Detection of Independently Moving Objects in Passive Video

Carlos Morimoto **Daniel DeMenthon** **Larry Davis** **Rama Chellappa**
Computer Vision Laboratory, Center for Automation Research
University of Maryland, College Park, MD 20742

Randal Nelson
Department of Computer Science
University of Rochester, Rochester, NY 14627

Abstract

We present two different approaches for the identification of independently moving objects (IMOs) and demonstrate their application to outdoor imagery taken from a moving autonomous vehicle. Both approaches involve image stabilization followed by an analysis of the stabilized image sequence. The stabilization reduces the effects of the movement of the autonomous vehicle, facilitating the detection of the IMOs. In the first approach, IMOs are detected based on a filtering approach that integrates the results of velocity tuned filters over several frames. In the second approach IMOs are identified by constraints on allowable values of the optic flow field after stabilization.

1 Introduction

Autonomous navigation tasks such as collision avoidance and maneuvering among vehicles can be addressed by a system that can detect independently moving objects from a moving platform, identify them as to object type (e.g., car, truck, pedestrian, animal) and estimate their motion relative to the motion of the vehicle.

One simple method for detecting IMOs from stationary cameras is to take the difference between image frames taken a short time apart and select the non-zero regions of the resulting image [1]. Assuming that the scene background does not change, the non-zero regions must correspond to moving objects. The problem becomes much harder when the camera is not stationary because the whole image is then subject to apparent motion.

The goal of stabilization is to eliminate from the image sequence all movements due to the camera itself (egomotion), so that the resulting image sequence would look like one taken from a stationary camera. While simple mechanical stabilization only removes the high frequency movements, our electronic stabilizer can remove the effects of a larger class of motions at a lower cost.

Image stabilization has been used for several other purposes including video compression [6] and recovery of egomotion [4]. A large variety of schemes have been described. For example, Burt et al. [2] use a 2D affine model and a hierarchical image registration algorithm which was implemented in special parallel hardware to perform stabilization in real-time, while Davis et al. [3] propose (among other methods) 3D models and the computation of flow to eliminate 3D rotation. Yao et al. [9] use multiple visual cues and an extended Kalman filter for the estimation of the desired 3D motion parameters that are used for stabilization. In order to obtain real-time performance, we adopt a 2D camera model, and implement an efficient and robust affine registration algorithm in a MV 200 Datacube board, a parallel pipeline image processing hardware, connected to a Sun SPARCStation 20/612, that serves as host for the Datacube and also runs part of the stabilization code.

After a reasonably stable sequence has been obtained, small discrepancies can be eliminated by means of a filtering scheme to make the detection of IMOs more robust and reliable. Our first approach to detect IMOs is based on *velocity tuned filters* (VTFs) that receive such sequences as input. As an alternative, our second approach uses a qualitative method for detecting independently moving

[0] The support of ARPA under contract DAAH-0493G049 and of Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) is gratefully acknowledged

objects based on available knowledge about the motion of the camera, which can be obtained from the computation of normal flow [8]. This approach requires a weaker form of stabilization in which only rotation effects need to be removed from the sequence. IMOs are identified by constraints on allowable values of the optical flow field for the resultant translation-only image sequences.

The following section describes our image stabilization system that is used for both approaches to the detection of IMOs. The first approach, based on VTFs, is presented in section 3 and the second, based on qualitative knowledge of the camera motion, is presented in section 4.

2 Image Stabilization

The main component of our image stabilization system is a frame to frame registration algorithm similar to the one developed by Zheng and Chellappa [10], modified in order to obtain near real-time performance.

Taking the first frame of the image sequence as reference, the stabilization process consists of applying the registration algorithm to the incoming frames, computing the camera motion parameters from the current frame f_i to the reference frame and then transforming f_i using the estimated parameters (warping the images) back to the reference frame. The reference frame can be automatically changed, based on user specified limits on the motion parameters.

The registration algorithm is based on a 2D model that involves the estimation of four motion parameters (two for translation plus rotation and scaling). Under this model, the transformation between two image frames is given by:

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = s \begin{pmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} + \begin{pmatrix} \Delta X_2 \\ \Delta Y_2 \end{pmatrix}$$

where (X_i, Y_i) are the image frame coordinates at time t_i , for $i = \{1, 2\}$, $(\Delta X_2, \Delta Y_2)$ is the translation measured in the image coordinate system of frame t_2 , Θ is the rotation angle between the two frames and s is the scaling factor.

This model is appropriate for image sequences of distant scenes, where perspective distortions can be neglected. The camera motion parameters are computed by matching a small number of feature points between two frames. The current implementation relies on the assumption that features high in the image are due to the horizon, but the user can help the system by specifying a window wherein the features are searched for. We are currently working on a better way of dynamically selecting features based on an optical flow computation algorithm developed by Liu et al. [7]. When forward-translation motion is dominant, regions where the optical flow is small correspond in general to distant parts of the scene. By constraining the selection of features to these regions, we avoid using features that are close to the camera, that have larger image displacements, and that introduce big distortions in our model.

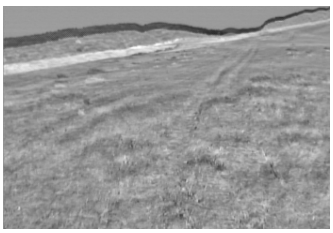
The extraction of features is done by a 5×5 convolution operator (corner detector) over seven vertical partitions of the search feature window. We use only three features to compute the affine camera motion. The partitions are used to guarantee that the features are somewhat spread over the feature window in order to measure the overall flow within that region. The selection of features is terminated as soon as three reliable features are found. A reliable feature is one that returns a high correlation from the current frame f_i to the previous frame f_{i-1}. This same set of features is also used to correlate f_i to frame f_{i+1}, so that feature detection is only performed every other frame.

The correlation also gives the feature displacement between two frames. For a feature with image coordinate T_{xy} detected in frame f_i and being correlated to frame f_{i-1}, the 7×7 neighborhood around T_{xy} of f_i is correlated to all 11×11 points centered around T_{xy} of f_{i-1}. The point of maximum correlation is selected as best match, but the feature itself is selected as a reliable one only when the maximum correlation is higher than 90%. For reliable features, we further refine the feature displacement using the fast subpixel correlation matching described in Jähne [5], which fits a second order polynomial surface around the point of maximum correlation in the correlation image instead of using the original grey level images.

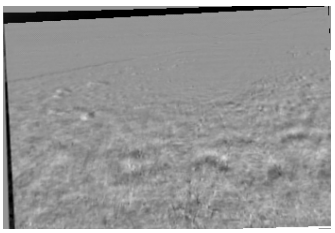
After the computation of a set of reliable features, we first estimate the scaling factor since the Euclidean distance between the feature points depends only on the scale between the two frames. This simplifies the estimation of the rotation and translation parameters, which is further simplified by linearizing the trigonometric terms of the equation above. These other parameters are computed

(a) Original 30^{th} frame.

(b) Original 60^{th} frame.



(c) Difference from original frames.



(d) Difference after stabilization.

Figure 1: Images a) and b) are from the original sequence; c) shows their difference and d) shows the difference between the two stabilized frames.

by solving the overconstrained linear system obtained from the set of matched features. Finally these estimates are used to warp the current frame to the reference frame using bilinear interpolation.

Figure 1a and 1b are image frames from a sequence taken with a camera mounted on a vehicle moving on rough terrain. Figure 1b is 30 frames or about one second apart from the frame in figure 1a. Figure 1c shows the difference between both frames. The dark stripe around the horizon gives an idea of the image displacement after one second. This displacement is very much reduced after stabilization, shown in figure 1d. Observe that the region around the horizon is very smooth. Since the image was warped (scaled, rotated and translated), there are parts of one frame that do not overlap the other, and these regions create the dark stripes near the top and left borders of figure 1d.

3 Velocity Tuned Filters (VTFs)

Image stabilization renders the background of the image approximately stationary. In order to overcome the effects of residual motions, we implemented a *temporal median filter* (TMF). This filter contains an image buffer that keeps the last four stabilized images, and outputs a median image that contains the median values of the images in the buffer. Determining the median grey level of a string of 4 grey levels can be performed very fast by table lookup: we use only 4 bit grey levels, so that the string of 4 grey levels is a 16 bit number; the 64K lookup table lists the 4 bit median for all 16 bit numbers.

In a string of grey levels from 4 frames, the grey levels from IMOs tend to be outliers with respect to the median of the string, so that IMOs at least partially disappear from the median image. A simple image difference scheme can then be applied to detect IMOs: the median image is subtracted from the current stabilized frame (also used to update the median filter buffer), and the result is thresholded. This operation tends to erase the background and to produce black spots at IMO locations.

Along with black spots for IMOs, an image resulting from the TMF still contains noise (due to imperfect stabilization) and spots corresponding to close scene objects that appear to move due to motion parallax. The goal of the VTFs described here is to reject the stabilization noise; motion parallax spots are also rejected if their apparent motion in the image is out of the velocity range for which the filter is tuned; rejection results are best in the upper half of the image.

The rationale behind VTFs is that ground vehicles should appear as black spots moving linearly at almost constant speed over a period of around one second. Vehicles do not make dramatic changes of direction in one second, and their speed generally does not change much either. Black spots that appear and vanish in small fractions

of a second are likely to result from noise and are rejected by the VTF.

Assume that frame f_i (after TMF processing) contains several black spots, and *we want to select only those that move linearly with a rate of 4 pixels per processed frame* [1]. If such spots also appeared in the previous frame, f_{i-1}, we know that by the time frame f_i is captured these spots have moved 4 pixels away from their f_{i-1} positions and are therefore located in frame f_i somewhere on *circles* centered at their f_{i-1} positions with radii 4 pixels. We process frame f_{i-1} to replace all the spot pixels with circles of predicted positions for the time of frame f_i. We can then superpose this image of predictions obtained from frame f_{i-1} with the image of actual spot positions of frame f_i. The spots of frame f_i which fall on circles of predicted positions are good candidates for being labeled as spots moving at 4 pixels per frame.

A fast way of detecting such candidates consists of summing the two images (after assigning suitable grey levels to binary image pixels to prevent overflow of the results), and detecting by thresholding the large values resulting from the addition of a pixel value from the prediction circle for frame f_{i-1} and a pixel value from an actual spot position for frame f_i.

Given a pixel P of a spot in a binary image for frame f_{i-1}, how can the prediction circle for a 4 pixel motion of this pixel between frame f_{i-1} and frame f_i be constructed? A convolution with a circle kernel provides the required result. Consider a 8×8 kernel with a circular pattern consisting of 1's located 4 pixels away from the kernel center, and 0's everywhere else [2]. When convolved with frame f_{i-1}, this kernel will generate nonzero values for pixels located 4 pixels away from pixel P of the spot. Convolving with this kernel thus transforms the spot pixels into circular patterns of radius 4 pixels around them, and produces the desired loci of predicted positions.

3.1 Implementation

Since we are more interested in evidence of motion from a sequence of several frames, the previous scheme is extended to involve more than two frames. For example, evidence of motion can also be obtained from frame f_{i-2} and added to the evidence of frame f_{i-1} and to the actual spot position found in frame f_i. Considering spot positions at frame f_{i-2}, the predictions for these spot pixels at frame f_i belong to circles of radii 8 pixels. These predictions could be obtained with a kernel of dimension 16×16 containing a circular pattern of nonzero values 8 pixels away from the center; however, it is less costly to obtain similar prediction results by scaling down frame f_{i-2} by a factor of 2, then convolving it with the same 8×8 kernel used for frame f_{i-1}. Similarly, predictions from frame f_{i-3} can be obtained with the same kernel and a frame f_{i-3} scaled down by a factor of 3, and predictions from frame f_{i-4} would be obtained by scaling down that frame by 4 before convolving with the kernel. The spots of frame f_i that fall on circles of predicted positions from all 4 previous frames are the best candidates for being labeled as spots linearly moving at 4 pixels per frame during 4 consecutive frames; their locations are obtained by thresholding the sum of frame f_i with the prediction images (the convolution results) from the 4 previous frames.

Therefore, once a new binary image is obtained from the temporal median process, the following steps are performed: (1) Add the result to the prediction images from the 4 previous frames, to flag the detected moving objects; (2) Compute a stack of 4 prediction images for the new frame; these prediction images will be used when step 1 is performed on upcoming frames.

A convolution kernel tuned to predict a wider range of spot motions, say from 2 to 4 pixels, instead of just 4 pixels as described above, is a 8×8 kernel with a wider crown of non zero elements, located between 2 and 4 pixels away from the kernel center, and 0's everywhere else.

If now we want to detect objects moving at faster or slower rate, we generally do not need to recompute new prediction images. For example, a prediction computed at frame f_i for a pixel displacement supposed to be completed 4 frames later will be valid only 2 frames later for an object that is twice as fast, and 8 frames later for an object that is twice as slow. Therefore, detecting objects moving at different pixel rates is only a matter of combining differently the prediction images from the prediction stack of each frame.

[1] The time elapsed between consecutive processed frames f_{i-1} and f_i is generally longer than 1/30 seconds, because the computer is unable to keep pace with the frame rate of the NTSC video signal.

[2] A 9×9 kernel would be more appropriate, but is less convenient for hardware convolutions in the Datacube

For a processing rate of 5 frames/sec on the Datacube, detecting objects moving 2 to 4 pixels per processed frame using 5 consecutive processed frames amounts to detecting objects moving 10 to 20 pixels per second during a full second. For 320×240 images and a 30 degree field of view, such objects would have a motion component of 6 to 12 km/h parallel to the image plane if they are located 100 m away from the camera.

We can also detect objects moving in the range of 12 to 24 km/h at 100 m by accumulating evidence from 3 consecutive processed frames. Trying to detect even faster objects would not be reliable because evidence from only 2 frames could be used.

4 Qualitative Detection

Nelson [8] suggests a qualitative method of detecting independently moving objects termed *constraint ray filtering*. It is based on the observation that the projected motion at any point on the image sphere is constrained to lie on a half line (ray), in the local 2-D space of projected velocity, whose parameters depend only on the observer motion and the location of the image point. On the other hand, the projected motion for an independently moving object is unconstrained and is unlikely to fall on this locus. Thus testing the motion field to determine whether it is consistent with the local constraint ray provides a means of detecting non-rigid motion.

To see how the constraint ray arises, consider the local projective plane centered at point p on the image sphere. The projected velocity at point p is expressed by the vector (u, v) where u and v are the apparent (angular) velocities parallel to the local x and y axes respectively. The rotational motion of the observer can be decomposed into components ω_X , ω_Y, and ω_Z parallel to the local X, Y, and Z axes. The projected velocity due to this rotation is given by $V_\omega = (\omega_x, \omega_y)$ independent of the distance to the world point projecting to p. Similarly, the translational motion of the observer can be decomposed into components v_x, v_y, and v_z, again parallel to the local axes. In this case, the projected velocity is given by $V_t = (v_x/Z, v_y/Z)$ where Z is the distance from the origin to the world point that projects to p. The net projected velocity is the sum of the two pieces given by $V = V_\omega + V_t = V_\omega + V_{XY}/Z$ where V_{XY} is (v_X, v_Y). For a given observer motion, both V_ω and V_{XY} are uniquely determined, and $1/Z$ runs from 0 to $+\infty$. The possible values for V thus lie on a ray in velocity space parallel to V_{XY} and with endpoint V_ω.

The local gradient-parallel component of the projected motion field can be efficiently computed from first order differential image properties. Combining these components into an accurate representation of the full motion field, however, can be unreliable and computationally expensive. Fortunately, it turns out that the gradient parallel component alone provides information that can be used to identify IMOs. If it is known that the nearest object is at least some minimum distance Z_0 away, then the projected motion is constrained to a finite segment in velocity space. It can be shown that if the motion is constrained to such a line segment, then the gradient-parallel components consistent with these motions must lie in a region generated by the exclusive OR of two circles whose diameters are formed by the origin and each end of the constraint segment.

The fraction of the plane representing gradient parallel components consistent with a rigid environment is frequently sufficiently small that an IMO has a good chance of generating gradient parallel components that fall outside of this region. This is particularly true if Z_0 can be bounded below. Thus a movement detector can be can be constructed that utilizes local results of a differential motion computation.

It was assumed in the above analysis that observer motion was known. In some situations, such information might be available from external sources. Determining the observer motion from an image sequence is difficult in the general case. However, for many practical problems, e.g. driving, the system spends most of its time executing only a few types of motion. This suggests that the necessary information about observer motion can be obtained by matching against a relatively small set of prototype motion fields.

4.1 Implementation

We have implemented a constraint ray movement detector that operates in real time (0.1 s latency), and detects independent motion in a variety of situations. A Datacube MV 200 image processing board is used to estimate the gradient-parallel component of the motion field by dividing the temporal derivative by the gradient magnitude. This array is then subsampled and downloaded to a Sun

Processor where a Hough transform technique is used to compute a coarse representation (here a 4×4 array quantized to one of 8 directions) of the true motion field. This is normalized to form a feature vector which is compared against a stored library of canonical motion fields. Currently the system recognizes same-direction and center divergent motion fields.

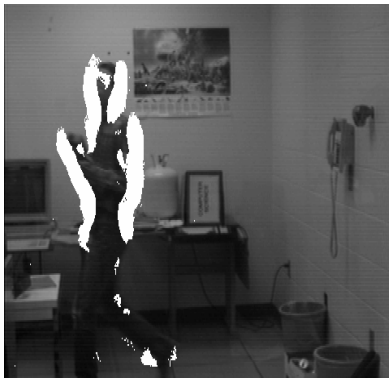


Figure 2: Detection of independently moving object by moving observer

The canonical field is used to generate a filter image which specifies, for every point in the image, the range of gradient parallel components consistent with the presumed motion. The filter image is then compared with the measured estimates of the gradient parallel component, and regions exhibiting inconsistent motion are marked as IMOs. Figure 2 shows the detector's response to a person walking across the field of view as the observer rotates and translates so that the entire scene appears to moving upward.

5 Conclusion

We are currently integrating the constraint ray filtering method (CRF) with the real-time stabilization algorithm. We have used a real time implementation of the CRF without stabilization on image sequences taped from a vehicle driving at low speeds (5-15 m/h) around the campus of the University of Rochester and on NIST grounds. The system performance was satisfactory most of the time, failing on situations where the movement of the camera was not smooth, leading us to believe that its performance could be greatly improved after integration with stabilization is completed.

References

[1] C.H.Anderson, P.J. Burt, and G.S. van der Wal. *Change Detection and Tracking Using Pyramid Transform Techniques.* Proc. SPIE Conference on Intelligent Robots and Computer Vision, Boston MA, 1985, 300-305.

[2] P. Burt and P. Anandan. *Image Stabilization by Registration to a Reference Mosaic.* In *Proc. of ARPA Image Understanding Workshop,* pages 425-434, Monterey, CA, November 1994.

[3] L.S.Davis, R. Bajcsky, R. Nelson, and M. Herman. *RSTA on the Move.* In *Proc. of ARPA Image Understanding Workshop,* pages 435-456, Monterey, CA, November 1994.

[4] M. Irani, B. Rousso, and S. Peleg. *Recovery of Ego-Motion Using Image Stabilization.* In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition,* pages 454-460, Seattle, WA, June 1994.

[5] B. Jähne. *Spatio-Temporal Image Processing.* Springer-Verlag, Berlin, Heidelberg, 1993.

[6] O.J. Kwon, R. Chellappa, and C.H. Morimoto. *Motion Compensated Subband Coding of Video Acquired from a Moving Platform.* In *Proc. of IEEE International Conf. on Acoustics, Speech, and Signal Processing,* Detroit, MI, 1995.

[7] H.C. Liu, T.H. Hong, M. Herman and R. Chellappa. *A General Motion Model and Spatio-Temporal Filters for Computing Optical Flow.* Technical Report, University of Maryland at College Park, Center for Automation Research, CAR-TR-741, October 1994.

[8] R. Nelson. *Qualitative Detection of Motion by a Moving Observer.* International Journal of Computer Vision 7, 33-46, November 1991.

[9] Y.S. Yao, P. Burlina, R. Chellappa, and T.H. Wu. *Electronic Image Stabilization.* International Conference on Image Processing, 1995.

[10] Q. Zheng, and R. Chellappa. *A Computational Vision Approach to Image Registration.* IEEE Transactions on Image Processing, vol. 2, no. 3, July 1993. pp. 311-326.