A Motion Calculation System Based on Background Motion Modeling

TieQi Chen[1], Yi L. Murphey[1], Grant Gerhart[2], and Robert Karlsen[2]

[1]Department of Electrical and Computer Engineering
The University of Michigan-Dearborn
Dearborn, Michigan 48128-1491, U.S.A.
Voice: 313-593-5028, Fax: 313-593-9967
E-Mail: yilu@umich.edu
[2]U. S. Army - TACOM, Warren, MI 48397-5000

Abstract. Motion calculation is often a necessary pre-processing step for moving object detection and tracking. It is a challenging task when the images are taken in outdoor scenes with cameras mounted on a moving vehicle. In this paper we present an accurate and efficient motion calculation system. The accuracy of the system is achieved by estimating background motions and eliminating those pixels that have similar motions to the background motion, and by calculating motion vectors using affine image transformation with Newton-Raphson style search method under subpixel resolution. Efficiency is achieved by concentrating on the regions of interests through a coarse-to-fine process.

Keywords: motion modeling, moving object detection

1. Introduction

Accurate and efficient motion calculation is an important component for intelligent vehicle systems that use vision sensors. Motion calculation is a pre-processing step for many moving object detection and tracking systems[4, 6, 11] that have applications including surveillance, intelligent vehicle systems and moving target tracking.

Motion calculation algorithms have been extensively investigated. Most of the motion detection algorithms are based on image correlation [1, 3, 12] and sum-of-squared-difference(SSD) methods [1, 7]. Motion vectors are obtained through the small inter-frame displacements, which are calculated by tracking a window with respect to translation and linear image deformation through the optimization of some matching criterion. In spite of many progresses [6, 8, 10], motion vector calculation is still considered a computationally expensive process and the results of most algorithms are not satisfactory in outdoor applications [7, 8]. This paper focuses on developing an accurate and efficient motion calculation system for applications in intelligent vehicle systems such as moving vehicle and pedestrian detection. Since such applications require on-board operation, a full search motion estimation

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

algorithm is usually too computationally costly, and low-cost motion estimation algorithms are considered as viable solutions [2].

The motion calculation system described in this paper is designed to achieve both accuracy and efficiency. The accuracy is achieved by effectively modeling of background motion and affine image transformation with subpixel resolution. The efficiency is achieved through a coarse-to-fine search process to find the motion vectors of moving objects in an efficient manner. Figure 1 illustrates the major computational steps involved in the proposed system.

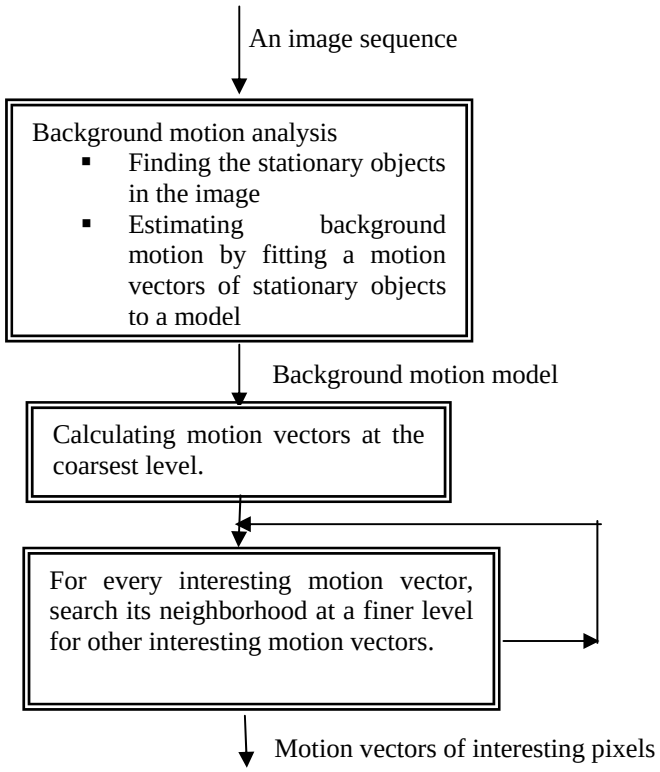


Fig. 1: An efficient and effective system for motion vector calculation.

The following sections describe the major algorithms employed in the motion calculation system.

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

2. A Background Motion Model

Background motion, also referred to as ego-motion, is the motion of the host vehicle, the vehicle with the vision camera system, relative to the road or to the stationary objects such as buildings, trees, etc. Accurate estimation of background motion is important in detecting truly moving objects. There are a number of publications in the literature devoted to background motion estimation. Shashua et al proposed a number of algorithms based on a *direct method* [5, 9]. In the *direct method,* each pixel contributes a measurement. Then a global probability function is used to estimate the parameters of the ego-motion model based on these measurements. To achieve a robust estimation model, Shashua et al reduced the number of estimated parameters from eight to a minimum of three to facilitate the use of *robust* estimation using sampling [8]. The following subsections describe our background motion model obtained by fitting motion vectors of stationary objects such as tree tops or buildings to an analytic model.

2.1 Estimate Background Motion

When driving a vehicle on the road, the driver sees stationary objects such as trees and buildings moving towards the edge of her/his vision at left and right side. That is the background motion we referred to, which is caused by the motion of the driver's own vehicle, i.e. the host vehicle. In most intelligent vehicle systems, the more interested objects are those moving in ways different from the background motion. Without the knowledge of the 3-D locations of the stationary objects in the field of view, it is difficult to calculate the background motion precisely. However, it is possible to obtain a good estimation using the following simplified model (see Fig. 2).

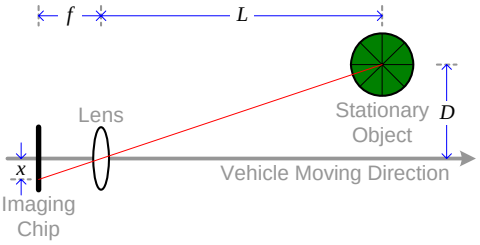


Fig. 2. Schematic drawing of background motion geometry.

We assume that the projected distance between the stationary object and the camera is described by L and D, where L is the distance in parallel to the vehicle moving direction and D in perpendicular to the vehicle moving direction. Let the distance between the lens and the imaging chip be f, and distance between the location on the image chip where the stationary object is projected and the imaging center be x. Based on this geometry we have,

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

$$\begin{cases} x = \dfrac{D \cdot f}{L} \\ \dfrac{\delta_x}{\delta_t} = \dfrac{D \cdot f}{L^2} \cdot \left(-\dfrac{\delta_L}{\delta_t} \right) = \dfrac{x^2}{D \cdot f} \cdot v_c = \left| \dfrac{v_c}{D \cdot f} \right| \cdot \text{sign}(x) \cdot x^2 \end{cases} \quad (1)$$

with vision sensors capturing the scene in image sequences. Equation (1) implies that:

- The stationary objects at the left half of the viewing plane appear to be moving towards left while the objects in the right half of the view plane appear to be moving towards right.
- The closer the stationary objects move towards the edge of the image plane, the faster they appear to be moving. The moving speed increases in a nonlinear order.

Usually v_c and D vary with the time. However during a small time interval such as the two consecutive image frames sampled at 15 to 20 fps v_c remains almost constant for all the stationary objects in the view. Furthermore, if we consider only the freeway scenes, most of the stationary objects are trees and the trees at the both sides of the road remain roughly the same distance to the center of the road. With this assumption, the background motion in an image can be approximated with two parabolic functions — the left one opens downward while the right one opens upward as follows:

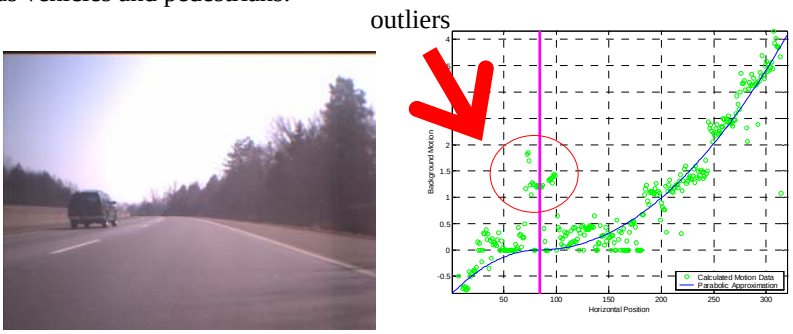
$$v_B(x) = \begin{cases} A \cdot (x - x_0)^2 & (x > x_0) \\ -B \cdot (x - x_0)^2 & (\text{otherwise}) \end{cases} \quad (2)$$

where $v_B(x)$ is the background motion along the horizontal direction, and A, B, and x_0 are the constant parameters to be determined. Note x_0 is the location where the left parabolic function ends and the right parabolic function begins. Both A and B are positive and usually $A > B$ because $|D|$ is smaller when $x > x_0$. This is caused by the fact that the host vehicle is usually driven on the right side of the road. Using the motion vectors calculated from the stationary objects such as tree or building tops in the image by the algorithm described in the next section, we are able to determine all the parameters in Equation (2) using the well-known least-square error method.

Figure 3 illustrates the background motion model. Figure 3 (a) is an image taken from a freeway scene, and Figure 3 (b) illustrates the motion data calculated from the treetops in the image and the parabolic approximation of the motion vectors. The horizontal axis in Figure 3 (b) is the same horizontal axis in the image plane, and the vertical axis is $v_B(x)$. The vertical line in Figure 3 (b) indicates where x_0 is, the spot where the left parabolic function ends and the right parabolic function begins. One can see that the parabolic function approximates the motion data points quite well except for the points indicated by the red arrow and the red circle where the treetops are blocked by a vehicle on the road. These points are considered outliers that can be eliminated by measuring the distance from each point to the parabolic curves: those have larger distances are eliminated from further process.

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

Another interesting thing we can observe from Figure 3 is that it is impossible to detect a moving object by looking at the magnitude of the motion vectors, especially when the motion vectors of the background can have such variety of magnitude. This is why it is important to generate a background motion model, although it is an approximate, in order to get more accurate motion vectors for true moving objects such as vehicles and pedestrians.



(a) (b)

Fig. 3. A background motion model. (a) is an original image N, (b) Illustration of background motion model.

2.2 Finding stationary objects using a sky segmentation algorithm

In most images of outdoor scenes, the sky region occupies a big portion at the top of the image. Usually directly below the sky region we can find stationary objects such as tree tops, telephone poles and building tops. By finding the regions of sky, we could quickly identify a region that contains stationary objects. The sky region is segmented based on edge strengths of pixels, which are obtained through a filtering process of applying four directional Sobel filters to the image. Figure 4 shows two examples of edge images.

(a) (b)

Fig. 4: Two examples of edge images.

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

The algorithm selects multiple seeds within the top of the edge image and then apply the well-known flood fill method to the seeds to find the regions of the sky. The pixels of strong edgeness located at the lower portion of the sky regions are identified as the sky boundary (see Figure 4). The regions containing stationary objects are those directly below the sky boundary.

3 Calculating Motion Vectors

Assume the image from the current frame is $I^t(x,y)$, the image from the previous frame is $I^{t-1}(x, y)$. The problem of calculating the motion vector of an image block is equivalent to minimizing the following equation:

$$\Psi(d_x, d_y) = \sum_i \left[I_t(x_i - d_x, y_i - d_y) - I_{t-1}(x_i, y_i) \right]^2$$

where (x_i, y_i) are the pixels within the image block, (d_x, d_y) is the motion vector to be calculated. By applying Taylor expansion and ignoring the high-order terms, we have:

$$\begin{aligned}
&\frac{\partial \Psi}{\partial d_x} \propto \sum_i [I^t(x_i - d_x, y_i - d_y) - I^{t-1}(x_i, y_i)] \cdot I_x^t(x_i - d_x, y_i - d_y) \\
&\approx \sum_i [S(x_i, y_i) - I_x^t(x_i, y_i) \cdot d_x - I_y^t(x_i, y_i) \cdot d_y] \cdot \\
&[I_x^t(x_i, y_i) - I_{xx}^t(x_i, y_i) \cdot d_x - I_{yy}^t(x_i, y_i) \cdot d_y] \\
&\approx \sum_i [S(x_i, y_i) \cdot I_x^t(x_i, y_i) - [S(x_i, y_i) \cdot I_{xx}^t(x_i, y_i) + (I_x^t(x_i, y_i))^2] \cdot d_x \\
&- [S(x_i, y_i) \cdot I_{xy}^t(x_i, y_i) + I_x^t(x_i, y_i) \cdot I_y^t(x_i, y_i)] \cdot d_y \\
&\approx \overline{S \cdot I_x^t} - \overline{S \cdot I_{xx}^t + (I_x^t)^2} \cdot d_x - \overline{S \cdot I_{xy}^t + I_x^t \cdot I_y^t} \cdot d_y
\end{aligned} \tag{3}$$

Similarly,

$$\frac{\partial \Psi}{\partial d_y} \propto \overline{S \cdot I_y^t} - \overline{S \cdot I_{yy}^t + (I_y^t)^2} \cdot d_y - \overline{S \cdot I_{xy}^t + I_x^t \cdot I_y^t} \cdot d_x \tag{4}$$

where

$$\begin{cases}
S(x, y) = I^t(x, y) - I^{t-1}(x, y) \\
I_x^t(x, y) = \dfrac{\partial I^t(x, y)}{\partial x} \\
I_y^t(x, y) = \dfrac{\partial I^t(x, y)}{\partial y} \\
I_{xx}^t(x, y) = \dfrac{\partial^2 I^t(x, y)}{\partial x^2} \\
I_{xy}^t(x, y) = \dfrac{\partial^2 I^t(x, y)}{\partial x \partial y} \\
I_{yy}^t(x, y) = \dfrac{\partial^2 I^t(x, y)}{\partial y^2}
\end{cases}, \tag{5}$$

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

and the notation $\overline{F}$ denotes the average value of F, namely, $\frac{1}{N}\sum_i F(x_i, y_i)$, and *N* is the total number of the pixels inside the image block. From $\frac{\partial \Psi}{\partial d_x} = \frac{\partial \Psi}{\partial d_y} = 0$, we have:

$$\begin{pmatrix} \overline{S \cdot I_{xx} + I_x^2} & \overline{S \cdot I_{xy} + I_x \cdot I_y} \\ \overline{S \cdot I_{xy} + I_x \cdot I_y} & \overline{S \cdot I_{yy} + I_y^2} \end{pmatrix} \cdot \begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} \overline{S \cdot I_x} \\ \overline{S \cdot I_y} \end{pmatrix} \quad (6)$$

Equation (6) makes the motion vector (d_x, d_y) calculation possible. However, since the high-order terms in the Taylor expansions are not included, Equation (6) works well only for small motion vectors. An accurate estimation of the motion vector can be obtained iteratively in a Newton-Raphson fashion. Our implementation is described as follows.

Algorithm 1: calculating motion vector

[step 1] Initially, we use the motion vector obtained at the previous frame as the current motion vector.
[step 2] Calculate all the elements of the matrices in Equation (6)
[step 3] Solve Equation (6) to obtain a further modification to the current motion vector.
[step 4] If the modification is small, then the algorithm converges. Otherwise, add the modification to the current motion vector and go to Step 2.

During the calculation of the matrices in Equation (6), the image colors and the image gradients at sub-pixel locations have to be used, which can be obtained using bilinear interpolation involving floating-point calculation. However, this algorithm usually does not require much more processing time than the conventional block-matching algorithm does because the calculation usually converges in less than 10 iterations, which is less than the number of possible motion vectors that the conventional block-matching algorithm has to try.

However, without dedicated hardware, calculating motion vectors for the entire image is still very time-consuming. We explored two strategies to speed up the process.

1) Skip pixels where there is not enough texture

Just as the accuracy of the conventional block-matching algorithm requires the uniqueness of the texture in the image block, the accuracy of the algorithm presented here requires a large determinant of the 2×2 matrix in Equation (6). The smaller this determinant is, the less accurate the solution to Equation (6) becomes. When this determinant becomes zero, the solution to Equation (6) becomes completely uncertain. It is obvious that this determinant directly reflects the texture strength of the image block. Therefore there is no point to calculate motion vector at every single pixel. Usually the majority of the pixels can be skipped if we accept only those pixels where the determinant is above a predefined threshold. However, calculating the

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

determinant of the 2×2 matrix in Equation (6) for every pixel is still very costly. Therefore it is more efficient to use a preprocess to select the pixels that have significant edge strength and only those pixels are used to calculate the determinant in Equation (6).

2) Extracting motion information from coarse to fine with focus on interesting regions only

To further speed up the process, we developed a coarse-to-fine process to extract detailed motion information only for the interesting regions where the motion vectors are significantly different from the background motion estimated using Equation (6).

The extracting process is first performed at a coarse level with a step size of 2^n pixels, namely every 2^nth pixel along each of *x* and *y* direction is evaluated using equation (6). The next scan is performed with step size of 2^{n-1} pixels and followed by step size of 2^{n-2}, ..., until step size equal to 1. Only the pixels that satisfy the following two conditions are processed at finer scales:

a) The pixels were not checked in the previous scans.
b) The pixels were close to the ones that have motion vectors significantly different from the estimated background motion in the previous steps.

We summarize the above computations in Algorithm 2, which is a recursive algorithm that performs coarse-to-fine motion vector calculation.

Algorithm 2: A coarse-to-fine recursive algorithm for calculating motion vectors in regions of interests.

[step 1] For a given initial pixel (x, y) at an initial scale s. Make current pixel p = (x, y).
[step 2] If the motion vector of p is already calculated, skip this pixel and exit.
[step 3] If the edge strength of p is below a predefined threshold, skip this pixel and exit.
[step 4] Calculate the determinant of the 2×2 matrix in Equation (6). If this determinant is below a predefined threshold, skip this pixel and exit.
[step 5] Calculate the motion vector using Algorithm 1.
[step 6] If the motion vector is significantly different from the estimated background motion and s>1, scan the immediate surrounding area pixel-by-pixel, and for each pixel (x, y) in the surrounding area call Algorithm 2 with scale s = s/2 as step size.
[step 7] exit the algorithm.

To combine the discussions in this and the last section, we derive the following system that calculates motion vectors for a given image, I^t, in an image sequence.

A System for calculating motion vectors of moving objects

[step 1] Calculate the edge and the gradient of the entire image I^t.
[step 2] Find the region of sky using edge information.
[step 3] Estimate background motion from the stationary objects, which are identified as the narrow regions directly below the region of sky.
[step 4] Find the parameters, A and B, that make Equation (2) best match the background motion for the motion vectors calculated by step 3. This can be accomplished by using a least-square fitting method.

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

[step 5] Scan the entire image (excluding the region of sky) and calculate motion vectors by calling Algorithm 2 at a pixel-by-pixel base at initial scale *s*.

4. Experimental Results and Conclusion

We have fully implemented Algorithm 1 through Algorithm 3 in C++ and tested them on many images captured in outdoor scene. Due to the space limit, we present two examples in Figure 5, and two more images in Figure 6.

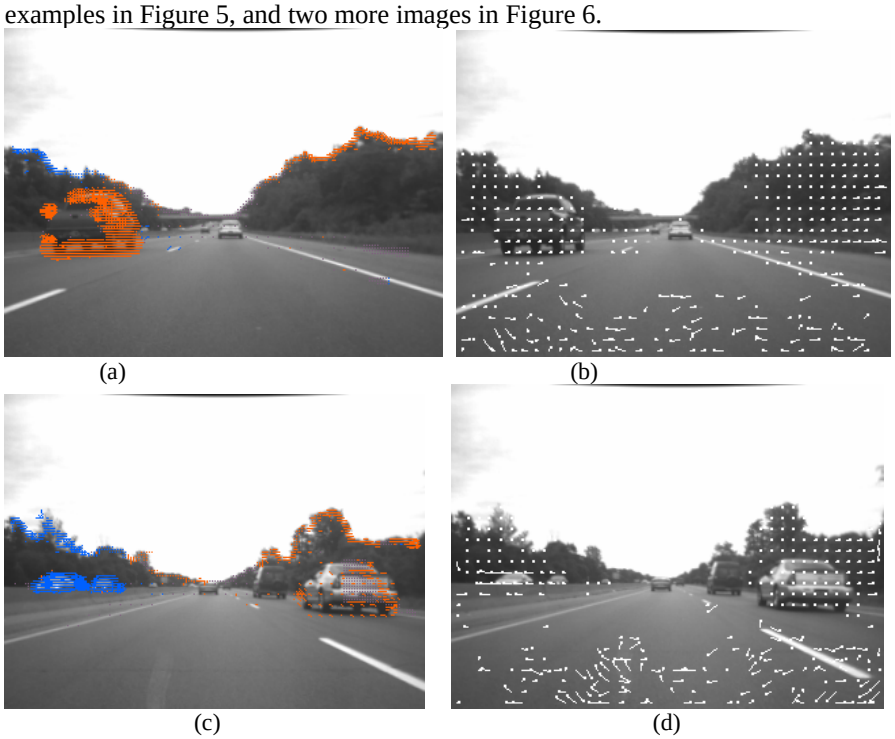


Fig. 5. Two examples of motion vector images generated by the proposed algorithm and by a standard SSD based method. (a) and (c) show the motion vectors generated by the proposed system on Image 1 and 2. (b) and (d) show the motion vectors generated by a SSD based correlation method on the same two images.

Figure 5 (a) and (c) show the motion vectors generated by the proposed system. The motion vectors illustrated in "blue" and "red" color are superimposed on the top of the objects in the original images. The red ones are the motion vectors pointing towards the right direction, while the blue ones towards the left. The motion vectors shown in (b) and (d) were generated by the standard SSD method with a matching window size of 8x8 pixels and a search window of 16x16. The motion vectors are illustrated in small arrows indicating their directions. It can be observed that the proposed algorithm eliminated most of the motion vectors belonging to various stationary objects. Motion vectors belonging to moving vehicles captured by the SSD method are also captured by the proposed algorithm. The result Figure 5 (c) shows

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

clearly the moving vehicles in both directions: motion vectors in "blue" are in the direction opposite to the moving direction of the host vehicle, the motion vectors in "red" are in the same direction as the host vehicle. Figure 6 shows two more images with the motion vectors generated by the proposed system. The motion vectors accurately capture the moving objects in the correction directions.

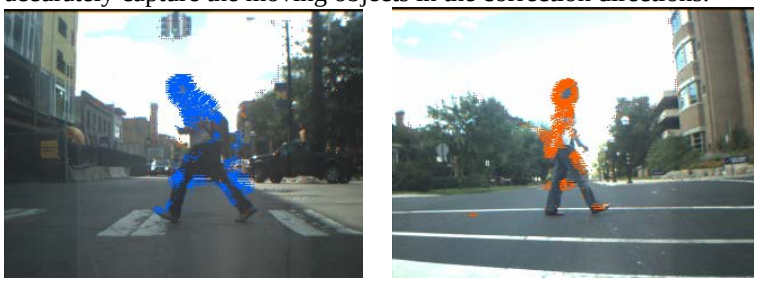


Fig. 6. Two more images showing the motion vectors generated by the proposed system.

We have presented a motion vector calculation system developed based on a background motion model, a motion calculation algorithm with a Newton-Raphson fashion, and a coarse-to-fine procedure to find areas of interesting motions. The experiment results presented in the paper show the proposed system is effective in calculating motion vectors of moving objects.

References

[1] P. J. Burt, C. Yen, and X. Xu. Local correlation measures for motion analysis: a comparative study. EE CPRIP, pp. 269-274, 1982

[2] Chimienti, A.; Ferraris, C.; Pau, D. A complexity-bounded motion estimation algorithm. *IEEE Transactions on Image Processing*, 11(4): 387 – 392, 2002.

[3] D. J. Connor and J. O. Limb. Properties of frame-difference signals generated by moving images. *IEEE Trans. COM*. 22(10):1564-1575, 1974.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Real-Time Tracking of Non-Rigid Objects Using Mean Shift. Proc. Conf. Computer Vision and Pattern Recognition, 2000.

[5] B. K. P. Horn and E. J. Weldon, Jr. Direct Methods for recovering motion. *International Journal of Computer vision*, 2: 51-76, 1988

[6] Jesse S. Jin, Zhigang Zhu, and Guangyou Xu. A Stable Vision System for Moving Vehicles. *IEEE Transactions On Intelligent Transportation Systems*, 1(1), 2000

[7] J. Shi and C. Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, Seattle, June 1994.

[8] Gideon P. Stein Ofer Mano and Amnon Shashua. A Robust Method for Computing Vehicle Ego-motion. Proceedings of IEEE Intelligent Vehicles Symposium, 2000.

[9] G. P. Stein and A. Shashua, "Model based brightness constraints: On direct estimation of structure and motion," In Proceedings of the IEEE Conference on Computer vision and Pattern Recognition, Puerto Rico, June 1997.

[10] T. Suzuki and T. Kanade. Measurement of vehicle motion and orientation using optical flow. In *IEEE* Conference on Intelligent Transportation Systems, Tokyo, Japan, October 1999.

[11] P. Viola, M. J. Jones, and D. Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. Proc. International Conference Computer Vision, pp. 734-741, Oct. 2003

[12] G. A. Wood. Realities of Automatic Correlation Problem. *Photogram. Eng. And Rem. Sens*., 49:537-538, 1983

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8