Robust monocular obstacle detection using a hybrid ground plane detection method

Will Gittoes
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
willgittoes@gmail.com

Richard Green
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand

***Abstract*— Most monocular vision algorithms usually work only under very specific constraints, or require a great deal of training. This paper proposes a method to use a novel combination of appearance-based and optical flow-based ground plane detection methods that work together to overcome situations that challenge each method individually. The proposed method overcomes several limitations encountered by other methods, and improves obstacle identification by 2.2%. The algorithm is then used to inform navigation of a small robot.**

I. INTRODUCTION

Monocular vision has been described as a worthwhile and popular method of obstacle detection, as opposed to using either dedicated range-finders or stereo-vision, for many reasons. Vision researches have cited many reasons for such a determination, including: the wide field of view offered by cameras [1, 2], low cost [2-4], compactness [3], ability to detect small obstacles [2], and the passive nature of receiving images [5, 6] as opposed to sending out a laser beam or other structured light.

There are two [2, 7] main approaches to detecting obstacles: using appearance or optical flow. Both of these methods aim to identify an area of the image as belonging to the ground plane, then map that area into world coordinates in order to build a map of occupied and unoccupied areas [5, 8].

A. Appearance-Based Obstacle Detection

Appearance-based obstacle detection methods operate on a single image [4] and use features such as colour [2, 5], edges [4, 8, 9], frequency [10] and texture [9] to identify an area as belonging to the ground plane.

One method to discover such a 'ground plane area' is to flood-fill the image starting from an estimated ground point [5]; the filled area is then classified as the ground plane. Another similar method involves segmenting the image into regions, and using colour and position to identify one region as the ground [11]. Several methods have been described that use vertical scan lines that run vertically upwards until an edge is detected [4, 8], defining the area below the detected edge points as the ground. There are also several methods that work by comparing pixel colours against pixel colours from a ground sample [2, 10], with some implementations using a machine learning approach to remember ground pixel appearances to aid in classifying new ground pixels [2, 10, 12].

The most important advantage offered by appearance-based methods is that no motion is required in order to calculate a result[4]. This means that even when a robot is motionless, obstacle information can be computed and used for path planning. Another major advantage is that the occupancy information is very high resolution [8]; ground planes can be discovered per-pixel. Appearance-based methods also work very well even on featureless surfaces [2, 13], and have been described as faster than optical flow [1, 14].

The main disadvantages of using an appearance-based approach are identification errors. All of the methods described need some starting knowledge about the ground, such as a point to fill from or pixel colour data to compare against [7] – this either requires a limited library (such as in [2] and [10]) and/or assuming a ground position close to the camera (such as in [2], [4], [5] and [8]). Textured ground can pose a problem [2, 15], as it is hard to fill or segment, one example being specular reflections causing a strong intensity anomaly [2]. Flood fills can also cause false negatives (incorrect identification of obstacles as belonging to the ground) due to 'bleeds' caused by leaking through ill-defined edges [5]. Finally, there is always the "perpetual colour constancy" problem [5, 16], where colours that appear the same to the human eye can vary wildly in absolute terms, which can also be exacerbated by cameras performing automatic white balance and shutter speed adjustment.

B. Optical Flow-Based Obstacle Detection

Obstacle flow-based (motion-based) approaches to obstacle detection rely on tracking feature points between two images. A ground-plane homography matrix can be calculated and then be used to allow comparison between the movement vectors and the expected movement vectors for the ground plane [16]. Each feature point can then be located in 3D [17], allowing classification as either belonging to the ground plane, or belonging to an obstacle.

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H^{-1} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

A ground plane homography matrix H is calculated, and can be used to map homogenous pixel coordinates u, v and w into ground plane coordinates X and Y [16].

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

H can be calculated given the following equations [16]:

$$h_{11} = u_0 \cos\varphi \qquad h_{12} = -\alpha_u$$

$$h_{13} = \alpha_u + u_0 (-\cos\varphi + c_z \sin\varphi)$$

$$h_{21} = -\alpha_u \sin\varphi + v_0 cos\varphi \qquad h_{22} = 0$$

$$h_{23} = \alpha(sin\varphi + c_z cos\varphi) \qquad h_{31} = cos\varphi$$

$$h_{32} = 0 \qquad h_{33} = -cos\varphi + c_z sin\varphi$$

Where v_t is the translational robot velocity, (u_0, v_0) is the camera centre, φ is the camera pitch and (α_u, α_v) are the focal length along the camera x and y axes respectively.

All optical-flow methods work this way, with most of the differentiation in features aimed at improving accuracy and resolution, or extending obstacle detection to full 3D Simultaneous Localization and Mapping (SLAM). For example: [16] analyses pixel colour gradients around feature points to try and extrapolate feature boundaries, and [1], [3] and [18] use Extended Kalman Filters (EKF) to increase feature tracking accuracy. [1], [3] and [18] also utilize optical flow for SLAM, allowing them to build a complete 3D textured model of the surrounding environment. While increasing accuracy of obstacle detection is broadly useful, it is important to note that full 3D environment mapping is "excessively complex" [19] when the requirement is only to detect obstacles, and that the problem can be reduced to the simpler issue of binary ground plane detection [16].

The main advantage of using a motion-based approach is that it does not require an external knowledge of the ground plane in the way that appearance-based approaches do [20]. It also works well on textured surfaces [2], and works despite colour changes (such as due to white balance adjustment or lighting changes) – as long as features can still be tracked.

The main disadvantage of optical flow is that, by definition, the usefulness of the data depends entirely on the quality of the features being tracked. Therefore, optical flow *requires* textured surfaces in order to produce good results [2, 21]. This also means that motion is required before any calculations can be performed at all. Good camera calibration is also required, as well as precise knowledge of the camera position and orientation relative to the ground plane.

II. CONSTRAINTS

The primary goal of this paper is to describe a more robust obstacle detection algorithm, therefore the constraints and assumptions that are required are fewer. Firstly, it is required that the ground plane is always flat[19], so that a homography matrix can be calculated. Secondly it is assumed that the camera position with respect to the ground plane does not move (except translationally in parallel to the ground) [19], again to ensure accurate calculation of the expected ground plane. Thirdly, we assume the vehicle moves at a constant, known velocity[19], as knowing our velocity and heading – and not needing to extrapolate these from the same data used for obstacle detection – makes it much easier to calculate obstacle positions[18]. Where the constraints are much relaxed from other papers is to do with ground plane appearance: neither a textured nor an untextured surface is required, nor is it required that the camera be travelling over a 'familiar' pre-learned surface texture.

III. METHOD

As can be seen from the above discussion of appearance-based and optical flow-based obstacle detection, each method has several important disadvantages. However, many disadvantages held by one method are situations in which the other method works very well. Consider the issue of ground plane texture: optical flow requires a ground plane texture in order to detect features [2], but appearance-based methods require an untextured ground plane, for easier segmentation or comparison. Also consider the issue where appearance-based methods all need a starting point on the ground, or a selection of ground pixels in order to work [7], and that colour inconsistencies between frames can cause these methods to fail [5]. However, optical flow analysis gives us absolute data points. In addition, optical flow requires the robot to be in motion, while appearance based methods can work on a single frame.

This method therefore uses a combination of appearance-based and optical-flow based methods, so that each method's advantages may cancel out the other's disadvantages. Firstly, optical flow is used to gather a selection of points that lie on the ground plane, and then these points are used to inform the assumptions of an appearance-based method. The appearance-based method will then yield a high resolution binary [22] obstacle mask (in image-space) that is then transformed into an obstacle map. The process is as follows:

1. Calculate optical flow, and select the locations of the features on the ground plane.
2. Use ground plane points to inform a reference area similar to the one used in [2].
3. Create a histogram of the colour (hue and intensity) of the reference pixels.
4. Classify all pixels in the image as either belonging to the ground or not belonging to the ground, based on comparison with the colour distribution generated in 3.
5. Convert the image-space obstacle map into a world-space obstacle map, and use this to create an overhead occupancy grid for navigation.

Each of these steps is described in more detail in the subsequent sections.

A. Optical Flow Analysis

To calculate optical flow, we compare the image-space movement vectors of a group of tracked features against the theoretical image-space movement of these features if they were on the ground plane. Vectors that correspond indicate that the feature lies on the ground plane; otherwise the feature lies on an obstacle.

Ground plane homography can be calculated once and reused indefinitely as long as the camera maintains a consistent orientation with respect to the ground plane [23]. A script is used that detects the corners of a chessboard and uses their relative positions to calculate camera pose with regards to the plane on which the corners lie. The chessboard is placed on the ground, therefore giving the ground plane homography matrix.

Feature points are selected using cvGoodFeaturesToTrack, with a quality threshold of 0.04 and a min distance of 8 pixels. This ensures feature points with high eigenvalues are selected, which are likely to remain translation and scaling invariant. The features are tracked using Lucas-Kanade.

These feature points are tracked between frames to create image-space movement vectors. Longer vectors will be less susceptible to noise; therefore two sets of features are tracked simultaneously on an alternating basis. Every 2 seconds, new features are selected, and every 1 second, the set of features that is chosen for optical flow calculation is swapped to ensure vectors with the least amount of noise are inspected. A set of tracked feature points and their image-space movement vectors is shown below in fig. 1.

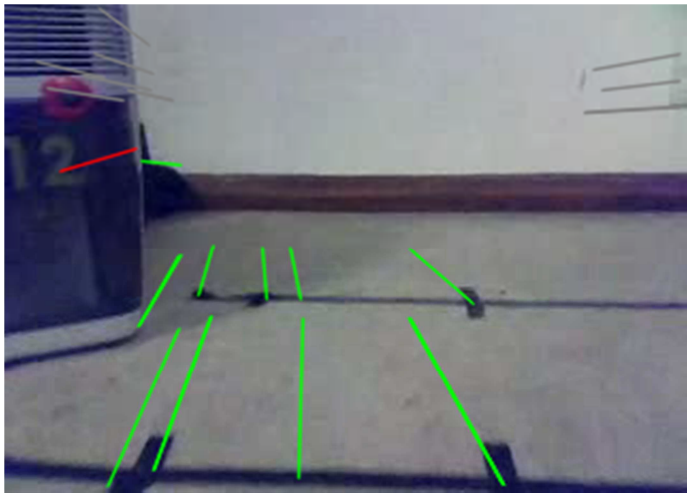

Figure 4. The screen-space movement vectors of several tracked points.

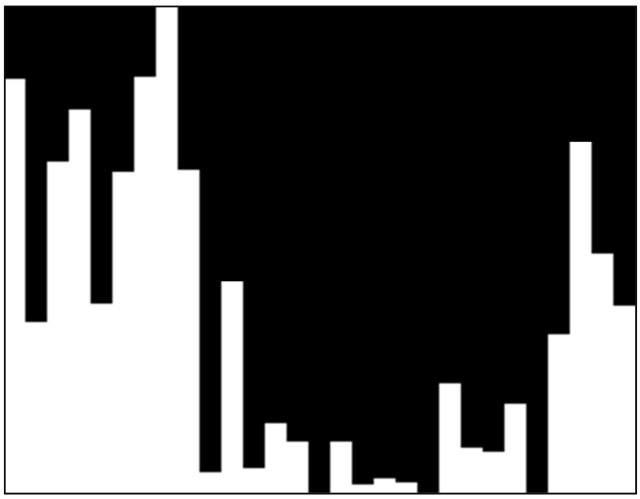

Figure 3. Hue distribution histogram for ground plane pixels in Figure 1

Figure 1. Value distribution histogram for ground plane pixels in Figure 1

Figure 2. Ground plane obstacle mask for the image in Figure 1. White areas are obstacles, black is the ground plane.

These vectors are then translated into world space coordinates using the inverse of the ground plane homography matrix as shown in equation (1). The world-space vectors are then compared against the known movement vector of the robot, and if they are within a certain threshold (4 cm), then the feature is considered to be on the ground.

In Figure 1, grey vectors are those that are above the horizon and are discarded, red vectors are obstacles and green vectors are for features that have been identified as belonging to the ground plane.

B. Ground Plane Reference Area Selection

The points obtained in A. are now used to inform selection of an area that is calculated to be on the ground plane. Regions around the current location of the features are selected and subtracted from a mask.

C. Ground Plane Pixel Value Histogram

The pixels selected in B. are now compiled into a histogram of hue and value. Pixels with low saturations are filtered from the hue histogram, to avoid noise, as in [16]. Figure 2 shows a hue histogram, while Figure 3 shows a value histogram.

D. Entire Image Classification

Each pixel in the entire image frame is now classified as either belonging to the ground plane or not belonging to the ground plane. The pixel's intensity and hue (should the saturation target be met) values are compared to the ground plane distribution, and if they are significantly inside that distribution, the pixel is classified as belonging to the ground, otherwise it belongs to an obstacle. The result is a binary image mask with the inverse of the entire ground plane highlighted.

Due to feature points often being attached to anomalously coloured sections of ground, a flood fill is performed at the screen location of each feature point that has been determined to be located on the ground plane. This ensures that the black tape visible in Figure 1 does not show up as an obstacle, as it would in most methods that use appearance alone.

Figure 4 shows the obstacle mask for the image in Figure 1.

E. Obstacle Occupancy Grid Creation

The image created in Section D. is now transformed into world-space by a perspective warp using the inverse ground plane homography matrix. The birds-eye obstacle map is then scaled, translated to a position relative to the robot's absolute world position, and then rotated to match the robot heating. Figure 5 shows the perspective transform of the obstacle map shown in Figure 4.

The obstacle map is represented by a grid at the resolution of 1 cell per 50cm^2. As the perspective warp of the obstacle mask becomes noisier as the distance from the robot increases [24], a set of obstacle map cells are chosen that are within 3 cells of the robot.

For each of these cells, a mask is computed that reveals just the area of the obstacle mask covered by the cell. This region is then examined, and if the number of obstacle pixels is greater than a certain threshold (25%), then the cell is considered occupied.

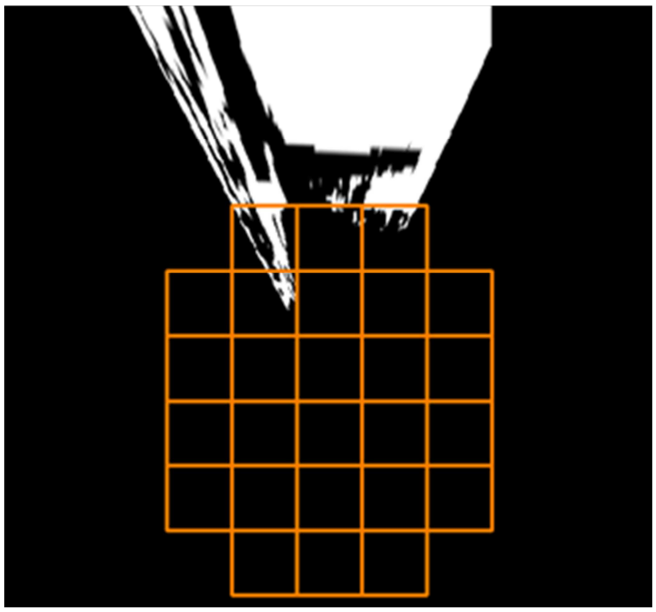

Figure 6. Transformed obstacle map from Figure 4

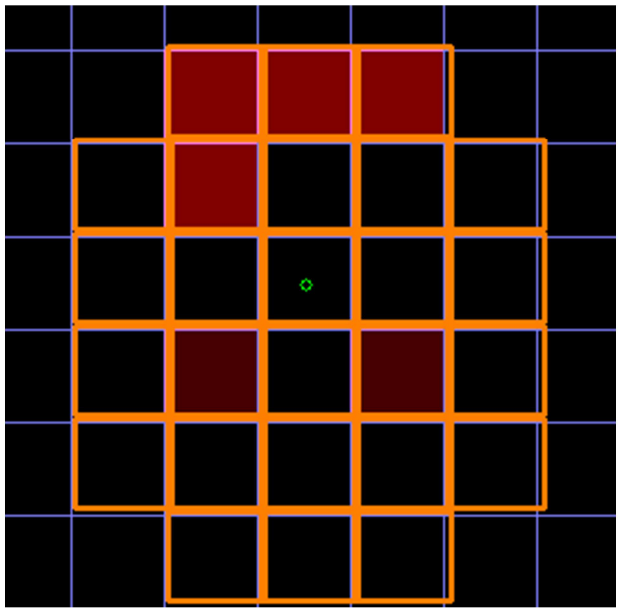

Figure 5. Obstacle occupancy map

Each cell has a probability value that represents the likelihood of that cell containing an obstacle. When a cell is found to be occupied, the associated probability value is increased. The robot navigation module uses a threshold (90%) to determine whether a cell is considered occupied or not.

As the robot moves through the environment, this overhead occupancy grid is continually filled out, and used in inform the navigation of the robot. Figure 6 shows the obstacle occupancy map at the time Figures 1-5 were taken.

The orange squares in figures 5 and 6 represent the area in which we are actively looking for obstacles, and are at a resolution of 50cm2 per cell. In Figure 6, the green circle represents the position of the robot, and in Figure 5 the robot is at the centre of the screen. Red squares in Figure 6 indicate the presence of an obstacle, and the robot will not try to navigate through these squares.

IV. Robot Platform

The robot platform is designed to be as simple as possible, allowing rapid prototyping and easy evaluation of the obstacle-detection algorithm.

A cheap, low-power netbook will be affixed to a small robot wheel base. The netbook will provide a webcam for image acquisition and the processing hardware to run the detection algorithms. The wheel base has two drivable voltage motors and a third omni-directional wheel that allow forward movement and on-the-spot rotation.

An ATmega328 microcontroller in an Arduino development board along with a DC motor controller board provides pulse-width modulation-based analogue control of the motor speed.

A. *Odometry and Heading Control*

Voltage motors were chosen over stepper motors because they provide far superior speed, however they lack precise control. While speed control is accurate enough to roughly calculate distance odometry, heading drift requires correction. This correction is calculated using the centrum of motion, in order to compensate the motor speeds and correct listing.

Firstly, the centrum of motion was calculated. This was done by interpolating of the intersection points of all the known-ground plane feature vectors. Only ground plane features, as only the movement vectors of coplanar points share a common centrum of motion.

Once the centrum of motion is calculated, the motor speeds can be adjusted. If the centrum is in the left field, then power to the left wheel is increased while power to the right wheel is decreased. If the centrum is in the right field, the opposite is done.

V. Results

A. *Quantitative Analysis*

To objectively classify the performance of the algorithm, the per-pixel accuracy of the image classification is measured.

Several images and their associated masks were recorded during testing of the robot. An obstacle mask was designated by hand for each of those images, with the assumption that these hand-crafted masks represent perfect accuracy. The obstacle mask as computed by the algorithm is then compared against the control mask, and the number and percentage of correctly and incorrectly classified pixels are calculated.

This method was chosen because it produces quantifiable results where the construction of the perfect comparison case is not biased by the algorithm. It may seem more appropriate to perform a similar pixel-counting analysis on the transformed, birds-eye obstacle map – but in such a case, the hand-calculated comparison would have to be based on the transformation of the base image – which would incur the same bias and inaccuracies as the mask. Comparing the final obstacle map against the actual environment has also been considered, but this would not provide sufficient resolution for the purposes of evaluation.

Figure 7 shows an example of a comparison mask, generated for the image in Figure 1, while Figure 5 shows the actual mask as generated by the algorithm. A white pixel indicates the presence of an obstacle, while a black pixel signifies the presence of the ground plane. Figure 8 highlights the false positives – pixels in the computed mask that are not in the comparison mask – while Figure 9 highlights the false negatives – pixels in the comparison mask that are not in the computed mask.

Figure 8. Hand-crafted ground truth obstacle mask for Figure 1.

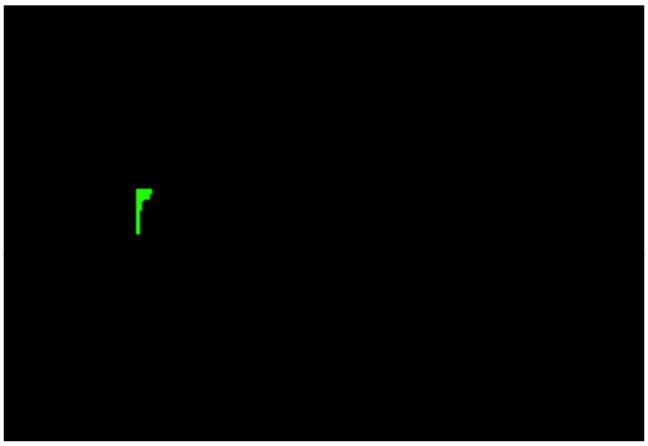

Figure 7. False positives in the obstacle mask

Figure 10. False negatives in the obstacle mask

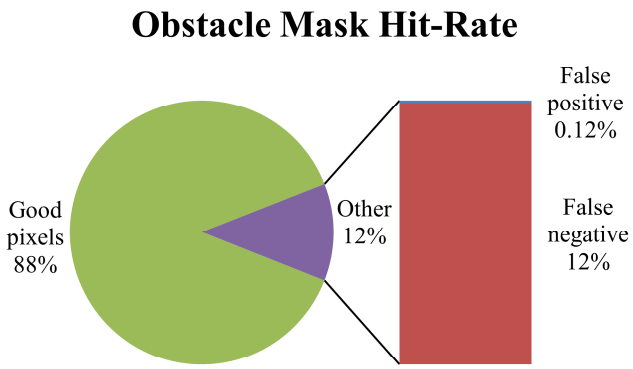

Figure 9. Relative number of pixels correctly and incorrectly classified in the obstacle mask

We can see that while very few areas of the ground were misidentified as obstacles, there were several areas that were falsely identified as belonging to the ground that do not. However, the false negatives are not too numerous to have a strong adverse effect on the detection of obstacle presence, and the majority of obstacle areas are bounded by mask pixels, ensuring that the robot will not try to navigate to those areas.

Figure 10 shows a graph of the percentage of pixels classified correctly and incorrectly, from several obstacle mask images.

Pixels were correctly classified in 88% of cases, with the vast majority of misclassified pixels being false negatives.

Very few previous monocular vision papers present quantitative results, so a direct comparison is difficult. In [19], a feature point miss rate of 14.2% is found, which is 2.2% worse than the 12% obstacle pixel miss rate found here.

B. Qualitative Analysis

While previous papers did not generally provide figures they did describe problems with their method; allowing a more qualitative comparison with the method presented here.

Previous monocular vision papers have stressed that appearance-based methods have problems dealing with textured floors [8]. The floor shown in Figure 1 was augmented with black tape that starkly contrasted with the normal floor colouring; appearance-based-only methods would struggle with such a scene, likely classifying the tape as an obstacle. The same qualities that make these areas difficult to segment using appearance also ensure that they are chosen as good feature points. The extra information provided by the optical flow analysis ensures that the algorithm is aware that these areas are on the ground plane, and therefore they are not misidentified as obstacles.

Another issue identified with appearance-based approaches is colour inconsistency, such as in [5] and [16], even between sequential frames in the same environment. This problem was not encountered using the hybrid approach, because the section of the algorithm using appearance works in a relative way within the confines of a single frame, and is therefore stateless with regards to time. This ensures that colour inconsistency is never an issue.

Some methods were unable to run in real-time, even on 2008 desktop hardware [4], however our method ran at 30 frames per second on similar hardware (2.6 GHz, single core), at a resolution of 320x240 pixels.

VI. Limitations and Future Work

While the algorithm ran at 30 frames per second on a desktop, performance on the target hardware was not as strong, running at only 4 frames per second. While this did not affect feature tracking or obstacle mask generation, it did negatively affect the cumulative creation of the obstacle map and the response time of the robot to obstacles. There exist laptops of a similar physical size with much higher performance, and it would be a simple future experiment to examine the performance of the algorithm on such a machine.

The algorithm was also strongly sensitive to a hardcoded value for robot speed. If the value was set incorrectly optical flow obstacle detection and obstacle map generation were strongly affected. Unfortunately, the variability of the motors

caused this to be an occasional issue. Optical flow can only be accurate either for motion calculation or obstacle detection, but not both [3], because both problems are strongly related [25]. A relatively simple solution would be the future addition of an odometry device, such as a mouse optical sensor or an odometer that measures wheel-rotation.

Kalman filters could also be applied to both feature tracking and robot position, in order to reduce noise and increase accuracy [26].

The robot supports only two kinds of motion – forwards motion and on-the-spot rotation. A future expansion could be to allow curved trajectories, however this would make optical flow calculation more complex.

VII. Conclusion

A monocular vision system has been developed for a small robot that uses a combination of optical flow processing and appearance inspection. The algorithm calculates the presence and pixel-location of obstacles in image frames with 88% accuracy. The obstacle data can be compiled into a 2D obstacle map of the robot environment, to allow navigation. The robot has been tested in environments designed to challenge standard optical flow-based and appearance-based methods, and has detected obstacles reliably when sufficient processing power was available from a nearby desktop PC.

References

[1] E. Einhorn, *et al.*, "A Hybrid Kalman Filter Based Algorithm for Real-time Visual Obstacle Detection ".

[2] I. Ulrich and N. Illah, "Appearance-Based Obstacle Detection with Monocular Colour Vision," in *AAAI National Conference on Artificial Intelligence*, Austin, Texas, 2000.

[3] E. Einhorn, *et al.*, "Monocular Scene Reconstruction for Reliable Obstacle Detection and Robot Navigation," in *European Conference on Mobile Robots*, Dubrovnik, 2009, pp. 7-12.

[4] Q. Zhan, *et al.*, "Automatic Navigation for A MobileReobot with Monocular Vision," IEEE, Beijing2008.

[5] T. Taylor, *et al.*, "Monocular Vision as a Range Sensor," in *CIMCA*, Gold Coast, 2004, pp. 566-575.

[6] B. Call, *et al.*, "Obstacle Avoidance For Unmanned Air Vehicles Using Image Feature Tracking," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Keystone, Colorado, 2006, pp. 1-9.

[7] H. Wang, *et al.*, "Real-Time Region-Based Obstacle Detection with Monocular Vision," in *IEEE International Conference on Robotics and Biometrics*, Beijing, 2005, pp. 615-619.

[8] S. Lenser and M. Veloso, "Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision," in *IEEE International Conference on Intelligent Robots and Systems*, Las Vegas, 2003, pp. 886-891.

[9] M. Blas, *et al.*, "Fask color/texture segmentation for outdoor robots," in *Intelligent Robots and Systems, IEEE Conference on*, 2008, pp. 4078-4085.

[10] T. W. Ubbens and D. C. Schuurman, "Vision-Based Obstacle Detection Using a Support Vector Machine," IEEE, Ontario, Canada2009.

[11] C.-H. Lin, *et al.*, "Robust Ground Plane Detection for Obstacle Avoidance of Mobile Robots Using a Monocular Camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010, pp. 3706-3711.

[12] D. Iakovidis, *et al.*, "Color Texture Recognition in Video Sequences using Wavelet Covariance Features and Support Vector Machines," in *Euromicro Conference 2003, Proceedings of*, 2003, pp. 199-204.

[13] Y. Chao and Z. Changan, "Obstacle detection using adaptive color segmentation and planar projection stereopsis for mobile robots," in *Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on*, 2003, pp. 1097-1101.

[14] R. Katz, *et al.*, "Dynamic Obstacle Detection Based on Probabilistic Moving Feature Recognition," *Field and Service Robotics*, pp. 83-91, 2008.

[15] Y. Shen, *et al.*, "Monocular Vision Based Obstacle Detection for Robot Navigation in Unstructured Environment," in *Advances in Neural Networks – ISNN 2007*, D. Liu, *et al.*, Eds., ed: Springer Berlin / Heidelberg, 2007, pp. 714-722.

[16] A. Jamal, *et al.*, "Real-time Ground Plane Segmentation and Obstacle Detection for Mobile Robot Navigation," IEEE, Bangalore, India2010.

[17] N. X. Dao, *et al.*, "Visual navigation for indoor mobile robots Using a single camera," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 2005, pp. 1992-1997.

[18] E. Einhorn, *et al.*, "Can't Take my Eye off You: Attention-Driven Monocular Obstacle Detection and 3D Mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010, pp. 816-821.

[19] M. Illic and S. Masciangelo, "Ground Plane Obstacle Detection from Optical Flow Anomalies: A Robust and Efficient Implementation," Genova, Italy.

[20] W. Kruger, *et al.*, "Real-time estimation and tracking of optical flow vectors for obstacle detection," in *Intelligent Vehicles '95 Symposium, Proceedings of*, 1995, pp. 304-309.

[21] K. Yamaguchi, *et al.*, "Moving Obstacle Detection using Monocular Vision," in *Intelligent Vehicles Symposium*, Tokyo, Japan, 2006, pp. 288-293.

[22] P. Batavia and S. Singh, "Obstacle detection using adaptive color segmentation and colour stereo homography," in *Robotics and Automation, 2001 Proceedings of*, 2001, pp. 705-710.

[23] J. Zhou and B. Li, "Robust Ground Plane Detection with Normalized Homography in Monocular Sequences from a Robot Platform," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 3017-3020.

[24] J. Weng, *et al.*, "Camera Calibration with Distortion Models and Accuracy Evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 965-980, October 1992.

[25] R. Munguia and A. Grau, "Monocular SLAM for Visual Odometry," in *Intelligent Signal Processing, IEEE International Symposim on*, 2007, pp. 1-6.

[26] S. Fu and G. Yang, "Uncalibrated monocular based simultaneous localization and mapping for indoor autonomous mobile robot navigation," in *Networking, Sensing and Control, ICNSC International Conference on*, 2009, pp. 663-668.