Real-Time Detection of Moving Objects by a Mobile Robot with an Omnidirectional Camera

Domagoj Herceg, Ivan Marković, and Ivan Petrović
University of Zagreb Faculty of Electrical Engineering and Computing
Department of Control and Computer Engineering
Zagreb, Croatia
Email: {domagoj.herceg, ivan.markovic, ivan.petrovic}@fer.hr

***Abstract*—Mobile robots equipped with an omnidirectional camera have gained a considerable attention over the last decade. Having an entire view of the scene can be very advantageous in numerous applications as all information is stored in a single frame. This paper is primarily concerned with detection of moving objects from optical flow field in cluttered indoor environments, which is necessary for safe navigation and collision avoidance. The algorithm is based on the comparison of the measured optical flow vectors with the generated ones. As depth information is not available, a novel method is proposed which iteratively generates optical flow vectors for different possible real world coordinates of the objects in the scene. This is necessary in order to incorporate motion estimates given by motor encoders. Back-projecting into image is then used to generate synthetic optical flow vectors needed for comparison. The algorithm was tested on a real system and was able to successfully localize a moving object under both artificial and natural lighting. The proposed algorithm can be implemented in real-time on any system with known calibrated model of the omnidirectional sensor and reliable motion estimation.**

I. Introduction

Omnidirectional cameras by definition provide a 360° view of its surrounding scene, thus making them a very attractive sensor. Such an enhanced field of view can by obtained by using several panoramic cameras, a combination of a standard camera and a mirror, or simply a standard camera with a wide-angle lens. This work is concentrated on an omnidirectional camera constructed from a standard camera and a hyperbolic mirror.

The advantages of omnidirectional cameras is followed by distortions caused by the mirror, and smaller resolution since the entire surrounding scene is fitted into a single standard camera frame. Furthermore, when detecting moving object by means of optical flow with non-stationary camera, e.g. a camera mounted on a mobile platform, the problem becomes more involved since the motion in the picture caused by the camera movement has to be taken into account.

When omnidirectional cameras are placed on a mobile platform, they are often used for estimating ego-motion or localization [1], [2], [3]. Furthermore, when tracking moving objects it is practical to pair them with a laser sensor [4], [5], and in order to detect moving objects from a sequence of images, optical flow field is often analyzed [6], [7].

This work was supported by the Ministry of Science, Education and Sports of the Republic of Croatia under grant No. 036-0363078-3018.

In this paper we are concentrated on detecting moving objects from the optical flow field recorded by an omnidirectional camera placed on top of a mobile robot platform as depicted in Fig 1. The obtained information could then be used for local navigation, obstacle avoidance etc. In [8] some properties of optical flow vectors for motion segmentation with a moving omnidirectional camera were analyzed. The results suggested that the angle of the optical flow field vectors could be used for moving object detection. In the present work, the proposed algorithm is based on searching for differences between the estimated optical flow field using mobile robot's odometry and the calculated optical flow field from the sequence of images. We are using only the omnidirectional camera and the wheel odometry, while the optical flow field is calculated in the raw omnidirectional image, i.e. no image unwarping is performed. The results show that the algorithm is capable of detecting moving objects in highly cluttered environments with both natural and artificial lighting.

II. Kinematic model of a differential drive mobile robot

In the present work we are using a Pioneer 3-DX mobile robot, with a differential drive and one passive caster wheel. Furthermore, we are using robot motion information, making it crucial to setup a motion model for estimation.

Calculating odometry solely on the motor encoders measurements is usually highly unreliable due to the accumulation of small errors over time, but in the present work we are only interested in difference between two consecutive discrete time steps. Moreover, for the purpose of image analysis, we assume that the robot motion consists only of translation and in-place rotation, but not both at the same time.

Assuming constant velocity between time indices $k-1$ and k, movement of a mobile robot can be modeled as follows:

$$\Delta x_k = D_k \cos\Theta_k \tag{1}$$

$$\Delta y_k = D_k \sin\Theta_k \tag{2}$$

$$\Delta\Theta_k = \omega_k T \tag{3}$$

$$D_k = v_k T \tag{4}$$

$$v_k = \frac{v_{L,k} + v_{R,k}}{2} = \frac{R \cdot \omega_{L,k} + R \cdot \omega_{R,k}}{2} \tag{5}$$

$$\omega_k = \frac{v_{R,k} - v_{L,k}}{b} = \frac{R \cdot \omega_{R,k} - R \cdot \omega_{L,k}}{b}, \quad (6)$$

where left and right wheel angular velocities at time index k are denoted by $\omega_{R,k}$ and $\omega_{L,k}$, respectively. Quantities Δx_k, Δy_k represent changes in coordinates of the center of mobile robots axis, while $\Delta\Theta_k$ is the angle between the vehicle axle and x-axis, and D_k is the traversed distance. Axle length is b while R is the wheel radius. Center of axle translational speed between time steps $k-1$ and k is v_{k-1}, and sampling period is denoted by T.

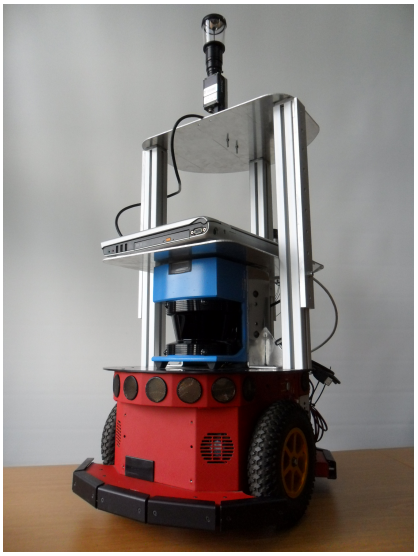

Fig. 1: Pioneer 3-DX equipped with an omnidirectional camera

III. Omnidirectional camera system

Our omnidirectional camera system is composed of a hyperbolic mirror and a standard camera. Such setup is called a catadioptric camera. One important property of catadioptric systems is the presence (or absence) of single effective viewpoint, a point in which all collected rays of light intersect. Systems that have a single effective viewpoint are called central projection cameras. This property is important because only central projection cameras can have completely sharp images. The mirror in our system satisfies these properties to an extent.

A. Image forming

In an omnidirectional camera, an image is formed by light passing through a system consisting of a mirror, lenses, and a digitization procedure. The following assumptions are commonly taken [2], [9]:

- The mirror is rotationally symmetric with respect to its axis. This symmetry is guaranteed by manufacturing.
- The mirror (lens) axis is perpendicular to the sensor plane.

An example of how the image is formed in an omidirectional camera is shown in Fig. 2. The reference coordinate system (X, Y, Z) is situated in the single effective viewpoint. Point

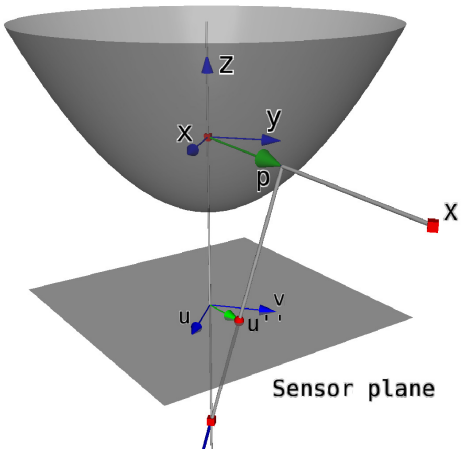

Fig. 2: Forming of an image in an omnidirectinal camera

$\mathbf{X}$ in the scene is mapped to the point $\mathbf{u}''$ in the sensor plane defined with coordinate system (u, v). In order to make scene analysis, we need to determine, for each point $\mathbf{u}''$, a vector $\mathbf{p}$ emanating from the coordinate system origin and pointing to the $\mathbf{X}$.

Firstly, we assume that the camera and mirror axes are perfectly aligned, we can see from Fig. 2 that real world coordinates (X, Y) of the point $\mathbf{X}$ in the scene, are proportional to the sensor plane coordinates (u, v) with some positive scalar α. We can now write $\mathbf{p} = [\alpha u, \alpha v, F(u, v)]$. It is important to notice that $\mathbf{p}$ is not a point, but a vector, which allows us to include scalar α into the $F(u, v)$ function, yielding:

$$\mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} u \\ v \\ F(u, v) \end{bmatrix}. \quad (7)$$

All points in the direction of this vector $\mathbf{p}$ will have same coordinates in the sensor plane.

Furthermore, the second assumption is that our mirror is rotationally symmetric, thus we can rewrite $F(u, v)$ as $F(\rho)$, where $\rho = \sqrt{u^2 + v^2}$. Function $F(\rho)$ is given in form of a polynomial of arbitrary order:

$$F(\rho) = a_0 + a_1\rho + a_2\rho^2 + a_3\rho^3 + a_4\rho^4 + ... \quad (8)$$

However, in [2] it is suggested that a polynomial of order four is best suited for most systems. Our primary interest is now in finding $F(\rho)$, which is achieved through camera calibration.

B. Camera calibration

Camera calibration is a necessary task for any vision system from which exact metric information is needed. The camera used in our experiments is a standard perspective Basler Scout camera. The mirror is hyperbolic in shape but the parameters of the hyperbola are unknown. In order to estimate camera's intrinsic and extrinsic parameters OCamCalib Toolbox proposed in [10] is used. Idea behind this method is to estimate the system parameters using a series of chessboard

TABLE I: Extrinsic and intrinsic system parameters

Extr. Param.	Value	Intr. Param.	Value
a_0	-195.889	x_{cen}	397.9
a_1	0	y_{cen}	513.5
a_2	$1.412 \cdot 10^{-3}$	c	0.99987
a_3	$-9.0145 \cdot 10^{-7}$	d	$-6.019 \cdot 10^{-6}$
a_4	$7.594 \cdot 10^{-10}$	e	$5.478 \cdot 10^{-6}$

pattern images with known dimensions. Corners of the patterns are extracted automatically and least squares minimization is used to estimate the function $F(\rho)$. Afterwards, using the symmetrical property of the mirror, projection center of the image (x_{cen}, y_{cen}) is found. To account for the the camera and axis misalignment and digitizing effects, a linear model is used. Full camera system is modeled as follows:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} x_{cen} \\ y_{cen} \end{bmatrix}. \tag{9}$$

Estimated parameters of the system are given in Table I.

IV. Moving object detection

In order to detect moving objects in the scene, a number of steps must be taken. The first step is to use the camera model (9) and robot motion estimates (1), (2) and (3) in order to estimate the possible optical flow vectors caused by motion and to segment out the vectors that are not considered to be parts of the static surrounding. Afterwards, it is necessary to classify segmented vectors to interpret the data and detect moving objects in the scene, in order to finally calculate measurement information which can be used for moving object tracking. The proposed algorithm is developed as a part of a multisensor fusion system for dynamic object tracking. Therefore, it should not be computationally demanding since it will be sharing CPU time with several other sensors. Some other approaches to motion segmentation based on structure-from-motion can be found in [11], [12].

A. Optical flow calculation

At a time index k we grab an image frame I_k and read odometry data. Image frame from the previous time index I_{k-1} is also kept in the memory. We need to extract features from I_{k-1} image which we will search for in I_k to form an optical flow field. Images are taken every $T_I = 0.2$ seconds. Shi and Tomasi [13] algorithm is used to extract the features, i.e. corners. Corners in the image I_{k-1} are denoted by C^i_{k-1}, where $i \in \{1, 2, ..., N_c\}$, with N_c being the number of corners searched in the image. Each corner C^i_k is given in terms of the coordinates in the image, $C^i_k = (u^i_k, v^{i'}_k)$.

The search for the corners in the current image C^i_k is done via pyramidal Lucas-Kanade feature tracker [14], which takes a list of corners C^i_{k-1} as input and tries to find those corners in the I_k image. The output of the algorithm is a list of estimated coordinates of corners in I_k image denoted by C^i_k.

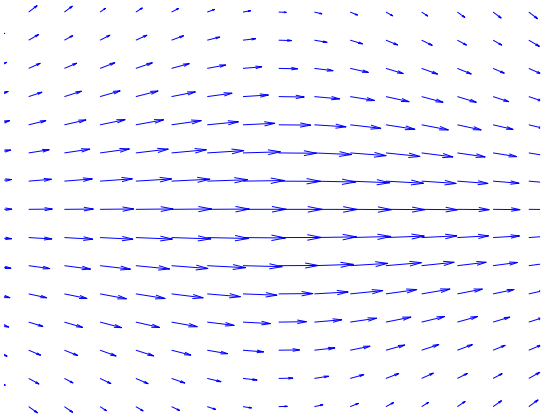

Fig. 3: An example of optical flow field caused by translation

B. Translation

Having tracked corners over two consecutive frames, we use the encoder data to estimate robot motion given by (1), (2) and (3). Total translation at time index k is given by $L_k = \sqrt{\Delta x_k^2 + \Delta y_k^2}$. This information is necessary because we are generating *synthetic optical flow*, calculated on the basis of robot motion estimates as opposed to measured optical flow which is provided by the optical flow algorithm. An example of simulated optical flow caused by translational motion form right to left in an omnidirectional image is shown in Fig. 3.

Aside from the robot motion, optical flow of a particular corner strongly depends on the real world coordinates of the object projected onto the image plane. The further away the point is from the mirror (robot coordinate system) the smaller the optical flow will be detected. As stated earlier, we have no depth information which makes our task somewhat more difficult. To circumvent this particular problem, we propose a novel approach. The idea is to iteratively back-project C^i_{k-1} onto $X - Y$ planes with different heights, which is necessary to incorporate motion estimates. Once we back-project C^i_{k-1} onto height Z^j_i, we can extract the other two real world coordinates of the point $\mathbf{X} = (X^j_i, Y^j_i, Z^j_i)$. At this moment, we can translate the point in the space by value of L_k, yielding new coordinates with respect to the mobile robot $\mathbf{X_n} = (X^j_i \pm L_k, Y^j_i, Z^j_i)$. Ambiguity of the sign of $L(k)$ can be solved by inspection of direction in which features moved. This requires to assume that there are more features that belong to stationary scenery than to moving objects, which we consider to be a sound assumption. Since only translation is performed, we are updating only the X^j_i coordinate in 3D space. By knowing all three translated real world coordinates, we can project the point back onto the image plane, thus giving us expected position of the corner in the current image $\hat{C}^i_k$.

Figure 4 illustrates the idea of projection onto different planes. By repeating this procedure for different heights, we are able to find the closest corner in pixels to the measured one. This enables us to evaluate if the optical flow vector belongs to a static object or to a moving one. For each corner we keep track of real world coordinates $\mathbf{X}$ at which best match

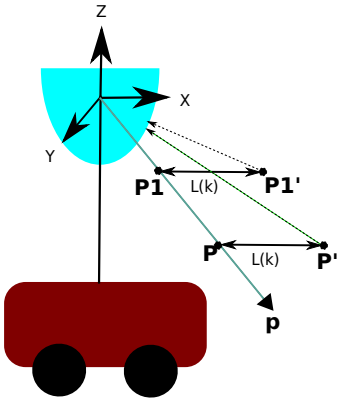

Fig. 4: Synthetic flow generation with odometry estimates

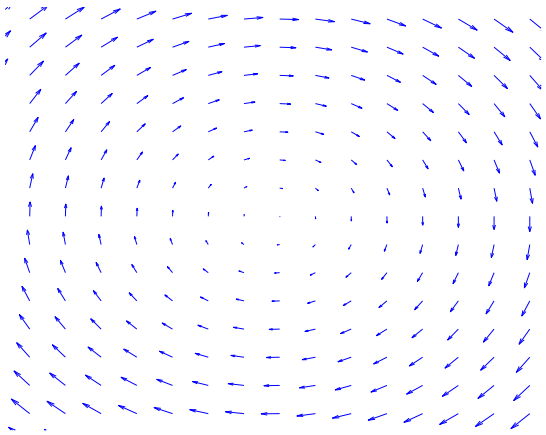

Fig. 5: Optical flow caused by in-place rotation

was found because we can use that information to estimate depth of the static object. The assumption that the mobile robot does not perform both translation and rotation at the same time is somewhat unrealistic. The fact is that robot will almost certainly have some unwanted rotation. As it turns out, this is very inconvenient, especially for the corners that have small optical flow vectors like the one lying on the line of motion. Therefore, it will be necessary to compensate for the rotation which can be done easily with no need for the camera model. Equations are given as follows:

$$r = \sqrt{(x - x_{cen})^2 + (y - y_{cen})^2} \tag{10}$$

$$\theta_{prev} = \arctan(\frac{y - y_{cen}}{x - x_{cen}}) \tag{11}$$

$$x_{new} = r\cos(\theta_{prev} + \Delta\Theta_k) + x_{cen} \tag{12}$$

$$y_{new} = r\sin(\theta_{prev} + \Delta\Theta_k) + y_{cen}, \tag{13}$$

where r is the distance of the pixel to the image center, θ is the angle of the pixel relative to (x_{cen}, y_{cen}), while (x_{new}, y_{new}) are the coordinates of the pixel compensated for rotation.

Each corner is rotated by the estimated angle from encoders data. In order to provide the needed robustness, we allow the ending point of the estimated optical flow vector $\hat{C}_k^i$ to be in some proximity of the measured C_k^i, i.e. look if the ending points of both vectors differ by more than some distance in pixels denoted by T_ϵ^{trans}. If a number of corners on restricted area shows significant similarities then they can be grouped together and marked as belonging to a moving object.

An immediate problem with this kind of detection is over-fitting. If we imagine a situation where the moving robot is overtaking a slightly slower robot with both robots moving in the same direction, then projecting vector on to different heights could yield a high consistency with measured optical flow vectors classifying them as stationary pixels with wrong height labels i.e. further away from our robot thus causing lesser optical flow. This kind of problem cannot be avoided without additional assumptions (or distance measurements). However it is unlikely that a moving object will mimic the movements of our robot for longer periods of time.

C. In-place rotation

An in-place rotation is a special case, which is fully covered in the previous subsection under rotation compensation, but it is important to treat it differently during computation. In our case, robot rotates only when changing its heading, but it is also crucial not to lose track of a moving object during that period as the change of mobile robots heading is quite often in real world situations. Since no information is gained about the robot translation, we cannot estimate height of the point in 3D space from optical flow vectors. It is important to take this into account in order to avoid false height labels. All we need to do is rotate the pixels in our image by estimated angle based on their distance from image center as given by (12) and (13). After the rotation, all points having Euclidean distance greater than T_{rot}^ϵ are considered to be outliers. Another distinct feature is that overfitting is no longer a concern which makes segmentation more reliable. An example of an optical flow field caused by an in-place counterclockwise motion is shown in Fig.5

D. Filtering and partitioning of segmented vectors

Once we have selected optical flow vectors that we consider to be caused by moving objects, we still need to make sense of that particular set. There still may be a lot of vectors that are segmented out due to poor optical flow calculation performance. For example, if we have a single outlier, we can safely assume that it is just calculation error. The reasoning behind this is intuitive since we are not tracking small objects, and are free to assume that objects sizes are comparable to those of the robot. Furthermore, vectors arising from single objects should have similarities (similar angle and module), and hence we remove vectors that have no similar neighbors. Due to the fact that our interest is in finding groups of similar vectors, a measure is adopted from [15]:

$$D_{sim}(\vec{v_1}, \vec{v_2}) = \frac{|\vec{v_1} - \vec{v_2}|}{\frac{|\vec{v_1}|+|\vec{v_2}|}{2} + \sigma}, \tag{14}$$

where σ is some arbitrary value to avoid division by zero problems. For each vector considered as an outlier, we search

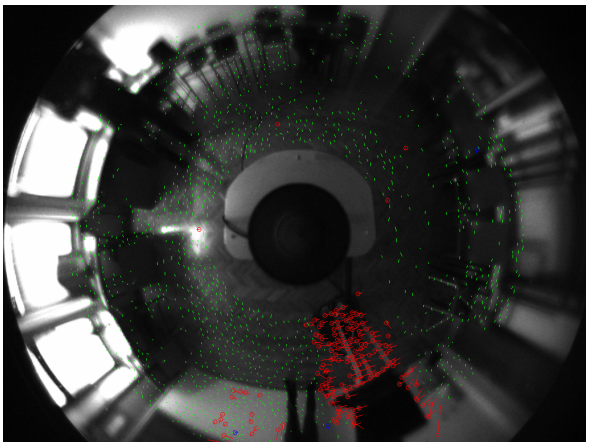

Fig. 6: Optical flow calculation and segmentation in the case of translation with natural lighting

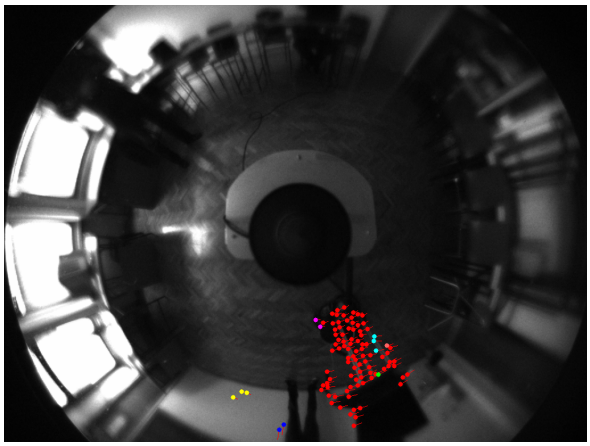

Fig. 7: Translation after filtration and partitioning

for another vector in the neighborhood of 30 pixels to make a *strong pair*. A strong pair are two nearby vectors whose similarity measure is less than 0.1. If we are not able to find another vector to make a strong pair, then we consider the first one to be a calculation error.

After the filtering stage, the vectors are partitioned in equivalence classes using disjoint set data structure and union find algorithm. To partition the vectors we need a predicate to tell us if the two vectors are certainly in the same group or if they may or may not be. Two vectors are in the same group if they form a strong pair. Afterwards, additional filtering is conducted. Examining the sets, all sets having number of vectors less or equal to T_n are considered insignificant. This threshold can be set to different value to match specific setups and camera models. In our case, we choose it to be three.

E. Measurement information

The partitioning result is a list of labels mapping a certain vector to a group. If a number of vectors is associated to a certain group, we can soundly assume that those vectors are indicating a moving object. From this information a bearing angle can be calculated. We propose to calculate the centroid of the group and use those coordinates as a measure of the bearing angle. For each group of vectors at time index k, its centroid coordinates are denoted by x_k^c and y_k^c and are calculated as follows:

$$x_k^c = \frac{1}{N_k}\sum_{i=0}^{N_k} x_i^k\,, \quad y_k^c = \frac{1}{N_k}\sum_{i=0}^{N_k} y_i^k, \tag{15}$$

where N_k is the number of vectors in the k-th group, x_i^k and y_i^k are ending points of the vector (output of Lucas-Kanade algorithm). The object bearing angle Θ_k , with respect to the robots coordinate frame, is expressed as

$$\Theta_k = \text{atan2}(y_k^c - y_{cen}, x_k^c - x_{cen}). \tag{16}$$

If needed, a convex hull can also be determined. Assuming that the tracked object is moving across the floor, it is possible estimate its distance from the robot. The idea is to look for the corner (ending point of optical flow vector) that is nearest to the image center (which also means that it is closest to the robot). By knowing robots height, real world coordinates can be calculated using our model (7). However, this is very uncertain since we are assuming that we have segmented the bottom of the moving object, which does not have to be necessarily true as it may happen that the bottom of the moving object is very similar to the ground and no corners were detected. The other obvious example is the case when the object is occluded by a static obstacle. Therefore, in the present work we are not estimating the range of the object.

V. Experimental results

The results of our experiments are presented in this section. The proposed algorithm showed good performance with both natural and artificial lighting in an indoor environment. It is beyond the scope of this paper to classify moving objects and distinguish between interesting objects and reflections or shadows that pose no threat of collision with mobile robot, but part of this issue has been addressed through filtering as shadows and reflections tend to have smaller spatial consistency. Figure 6 shows segmented vectors (red lines with circles at one end) and the ones we consider to fit in our model of static background (green lines). The algorithm was set to use $N_c = 2000$ corners.

As we can see in Fig. 6, there are a number of vectors that do not belong to the moving object (another Pioneer 3-DX mobile robot) – namely on the very light surface of the wall where optical flow algorithm performs badly due to shadows and sensitivity to changes in brightness. One vector is assigned to the reflection of the ground which is also a slow moving object in the image. There are also some vectors that are just evidence of incorrect optical flow calculation. But all these vectors show no spatial consistence and were discarded in the process of filtering.

Filtering and clustering results are depicted in Fig. 7. A large number of vectors that showed no spatial consistence were filtered out. There were a few which have their strong pairs and were not filtered out. However, they should be grouped

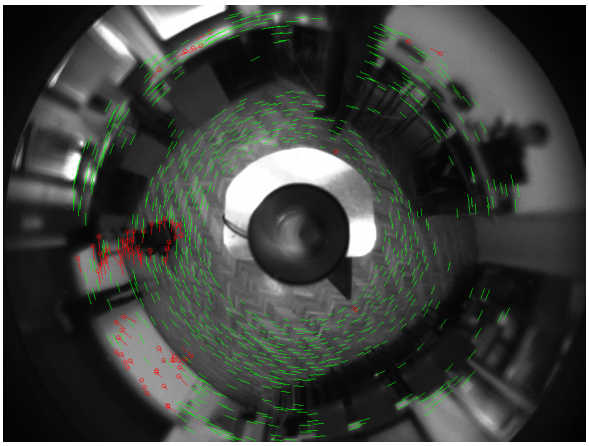

Fig. 8: Optical flow calculation and segmentation in case of rotation with artificial lighting

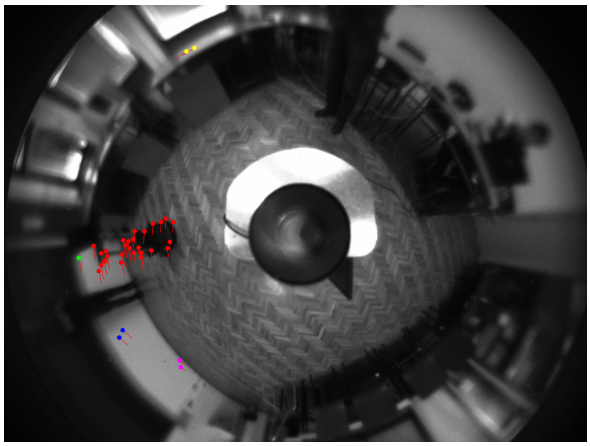

Fig. 9: Rotation after filtration and partitioning

together in small groups which is another step of filtering as the threshold is set to consider a group of pixels as a significant sign of a moving object. It can be clearly seen that the moving object had a number of vectors associated with it and was segmented correctly as a moving object. Each group of vectors were marked with different color.

Figure 8 shows a scene for the case of rotational motion and the results of the segmentation under artificial lightning are shown in Fig. 9. The algorithm was set to use $N_c = 1000$ corners.

Optical flow vectors can be seen clearly because rotation causes more pixel movement between two consecutive time indices. By analyzing Figs. 8 and 9 we can conclude that in the case of a rotational motion too, the algorithm successfully detects a moving object.

Experiments were performed on a 2.10 GHz laptop CPU with a C/C++ implementation. The average computational time of the complete algorithm was a bit less than 100 ms per frame on a 1024x768 pixel image.

VI. Conclusion

In this work we have analyzed an optical flow vector information from the pyramidal Lucas-Kanade algorithm applied to an omnidirectional image. Camera system model coupled with movement information from motor encoders was used to segment out vectors that do not belong to the static scene around mobile robot. We have proposed a novel approach to iteratively refine our match between measured and estimated optical flow vectors and to additionally filter vectors potentially indicating a moving object. The measurement of the sensor, as the bearing of the object, was calculated from the centroids of the grouped optical flow field vectors. The experiments showed that the algorithm is able to detect a moving object in an adverse scenario under natural and artificial lighting.

Acknowledgment

The authors would like to thank Hrvoje Šamija for fruitful discussions on the analyzed problematics.

References

[1] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro, "Image-based Monte Carlo localisation with omnidirectional images," *Robotics and Autonomous Systems*, vol. 48, pp. 17–30, 2004.

[2] D. Scaramuzza, "Omnidirectional Vision: From Calibration to Robot Motion Estimation," Ph.D. dissertation, ETH Zurich, 2008.

[3] D. Scaramuzza, F. Fraundorfer, and M. Pollefeys, "Closing the loop in appearance-guided omnidirectional visual odometry by using vocabulary trees," *Robotics and Autonomous Systems*, vol. 58, no. 6, pp. 820–827, 2010.

[4] M. Kobilarov, G. Sukhatme, J. Hyams, and P. Batavia, "People tracking and following with mobile robot using an omnidirectional camera and a laser," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, no. May, 2006, pp. 557–562.

[5] C. Plagemann, C. Stachniss, J. Hess, F. Endres, and N. Franklin, "A nonparametric learning approach to range sensing from omnidirectional vision," *Robotics and Autonomous Systems*, vol. 58, no. 6, pp. 762–772, 2010.

[6] W. Enkelmann, "Obstacle Detection by Evaluation of Optical Flow Field from Image Sequences," *Image and Vision Computing*, vol. 9, pp. 160–168, 1991.

[7] A. Talukder, S. Goldberg, L. Matthies, and A. Ansar, "Real-time detection of moving objects in a dynamic scene from moving robotic vehicles," in *Proceedings of the International Conference on Intelligent Robots and Systems 2003*, no. 818, 2003, pp. 27–31.

[8] H. Šamija, I. Marković, and I. Petrović, "Optical Flow Field Segmentation in an Omnidirectional Camera Image Based on Known Camera Motion," in *International Convention on Information and Communication Technology, Electronics and Microelectronics*, 2011.

[9] B. Mičušík, "Two-view geometry of Omnidirectional Cameras," Ph.D. dissertation, Czech Technical University, 2004.

[10] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for Easily Calibrating Omnidirectional Cameras," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 5695–5701.

[11] K. Ozden, K. Schindler, and L. Van Gool, "Multibody Structure-from-Motion in Practice," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1134–1141, 2010.

[12] A. Kundu, K. Madhava Krishna, and C. Jawahar, "Realtime Motion Segmentation based Multibody Visual SLAM," in *Seventh Indian Conference on Computer Vision, Graphics and Image Processing*, 2010.

[13] J. Shi and C. Tomasi, "Good Features to Track," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 593–600, Jun. 1994.

[14] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," *Intel Corporation Microprocessor Research Labs*, 2000.

[15] S. Smith, "Asset-2: Real-time motion segmentation and object tracking," Oxford Centre for Functional Magnetic Resonance Imaging of the Brain, Tech. Rep., 1994.