Neural Network-Based Stereo Vision Outlier Removal

March Strauss[1*], and *Corné E.* van Daalen[1]

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

Abstract. Stereo vision systems rely on accurate feature matching to provide valid stereo reconstruction and pose estimation. This accuracy is achieved through outlier removal techniques, such as RANSAC. However, images also contain semantic information, which can be extracted using neural networks. This paper proposes an additional outlier removal method, where the images are semantically segmented using a neural network, before the features identified are assigned semantic identifiers using a probabilistic data association technique, and matches are evaluated based on this added semantic information. This blending of feature-based techniques with dense semantic maps allows for more information to be tied to each feature, not just its position in the image. This opens paths to applications like class-based clustering. The approach proposed is compared to a traditional outlier removal system by comparing the produced disparity values to known ground truth measurements, and assessed for accuracy and execution speed. It is shown how the addition of semantic segmentation does improve the accuracy of disparity measurements in stereo images, with a loss in processing speed. However, this loss can be mitigated by utilising more specialised hardware.

1 Introduction

Stereo vision is used in many autonomous navigation systems, due to its availability and low cost of implementation. Such systems are currently finding applications which require 3-dimensional maps of environments provided by stereo cameras, such as for quality control systems [1] and biomedical applications [2].

Stereo vision systems typically require that common points between the two captured images be accurately identified and matched between the images. Several approaches to determining these points of interest exist, ranging from sparse methods where the matched points are a few pairs of common features present in both images, to dense solutions that try to match all corresponding pixels in both images. This paper focuses on a feature-based approach.

These matched features are needed to accurately determine the depth of specific 3-D points in the captured scene, a process known as stereo reconstruction. The image location of these features, if accurately matched, can also be used to determine the relative pose

* Corresponding author: 21616086@sun.ac.za

between two consecutive viewpoints. It allows for an estimation of the motion of the stereo cameras to be inferred across time.

Many common feature-matching systems, however, produce a large number of incorrect matches, along with accurate ones. Such outliers can be detrimental to the performance of a stereo vision system, producing spurious and inaccurate depth measurements. It is for this reason that outlier removal methods are commonly employed to remove as many of these mismatched points as possible before attempting to use the remaining matched features to perform stereo reconstruction and/or relative pose estimation.

These outlier removal algorithms usually depend on statistical methods, such as the least median of squares or random sample consensus (RANSAC) [3] to filter out as many outliers as possible. Often these methods function more efficiently with fewer as it gave more accurate matches than with a large amount of less accurate ones. Applying a separate outlier removal method to the initial proposed matches, before applying the statistical methods, could therefore improve the quality of the final returned matches.

Images also contain a large amount of semantic information, encoded within the objects and environments captured in the image. If two features are matched, but the features are located on different object classes, this potential match should be discarded as an outlier.

This paper proposes extracting this semantic information using a trained neural network, and then applying the extracted information to the initial set of matched features. This information can be used to discard matches where the semantic classification of the features does not correspond, before further refining the matches using more traditional methods.

In doing so, this paper presents a novel approach to integrating semantic information into traditional feature-based stereo vision through a probabilistic data association technique. This is in contrast to other applications, where the semantic information and the featural information are kept separate. This mixture of semantic and featural data also allows for further work in feature-based systems, such as clustering features based on semantic information, and visual odometry through semantic identification of dynamic objects.

The rest of the paper is organised as follows: To better contextualise the contributions made by the work presented in this paper, some related applications employing similar techniques are presented (Section 2). This is then followed by an overview of various techniques which are used to extract information from still images (Section 3). To successfully use the extracted information, a method of associating one set of data with another must be utilised (Section 4).

With this background knowledge, the paper then discusses the methodology utilised in the testing and experimentation phase (Section 5), including selected datasets and the implementation of each of the aspects explained in the previous sections. This is followed by an analysis and discussion of the results acquired from these tests (Section 6), which allow for specific conclusions to be drawn (Section 7), as well as discussions of potential further work on this topic (Section 8).

2 Related work

Semantic information has begun to be incorporated into a number of different areas in artificial intelligence and autonomous vehicle navigation.

A common application of semantic information is simultaneous localisation and mapping (semantic SLAM). Semantic SLAM uses semantically differentiable objects, to serve as landmarks in robot or vehicle pose estimation and mapping applications [4]. These algorithms use the semantic information in the scene to identify objects that can serve as landmarks from multiple views and poses, such as doors in hallways, or parked vehicles in car parks. Another approach to SLAM that uses both semantic information image features

proposes a hybrid landmark-based and semantic-based SLAM system to generate more landmarks [5].

In contrast to these approaches where semantic information is incorporated into stereo vision systems, this paper proposes a novel combination of the sparse features-based information with a dense semantic map. This combination enriches the feature-based approach, while ensuring that each feature that is found is guaranteed to have semantic data associated with it.

3 Overview of image processing techniques

This section outlines the image-processing techniques used to extract this information, and how they are applied.

3.1 Feature detection and matching

Many different feature detection algorithms exist, ranging from basic corner detectors to more sophisticated algorithms, such as SURF, FAST, and BRIEF, each of which has its own benefits [6]. The two detectors that are expanded on in the following sections, SIFT and ORB, were chosen as common, representative options, and will be utilised going forward.

The scale invariant feature transform (SIFT) algorithm [7] produces very accurate matches that are robustly invariant to image rotation and scale, but this comes at a comparatively high computational cost. In comparison, an oriented FAST and rotated BRIEF (ORB) [8] feature detector is much faster computationally, but produces less accurate feature matches.

In both cases, the features found in each image are described using a descriptor, which includes its size and orientation, as well as an x and y location on the image. An example of an image with a number of detected features highlighted is shown in Figure 1. The features are shown in magenta, drawn as a circle with a diameter equal to the feature size, and a line showing its orientation.

Fig. 1. An example of features detected using the SIFT algorithm.

Features are matched by comparing their descriptors and matching features with the closest corresponding descriptors. A commonly used feature-matching method is brute force matching, where every feature in one image is compared to every feature in the other, and the absolute best match is chosen. This produces good matches, but becomes slow for substantial numbers of points needing matches.

Another algorithm used for feature matching is an approximate nearest-neighbour search [9], which is faster when applied to large datasets, but is not guaranteed to produce the best matches from all the points. An example of the output of a feature-matching algorithm is shown in Figure 2.

Fig. 2. The output of brute force feature-matching algorithm using SIFT features. Each match is highlighted in a random colour.

3.2 Semantic segmentation

Semantic segmentation is a process whereby each pixel in an image is assigned a specific class label or a likelihood over a number of class labels. An example of the results of such a process is given in Figure 3.

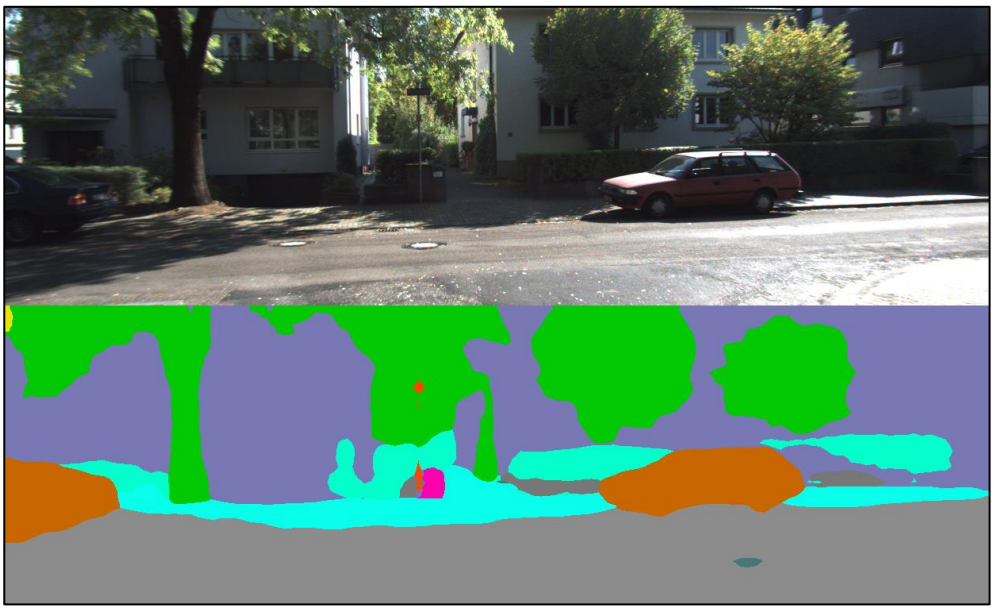


Fig. 3. Example of an image before and after semantic segmentation with a ResNet50 model. The base image is shown above. The output is shown below, with each pixel shaded in a different colour based on a class label with the highest probability.

Semantic segmentation algorithms have become far easier to design and implement with the advent of modern deep convolutional neural networks (DCNNs), which are well optimised for image processing and can be trained for a number of specialised applications given the right data, such as for automated medical diagnosis and organ segmentation [10].

The main limitations of using DCNNs are twofold: firstly, the performance of such a network is heavily restricted by the data it was trained on. A network trained on indoor scenes is going to struggle to parse scenes in outdoor scenarios, and is likely to mislabel such a scene. Secondly, it requires a substantial amount of computing power to perform inference using such models. This can make systems utilising DCNN's severely hardware limited. Performance can be greatly improved if run on a dedicated hardware device, such as graphics processing unit (GPU).

Several low-overhead models have been developed for low-power computing devices, such as Google's MobileNet architecture [11], but this comes at the cost of the accuracy of the inferred class regions. Another, more powerful network architectures is ResNet [12], which allows for more processing layers, and thus a deeper network, to be implemented without increasing the training difficulty by utilising residual learning techniques. U-Net [13] is an architecture designed to ease training by utilising available training data more efficiently. This architecture was designed originally for use in biomedical image segmentation, where the availability of large amounts of training data is limited.

For this paper, several pretrained segmentation models were selected for experimentation, each of which had different levels of accuracy and processing performance. More details about each of the chosen models are given in Section 5.

4 Semantic feature filtering

This section outlines the novel approach to constructing a semantic description for image features, which forms the basis for the method of outlier removal that this paper presents. This derivation assumes certain inputs are present:

- The location of image features in the image. This is usually output from a feature detection algorithm (Subsection 3.1).
- A pixel-wise distribution over a number of semantic class labels. This is usually the output produced by a semantic segmentation neural network (Subsection 3.2).

4.1 Derivation of feature class label

A naïve approach to determining the label of a feature once its location in the image has been determined is to use the distribution over the semantic class labels of the pixel at the detected feature location. However, many feature detection algorithms return sub-pixel accurate feature locations, requiring that an assumption of which pixel a feature is located in be made. Features can also be detected in regions of the image on the edges of semantic class regions, such as the edges of a roof on a building. Here the likelihood that a single pixel can capture the semantic quality of the feature is reduced. It is for these reasons that a method that considers a region of pixels was developed.

Once a feature has been located in an image, and a distribution has been found over all of the class labels for each pixel in the image, each feature needs to be assigned its own probability distribution over the class labels, $P(L_F|I)$, where:

- L_F is the class label of the feature
- I is the given image

Due to the effect of image noise in feature detection, there is uncertainty about the exact location of the feature on the image. This uncertainty can be expressed as a distribution, $P(p_F)$, which, when considering only a small region around the detected feature location, can be expressed as a discrete distribution describing the probability that the feature is located within that specific pixel. This is expressed as

$$P(p_F|I) = \gamma_n \text{ if } p_F = p_n. \quad (1)$$

One can then compare any feature location p_F with every possible pixel location p_n, and for the p_n that matches p_F, assign the probability value γ_n to the output of the function.

The distribution over the semantic labels at the image location of the feature $P(L_F|I)$ can be expressed via marginalisation by

$$P(L_F|I) = \sum_{p_F} P(L_F, p_F|I) \quad (2)$$

which can in turn be expressed in terms of the individual pixel labels L_n through marginalisation and conditional probability as

$$P(L_F, p_n|I) = \sum_{L_n} P(L_F|L_n, p_n, I)P(L_n, p_n|I). \quad (3)$$

Using the assumption that the semantic label of a pixel is conditionally independent of the feature location given the image, and again using the definition of a conditional distribution, Equation 3 can be expressed as

$$P(L_F, p_n|I) = \sum_{L_n} P(L_F|L_n, p_n, I)P(L_n|I)P(p_n|I). \quad (4)$$

with the term $P(L_F|L_n, p_n, I)$ is an indicator function that is equal to 1 when the class of the pixel and class of the feature match, and 0 in all other cases. Thus, when summed over all the class labels L_n, this reduces the expression in Equation 4 to the form

$$P(L_F, p_n|I) = \gamma_n P(L_n|I). \quad (5)$$

If Equation 5 is applied over all the pixels in consideration, this transforms the expression in Equation 2 into a simple weighted sum over the pixel distributions for each pixel considered part of the feature, allowing a probability distribution over the semantic class labels to be assigned to each feature.

4.2 Determining pixel weights

The derivation in Section 4.1 shows that the values needed to turn the output of the semantic segmentation algorithm into a distribution over the classes for each feature are the specific weight values γ_n assigned to each pixel. To better contextualise the applications of these weights, a slight shift in perspective is needed. When considering Equation 5, one can also interpret the weight γ_n as the contribution that pixel *n*'s label distribution makes to the features label distribution, as opposed to the feature's positional uncertainty.

In order to calculate these values, a good assumption to make is that the contribution each pixel in the feature makes to the overall feature class reduces as one moves further away from the feature's measured location. To account for this, a 2-D pixel-wise Gaussian distribution was applied over the pixel-grid, centred on the pixel closest to the feature's location, similar to the technique used in Gaussian blur filters [14, *pp. 137-150*].

Normally, 2-D Gaussian distributions require an infinite plane to be accurately modelled. However, in practice, pixels further than 3 standard deviations from the mean contribute little to the final value, such that the distribution of pixel weights, often called a 'kernel', can be cut off beyond this point. This Gaussian kernel can be adaptively sized to account for the size of the specific feature, or can be fixed to reduce the amount of computation required to calculate the feature distribution.

4.3 Comparing features semantically

Once the class distribution for each feature has been determined, a means of comparing features with each other based on these distributions is needed to quantify the semantic similarity of them. The measure chosen for this paper is the discrete Kullback-Leibler (KL)

divergence, which measures how much one probability distribution differs from another, expressed as

$$D_{KL}(P||Q) = \sum_x P(x) log\left(\frac{P(x)}{Q(x)}\right). \quad (6)$$

When the distributions *P(x)* and *Q(x)* are identical, the KL divergence has a value of 0, with the value increasing as the difference between the distributions becomes greater. Since there is no reason to prefer one of the features as the reference *P*, and to avoid the non-symmetric nature of the KL divergence, a better measure of the difference between two features would be to use the symmetrised KL divergence, which is often expressed as

$$D_{KL}(P||Q) + D_{KL}(Q||P) \quad (7)$$

allowing for the distributions of the matched features to be compared, taking the entire distribution into account. Any matches whose class distributions differ too much can be excluded.

5 Implementation and experimental setup

This section lays out how the various methods of extracting information from images that were described in Section 3 and the semantic feature filter described in section 4 were combined into a working experimental system, and the methods employed to validate this system.

5.1 Testing data

The dataset chosen for the evaluation of the proposed outlier removal technique is the KITTI stereo vision and optical flow dataset [15], specifically the 194 validation image pairs, each of which is 1226 pixels wide by 370 pixels high. This set of images was chosen for a number of reasons: Firstly, its ubiquity as a testing and validation set in stereo vision applications means that it is well suited to the task.

Secondly, it provides known ground-truth measurements taken by a separate LIDAR sensor, allowing for a comparison between the extracted values and the true values to be analysed. These values are stored as a sparse disparity map for the stereo-matching case, and as a sparse *x* and *y* vector map for optical flow.

Finally, the images are well calibrated and stereo rectified, meaning that no additional image processing or correction is required before attempting the stereo matching, which could have an impact on the accuracy of the results.

5.2 Implementation of the semantic feature filter

The testing script, which incorporates each of the functions required to perform feature matching and outlier removal, was created in Python. Each of the required aspects of the system is outlined in the following sections.

5.2.1 Feature detection

Two different feature detectors have been implemented for evaluation, namely SIFT and ORB, as both have a standard implementation in the OpenCV library [16], which forms a core part of the testing script's image-processing pipeline. Both feature detectors were

constrained to detect only 500 features per image, as the OpenCV SIFT detector usually detects far more features than the ORB one if not limited in the code.

5.2.2 Feature matching

Feature matching is done using OpenCV's brute-force matching algorithm. While this is not the most efficient matching algorithm, it is simple to execute and produces good results when the number of features is low.

5.2.3 Semantic segmentation

Semantic segmentation was performed using the MIT semantic segmentation library for Python [17]. This module is built using the PyTorch framework, and was designed for use with the ADE20K dataset [18], which features a wide number of scenes in differing contexts, including many outdoor and urban environments. This dataset produces models well suited for evaluation using the KITTI dataset, as they share many similar image contexts. The ADE20K dataset has a base of 150 different class labels.

Three different neural network architectures were chosen, each offering different levels of segmentation accuracy at the cost of inference speed. The MobileNetV2 architecture was chosen as the lowest-accuracy model, as it is designed for speed and efficiency on low-power devices. Two separate ResNet models were selected, one with 18 layers, and one with 50 layers, to have a middle- and high-accuracy model for comparison with the MobileNet.

For inference, three different pre-trained models were used, each offering different advantages and disadvantages. These models are listed in Table 1, in order from fastest performing to slowest performing. These values were measured on a single NVIDIA Titan Xp GPU running on the Pascal architecture.

Table 1. Performance of semantic segmentation models [19]

Encoding stage	Decoding stage	Pixel accuracy (%)	Inference speed (fps)
MobileNetV2 – Dilated [11, 17]	Single convolution + deep supervision	75.75	17.20
ResNet18 – Dilated [12, 17]	Pyramid pooling module [20] + deep supervision	78.64	11.70
ResNet50 – Dilated [12, 17]	Pyramid pooling module [20] + deep supervision	79.73	8.30

Following segmentation, the matches found in the previous step will be compared and potential outliers removed as per the technique outlined in Section 4. For the evaluation performed in the testing script, the semantic similarity threshold is set to 0.2 for stereo matching, and 0.5 for matching across timesteps.

5.2.4 Traditional outlier removal

Traditional outlier removal was performed regardless of whether the semantic feature filter was used or not, as the semantic filter would remove only matches that are semantically mismatched, while still leaving in many other potentially mismatched points. These

traditional methods were implemented in one of two ways, depending on the relationship between the images in the matched pair.

For stereo matching of features taken at the same time instant, outlier removal is straightforward. Since the images are already stereo rectified, the epipolar geometry [14, *pp. 395-403*] of the image can be used to find matches that could be considered outliers. To allow for slight variations in feature location, the horizontal position of matched points was allowed to differ by a maximum of 2 pixels.

For matched images taken at different timesteps, a RANSAC-based approach was undertaken [21]. Any matches that do not conform to this relationship are then considered outliers. The RANSAC algorithm is a robust method for removing outliers. However, the number of algorithm iterations required to get a 99% confidence in the inliers increases hyperbolically with the proportion of outliers in the data [22], increasing the need for other methods of outlier removal if the initial proportion of outliers is high.

The RANSAC parameters used in the experimental script allowed for a maximum of 2000 iterations of the algorithm to be performed, aiming to achieve a certainty of 99.5%. As with the stereo-matching case, the RANSAC algorithm was allowed a 2-pixel threshold for determining if a point was an inlier or an outlier.

5.3 Hardware

All experiments were performed on an Ubuntu-based PC, with an Intel Core i7-11700 and 16 GB of RAM. All operations, including semantic segmentation, were performed using the CPU. While performing the semantic segmentation in parallel on a separate GPU would have improved performance, in testing it was decided that running the inference in a 'worst-case scenario' would still allow for a good comparison.

5.4 Testing procedure

For each combination of feature detector and segmentation network, the testing script was executed and performed a full pass of stereo feature-matching and time-displaced feature-matching on each of the 194 images contained in the dataset. The information for each match, including the corresponding ground truth (if available) was then recorded for later analysis. Each aspect of the system, such as the semantic segmentation process, the feature-matching process, and the outlier-removal process, was timed to acquire a measure of the relative performance of each component during the execution.

6 Results

After running all 194 image sets through the testing script defined in Section 5, several sets of results were generated. These are broken down into several different categories, each pertaining to a different application of feature-matching and outlier removal.

First, some observations were noticed in all the different result sets, regardless of the experimental configuration. These include images that cannot be used as part of the analysis, as they have no valid ground-truth values for comparison, as well as images with points that are mismatched even after applying outlier removal.

Second, the measures of accuracy with respect to stereo-matching are given. These results made use of the disparity between matched features, which is the difference in the horizontal direction between the feature's location in the left image vs the right image. The 3-D depth of the feature can be calculated from the disparity, so more accurate disparity values result in more accurate depth measurements.

Third, measures of accuracy with respect to matching across timesteps are given. These results were evaluated by measuring the optical flow of each feature between timesteps, which shows how the feature moves between one timestep and another. Accurate tracking of points across timesteps can be used to infer the camera's motion between the two instants, and more accurate flow values result in more accurate inferred motion.

Finally, the execution time for each stage of the script is given. These values present a measure of the relative performance of each different experimental configuration, allowing for an evaluation of the change of accuracy against the loss in performance.

6.1 General observations

As specified in Section 5.2.1, each image has 500 features detected before matching. The mean number of matches for the different configurations across all 194 image sets is given in Table 2. This table also shows the number of remaining matches after outlier removal for each of the different segmentation models.

Several of the evaluated images also had all of the detected features with no associated ground-truth values, as discussed in Section 5.1. This occurs when the features are found on objects that are very distant in the image, beyond the range of the LIDAR detector. An example of this kind of image is given in Figure 4.

Table 2. Mean amount of feature matches found in image pairs. **No segmentation:** Only epipolar constraints (for stereo-matching) or RANSAC (for matching across timesteps). **MobileNet, ResNet18, ResNet50:** Different semantic filters, followed by traditional outlier removal, as indicated. Values represent the number of points remaining after each stage, separated by a slash (/)

	Stereo-matching				
	Initial matches	Epipolar constraints only	MobileNet / Epipolar constraints	ResNet18 / Epipolar constraints	ResNet50 / Epipolar constraints
SIFT	185	171	165 / 157	164 / 155	160 / 151
ORB	252	201	211 / 185	209 / 183	200 / 176
	Matching across timesteps				
	Initial matches	RANSAC only	MobileNet / RANSAC	ResNet18 / RANSAC	ResNet50 / RANSAC
SIFT	234	146	228 / 145	227 / 143	224 / 143
ORB	297	163	286 / 163	285 / 162	281 / 162

Fig. 4. Example of image with no equivalent ground truth, as no features have been identified on the ground within range of the LIDAR sensor. Almost all of the detected features, whose locations are shown in magenta, are placed in the region where the tree canopy meets the sky.

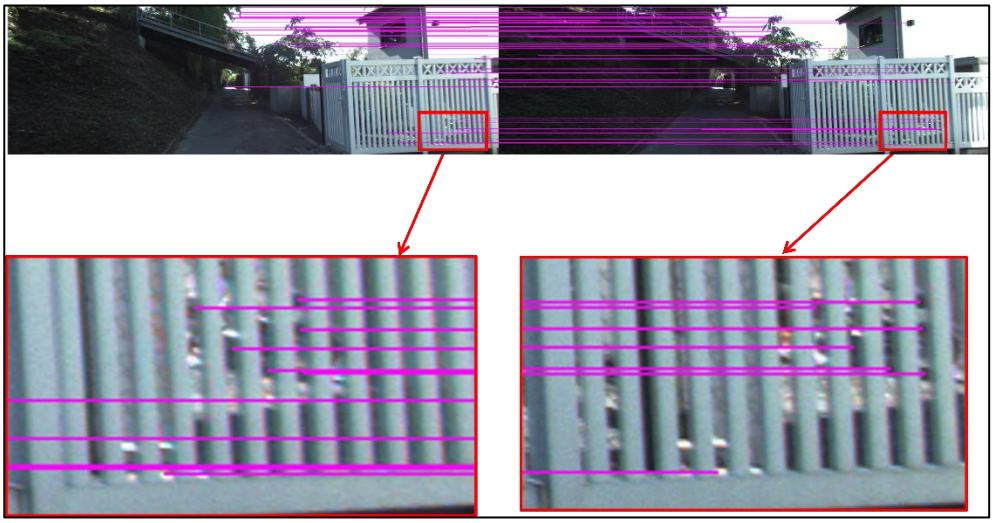


Fig. 5. Final stereo matches after both semantic and traditional outlier removal using epipolar constraints. Several incorrect matches are still present on objects with multiple similar geometric features.

Another common issue is semantically and geometrically ambiguous points being considered as matches. This can occur on objects with strong geometric features, such as the straight, parallel lines seen on fences and gates. The features behind these objects often cannot be identified by the semantic segmentation algorithm, leading to matches that may not be caught by either the semantic filter or the traditional outlier removal techniques. An example is shown in Figure 5.

6.2 Stereo-matching and feature disparity

These results look at the disparity, or the difference in a horizontal position, between features found in the left image and those found in the right image of a stereo-image pair taken at a specific time-instant.

6.2.1 Match accuracy compared with ground-truth values

For each match found in each image, the disparity value was computed. Then, the sparse disparity map containing ground-truth values taken by a LIDAR scanner was loaded, and the ground-truth disparity at the pixel location of the detected feature was extracted. If no ground-truth value was present, then the match was excluded from the comparison.

Table 3 shows the mean absolute disparity error between the measured points and the ground truth for each combination of feature detector and segmentation network.

Table 3. Mean absolute disparity error. **No segmentation:** Only epipolar constraints applied to remove outliers. **MobileNet, ResNet18, ResNet50:** Different semantic filters are applied to remove outliers, followed by traditional outlier removal, as per No Segmentation. Units in pixels.

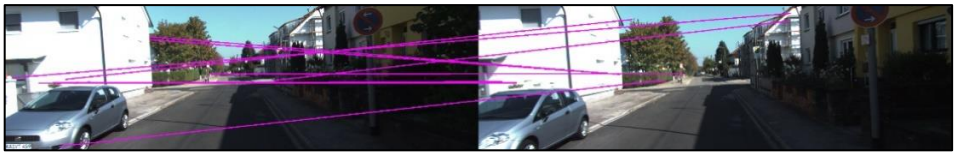
		No segmentation	**MobileNet**	**ResNet18**	**ResNet50**
SIFT	Measured value	0.8255	0.7215	0.7249	0.6957
	% Improvement	**+ 0%**	**+ 12.60%**	**+ 12.49%**	**+ 15.72%**
ORB	Measured value	1.515	1.193	1.182	1.168
	% Improvement	**+ 0%**	**+ 21.25%**	**+ 21.98%**	**+ 22.90%**

Table 3 shows how the addition of the semantic outlier removal before application of traditional outlier removal improves the accuracy of the matched features overall. It also reduced the spread of outlier values, as shown in Table 4, which illustrates the largest absolute disparity error found for each model.

Table 4. Maximum absolute disparity error. **No segmentation:** Only epipolar constraints are applied to remove outliers. **MobileNet, ResNet18, ResNet50:** Different semantic filters applied to remove outliers, followed by traditional outlier removal, as per No Segmentation. Units in pixels.

	No segmentation	**MobileNet**	**ResNet18**	**ResNet50**
SIFT	304.25	41.08	89.26	89.26
ORB	777.22	439.34	439.34	439.34

This illustrates how the application of a semantic segmentation outlier removal layer to the stereo-matching pipeline improves the quality of the matches retained.



(A)

(B)

(C)

Fig. 6. Comparison of matches removed by different filtering methods, using SIFT features and MobileNet for semantic segmentation. **A)** Matches removed by the semantic filter. **B)** Matches removed by traditional outlier removal methods after semantic filtering. **C)** Matches removed by traditional methods without semantic filtering.

6.2.2 Quality of removed matches

To further assess the results of the system, a qualitative analysis of the resulting matches is performed. Inlier matches in stereo-rectified images should be at the same height, with the point in the left-hand image being further to the left than the equivalent point in the right-hand image.

An example of the matches removed by the semantic segmentation filter is given in Figure 6. This illustrates the quality of the matches removed by the semantic outlier removal process. Many of these points are outliers, though some appear to be valid inlier matches.

Since the number of matches removed is relatively small compared to the total matches, the inlier matches removed by the semantic segmentation filter should not have a significant effect on the overall accuracy of the remaining matches.

When analysed alongside the quantitative results, the way in which the mean disparity error is reduced can be surmised. It is likely that the semantic filter removes a number of potentially geometric valid points that are also still mismatched in the scene. This helps to remove a greater number of outlier points when the two different filters are combined.

6.3 Matching across time-steps and optical flow

These results measure the displacement of features in the *x* and *y* pixel directions between two images taken at different times, usually referred to as sparse optical flow.

6.3.1 Match accuracy compared with ground-truth values

Table 5 shows the mean absolute error in flow values between features in images taken at different time-steps in both the *x* and *y* directions.

Unlike the stereo-matching case, where basic geometrical relationships helped to remove outliers, this outlier removal method used RANSAC to help remove outlier matches. The differences between having the semantic outlier removal in place versus not having it are marginal, offering only a 0.84% improvement in error reduction at best. However, these results show a high degree of variability, with increases and decreases in accuracy changing across similar tests.

Table 6 shows the absolute maximum displacement error encountered in both the *x* and *y* directions. This again shows inconsistent differences, with the semantic outlier removal technique occasionally increasing the spread of errors between matches and the ground truth in the images.

Table 5. Mean absolute disparity error. **No segmentation:** Only RANSAC applied to remove outliers. **MobileNet, ResNet18, ResNet50:** Different semantic filters applied to remove outliers, followed by traditional outlier removal, as per No Segmentation. Units in pixels.

		No segmentation	**MobileNet**	**ResNet18**	**ResNet50**
***x* direction**					
SIFT	Measured value	0.3333	0.3318	0.3401	0.3305
	% Improvement	**+ 0%**	**+ 0.45%**	**- 2.04%**	**+ 0.84%**
ORB	Measured value	0.7050	0.7225	0.7049	0.7090
	% Improvement	**+ 0%**	**- 2.48%**	**+ 0.01%**	**- 0.57%**
***y* direction**					
SIFT	Measured value	0.1833	0.1866	0.1881	0.1826
	% Improvement	**+ 0%**	**- 1.80%**	**- 2.62%**	**+ 0.38%**
ORB	Measured value	0.5347	0.5470	0.5416	0.5456
	% Improvement	**+ 0%**	**- 2.30%**	**- 1.29%**	**- 2.04%**

(A)

(B)

(C)

Fig. 7. Illustration of removed outlier points in optical flow case, using SIFT features and MobileNet for semantic segmentation. **A)** Matches removed by the semantic filter. **B)** Matches removed using a RANSAC-based filter, after semantic filtering. **C)** Matches removed by a RANSAC-based filter with no semantic filtering.

Table 6. Maximum absolute disparity error. **No segmentation:** Only RANSAC applied to remove outliers. **MobileNet, ResNet18, ResNet50:** Different semantic filters applied to remove outliers, followed by traditional outlier removal, as per No Segmentation. Units in pixels.

	No segmentation	**MobileNet**	**ResNet18**	**ResNet50**
***x* direction**				
SIFT	24.61	24.61	26.28	24.61
ORB	10.36	17.98	10.36	24.78
***y* direction**				
SIFT	3.54	5.77	6.36	2.64
ORB	7.00	7.0	7.0	3.92

6.3.2 Quality of removed matches

Like in the case with stereo-matching, the quality of the removed points in each image can be assessed to check if the removed points are truly outliers, or if any of them appear to be inlier points. This is visualised in Figure 7, which shows the removed points for each of the outlier removal stages in the program. Figure 7 also illustrates, in a comparable way to Figure 6, how the semantic filter removes a majority of the worst outliers.

Unlike the stereo-matching case, the RANSAC-based outlier removal method removes the majority of removed matches. Matches with mismatched semantic labels are far more likely to not conform to the needed homographic transformation, and thus are much more likely also to be removed by the RANSAC algorithm.

However, just these values alone do not paint the whole picture, mostly due to the behaviour of the RANSAC algorithm. The RANSAC algorithm's computational performance decreases if there is a large number of outliers. Thus, while no gains in accuracy may be apparent, the performance and convergence time of RANSAC would likely be improved, especially on data polluted with large amounts of outliers.

6.4 Execution time

The mean time required to segment each image using the system described in Section 5 is given in Table 7. These results correspond well with the expected relative performance of each model shown in Table 1. The processing time of the feature detections and matching, as well as outlier removal is given in Table 8, for both the fixed kernel used in the results given in the previous sections, as well as a comparative experiment completed using a dynamically sized kernel.

Table 7. Mean single image segmentation time using experimental set-up. Units in seconds

Model	**Time**
MobileNet	2.188
ResNet18	3.530
ResNet50	6.390

These results show that the addition of the semantic segmentation filter to the outlier-removal pipeline does significantly increase the execution time of the system. The segmentation algorithm takes the longest time to process, when compared to feature-matching and outlier removal, though this can be improved with better hardware and parallelisation. The outlier-removal process also slowed the performance of the system, though not to the extent that adding the semantic segmentation does.

Table 8. Mean processing times for combined stereo-matching (one image pair) and optical flow (one image pair) across all different segmentation models. Outlier removal includes both RANSAC and epipolar constraints, as well as semantic filters in the segmentation cases. Units in seconds

		No segmentation	With segmentation (fixed kernel)	With segmentation (dynamic kernel)
SIFT	Feature detection and matching	0.2071	0.2089	0.2032
	Outlier removal	**0.07901**	**0.8345**	**0.9537**
ORB	Feature detection and matching	0.0320	0.03366	0.03037
	Outlier removal	**0.1268**	**1.1407**	**6.4232**

However, Table 8 shows that the use of a fixed kernel had better performance than the dynamic kernel, especially in the case of ORB features. This is likely due to the fact that ORB features tend to have larger diameters, on average. This increases the size of the feature descriptor, which leads to a more pronounced increase in the execution time, due to the quadratic relationship between the number of pixels in a feature and the feature's diameter. The difference in accuracy seen in the stereo-matching and optical flow tests between the fixed and dynamic kernel was marginal, never differing by more than 2.5% in all cases.

7 Conclusion

The analysis of the results given in Section 6 showed that the addition of a semantic filter to a stereo-matching pipeline reduced the mean disparity error compared to ground-truth values. Since the underlying matching algorithms were not changed, this reduction showed that the number of outlier measurements polluting the mean must have been reduced.

For the case of matching points between different poses at different timesteps, the overall reduction in outliers was not as apparent. However, these results did not paint the whole picture, as these results were more greatly affected by the behaviour of the RANSAC algorithm. While the effect on the accuracy values was small, the potential boost given to the performance of the RANSAC algorithm due to the removal of outliers was also noted.

In both cases, the extra processing needed to perform the semantic segmentation on the various image pairs did significantly slow the rate at which images could be processed. The speed of semantic segmentation could be improved through the use of better and more specialised hardware. General processing performance could also be improved by improving the efficiency of the code implementation, such as switching from an interpreted language like Python, which was used throughout this paper, to a compiled language like C++. If the semantic data, however, was already available from some other source, or was being used for some other purpose, then the addition of a semantic filter to a stereo-matching pipeline would add far less performance cost, with a much more noticeable gain in accuracy.

When combined with better hardware, the addition of semantic information to feature-based methods for stereo-matching did add further possibilities for data collection and inference. Many applications rely on extracting as much information as possible from still images, and having more of that information readily available for other applications is an attractive option.

8 Future work

Although the results presented in this paper show promise, the performance costs associated with achieving them are an undesirable outcome, especially for real-world applications. However, these experiments were performed using non-ideal hardware systems. Thus, in

order to better measure the ideal performance of the system as designed, further experimentation using more powerful hardware, better designed for DNN-based inference, should be used. This would also allow for better parallelisation of the segmentation operation, working on both the left and right images simultaneously, rather than sequentially. These hardware improvements could also be combined with software optimisations, to make the inference and filtering operations more efficient, and thus improve the performance even more.

To better evaluate the use of semantic segmentation in RANSAC-based approaches to outlier removal, experiments analysing the effect on RANSAC performance with fewer points, and larger proportions of outliers present, can be performed. These experiments could also better illustrate the stochasticity inherent in the RANSAC algorithm, and measure what effect this has on the accuracy with and without the semantic filter.

Finally, the addition of semantic information to more traditional landmark-based vision systems, such as SLAM, could have benefits that go beyond just improved feature matching. The semantic class of each landmark could be used to discriminate between static features and features located on potentially dynamic objects.

References

1. J.J. Aguilar, F. Torres, M.A. Lope, *Stereo vision for 3D measurement: accuracy analysis, calibration and industrial applications*, *Measurement*, **18**, 193-200 (1996)
2. H. Suenaga, H.H. Tran, H. Liao, K. Masamune, T. Dohi, K. Hoshi, T. Takato, *Vision-based markerless registration using stereo vision and an augmented reality surgical navigation system: a pilot study*, BMC Medical Imaging, **15** (2015)
3. M.A. Fischler, R.C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, in Communications of the ACM, **24**, 381-395 (1981)
4. K.J. Doherty, D.P. Baxter, E. Schneeweiss, J.J. Leonard, *Probabilistic Data Association via Mixture Models for Robust Semantic SLAM*, ICRA (2020)
5. S.L. Bowman, N. Atanasov, K. Daniilidis, G.J. Pappas, *Probabilistic Data Association for Semantic SLAM*, ICRA (2017)
6. S.H.K. Tareen, Z. Saleem, *A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK*, iCoMET, 1-10 (2018)
7. D.G. Lowe, *Distinctive image features from scale-invariant keypoints*, IJCV, **60**, 91-110 (2004)
8. E. Rublee, V. Rabaud, K. Konolige, G. Bradski, *ORB: An efficient alternative to SIFT or SURF*, IJCV, 2564-2571 (2011)
9. M. Muja, D.G. Lowe, *Fast approximate nearest neighbours with automatic algorithm configuration*, VISAPP, **1**, 331-340 (2009)
10. C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, Z. Xie, *Deep Learning and Its Applications in Biomedicine*, Genomics, Proteomics & Bioinformatics, **16**, 17-32 (2018)
11. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.C. Chen, *MobileNetV2: Inverted Residuals and Linear Bottlenecks* (2018)
12. K. He, X. Zhang, S. Ren, J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, ICCVPR, 770-778 (2016)

13. O. Ronneberger, P. Fischer, T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, MICCAI, 234-241 (2015)
14. L. G. Shapiro, G.C. Stockman, *Computer Vision* (2001)
15. A. Geiger, P. Lenz, R. Urtasun, *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*, IJCV (2012)
16. G. Bradski, *The OpenCV Library*, Dr. Dobbs Journal of Software Tools (2000)
17. B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, A. Torralba, *Semantic understanding of scenes through the ADE20K dataset*, IJCV (2018)
18. B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, A. Torralba, *Scene Parsing through ADE20K Dataset*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, ICCVPR (2017)
19. CSAILVision, *PyTorch Implementation for Semantic Segmentation/Scene Parsing on MIT ADE20K dataset*, Available at: https://github.com/CSAILVision/semantic-segmentation-pytorch (2020) (Accessed: 14 April 2022)
20. H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, *Pyramid Scene Parsing Network*, CVPR (2017)
21. W. Brink, *Stereo vision for simultaneous localization and mapping* (Stellenbosch University, 2012)
22. K.G. Derpanis, *Overview of the RANSAC Algorithm*, (2010)