GLOBAL MOTION ESTIMATION FROM SEMI-DYNAMIC VIDEO USING MOTION SENSORS

Ramazan Savaş Aygün and Aidong Zhang

Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260-2000
{aygun,azhang}@cse.buffalo.edu

ABSTRACT

Global Motion Estimation(GME) techniques have been developed and usually applied on video that motion takes place often. Although these methods produce accurate results where frequent motion occurs, they turn out to be inefficient if motion is not so often in the video as in semi-dynamic videos. In this paper, we propose motion sensors that will indicate the existence of motion and yield quick approximation to the motion when motion exists thus removing the computations of the hierarchical evaluation of low-pass filtered images as in iterative descent methods.

1. INTRODUCTION

Global Motion Estimation (GME) techniques play an important role in video compression methods. GME is usually used to describe the camera motion in a video. The motion is usually modeled with perspective, affine, translation-zoom-rotation or translational motion models. Most of the GME techniques developed concentrate on the accuracy of motion parameters of the chosen motion models [1, 2, 3]. These methods usually include an initial estimation of the subset of the motion parameters and then adjusting of the motion parameters using a hierarchical pyramid of low-pass filtered images. These methods are usually tested on dynamic video where there is almost always motion in the video. In semi-dynamic video applications, like distance education, the motion does not happen often. The motion usually happens at intervals and then the camera stabilizes.

A hierarchical gradient descent method which uses M-estimators has been used to perform GME [1, 2, 3, 4]. An initial matching is necessary to avoid being trapped in local minima. The iterative descent is used to adjust the motion parameters at each level of the pyramid. There are two drawbacks of this kind of approaches: error in initial estimation and hierarchical computation of iterative descent. First drawback causes the technique to be trapped in local minima and the next one introduces significant computation. Tomasi and Shi [5] presents feature selection process based on a dissimilarity of feature selection. The features are selected based on the initial frame and the current frame thus depending on the motion between two frames. There are features presented based on edges (high gradients), corners, block having high spatial frequency. Features are selected using the Laplace operator with its FIR filter coefficients $1, -2, 1$ [6]. This type of features are selected according to the neighboring pixels.

In this paper, we propose motion sensors which are sensitive to motion that may take place. The motion sensors are expected to displace their positions in any type of motion and should be enough to describe the motion. For example, 4 motion sensors should yield information about perspective motion and 3 motion sensors should yield information about affine motion. For each pixel within the initial frame, a block search is performed. We used two kinds of masks: square and circular. The motion sensors are not only edges having high gradients. They are obtained by using more general information than gradients and carry more information in case of motion.

This paper is organized as follows. Section 2 following section explains motion sensors. The GME is discussed in Section 3. Section 4 explains our experiments and the last section concludes our paper.

2. MOTION SENSORS

Our experiments showed that motion estimation methods which process all the pixels that are available are slow for video. So, rather a set of feature points are selected and they are tracked in each frame. Mapping feature points in two frames is not easy since there may be several feature points that share the same characteristics.

This method aims to match blocks that have motion sensors as their center points rather than mapping feature points

This research is supported by NSF grant IIS-9733730.

themselves. This approach does not require the exact mapping of the feature point. The error function used in block matching introduces flexibility in tracking of the motion sensors points even though feature points are not located as they are expected.

Motion sensors are feature points which are sensitive to motion. The motion sensors are expected to displace their positions in any type of motion. 2 motion sensors should yield information about translation-zoom-rotation motion and 1 motion sensor should be enough to detect translational motion.

Edges having high gradients are likely to be candidates for motion sensors. Unfortunately, gradient contains information within ± 1 pixel distance which is usually not enough to detect motion due to existence of patterns or aperture problem. Another problem with the edges is the mapping of edges in two frames. Because another edge may possess similar information to the desired edge and makes it difficult to detect the motion.

It is very hard to find absolute motion sensors that will change their locations after every type of motion. The goal is to find the motion sensors that will displace their locations in most types of motion. In Fig. 1, there are two lines intersecting each other. Every point lying on lines l_1 and l_2 is likely to displace to their positions after a motion. Let the slopes for l_1 and l_2 be m_1 and m_2, respectively. If there is a motion in the direction of m_1, the points lying on line l_1 will be useless and moreover, make the motion estimation difficult. Similar statement is also true for line l_2. Which points carry the highest information for motion? The answer is the intersection of line l_1 and line l_2 because the p will always change its location either there is a motion in the direction of m_1 or m_2.

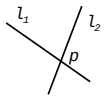

Fig. 1. Intersecting lines.

In real video, the existence of lines and their intersection are not guaranteed. Even though, the lines may exist, they may not be straight lines or may be hard to detect. Therefore, we propose a method that will work without detection of lines and using gradients (since they carry limited information about the surrounding pixels).

2.1. Detection of Motion Sensors

A feature point is distinguishable by its surrounding pixels. A block matching process is performed for a block containing motion sensor ms as its center within the same frame. The block search operation is performed within distance of d. Each block size has a height and width of n pixels. For the block search operation, n-step search can be used. As an error function, we use the sum of absolute error differences (SAD)

$$\varepsilon(B^p, B^t)_{p \neq t} = \Sigma_{i=1}^{n} \Sigma_{j=1}^{j=n} |B^{ms}(i,j) - B^t(i,j)| \quad (1)$$

where B^p represents the block where p is its center and B^t represents a block within the search distance d. Let x_p and y_p denote the x and y coordinates of a pixel p. The sensitivity of a pixel is determined by $s(p)$ which is computed by

$$min_{(x_p-d) \leq x_t \leq (x_p+d), (y_p-d) \leq y_t \leq (y_p+d)} (\varepsilon(B^p, B^t)); \quad (2)$$

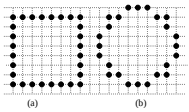

Fig. 2. Square and circular masks.

We have used two different masks to detect motion sensors: square and circular. Figure 2 depicts 8x8 square mask and a circular mask of radius 4. Square mask contains 64 pixels whereas circular mask has 61 pixels. Circular mask is a better approximation than square masks, since pixels within a radius is considered. In real examples, the difference is not distinguishable when either of these masks is used. Fig. 3 shows motion sensors detected for a frame from a mobile & calendar frame by square and circular masks.

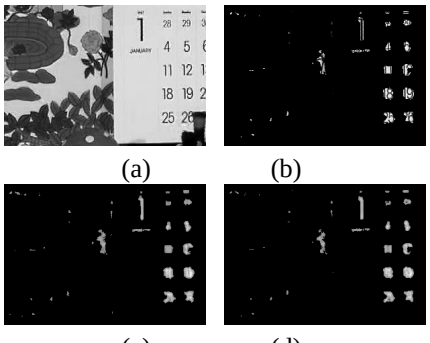

Fig. 3. Mobile & Calendar example for motion sensors a) original frame b) motion sensors in the frame from circular mask c) application of square mask d) application of circular mask.

One problem in matching of the blocks is the significant displacement between two frames. In most cases, the motion between two consecutive frames are not huge.

2.2. Optimization in Detection of Motion Sensors

Although motion sensor detection is performed at the beginning, the comparison of blocks is the most expensive part and it should be optimized. There are a couple of optimizations that can be performed. Since the goal is not to find all the motion sensors, there is no problem if some of them are missed. In most cases, a motion sensor has a neighboring motion sensor and an ordinary pixel has a neighboring ordinary pixel. Therefore, motion sensor detection can be performed at intervals. To increase the performance, every other pixel is skipped during detection.

Note that the distance between two blocks are symmetric. If the distance between two blocks are already computed, there is no need to compute the distance between those two blocks again. If the search distance is d, there are initially $(2d)^2 - 1$ SAD computations. Due to symmetry, this is reduced to $d^2 - 1$.

The sensitivity of a pixel will be high if it is a motion sensor. Otherwise, the pixel's region is similar to its surrounding. If the sensitivity is low when comparing the blocks, the further blocks do not need to be compared. Because that pixel can no longer be a motion sensor.

3. GLOBAL MOTION ESTIMATION USING MOTION SENSORS

There are different types of motion models that are used in global motion estimation depending on the camera operations and the structure of the scene. In this paper, our goal is to detect the camera motion which is parameterized by perspective motion model:

$$\begin{aligned} x_i' &= \frac{a_0 + a_2 x_i + a_3 y_i}{a_6 x_i + a_7 y_i + 1} \\ y_i' &= \frac{a_1 + a_4 x_i + a_5 y_i}{a_6 x_i + a_7 y_i + 1} \end{aligned} \tag{3}$$

where a_0, a_1, a_2, a_3, a_4, a_5, a_6, and a_7 are motion parameters and (x_i', y_i') is the new location of (x_i, y_i). This model turns into affine motion when $(a_6 = 0,\ a_7 = 0)$, translation-zoom-rotation motion when $(a_4 = -a_3,\ a_5 = a_2,\ a_6 = 0,\ a_7 = 0)$, and translational motion when $(a_2 = 1,\ a_5 = 1,\ a_4 = 0,\ a_5 = 0,\ a_6 = 0,\ a_7 = 0)$.

The error between two frames can be declared as

$$\varepsilon = \sum^N e_i^2 \quad where\ e_i = I'(x_i', y_i') - I(x_i, y_i) \tag{4}$$

where $I(x_i, y_i)$ is the intensity at (x_i, y_i) in the previous frame and its corresponding in the current frame is $I'(x_i', y_i')$. Error ε is computed for pixels overlapping in two frames.

The iterative descent methods are likely to be trapped in local minima when they try to minimize Equation 4. The hierarchical (iterative descent) approach is usually applied to detect the motion parameters. Initial estimation of translational parameters is necessary to avoid local minima. This method requires generation of low-pass filtering of a frame, computation of gradients and computing gradient descent for each layer. In our experiments, motion sensors approximately give the motion parameters without generation of hierarchical pyramid.

The mapping of a motion sensor is computed using block matching. For finding the best matching block, the full searching or n-step searching can be applied. Since the number of feature points is usually very few, the process of full searching is still fast and sometimes is desired if accuracy is needed. For perspective, affine, translation-zoom-rotation and translational motion at least 4, 3, 2 and 1 motion sensors are required, respectively. Extra motion sensors can be used to check the correctness of motion.

In our implementation, we choose 3 motion sensors to estimate the motion. Since 3 motion sensors are used, only the parameters for affine motion can be estimated. The parameters for translational and translation-zoom-rotation are subsets of these parameters. Since perspective motion is sensitive to slight changes in the parameters and iterative descent methods can be trapped in local minima, the initial guess of perspective motion parameters is not performed by motion sensors. The general structure of the motion estimation algorithm is depicted in Fig. 4.

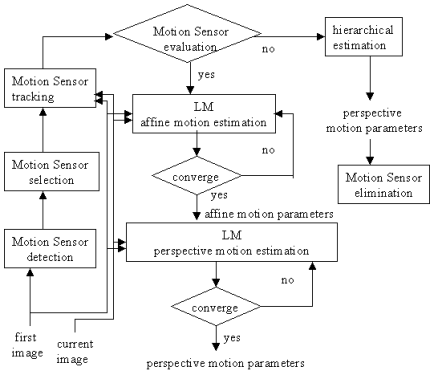

Fig. 4. Global Motion Estimation Algorithm.

If the motion detection by motion sensors is confirmed, affine motion is estimated using Levenberg-Marquardt(LM) iterative nonlinear minimization algorithm. Once the parameters are estimated for affine motion, these parameters are again fed into LM algorithm to estimate the perspective motion parameters.

If the motion sensors cannot uniquely identify a global motion, the motion estimation is determined as in [1]. After the motion parameters are obtained, motion sensors which do not conform to global motion are eliminated and not used in the subsequent motion detection.

To increase the robustness of motion estimation, M-estimators

(a) (b) (c) (d)

Fig. 5. Experiments. a) Background sprite from coastguard b) dynamic sprite from coastguard c) a frame from a lecture d) the dynamic sprite of the frame.

are used [1, 2] and the error is expressed as:

$$\sum^{N} \rho(e_i) \tag{5}$$

where $\rho(e_i) = e_i^2$ in the original formulation. Since this function gives more weight to large errors, it is biased by local motion (which are outliers for global motion). To decrease the effect of outliers, the truncated quadratic motion is used:

$$\rho(e_i) = \begin{cases} e_i^2, & \text{if } |e_i| \leq t \\ 0, & \text{if } |e_i| > t \end{cases} \tag{6}$$

where t is a threshold selected according to the histogram of the errors.

4. EXPERIMENTS

Our test database consists of the lectures recorded and digitized at SUNY/Buffalo. The lectures take place in a classroom which is a restricted environment. The camera motion is not often and less than 10% of the frames contain camera motion. The global motion is usually camera motion and the perspective motion model is chosen for motion estimation. To show the effects of our experiments, we have also tested on MPEG-4 test sequences like coastguard which contains almost continuous motion in all frames.

The difficulty with this type of video is that the motion sensors may be occluded by moving objects or may disappear from the scene due to camera motion. In those cases, motion sensors will significantly increase the error in Equation 4. The threshold t_{error} determines that the maximum average e_i^2 could be between consecutive frames. In our test environment t_{error} is chosen as 100. Once the error exceeds t_{error}, motion sensors are recomputed for the new frame. The dynamic and static mosaics [7] are generated to evaluate the correctness of the algorithm. If sprites can be generated properly, the GME is assumed to be right. Figure 5 shows the sprites generated from coastguard example and lecture examples.

5. CONCLUSION

In this paper, we introduced a method to perform the global motion estimation method from semi-dynamic video. If the video does not contain continuous motion, the existence of the motion can be detected by motion sensors. Moreover, motion sensors also give good approximation to the motion model parameters. This initial estimation reduces the number of computation at the levels of pyramid. Although initial detection of the motion sensors is costly, it is usually done once at the beginning and can be optimized by the methods given in Section 2.2.

6. REFERENCES

[1] F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 497–501, March2000 2000.

[2] T. Sikora A. Smolic and J.-R. Ohm, "Long-term global motion estimation and its application for sprite coding, content description and segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, pp. 1227–1242, December 1999.

[3] A. Smolic and J.-R. Ohm, "Robust global motion estimation using a simplified m-estimator approach," in *Proc. ICIP2000, IEEE International Conference on Image Processing*, September 2000.

[4] Wen Gao Yan Lu and Feng Wu, "Fast and robust sprite generation for mpeg-4 video coding," in *The second IEEE Pacific-Rim conference on Multimedia (PCM)*, October 2001, pp. 118–125.

[5] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*. Seattle, June 1994.

[6] B. Stabernack H. Richter, A. Smolic and E. Mller, "Real time global motion estimation for an mpeg-4 video encoder," in *Proc. PCS'2001, Picture Coding Symposium*, April 2001.

[7] M. Irani and P. Anandan, "Video indexing based on mosaic representations," in *Proceedings of IEEE*, May 1998, pp. 905–921.