REAL-TIME IMAGE MOSAICING FROM A VIDEO SEQUENCE

Masakatsu Kourogi, Takeshi Kurata†, Jun'ichi Hoshino‡ and Yoichi Muraoka

Waseda University, School of Sci. & Eng. 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555, JAPAN.
†Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba, Ibaraki, 305-8568, JAPAN.
‡SECOM IS Laboratory, 8-10-16 Shimorenjaku, Mitaka, Tokyo, 181-8528, JAPAN.
E-mail:kourogi@muraoka.info.waseda.ac.jp

ABSTRACT

This paper describes a fast and robust image registration method that can be used to create a panoramic image/video from video sequences. To estimate alignment parameters for image registration, the method computes pseudo motion vectors that are rough estimates of optical flows at each selected pixels. Using the proposed method, we implemented a software system that can, with a low-cost PC, create and display panoramic images/videos in real-time.

1. INTRODUCTION

Creating panoramic images from video sequences is useful for many applications such as image browsing [8], video surveillance [1] and virtual reality [3][7]. Its real-time processing is of great significance because it enables video mosaicing to be performed without accumulating video sequences and it enables online user to specify the way that panoramic images are created. It also enables us to create a panoramic image on which live video frames are overlaid. Such panoramic videos are very suitable for presentation of a wide scene because they let a guide move around while explaining in the scene without causing the viewer to lose sight of the global scene. In order to overlay live video frames on a panoramic image, it is necessary to achieve real-time image registration for scenes including objects that are moving or are not included in the panoramic image (that are outliers).

Although previous works [2] have dealt with image registration when the scenes include objects that are outliers, the previous method requires intensive computational cost that hinders real-time software implementation. On the other hand, some previous methods [9] can provide nearly real-time processing. However they assume the scenes in video sequences not to include outliers, and implementation issues have not covered thoroughly.

In this paper, we describe a fast and robust method for frame-to-frame and frame-to-mosaic image registration under affine and projective motion model, and consider implementation issues for real-time processing.

We assume the influence of motion parallax in the scenes to be relatively small, and compute pseudo motion vectors that are rough estimates of optical flows at each selected pixels. Pixels are selected according to the gradient of image brightness, and the number of pixels is controlled to guarantee real-time processing. Then the pseudo motion vectors are verified by pixel-wise matching, and alignment parameters are estimated from verified motion vectors. The M-estimator is used to exclude outliers.

Using this method, we implemented a software system that can create and display a panoramic image/video from video sequences in real-time with a low-cost PC.

2. IMAGE REGISTRATION

2.1. Frame-to-frame image registration

The proposed method estimates frame-to-frame alignment parameters by the following computationally lightweight steps.

1. Compute pseudo motion vectors that are rough estimates of the optical flows at each pixel.
2. Verify the computed motion vectors by pixel-wise matching.
3. Estimate the motion parameters from verified motion vectors.
4. Compensate the global motion between successive frames and repeat steps 1–4 on the compensated frames until either the error of motion compensation is below a threshold or a fixed number of iterations has been completed.
5. Update the motion parameters by using the M-estimator.

0-7803-5467-2/99/ $ 10.00 © 1999 IEEE

2.1.1. Pseudo motion vectors and pixel-wise matching

The computation of pseudo motion vectors at each pixel is based on the spatio-temporal gradient of image brightness. Let $I(x, y, t)$ be the brightness of a point (x, y) at time t, and we define the pseudo motion vector (u_p, v_p) by using the following equations:

$$u_p = -\frac{\partial I}{\partial t}/\frac{\partial I}{\partial x}, \quad v_p = -\frac{\partial I}{\partial t}/\frac{\partial I}{\partial y}. \tag{1}$$

Since the pseudo motion vectors at each pixel can be computed by only the spatio-temporal gradients at the pixel, its computational cost is quite small.

The qualitative characteristics of these equations can be explained as follows. If the true motion vector (u, v) at the point (x, y) is sufficiently small, the well-known constraint equation $\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$ should be satisfied. Using this equation, the pseudo motion vector can be written as follows:

$$u_p = -\frac{\partial I}{\partial t}/\frac{\partial I}{\partial x} = u + (\frac{\partial I}{\partial y}/\frac{\partial I}{\partial x}) \cdot v,$$
$$v_p = -\frac{\partial I}{\partial t}/\frac{\partial I}{\partial y} = v + (\frac{\partial I}{\partial x}/\frac{\partial I}{\partial y}) \cdot u.$$

The ratios of horizontal-to-vertical and vertical-to-horizontal gradients of image brightness, $\frac{\partial I}{\partial y}/\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial x}/\frac{\partial I}{\partial y}$, are empirically known to have bell-shaped distributions with peaks at zero. Thus the distribution of pseudo motion vectors (u_p, v_p) has its peak near the true motion vector (u, v). Although the constraint equation does not generally hold if the motion vector (u, v) is large, the distribution of pseudo motion vectors (u_p, v_p) is also empirically biased to the true motion vector.

When dealing with actual discrete image sequences, we use the following equations to compute the horizontal, vertical, and time gradients of image brightness. Let $I_r(x, y)$ and $I_c(x, y)$ be the image brightnesses of the reference frame and the current frame at a point (x, y):

$$\frac{\partial I}{\partial x} = (I_r(x + \delta x, y) - I_r(x - \delta x, y))/2\delta x,$$
$$\frac{\partial I}{\partial y} = (I_r(x, y + \delta y) - I_r(x, y - \delta y))/2\delta y,$$
$$\frac{\partial I}{\partial t} = I_c(x, y) - I_r(x, y),$$

where δx and δy are horizontal and vertical sampling units of image, respectively.

Given a threshold T, we can select a pseudo motion vector (u_p, v_p) by testing the following condition:

$$|I_c(x + u_p, y + v_p) - I_r(x, y)| < T. \tag{2}$$

Since the true motion vector should satisfy this equation, it is possible to verify the pseudo motion vectors by testing this condition.

2.1.2. Parameters estimation

In the next step, the global motion parameters are estimated from the selected pseudo motion vectors by using the least-squares method. In this paper, we estimate affine motion model because of its stability of parameters estimation. Then the estimated global motion between the frames is compensated and the motion parameters are re-estimated from the compensated frames. There is, however, heavy computational cost for compensating motion between frames at each iteration. Thus we use the following strategy in order to avoid the motion compensation. Let the previous estimation result of affine parameters be $(a_1, a_2, .., a_6)$. A compensation motion vector (u_e, v_e) can be computed at each pixel (x, y) by calculating

$$u_e = a_1 x + a_2 y + a_3,$$
$$v_e = a_4 x + a_5 y + a_6.$$

We can compute pseudo motion vectors and time derivatives between compensated frames without actually compensating by using the following equations:

$$u_p = -\frac{\partial I}{\partial t}/\frac{\partial I}{\partial x} + u_e, \quad v_p = -\frac{\partial I}{\partial t}/\frac{\partial I}{\partial y} + v_e, \tag{3}$$

where

$$\frac{\partial I}{\partial t} = I_c(x + u_e, y + v_e) - I_r(x, y). \tag{4}$$

Computation and selection of pseudo motion vectors and the estimation of the global motion parameters are repeated until either the average error of pixel-wise matching is below a threshold or a fixed number of iterations has been completed. The selected pseudo motion vectors, however, include two types of outliers: (a) those caused by moving objects, (b) those caused by the wrong pixel-wise matching. We therefore use the M-estimator to reject those outliers. We use the Cauchy weight function $w(\epsilon) = \frac{1}{1+(\epsilon/c)^2}$ because its computational cost is relatively low.

By giving the estimated affine parameters as an initial estimate, we can find projective motion parameters by using Szeliski's well-known algorithm [6]. Since the initial estimate is close to the solution of projective parameters, the algorithm should converge to it in a relatively small number of iterations.

A diagram of the frame-to-frame image registration is shown in Figure 1.

2.2. Frame-to-mosaic image registration

In estimating frame-to-mosaic alignment parameters, it is necessary to handle large displacement because the

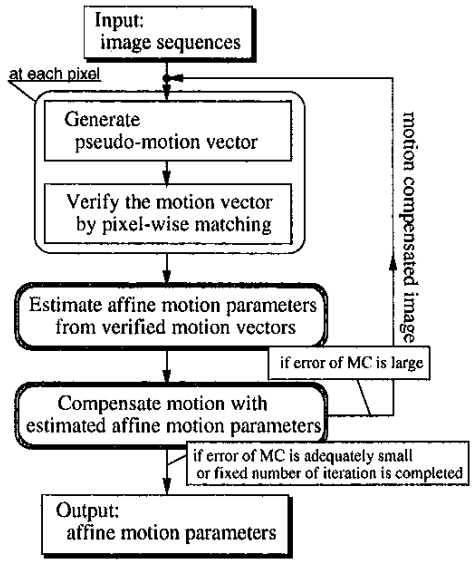

Figure 1: A diagram of image registration

panoramic image is much larger than the frame. The previous method of frame-to-frame image registration, however, requires the initial estimate to be relatively close to the parameters because the method is based on the spatio-temporal gradient of image brightness.

We therefore give multiple initial estimates so that the motion parameters to be estimated are sufficiently close to at least one of the initial estimates. By taking the estimated parameters that gives the smallest MSE (mean squared error), we can estimate the frame-to-mosaic alignment parameters for image registration. Int this paper, we estimate affine parameters to create a cylindrical panorama and estimate projective parameters to create a planar panorama.

Once the alignment parameters are estimated, subsequent estimation finds the parameters by using the previous result as an initial estimate.

2.3. Computational cost

Most of the computational cost of the proposed method is due to the computation of pseudo motion vectors and of pixel-wise matching, since they are computed at each pixel and at each iteration.

The computation of pseudo motion vectors requires the calculation of the time derivative of image brightness and requires the derivative to be divided by the horizontal and vertical gradient of image brightness in Eq. (3). Because the gradients are fixed throughout iteration, by computing the reciprocal of gradients $1/\frac{\partial I}{\partial x}$ and $1/\frac{\partial I}{\partial y}$, we can calculate the pseudo motion vectors at each iteration without the division operation by using the following equations:

$$u_p = -\frac{\partial I}{\partial t} \cdot (1/\frac{\partial I}{\partial x}) + u_e, \quad v_p = -\frac{\partial I}{\partial t} \cdot (1/\frac{\partial I}{\partial y}) + u_e.$$

Since for most CPUs division is much more expensive operation than multiplication, the use of these equations can be expected to reduce the computational cost.

Sub-pixel image brightness is required for pixel-wise matching in Eq. (2) and computation of time derivative in Eq. (4). We use linear interpolation to compute sub-pixel brightness.

Let N be the number of pixels and M be the number of iterations. The numbers of operations required are listed in Table 1.

operation	# of operation
compute reciprocal of gradient	$2 \times N$
compute compensation vector	$N \times M$
linear interpolation	$2 \times N \times M$
parameters estimation	M

Table 1: Computational cost for image registration

3. IMPLEMENTATION ISSUES

We built a software system that (1) creates and displays new panoramic images from video sequences, and (2) overlays the video frame on a panoramic image that has already been created.

3.1. System overview

The system creates and displays panoramic images/videos from video sequences by going through the following steps.

1. Capturing a video frame.
2. Image registration.
3. Merging the frame to the panoramic image/video.
4. Displaying the panoramic image/video.

To create a new panoramic image from video sequences, we estimate frame-to-frame alignment parameters between captured frames, and merge each frame to a mapped plane by warping it with the estimated parameters. We estimate affine parameters to map each frame to a cylindrical panorama and estimate projective parameters to map each frame to a planar panorama.

The cylindrical panorama is used for creating wide-view panoramic images. The vertical slit at the image

center of each frame is merged to the panoramic image since the slit is approximately tangent plane of the cylindrical panorama.

To overlay video frames on a panoramic image that has already been created, we find frame-to-mosaic alignment parameters by using the proposed method and we overlay the frames by warping them with the parameters.

3.2. Computational resources for implementation

We implemented the system on a low-cost PC (OS: Linux-2.2.4, CPU: PentiumII-450MHz [Dual CPU]). The software uses Video for Linux API for capturing full-rate video frames (size: 320 × 240, 24-bit RGB), and X11R6 library for displaying a panoramic image/video.

3.3. System architecture

The four steps explained in Section 3.1 were implemented as a pipeline of UNIX processes, each communicating with the other processes via shared memory facilities. If the OS supports multiprocessing function, the processing can be speeded-up by the OS without rewriting or recompiling code as the number of CPUs increases since most of the multiprocessing-supported OSes regard a UNIX process as a unit of scheduling. The leading UNIX OSes such as Linux, IRIX and Solaris are aggressively supporting multiprocessing.

3.4. System optimization

A naive implementation of the pipeline processing described in Section 3.1 limits the frame-rate of all the processes to that of the most time-consuming process. In fact, on the computational resource described in Section 3.2, processes 1 and 2 require 30 msec. per frame, while processes 3 and 4 respectively require 80 and 70 msec. This means that process 1 and 2 must wait 50 msec. for the completion of process 3. That is critical because image registration between successive frames becomes more challenging as the frame rate is lower (because the displacement can be larger). We therefore improved the pipeline processing so that it would not stall the image registration process. We did this as follows:

If process 3 is not completed when process 2 is completed, process 2 continues image registration between the next frames and adds the estimated parameters to the affine parameters until process 3 is completed. When process 3 is completed, process 2 delivers the accumulated parameters and the last frame to process 3 for merging. This improvement lets the image registration process run without stalling.

4. EXPERIMENTAL RESULTS

The experimental results described in this section show the effectiveness of the proposed method in creating panoramic images/videos from video sequences on a low-cost PC.

4.1. Online creation of panoramic images

We create the panoramic image with projective parameters, from live input video taken from a pan/tilt/zoom camera. For image registration, we use a threshold of pixel-wise matching $T = 5.0$, and we set the number of pixels $N = 2000$, and the number of iterations $M = 30$. Those values are empirically suitable for both real-time processing and stable parameter estimation.

The panoramic image created (size: 900 × 310) is shown in Figure 2. The processing times required for each process and for overall throughput are listed in Table 2.

Running the software on the multiprocessor PC (2 CPUs) reduces the throughput by almost 50%. This indicates that the processing power of two CPUs can be fully exploited.

Process	processing time
Image registration	30 msec
Merging frame to mosaic	80 msec
Displaying mosaic	70 msec
Total	180 msec
Throughput	90-100 msec

Table 2: Processing times and throughput for the proposed method

Figure 2: Input video frames (above) and a panoramic image created (below).

4.2. Overlaying video on panoramic image

We created a panoramic image (cylindrical panorama) on which input video frames were overlaid. The panoramic image (size: 1172×240) and the input frames (size: 320×240) are shown in Figure 3, and the overlaid panoramic video is shown in Figure 4. It took 800 m-sec. to establish frame-to-mosaic image registration, because we gave 20 initial estimates. Once the alignment parameters were estimated, the system could overlay input video frames at 10-12 frames per sec.

In this video sequence, a guide is introducing the building while moving around the scene. The created overlaid panoramic video delivers and depicts the scene better than the input video frames do.

Figure 3: A panoramic image (above) and input video frames (below).

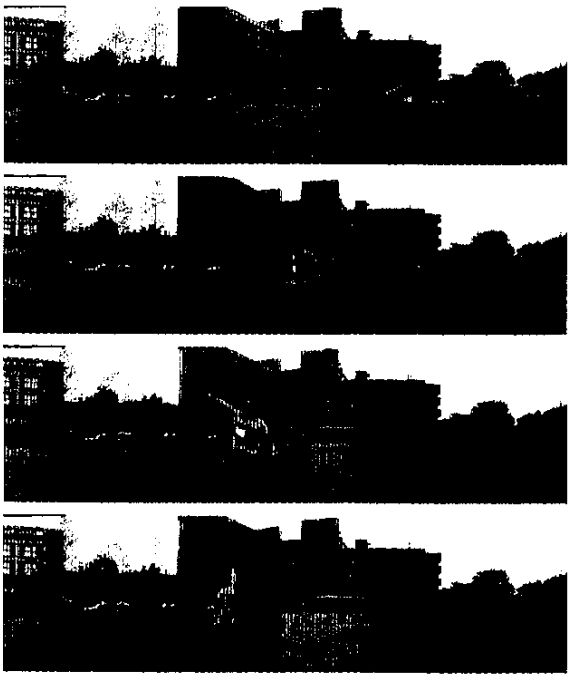

Figure 4: A video overlaid on the panoramic image.

5. CONCLUSION

We developed a fast and robust method for estimating frame-to-frame and frame-to-mosaic alignment affine and projective parameters for the online creating and displaying of panoramic images/videos from video sequences. We also considered some implementation issues to for real-time processing. Experimental results show that the proposed method can, with a low-cost PC, create and display panoramic images/videos from video sequences in real-time.

6. REFERENCES

[1] L. G. Brown, "A survey of image registration techniques," ACM Computing Surveys, Vol. 24, No. 4, pp. 325-376, 1992.

[2] M. Irani, P. Anandan, and S. Hsu, "Mosaic based representations of video sequences and their applications," ICCV95, pp. 605-611, 1995.

[3] S. E. Chen, "QuickTime VR – an image-based approach to virtual environment navigation," SIGGRAPH'95, pp. 29-38, 1995.

[4] H. S. Sawhney and S. Ayer, "Compact representation of videos through dominant and multiple motion estimation," IEEE PAMI, Vol. 18, No. 8, pp. 814-830, 1996.

[5] H. S. Sawhney and R. Kumar, "True multi-image alignment and its application to mosaicing and lens distortion correction," IEEE PAMI, Vol. 21, No. 3, pp. 235-243, 1999.

[6] R. Szeliski, "Video mosaics for virtual environments," IEEE Computer Graphics and Applications, 16(2):22-30, 1996.

[7] R. Szeliski, H. Y. Shum, "Creating full view panoramic image mosaic and environment maps," Proc. SIGGRAPH '97, 1997.

[8] Y. Taniguchi, A. Akutsu, and Y. Tonomura, "PanoramaExcerpts: Extracting and packing panoramas for video browsing," Proc. ACM Multimedia '93, pp. 39-46, 1997.

[9] H. S. Sawhney, R. Kumar, G. Gendel, J. Bergen, D. Dixon, and V. Paragano, "VideoBrush: Experiences with Consumer Video Mosaicing," Proc. of Fourth IEEE Workshop on Applications of Computer Vision, pp. 56-62, 1999.