whether objects are static or dynamic (motion segmentation). This is difficult to do as knowledge of static objects is required for localisation which is then used to track the trajectories of dynamic objects. This paper proposes a motion segmentation technique that classifies objects as static or dynamic by measuring the change in distance between them across many time steps; this removes the need for localisation information. The technique is adapted from a probabilistic method for outlier removal and existing motion segmentation techniques. A simple, 1D environment is simulated to show proof of concept. Additionally, a few strategies for PGM model construction are investigated where the results show a clear relationship between accuracy and computational times.

1 Introduction

A robot moving through an unknown environment must be able to assess its environment to successfully navigate. For a robot equipped with stereo cameras, this is typically done by assessing the images taken by these cameras. Features, such as points or edges, are identified from these images and tracked over time. This is necessary as features on static objects are used to reconstruct the robot's trajectory. Using the knowledge of this trajectory, features on dynamic objects in three-dimensional (3D) space can be tracked. By doing so, the trajectories of the dynamic objects can be predicted and used for collision avoidance. This forms the basis for motion segmentation: classifying objects as static or dynamic for navigation and collision avoidance.

Various methods have been proposed for motion segmentation. Dense methods focus on analysing the pixels in video frames. The commonly used method of image differencing calculates the change in intensity of pixels over time-consecutive frames [1]; it is quite useful for applications with a static camera. However, for a moving camera where even the static objects appear to be dynamic, the frame rate should be high enough to extract useful information [2]. Such a method is susceptible to noise. Another method, called layering,

* Corresponding author: katiyar.anchal@gmail.com

involves separating a frame into layers where each layer contains information about the object moving in the layer and the depth the object is perceived in [3]. Moving objects are extracted by identifying patches of pixels that undergo transformations from one frame to the next. This method can be used for moving cameras but requires prior knowledge of the robot's trajectory. Also, it assumes that the viewpoint of the objects from the cameras does not vary. This paper will not consider these methods due to the restrictions on moving cameras and visual perspectives of objects, respectively.

Feature-based methods use features identified in frames; these methods tend to be less computationally intensive when compared to dense methods. This is due to the number of identified features being commonly less than the number of pixels in a frame. It can track the object through different frames using a known trajectory of the robot, thus useful for applications with a moving robot. The knowledge of the robot's trajectory can be obtained by using motion sensors or a global positioning system. However, obtaining knowledge of the robot's trajectory from sources other than cameras limits the applicability of such methods.

Brink [4] uses a probabilistic feature-based method, more specifically a *probabilistic graphical model*, with a global positioning system that tracks the robot's trajectory. Brink calculates the *velocities* of features to distinguish whether they are dynamic or not; a feature with a velocity of 0 is static. We adapt this technique by calculating the *relative distance* between two features. For two static features, this distance should not change, regardless of the viewpoint. On the contrary, the relative distance between a static and dynamic feature or two dynamic features would most likely vary over time. This is the principle that is used for our approach to motion segmentation. Consequently, obtaining knowledge of the robot's trajectory from other sources is not necessary for this method.

A similar core principle is used for outlier removal by Chiu [5]. Outlier removal is the removal of erroneous data (outliers) that is inconsistent with true data (inliers). When features are identified from images, they are matched with features on other images; the features can then be tracked over time. However, noisy measurements (a typical source for outliers) can lead to misidentification of features or mismatched features. This leads to erroneous calculations, and the outliers are removed. Chiu uses the knowledge of the shape created by a set of 3D points to identify inliers. For example, the triangle formed by three features in 3D space would remain unchanged if all features are inliers. If at least one feature is an outlier, the shape changes. Note that Chiu assumes that all features are static. If this method is used in a dynamic environment, a dynamic feature would be viewed as an outlier. Thus, this method can be exploited to perform motion segmentation as dynamic features would be identified as outliers.

Chiu's method was used in a 3D environment with practical data. Although these factors are important to develop the method for a real-world problem, we reduce the complexity of these factors to address the core concept. We simplify the problem by simulating a 1D environment with distinct objects rather than features on objects. Here, the shape created by the minimal set of objects (two) is a line. Consequently, an unchanged line, or distance, indicates inliers (static objects) whereas a varying line indicates at least one outlier (dynamic object). This simplified method cannot be applied to the real-world as it is; therefore, future work includes expanding the method to a 3D environment with practical, rather than simulated, data.

Lastly, Chiu and Brink employ probabilistic methods, or more specifically probabilistic graphical models (PGMs), to reason about their problems. These models work well with feature-based approaches and are also suited to our purpose.

In conclusion, this paper proposes a motion segmentation method using PGMs that does not require knowledge of the robot's trajectory from sources other than cameras. It adapts Brink's motion segmentation technique by observing relative distances between objects

rather than their velocities. It also adapts Chiu's outlier detection technique by using outliers to represent dynamic objects. Furthermore, we also compare a few graph construction strategies to investigate the relationship between their execution times and accuracy rates.

The paper is organised as follows: Section 2 presents the model design and inference whereas Section 3 looks at the implementation thereof. Section 4 presents the results, and the paper is concluded in Section 5.

2 Algorithm Design

We discuss the design of the PGM-based motion segmentation algorithm where static and dynamic objects are identified using the change in relative distance between them. This section first provides a brief background on PGMs, before explaining the setup of the problem and the environment. Thereafter, the modelling and inference processes for PGMs are discussed as well as how they are used for our algorithm.

2.1 PGM Background

A probabilistic graphical model (PGM) is a tool that is useful for reasoning about problems that have numerous complex relationships between uncertain quantities [6], [7]. It is typically used to represent a probabilistic model of a system with uncertain quantities, denoted as random variables (RVs), in the form of a graph. An RV can take on several values, discrete or continuous. They also have statistical dependencies on each other that define how the values of an RV affect other RVs. These dependencies are described with probability distributions that represent one's belief of the values that the RV can take on. This creates an excellent reasoning system with the following procedure: observed data fixes the value of some RVs, their effect on other RVs is calculated, and the system's knowledge of the RVs we want to reason about are extracted. This is the process of inference, the second step for using a PGM. It is done after the probability distributions over the RVs and their dependencies have been defined, otherwise known as the modelling step. To use PGMs, these two main steps are followed. Their detailed explanations are discussed in Subsections (2.3) and (2.4), after the problem setup in Subsection (2.2).

2.2 Problem Setup

We want to perform motion segmentation for mobile robots. For this situation, a robot, or an observer, moves among static and dynamic objects. As the robot moves between timesteps, it takes noisy measurements, or observations, of the locations of the objects in its environment. The measurements can then be used to calculate the change in relative distance between the objects, as depicted in Figure (1).

For this setup, a few assumptions are made: the robot can always view all objects and associate the correct measurement with the correct object, in other words, there are no incorrect measurements or outliers. The robot and the objects are rigid bodies: they do not change form or shape at any point. The robot also moves independently of the objects. The dynamic objects also move independently of each other; they can neither collide nor avoid

collisions. These assumptions are used in the modelling step, which is discussed in the following subsection.

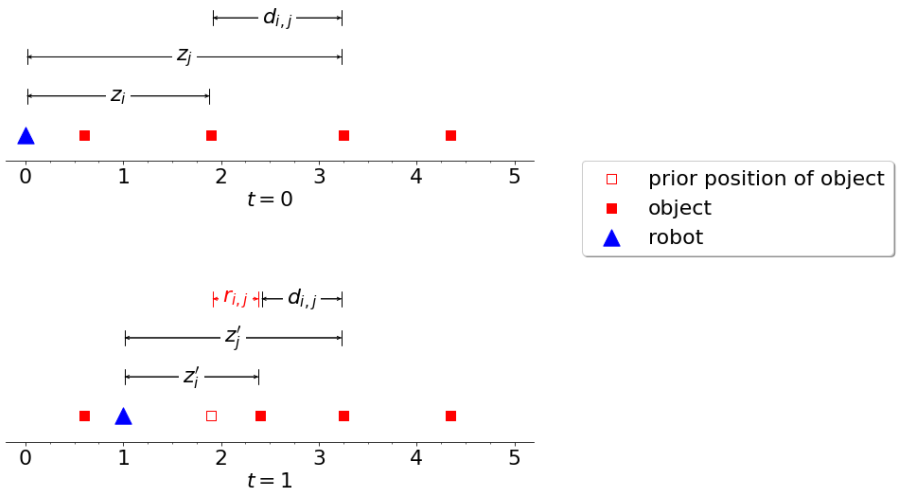


Figure 1: Problem environment set-up where d_i and d_j represent the measurements taken of object i and object j by the robot. The change in relative distance between objects i and j is calculated as the difference between the two measurements over multiple timesteps; it is represented by $r_{i,j}$.

2.3 Modelling

After describing the problem setup, we move onto modelling the problem using PGMs. Modelling follows a few steps: selecting random variables (RVs), defining their domains, defining the relationships between RVs with probability distributions and lastly, consolidating these elements into a graph.

First in PGM modelling is the selection of RVs to represent quantities with uncertainty, which may be unobserved or observed. We start off by choosing a classification label for object i as the random variable m_i. This discrete RV has a domain $\{0, 1\}$ where the state 0 denotes an object as static and 1 as dynamic. We also choose an RV, $\hat{r}_{i,j}$, to take on continuous values and represent the *measured* change in relative distance between a pair of objects. These measurements are generally contaminated with noise which also needs to be accounted for. Hence, we also choose an RV, $r_{i,j}$, to represent the *actual* change in relative distance between the sampled objects. Both $\hat{r}_{i,j}$ and $r_{i,j}$ are assumed to be Gaussian distributed which have a closed form under basic operations [7]. The final RV, $s_{i,j}$, connects the continuous RV, $r_{i,j}$, to a pair of discrete RVs, m_i and m_j. This RV has a domain $\{0, 1\}$ which denotes a pair of static objects with 0 and at least one dynamic object with 1. These RVs are represented by nodes in a PGM, as depicted in Figure (2).

After defining the RVs, we now define the relationships between them. To describe causal relationships between RVs, conditional probability distributions are used. We begin with the continuous RVs, $\hat{r}_{i,j}$ and $r_{i,j}$. The measured change in relative distance between objects is the actual change in relative distance with measurement noise; this relationship is described by

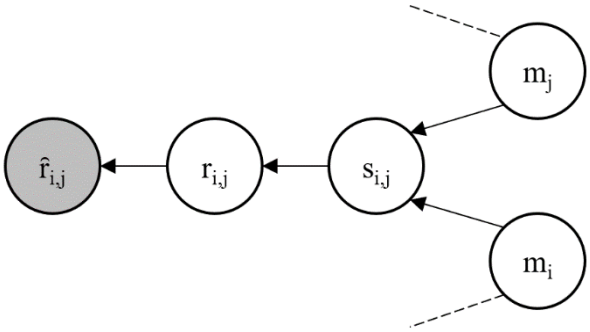


Figure 2: Bayesian network representing the structure for the motion segmentation algorithm (Shaded nodes depict observed RVs. Dotted lines represent connections to other pairs.)

$$\hat{r}_{i,j} = r_{i,j} + v_{i,j}, \tag{1}$$

where $v_{i,j}$ represents noise as a zero-mean Gaussian distribution with variance Σ_v. As such, the measured quantity is dependent on the actual quantity. This can be described with a conditional probability distribution as,

$$p(\hat{r}_{i,j} \mid r_{i,j}) = \mathcal{N}(r_{i,j}, \Sigma_{\hat{r},r}), \tag{2}$$

where $\mathcal{N}(r_{i,j}, \Sigma_{\hat{r},r})$ represents a Gaussian distribution with mean $r_{i,j}$ and variance $\Sigma_{\hat{r},r}$.

Continuing onto discrete RVs, the relationship between $s_{i,j}$, m_i and m_j can be described by a conditional probability table. Table 1 shows the normalised values for the conditional probability distribution over $s_{i,j}$ (which detects a pair of static objects), m_i (object label i) and m_j (object label j). The table shows that $s_{i,j}$ is 0 only when both object i and j are static and the distance between them is constant. For all other combinations of static and dynamic objects, $s_{i,j}$ is 1.

Table 1: Conditional probability distribution over labels for objects i and j, m_i and m_j, and the RV for pairs of objects, $s_{i,j}$

m_i	m_j	$s_{i,j}$	$p(s_{i,j} \mid m_i, m_j)$
0	0	0	1
0	1	1	1
1	0	1	1
1	1	1	1
Other combinations			0

Lastly, we discuss the relationship between $r_{i,j}$ and $s_{i,j}$, the actual change in relative distance and the RV that looks at pairs of objects. The actual change in distance between a pair of objects is dependent on whether the pair contains two static objects or not. As such, the distribution of $r_{i,j}$ varies for each state of $s_{i,j}$ (adapted from Brink):

$$p(r_{i,j} \mid s_{i,j} = 0) = \delta(r_{i,j}) \tag{3}$$
$$p(r_{i,j} \mid s_{i,j} = 1) = \mathcal{N}(0, \Sigma_{r,s}). \tag{4}$$

This means that given that $s_{i,j}$ is 0 for a pair of static objects, $r_{i,j}$ (the actual change in distance between the pair of objects) takes on a value of 0 with a probability of 1; that is, we are certain that the actual change in relative distance between two static objects is constant, as represented by the Dirac delta distribution. Furthermore, when $s_{i,j}$ is 1, a zero-mean Gaussian distribution represents the variance of the change of relative distance. The zero-mean property allows for negative quantities when objects move further apart.

With this, we have described the full network – random variables and their relationships. We can now model these into a network. The most common types of PGMs used for modelling are the Bayesian networks [8] and Markov random fields [9]. These graphs represent the RVs as nodes and relationships as links between nodes. For causal relationships described by conditional probability distributions, directed links are used to show which RV depends on which other RV. The most suited graph for such relationships is the Bayesian network, a directed graph. On the other hand, Markov random fields use undirected links for non-causal relationships described by factors. As all the relationships described for the problem are causal, we use a Bayesian network as depicted in Figure (2).

The full network can be described as a product of all conditional probability distributions, also called factors. With this, we have completed the modelling step of PGMs and continue onto inference.

2.4 Inference

Inference is the process of calculating beliefs over random variables given some evidence. Efficient algorithms have been developed for inference; however, the implementation of these algorithms requires PGMs, such as the Bayesian network, to be converted to another type of graph, such as a cluster [7] or factor graph. These graphs are useful as their structures allow for a technique called *message passing* to be implemented. For example, in a cluster graph the nodes are called clusters, which contain information about groups of RVs as factors. When clusters have RVs in common, an undirected link is used to connect them. Between two connected clusters, messages can be sent to pass information about the RVs that they share. The messages are calculated by summing, or integrating, out RVs that are not shared between the clusters; this eliminates the RV and results in a smaller factor. The message is then sent from one cluster and absorbed by the neighbouring cluster by multiplying the message in with the cluster's factors. As more messages are passed between clusters, more information is shared. Eventually, the clusters will reach a steady state about the values of their RVs. The information can then be extracted to determine the beliefs over the RVs we are interested in. This describes the belief-propagation algorithm [8], also known as the sum-product or the Shafer-Shenoy algorithm. In a factor graph, the algorithm works similarly other than having two types of messages. However, we use cluster graphs for their superior, if not similar, performance for complex problems [10].

A cluster graph is presented in Figure (3) for our problem. The factors are the conditional probability distributions over the RVs described in the previous section. The clusters are linked via the set of shared RVs, known as the separation set or the sepset.

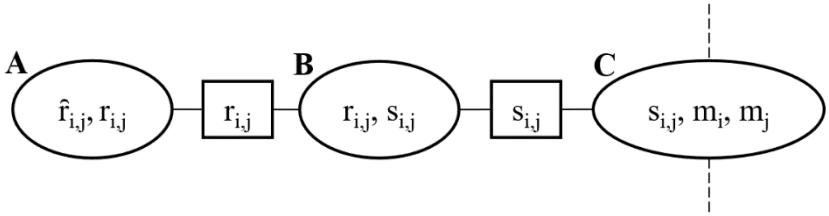


Figure 3: Cluster graph showing the clusters (ellipses) and their shared RVs (squares), known as the sepset (Dotted lines represent connections to other clusters.)

To calculate the messages between the clusters, we label the clusters from left to right, A, B and C, respectively. Note in Figure (3) that cluster C contains the only RVs that we want to reason about, namely the object labels m_i and m_j. Thus, we only need to send messages from cluster A to cluster C to calculate beliefs over the object labels. Furthermore, only cluster C connects to other parts of the cluster graph. This means that the messages sent from cluster A to cluster C are constant and do not receive any information from other parts of the graph. Therefore, once these messages are calculated and sent, clusters A and B can be discarded. It should be noted that by calculating the messages by hand, the complexity of computing hybrid clusters, which contains a mix of discrete and continuous RVs, is removed and the number of RVs in the system is reduced.

The message from cluster A to cluster B is the probability distribution from Equation (2),

$$\mu_{A\to B}(r_{i,j}) = p(\hat{r}_{i,j} \mid r_{i,j}), \tag{5}$$

where $\mu_{A\to B}$ is the message sent from source A to destination B. Note that $\hat{r}_{i,j}$ cannot be marginalised as it is an observed RV. The following message from cluster B to cluster C is

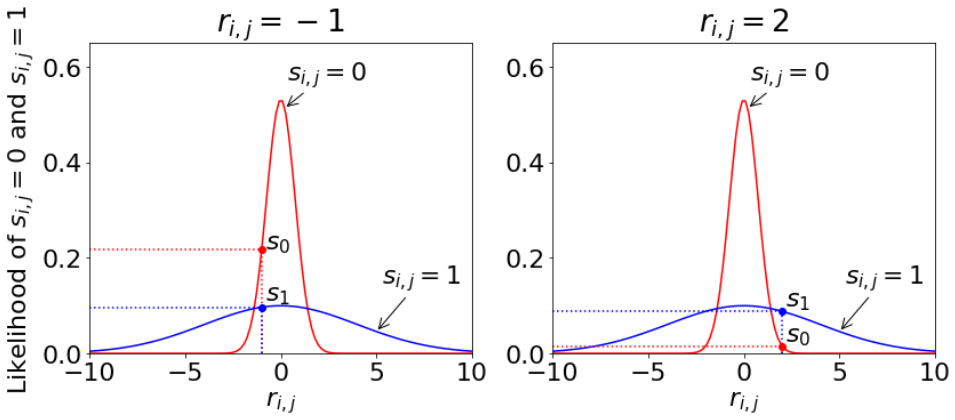
$$\mu_{B\to C}(s_{i,j}) = \int p(\hat{r}_{i,j} \mid r_{i,j})\, p(r_{i,j} \mid s_{i,j})\, d\, r_{i,j}, \tag{6}$$

where we can substitute the distributions from Equations (2), (3) and (4) to obtain

$$\mu_{B\to C}(s_{i,j} = 0) = \mathcal{N}(0, \Sigma_{\hat{r},r}) \tag{7}$$
$$\mu_{B\to C}(s_{i,j} = 1) = \mathcal{N}(0, \Sigma_{\hat{r},r} + \Sigma_{r,s}). \tag{8}$$

These messages are equivalent to Gaussian distributions where one distribution is a 'wider' version of the other, in other words, it has a larger variance than the other. As the RV $\hat{r}_{i,j}$ is observed, the message $\mu_{B\to C}$ can be directly calculated. For example, in Figure (4a), the likelihood of $s_{i,j}$ taking on the value of 1 is higher than $s_{i,j}$ taking on the value of 0: this indicates that it is more likely for the pair to contain a dynamic object than not. The vice-versa case is shown in Figure (4b). After the likelihood for each state of $s_{i,j}$ is calculated, it is incorporated into cluster C.

It should be noted that when messages have passed from cluster A to cluster C, only discrete RVs are left in cluster C. Messages can then be passed easily between the remaining discrete clusters in the graph for inference over RVs that we are interested in, namely the object labels, m – this is discussed in the following section.



(a) (b)

Figure 4: The message from cluster B to C can be depicted as two Gaussian functions where each function models the likelihood of a particular state of $s_{i,j}$. They can be calculated directly with $\hat{r}_{i,j}$ and $s_{i,j}$. (a) The likelihood of $s_{i,j} = 1$ is higher than $s_{i,j} = 0$ when $\hat{r}_{i,j}$ is 2.5. (b) The likelihood of $s_{i,j} = 0$ is higher than $s_{i,j} = 1$ when $\hat{r}_{i,j}$ is -1.

3 Implementation

We have modelled the motion segmentation algorithm and discussed the equations; we now apply it to the problem environment. We simplify a practical, 3D environment to a simulated, 1D environment which is depicted in Figure (1). The environment is simulated and the number of timesteps and objects are specified. After specifying a non-minimal number of timesteps (more timesteps allow for more comparisons between a pair of objects), the number of objects was chosen to maximise the simulation computer potential. The simulation computer uses a 4-core 8-thread 2.8 GHz Intel Core i7 processor with 16GB RAM.

There are ten data sets that were generated. The number of timesteps and objects, and the ratio of static to dynamic objects (1:1) were kept constant throughout the data sets. Other quantities, such as object labels, object locations, dynamic object speeds and control inputs, were randomly generated for each set. The measurements were calculated by adding noise to the difference between the robot position and object position.

After setting up the environment, we consider the implementation of the algorithm. The model and inference for calculating the change in relative distance between a pair of objects have been discussed; however, how the objects are paired up still needs to be explored. We explore a few options for pairing objects, allowing us to determine an optimum strategy in terms of accuracy rates and execution times. For a simple start, we choose to pair every object with every other object. This strategy allows for maximum number of pairings possible per object, therefore allowing every pair of objects to share information with every other pair. Thereafter, we consider reducing the number of connections in three different ways: by randomly pairing objects, pairing objects to other objects that are closest to them, and choosing a few ‘main’ objects to pair other objects with. Firstly, we randomly pair objects with each other; the number of pairings per object was chosen to be five percent of the total number of pairings possible. We call this strategy, “Random”. Secondly, we pair objects with their neighbours, that is, the closest object on either side of it. The neighbours are calculated in the first timestep and the pairs are created. We call this strategy, “Neighbour”.

Thirdly, we choose a few 'main' objects, five percent of the total number of objects, and we pair each object with the next object as they are chosen. The rest of the objects are divided up into groups of equal size and paired with a 'main' object. We call this strategy, "Main".

These strategies are then used to construct the graph; we can also call them model construction strategies. After construction, the measurements are inserted into the graph as evidence and inference is performed. Construction and inference are done with a C++ library developed by the University of Stellenbosch, called EMDW. All strategies are executed and tested with the ten sets of data. The process is timed, and the results are compared to measure accuracy rates.

We expect the first strategy to have the highest accuracy, as it constructs the maximum number of pairings possible per object. Consequently, there will be multiple messages containing information on the state of every object; this strategy will, therefore, see high execution times. Thus, we consider this strategy as a baseline to compare other strategies with, naming it the 'Baseline' strategy. The results of implementation are reviewed in the following section.

4 Results

The four model construction strategies, namely: the baseline, random, main, and neighbour strategies, are shown in Figure (5) which depicts the execution times (in milliseconds) and the accuracy (in percentages). The execution times are depicted on an exponential scale due to their exponential differences. The accuracy is calculated by comparing the beliefs to the ground truth, that is, if the probability of a known static object is more than 0.5, it is correct. The metrics are calculated for ten distinct runs that use the ten data sets.

As per Figure (5), the strategy with the highest accuracy is the baseline strategy. This makes sense intuitively as this strategy contains the highest number of pairings per object – there are multiple sources of information to calculate a belief. As expected, it also has the highest execution times – in the range of 55 to 65 seconds. This strategy contrasts with the neighbour strategy which has execution times in the range of 5 to 6 milliseconds, almost 10 000 times less than the baseline strategy. After observing other strategies, a trend that can be

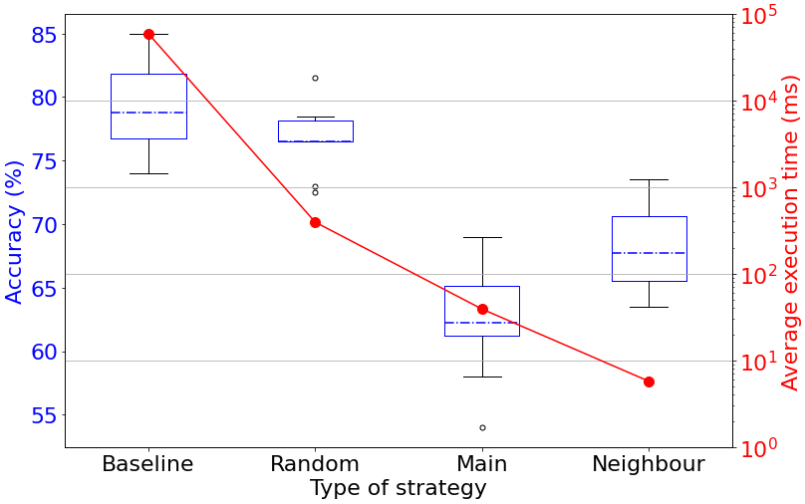


Figure 5: A comparison of the accuracy (red) of the strategies to the ground truth and the average execution times (red) is shown. A clear trend can be seen: the more accurate the beliefs, the exponentially larger the execution times.

clearly seen is that higher accuracy comes at the cost of exponentially high execution times. Additionally, the main strategy's accuracy varies significantly, clocking the worst accuracy of all. This can be caused by the random selection of main points; if the selection does not contain a pair of static objects, the model cannot send many useful messages.

A comparison was also made using the marginal beliefs retrieved from the implementation of the strategies against the beliefs from the baseline strategy. The Kullbuck-Liebler (KL) divergence is used for this purpose as it is a widely used standard for comparing the proximity of a distribution to another [6]. The divergence for each data set was averaged and divergences from all ten data sets were collated, as portrayed in Figure (6).

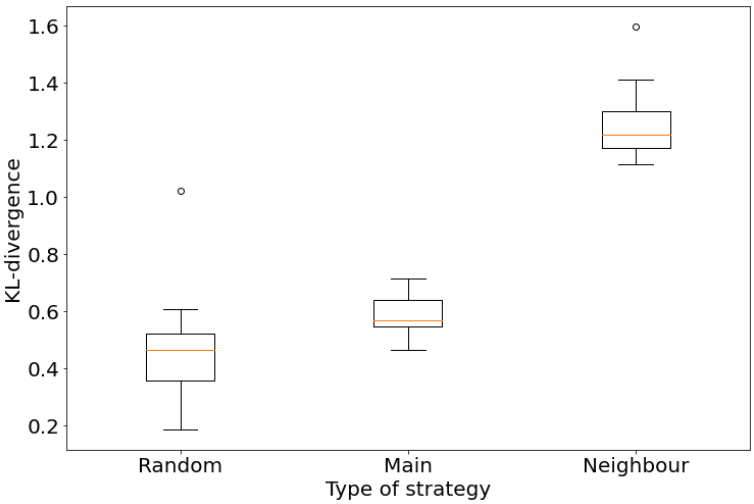


Figure 6: The distributions obtained from various strategies are compared to the baseline strategy using the averaged KL divergence over ten sets of data

A large, positive divergence indicates that the strategy is often less certain about its marginal beliefs than the baseline strategy's beliefs. A small, positive divergence indicates that it is seldom less certain than the baseline strategy's beliefs. The random strategy shows the lowest average divergence, implying that it is the most reliable strategy after the baseline strategy; this is also reflected in Figure (5). Additionally, the neighbour strategy has the highest average divergence even though it performed better than the main strategy in terms of accuracy. This means that the marginal beliefs of the neighbour strategy were less certain than that of the main strategy, but there were more correct beliefs.

5 Conclusion

Motion segmentation algorithms are used to distinguish between static and dynamic objects. It is typically done by analysing images taken by a robot that is equipped with cameras. Furthermore, additional (expensive) sensors are also generally required to obtain knowledge of the robot's trajectory before motion segmentation can be performed; however, the requirement could restrict applicability of the algorithm if this information is not available. Our approach removes the need for this information by calculating the change in the relative distance between objects to determine whether the objects are static or dynamic. Thus, our method can be used for robots that are equipped solely with cameras as sensors.

Our developed algorithm adapts a PGM-based outlier removal technique and an existing motion segmentation technique to solve motion segmentation for mobile robots. The

algorithm was designed and implemented for a simulated 1D environment. The graph was constructed, inference was performed, and beliefs extracted. A few model construction strategies were also explored. The strategies were compared for their accuracy rates and execution times and evaluated for their performance.

The baseline strategy performed best in terms of accuracy rates, and worst in terms of execution times; thus, it is not a very practical solution. If high accuracy is required, the random strategy can be recommended for its significantly lower execution times for slightly less accuracy rates than the baseline strategy. If execution times are prioritised, the neighbour strategy performed the best out of all but also did not sacrifice its accuracy entirely, like the main strategy.

Although the algorithm performed well for a simulated environment, aspects in the practical environment may need to be addressed as well. Complexities can be added, such as a 2D or a 3D environment or the use of practical data from images taken by a robot. Outlier detection and removal is a key aspect for robust measurement processing which should also be considered. Lastly, the accuracy and execution times were the only metrics measured; however, additional metrics, such as sensitivity to noise, can also be measured.

References

[1] R. Jain and H.-H. Nagel, 'On the Analysis of Accumulative Difference Pictures from Image Sequences of Real World Scenes', *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 206–214, Apr. 1979, doi: 10.1109/TPAMI.1979.4766907.

[2] L. Zappella, X. Lladó, E. Provenzi, and J. Salvi, 'Enhanced Local Subspace Affinity for feature-based motion segmentation', *Pattern Recognit.*, vol. 44, no. 2, pp. 454–470, Feb. 2011, doi: 10.1016/j.patcog.2010.08.015.

[3] M. Pawan Kumar, P. H. S. Torr, and A. Zisserman, 'Learning Layered Motion Segmentations of Video', *Int. J. Comput. Vis.*, vol. 76, no. 3, pp. 301–319, Mar. 2008, doi: 10.1007/s11263-007-0064-x.

[4] D. Brink, 'Using probabilistic graphical models to detect dynamic objects for mobile robots', Stellenbosch University, Stellenbosch, 2016.

[5] A. Chiu, 'Probabilistic Outlier Removal for Stereo Visual Odometry', p. 110.

[6] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge: Cambridge University Press, 2011. doi: 10.1017/CBO9780511804779.

[7] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. Cambridge, MA: MIT Press, 2009.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[9] R. Kindermann and J. L. Snell, *Markov random fields and their applications*. Providence, R.I: American Mathematical Society, 1980.

[10] S. Streicher and J. du Preez, 'Graph Coloring: Comparing Cluster Graphs to Factor Graphs', in *Proceedings of the ACM Multimedia 2017 Workshop on South African Academic Participation - SAWACMMM '17*, Mountain View, California, USA, 2017, pp. 35–42. doi: 10.1145/3132711.3132717.