Motion Detection in the Presence of Egomotion Using the Fourier-Mellin Transform

Santosh Thoduka(✉), Frederik Hegger, Gerhard K. Kraetzschmar, and Paul G. Plöger

Department of Computer Science, Bonn-Rhein-Sieg University of Applied Sciences, Grantham-Allee 20, 53757 Sankt Augustin, Germany
{santosh.thoduka,frederik.hegger,gerhardk.kraetzschmar, paulg.ploger}@h-brs.de

Abstract. Vision-based motion detection, an important skill for an autonomous mobile robot operating in dynamic environments, is particularly challenging when the robot's camera is in motion. In this paper, we use a Fourier-Mellin transform-based image registration method to compensate for camera motion before applying temporal differencing for motion detection. The approach is evaluated online as well as offline on a set of sequences recorded with a Care-O-bot 3, and compared with a feature-based method for image registration. In comparison to the feature-based method, our method performs better both in terms of robustness of the registration and the false discovery rate.

Keywords: Motion detection · Mobile robots
Egomotion compensation · Fourier-Mellin transform

1 Introduction

Autonomous mobile robots are expected to operate in dynamic environments with other agents such as humans, robots, pets etc. This is especially true for service robots that work in a home environment, such as in RoboCup@Home, and for outdoor robots that need to navigate through traffic and pedestrians. Moving objects in such environments can add to the complexity of certain tasks such as navigation, scene understanding and action monitoring; however, motion can also be an important cue for interpreting the environment and understanding the effects of actions.

While motion can be detected by infrared sensors, radars etc., vision-based motion detection is a practical and cheap solution for robots, and is a well-studied field, especially in the context of video surveillance systems. For a stationary camera, the methods used in video surveillance systems can be directly applied and used for detecting motion. However, robots often need to detect motion while they are in motion (egomotion); in such cases methods that rely on a static

H. Akiyama et al. (Eds.): RoboCup 2017, LNAI 11175, pp. 252–264, 2018.
https://doi.org/10.1007/978-3-030-00308-1_21

camera cannot be used directly because from the point of view of the robot, the entire image of the scene appears to be moving as the robot is moving.

Consider the two images in Fig. 1, which includes forward motion of the camera and a falling toy. As humans, our attention is naturally drawn to the toy even when we are moving, since it provides us with a visually salient stimulus [1, p. 41]. However, from a 2D image perspective, the entire scene has changed; the red arrows indicate the direction of apparent motion in the image due to camera motion and the green arrow indicates the direction of motion of the toy. The apparent motion of the scene follows a pattern, while the motion of the toy can be seen as an anomaly in this pattern (i.e. it is contrary to the expected egomotion-induced optical flow), and hence an indicator for the robot that some action is required. Detecting this motion is a challenging task for a robot as it involves being able to distinguish between global changes in the scene caused to its own motion and local changes due to the movement of other objects.

(a) Frame 1 (b) Frame 2

Fig. 1. Camera motion and independent motion (yellow circle) between frames. (Color figure online)

This paper tackles the problem of using 2D vision methods for detecting independent motions in the environment when the observer (a robot) *is also moving* through the environment. We approach this problem by combining existing methods: the Fourier-Mellin transform (FMT) [2] for compensating camera motion and temporal differencing for motion detection. The algorithm is able to run close to real-time on a robot.

The paper is structured as follows: Sect. 2 discusses related work, Sect. 3 explains the proposed approach, Sect. 4 discusses the results, along with a comparison to a feature-based method, and finally conclusions and future work are discussed in Sect. 5.

2 Related Work

The majority of the literature regarding motion detection considers scenes from a static camera, with some approaches allowing for slight camera motions; however,

there has been an increase in vision-related work with camera motion due to the importance of autonomous driving and driver assistance systems [3].

For static cameras, methods such as background subtraction, temporal differencing and optical flow are well known for motion detection. However, none of these methods, used in isolation, are applicable for moving cameras since the majority of the changes in the scene would be caused by camera motion. Some methods [4–6] compensate for camera motion before applying differencing or optical flow to detect motion. Camera motion is typically compensated by first calculating optical flow or tracking features between frames, followed by selection of inlier vectors that best describe the dominant motion. The transformation estimated using the inlier vectors is then used to align the frames, minimizing the differences between the two frames due to camera motion.

Badino et al. [6] estimate 6D egomotion using optical flow calculated from stereo cameras. The computed egomotion is used to compensate the flow vectors such that only moving objects have large vectors. Kim et al. [7] use the Lukas-Kanade tracker both for camera motion compensation and for continuously updating a background model, and subsequently use background subtraction for motion detection. Meier et al. [8] use inertial-optical flow for egomotion and object motion estimation. Tracked features are classified as outliers and inliers based on whether they agree with the global motion seen in the image. The authors use the outliers specifically to detect and characterize the motion of independently moving objects. Kumar et al. [9] use machine learning to predict optical flow statistics (mean and covariance) given the head and eye velocities of the iCub robot. Moving objects are detected by comparing their flow vectors to the learned statistics.

Convolutional Neural Networks (CNNs) have also been used for compensation egomotion: Tokmakov et al. [10] train a network that uses ground truth optical flow to detect independently moving objects from a moving camera. Rezazadegan et al. [11] perform action recognition on moving cameras by first centering the person of interest. Agrawal et al. [12] compute the transformation between sequential images due to camera motion by using a Top-CNN which receives input from two identical Base-CNNs (one for each image).

A common theme in the related work is the use of optical flow or feature tracking methods, which rely on detecting good features. Apart from the work by Kumar et al. [9], there are no published results of extensive evaluation in different scenes or with varying parameters. Additionally, the hardware, frame rate, image size are often not reported, making it difficult to determine whether the methods can be run on a mobile robot.

Adjacent fields, such as visual odometry and structure-from-motion, essentially try to solve a similar problem: that of estimating camera motion. Most approaches in these fields also rely on detecting and tracking good features, but other methods have also been explored. For example, FMT has been used for visual odometry [13,14] and found to be equally good or better than optical flow methods. The advantage of Fourier-based methods is that they are not dependent on finding good features and are robust to some lighting changes and

frequency-dependent noise. The FMT approach calculates the similarity transform, which is a simplification of the affine and perspective transforms, which also makes it a better candidate for running on a robot with limited resources. We assume, as in other related work, that moving objects constitute less than half the image; i.e. the apparent motion of the scene due to camera motion is the dominant motion.

3 Approach

Since it has already been successfully used in visual odometry, we decided to use FMT to compensate camera motion. Additionally, in order to avoid calculation of features and optical flow, we then use temporal differencing for detecting independent motions. Our approach is based on a vision pipeline that has two stages: (1) FMT is used to compute the transform between consecutive frames, which is then used to align them, hence compensating camera motion. (2) Temporal differencing is performed on the aligned frames to detect moving objects.

I. Image registration using FMT: Image registration is the process of transforming an image to geometrically align it with a reference image. FMT was first used for image registration by Reddy et al. [2]. It is an extension of the phase correlation method and can simultaneously retrieve the rotation, scale and translation between images; i.e. it estimates the similarity transform for a given pair of images. The steps of the method can be seen in Fig. 2.

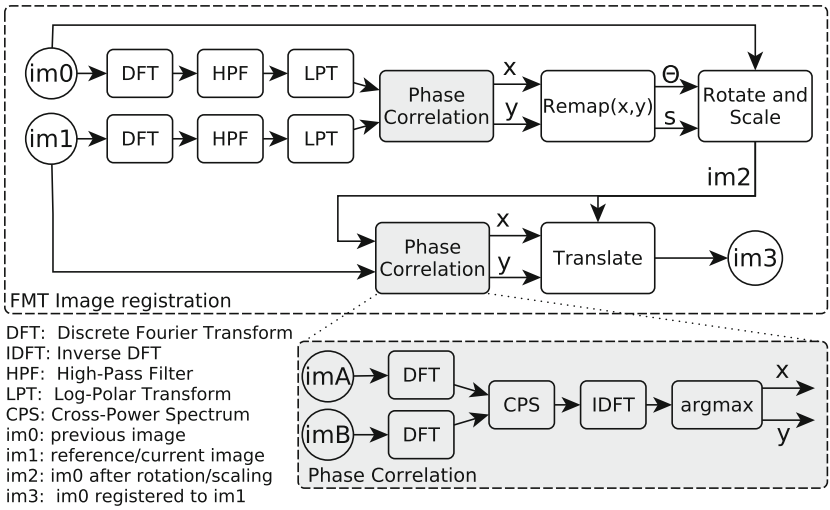


Fig. 2. Top: FMT image registration pipeline. Bottom: Phase correlation block which is used twice in the registration pipeline.

FMT-based image registration itself is a two-step process; the first step estimates the rotation and scale and the second estimates the translation. In both steps, phase correlation is used to find the translation between two images. In the first step, the input images to the phase correlation step are the log-polar transforms of the discrete Fourier transforms of the grayscale images ($im0$ and $im1$) to be registered. The estimated shift in the log-polar images (x, y) is converted into rotation θ and scale s and used to transform $im0$.

This transformed image $im2$ and the reference image $im1$ are the inputs to the second phase correlation step. The estimated translation from the second phase correlation step is used to transform $im2$ again. This resultant image $im3$ is now registered with the reference image $im1$.

Fourier-Mellin Transform: The FMT of a function $f(r, \theta)$ is given by [13]:

$$M_f(\boldsymbol{u}, v) = \frac{1}{2\pi} \int_0^{\infty} \int_0^{2\pi} f(r, \theta) \boldsymbol{r^{-ju}} e^{-jv\theta} d\theta \frac{\boldsymbol{dr}}{\boldsymbol{r}} \tag{1}$$

where the elements in bold are the Mellin transform parameters and the remaining ones are the Fourier transform parameters.

By substituting $r = e^{\rho}$, the FMT can be expressed as just a Fourier transformation [13]:

$$M_f(u, v) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_0^{2\pi} f(e^{\rho}, \theta) e^{-ju\rho} e^{-jv\theta} d\theta d\rho \tag{2}$$

Log-Polar Transform: In practice, the variable substitution is realized using the log-polar transform. The log-polar transform is performed by remapping points from the 2D Cartesian coordinate system (x, y) to the 2D log-polar coordinate system (ρ, θ) [15]:

$$\begin{aligned} \rho &= \log(\sqrt{(x - x_c)^2 + (y - y_c)^2}) \\ \theta &= \text{atan2}(y - y_c, x - x_c) \end{aligned} \tag{3}$$

where ρ is the logarithm of the distance of a given point, (x, y), from the centre, (x_c, y_c), and θ is the angle of the line through the point and the centre. This transform converts rotation and scaling in the Cartesian coordinate system to translations in the log-polar coordinate system.

Phase Correlation: Phase correlation, introduced by Kuglin et al. [16], is a global method that can retrieve the translation between two images. The method is based on the Fourier shift theorem, which states that shifting a signal by τ in the time/space domain multiples the Fourier transform by $e^{-j\omega\tau}$. The phase difference can be calculated by using the normalized Cross Power Spectrum (CPS) in Eq. 4 [2],

$$\begin{aligned} f_2(x, y) &= f_1(x - t_x, y - t_y) \\ F_2(\xi, \eta) &= e^{-j2\pi(\xi t_x + \eta t_y)} F_1(\xi, \eta) \\ CPS = e^{-j2\pi(\xi t_x + \eta t_y)} &= \frac{F_1(\xi, \eta) F_2^*(\xi, \eta)}{|F_1(\xi, \eta) F_2(\xi, \eta)|} \end{aligned} \tag{4}$$

where F_1 and F_2 are the Fourier transforms of f_1 and f_2, and ξ and η are the spatial frequencies in x and y.

The term $e^{-j2\pi(\xi t_x+\eta t_y)}$ is equivalent to the Fourier transform of a shifted Dirac delta function; hence, if we take the inverse Fourier transform of CPS, the result is a signal with a peak at (t_x, t_y). By finding the location of the peak we retrieve the translation between the two images.

Rotation Ambiguity: Since the Fourier magnitude plots are conjugate symmetric, there is an ambiguity in the recovered rotation. If the calculated rotation angle is θ, the actual rotation of the image could be θ or $\theta + \pi$. For this application, we assume that consecutive frames are never rotated by more than π radians and hence do not perform an extra step to resolve the ambiguity.

High-pass Filtering: During the phase-correlation step, apart from the peak at the actual rotation angle, there are additional peaks at multiples of 90°. Sometimes the peak at 0° is higher than the peak at the required rotation angle. Both [17] and [2] suggest applying a high-pass filter in the Fourier domain to prevent the false peak at 0°.

II. Motion Detection Using Temporal Differencing: We use temporal differencing for motion detection, and additional operations, such as thresholding, edge masking and clustering of contours, are used to eventually output a set of bounding boxes representing independently moving objects in the scene (see Fig. 3). Temporal differencing is performed by taking the absolute difference of the pixel intensities between the registered frame $im3$ and the current frame $im1$. A binary threshold is applied on the difference image with intensities above the threshold being set to 255 (white) and those below being set to 0 (black), hence selecting pixels where a large difference is seen.

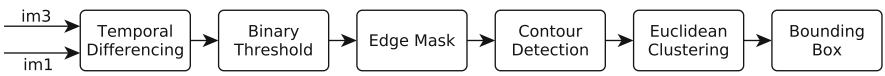
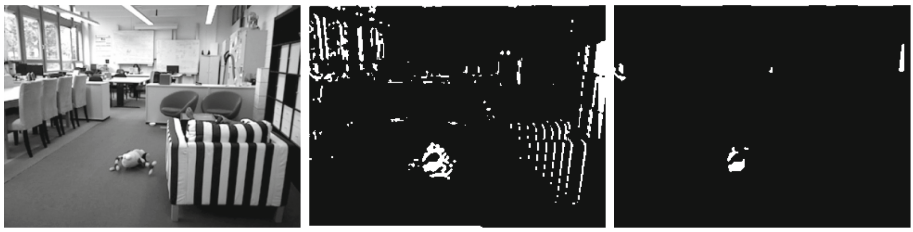


Fig. 3. Motion detection pipeline

Edge Mask: Edges are regions in the image where there are discontinuities in the pixel intensities. If the images are imprecisely registered, the edges might not overlap exactly with each other. This will result in large values in the difference image and will likely be present in the thresholded image (such as the stripes of the sofa in Fig. 4b). In order to remove the edges (false detections), we construct an edge mask from the thresholded image by first applying Canny edge detection to the thresholded image and fitting contours to them. Oriented rectangles are then fit to the contours and are classified as edges if the aspect ratio is very high or very low. The rectangles which are classified as edges are then masked out as seen in Fig. 4c. It is also worth noting that this process causes some degradation in the detection of the moving stuffed toy as well; this is discussed further in the evaluation.



(a) Scene (b) Before mask (c) After mask

Fig. 4. Edge mask on a scene with egomotion and independent motion

Clustering: In order to separate the resultant threshold image into a set of regions, we cluster the white pixels based on their distance from each other. This is done by first finding contours, and then applying Euclidean clustering on the contour points.

Small clusters are discarded and bounding boxes are fit to each cluster. Some bounding boxes are ignored if the ratio of white to black pixels is low. The intermediate outputs from the motion detection pipeline can be seen in Fig. 5.

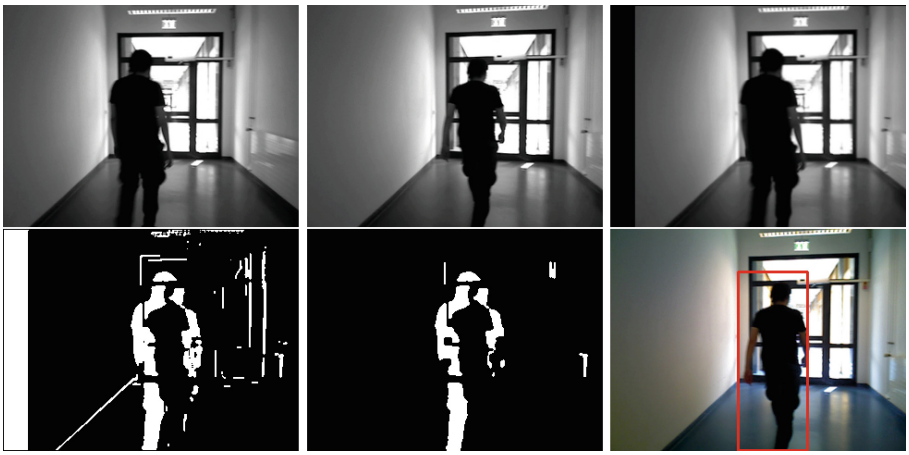


Fig. 5. The main steps of the motion detection pipeline: **Top:** previous, current and registered frame. **Bottom:** thresholded image, edge-masked image, bounding box of clustered contour points

Runtime: An open-source Python implementation of the FMT image registration method[1] was found to be too slow for our application (1.6 Hz). Our C++

[1] https://github.com/matejak/imreg_dft.

port[2] is able to process 320×240 images at 14.5 Hz on an Intel Core i3, 1.7 GHz processor, while the overall motion detection pipeline runs at 11 Hz.

4 Evaluation

For evaluation, we collected fifteen image sequences and annotated them with ground truth (GT) bounding boxes of moving objects. A Care-O-bot-3 with a head-mounted ASUS Xtion Pro 3D camera was used for recording the sequences. All sequences involve robot egomotion (linear: 0–0.3 m/s, angular: 0–0.3 rad/s) and the set of moving objects in the scene includes humans, doors and a stuffed toy. The experiments are run on the recorded sequences, but the algorithm is able to run on the robot as well with a ROS (Robot Operating System) wrapper.

In [18], the authors discuss evaluation methods and metrics for motion detection algorithms, suggesting the following for object-based metrics:

- True Positive (TP): "A detected foreground blob which overlaps a GT bounding box, where the area of overlap is greater than a proportion Ω_b of the blob area and greater than a proportion Ω_g of the GT box area."
- False Negative (FN): "A GT bounding box not overlapped by any detected object."
- False Positive (FP): "A detected foreground object which does not overlap a GT bounding box."

We use these definitions and choose $\Omega_b = \Omega_g = 0.5$. We regard a single GT object that is overlapped by multiple detected bounding boxes as a true positive (given that the total overlap proportion criteria holds). Since the definitions have been modified, $TP + FN$ does not equal the number of GT objects (as is usually the case). A TP in this case is an object that has been reliably detected, whereas a FN is an object that has not been detected at all. This gives us two ways of interpreting the results: one which considers objects reliably detected (true positive rate, TPR), and one which considers objects not detected at all (false negative rate, FNR). These two metrics and the false discovery rate, FDR, are defined as:

$$TPR = \frac{N_{TP}}{N_{gt}}, FNR = \frac{N_{FN}}{N_{gt}}, FDR = \frac{N_{FP}}{N_d} \tag{5}$$

where N_{TP}, N_{FN}, N_{FP}, N_{gt} and N_d are the number of true positives, false negatives, false positives, ground truth objects and detected objects.

4.1 Experiments

Six sequences are chosen for the first two experiments since they cover both translational and rotational robot motion, and different types of object motions.

[2] https://github.com/sthoduka/fmt_motion_detection.

Experiment 1: Frame Rate. For this experiment, the frame rate of the camera is kept constant at 30 Hz, but, by skipping frames, the effective frame rate of the algorithm is altered. This results in larger differences between frames (due to motion) at slower frame rates without additional effects like motion blur. As seen in Fig. 6, the frame rate has a large impact on the performance, most significantly when halving the frame rate to 15 Hz. Although in most cases 6 Hz results in the best TPR, the FDR also rises significantly at this frame rate. The object-level annotation is also a cause of the poor performance at higher frame rates: at 30 Hz, the small motions of parts of objects are detected instead of the motion of the entire object, which is observable at lower frame rates. Overall, this result suggests that the frame rate of the algorithm should be dynamically altered based on the speed of the robot and expected speed of the objects.

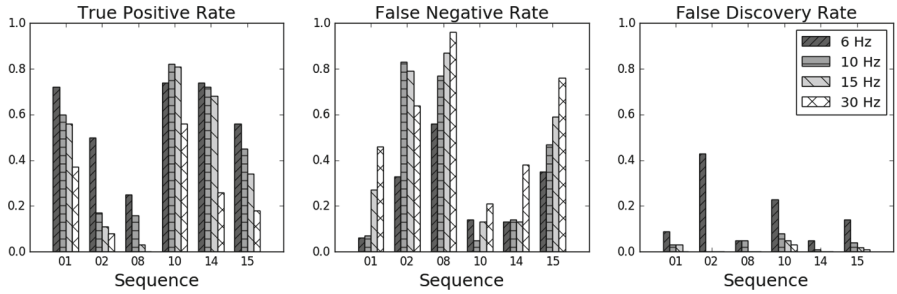


Fig. 6. Impact of altering time between consecutive frames

Experiment 2: Edge Mask. In general, applying the edge mask to the thresholded image reduces both the TPR and FDR, as seen in Fig. 7. Depending on the application, reducing the false detection rate to nearly zero at the expense of a decrease in true positive rate can be an acceptable compromise.

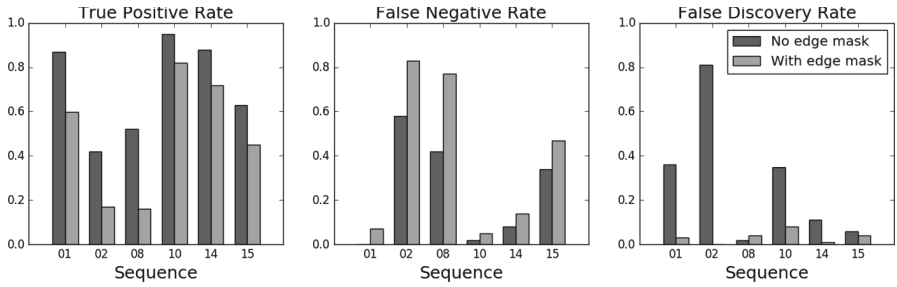


Fig. 7. Impact of applying an edge mask on the difference image

Comparison with a Feature-Based Approach. Here, we compare the results of the FMT method to a feature-based method. The monocular camera code from LIBVISO2 [19] is used for feature extraction and matching and the inlier vectors are used to estimate an affine transformation between consecutive frames. This affine transform is used to register the frames, after which the temporal differencing pipeline is used for motion detection. For both methods, we use the same parameters for temporal differencing, use frames of size 320×240, and skip every two frames (10 Hz effective frame rate). The experiments are run on a PC with an Intel Core i3, 1.7 GHz processor and 4 GB of RAM. The results of the comparison are seen in Table 1[3].

Table 1. Detection rates for the FMT and feature-based methods

Sequence	FMT			Feature-based		
	TPR	FNR	FDR	TPR	FNR	FDR
01	0.6	0.07	**0.03**	0.47	0.1	**0.24**
02	**0.17**	**0.83**	0.0	**0.25**	**0.75**	0.0
03	0.38	0.59	0.07	0.31	0.5	0.0
04	0.38	0.37	0.05	0.33	0.45	0.02
05	0.3	0.29	0.09	0.23	0.46	0.1
06	0.5	0.39	**0.0**	0.48	0.41	**0.11**
07	0.59	0.29	**0.09**	0.62	0.28	**0.18**
08	**0.16**	**0.77**	0.04	**0.06**	**0.8**	**0.15**
09	0.29	0.52	0.0	0.3	0.5	0.03
10	0.82	0.05	**0.08**	0.79	0.07	**0.19**
11	0.46	0.12	**0.04**	0.38	0.12	**0.14**
12	0.48	0.52	0.0	0.45	0.48	0.0
13	**0.03**	**0.95**	0.0	**0.03**	**0.95**	0.0
14	0.72	0.14	0.01	0.74	0.13	0.04
15	0.45	0.47	0.04	0.41	0.52	0.05

The TPR and FNR are comparable for the two methods; the sequences where the motions are far away or short and quick tend to have low TPR and high FNR in both cases. The reason for the higher FDR for the feature-based method is evident from Fig. 8, which shows the x and y translations for sequence 10. The sequence consists of the robot rotating to the left at a constant speed (0.3 rad/s), with some people moving in the scene. The FMT method shows a relatively constant translation in x and no translation in y, whereas the feature-based method is a lot noisier, with some large spikes; these spikes most likely account for the increased FDR. Similar behaviour was seen in the other sequences as

[3] All output videos can be found here: https://www.youtube.com/playlist?list=PL1rZfrn4gV_jc-Y3FdsEujE6RfWYi20gy.

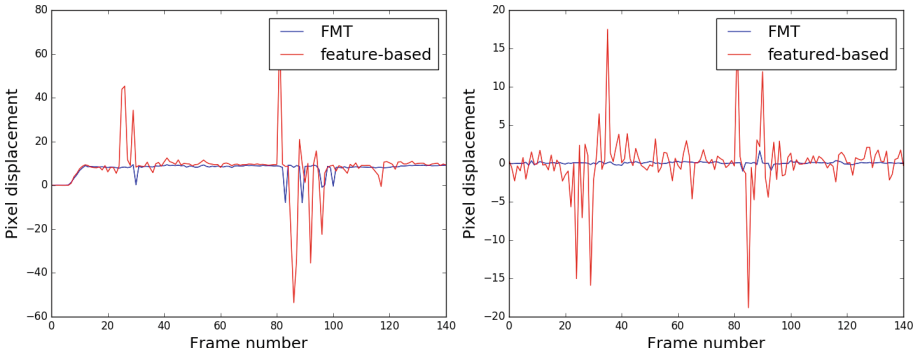


Fig. 8. Pixel translations for sequence 10 along the x-axis (left) and y-axis (right) for the FMT and feature-based methods.

well. The FMT method is able to run at about 11 Hz, while the feature-based method runs at 12 Hz, with comparable CPU and memory usage. This suggests that the FMT method is more robust while using the same amount of resources.

5 Conclusions and Future Work

In this paper, we combined existing approaches for independent motion detection from a moving camera. The Fourier-Mellin-based image registration method was used for egomotion compensation and temporal differencing for motion detection. Unlike other methods, FMT does not rely on the detection of good features, which is one of its advantages. For the set of sequences evaluated, a frame rate of 10–15 Hz was found to be ideal for the detection rate; at 30 Hz, motions between frames are sometimes too small to be detected. The algorithm processes frames at 11 Hz, which allows it to be run close to real-time on a robot within the range of the ideal frame rate. In comparison to a feature-based method, it performs better in terms of robustness of the registration and the false discovery rate. A more systematic evaluation is required to determine the limits on object and camera motion speed, depth variance of the scene and depth of the moving objects. Dynamically varying the frame rate based on the speed of the robot and applying the motion detection output to a task of the robot, such as safe navigation or turning towards a waving person, is future work.

Acknowledgements. We gratefully acknowledge the continued support by the b-it Bonn-Aachen International Center for Information Technology and the Bonn-Rhein-Sieg University of Applied Sciences. We also thank Alex, Argentina and Deebul for proof-reading the paper.

References

1. Frintrop, S.: VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search. LNCS (LNAI), vol. 3899. Springer, Heidelberg (2006). https://doi.org/10.1007/11682110
2. Reddy, B.S., Chatterji, B.N.: An FFT-based technique for translation, rotation, and scale-invariant image registration. IEEE Trans. Image Process. **5**(8), 1266–1271 (1996)
3. Gehrig, S., Franke, U.: Stereovision for ADAS. In: Winner, H., Hakuli, S., Lotz, F., Singer, C. (eds.) Handbook of Driver Assistance Systems, pp. 495–524. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-12352-3_22
4. Kim, J., Ye, G., Kim, D.: Moving object detection under free-moving camera. In: 2010 IEEE International Conference on Image Processing, pp. 4669–4672. IEEE (2010)
5. Hariyono, J., Hoang, V.-D., Jo, K.-H.: Moving object localization using optical flow for pedestrian detection from a moving vehicle. Sci. World J. **2014** (2014)
6. Badino, H., Franke, U., Rabe, C., Gehrig, S.: Stereo vision-based detection of moving objects under strong camera motion. In: Proceedings of the First International Conference on Computer Vision Theory and Applications, pp. 253–260 (2006)
7. Kim, S.W., Yun, K., Yi, K.M., Kim, S.J., Choi, J.Y.: Detection of moving objects with a moving camera using non-panoramic background model. Mach. Vis. Appl. **24**(5), 1015–1028 (2013)
8. Meier, D., Brockers, R., Matthies, L., Siegwart, R., Weiss, S.: Detection and characterization of moving objects with aerial vehicles using inertial-optical flow. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2473–2480. IEEE (2015)
9. Kumar, S., Odone, F., Noceti, N., Natale, L.: Object segmentation using independent motion detection. In: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pp. 94–100. IEEE (2015)
10. Tokmakov, P., Alahari, K., Schmid, C.: Learning Motion Patterns in Videos. CoRR, vol. abs/1612.07217 (2016)
11. Rezazadegan, F., Shirazi, S., Upcroft, B., Milford, M.: Action Recognition: From Static Datasets to Moving Robots. CoRR, vol. abs/1701.04925 (2017)
12. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 37–45 (2015)
13. Kazik, T., Göktoğan, A.H.: Visual odometry based on the Fourier-Mellin transform for a rover using a monocular ground-facing camera. In: 2011 IEEE International Conference on Mechatronics (ICM), pp. 469–474. IEEE (2011)
14. Goecke, R., Asthana, A., Pettersson, N., Petersson, L.: Visual vehicle egomotion estimation using the Fourier-Mellin transform. In: 2007 IEEE Intelligent Vehicles Symposium, pp. 450–455. IEEE (2007)
15. Sarvaiya, J.N., Patnaik, S., Bombaywala, S.: Image registration using log-polar transform and phase correlation. In: TENCON 2009-2009 IEEE Region 10 Conference, pp. 1–5. IEEE (2009)
16. Kuglin, C.D., Hines, D.C.: The phase correlation image alignment method. In: Proceeding of IEEE International Conference on Cybernetics and Society, pp. 163–165 (1975)
17. Stone, H.S., Tao, B., McGuire, M.: Analysis of image registration noise due to rotationally dependent aliasing. J. Vis. Commun. Image Represent. **14**(2), 114–135 (2003)

18. Lazarevic-McManus, N., Renno, J., Makris, D., Jones, G.: Designing evaluation methodologies: the case of motion detection. In: Proceedings of 9th IEEE International Workshop on PETS, pp. 23–30 (2006)
19. Geiger, A., Ziegler, J., Stiller, C.: StereoScan: dense 3D reconstruction in real-time. In: Intelligent Vehicles Symposium (IV) (2011)