2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

# Real-time Feature-based Video Mosaicing at 500 fps

Ken-ichi Okumura, Sushil Raut, Qingyi Gu, Tadayoshi Aoyama, Takeshi Takaki, and Idaku Ishii

*Abstract*— **We conducted high-frame-rate (HFR) video mosaicing for real-time synthesis of a panoramic image by implementing an improved feature-based video mosaicing algorithm on a field-programmable gate array (FPGA)-based high-speed vision platform. In the implementation of the mosaicing algorithm, feature point extraction was accelerated by implementing a parallel processing circuit module for Harris corner detection in the FPGA on the high-speed vision platform. Feature point correspondence matching can be executed for hundreds of selected feature points in the current frame by searching those in the previous frame in their neighbor ranges, assuming that frame-to-frame image displacement becomes considerably smaller in HFR vision. The system we developed can mosaic 512×512 images at 500 fps as a single synthesized image in real time by stitching the images based on their estimated frame-to-frame changes in displacement and orientation. The results of an experiment conducted, in which an outdoor scene was captured using a hand-held camera-head that was quickly moved by hand, verify the performance of our system.**

## I. Introduction

Video mosaicing is a video processing technique in which multiple images are merged into a composite image that covers a larger, more seamless view than the field of view of the camera. Video mosaicing has been used to generate panoramic pictures in many applications such as tracking, surveillance, and augmented reality. Most video mosaicing methods [1] determine the transform parameters between images by calculating feature points using feature extractors such as the KLT tracker [2] and SIFT features [3] to match feature point correspondences between images. Many real-time video mosaicing approaches for fast image registration have been proposed. Kourogi et al. proposed a fast image registration method that uses pseudo motion vectors estimated as gradient-based optical flow [4]. Civera et al. developed a drift-free video mosaicing system that obtains a consistent image from 320×240 images at 30 fps using an EKF-SLAM approach when previously viewed scenes are revisited [5]. de Souza et al. performed real-time video mosaicing without over-deformation of mosaics using a non-rigid deformation model [6]. Most of these proposals only apply to standard videos at relatively low frame rates (e.g., NTSC 30 fps). Further, the camera has to be operated slowly, without any large displacement between frames, in order to prevent difficulties in matching feature point correspondences. However, in scenarios involving rapid camera motion, video images at such low frame rates are insufficient for tracking feature points in the images. Thus, there is a demand for real-time high-frame-rate (HFR) video mosaicing at rates of several hundred frames per second and greater.

K. Okumura, S. Raut, Q. Gu, T. Aoyama, T. Takaki, and I. Ishii. are with Hiroshima University, Hiroshima 739-8527, Japan (corresponding author (I. Ishii) Tel: +81-82-424-7692; e-mail: okumura@robotics.hiroshima-u.ac.jp).

Many field-programmable gate array (FPGA)-based HFR vision systems have been developed for real-time video processing at hundreds of frames per second and greater [7]–[9]. Although HFR vision systems can simultaneously calculate scalar image features, they cannot transfer the entire image at high speed to a personal computer (PC) owing to the limitations in the transfer speed of its inner bus. We recently developed IDP Express [10], an HFR vision platform that can simultaneously process an HFR video and directly map it onto memory allocated in a PC. Implementing a real-time function to extract feature points in images and match their correspondences between frames on such an FPGA-based high-speed vision platform would make it possible to accelerate video mosaicing at a higher frame rate than 30 fps, even when the camera moves quickly.

In this study, we developed a real-time video mosaicing system that operates 512×512 images at 500 fps. The system can simultaneously extract and enable correspondence between feature points in images for estimation of frame-to-frame displacement, and stitch the captured images into a panoramic image to give a seamless wider field of view.

## II. Algorithm

Most video mosaicing algorithms are realized by executing the following processes [1]: (1) feature point detection, (2) feature point matching, (3) transform estimation, and (4) image composition. In many applications, image composition at dozens of frames per second is sufficient for the human eye to monitor a panoramic image. However, feature tracking at dozens of frames per second is not always accurate when the camera is moving rapidly, because most feature points are mistracked when image displacements between frames become large. In HFR vision, frame-to-frame displacement in feature points becomes considerably smaller. Narrowing the search range by assuming HFRs reduces the inaccuracy and computational load needed to match feature points between frames even when hundreds of feature points are observed from a rapidly moving camera, whereas image composition at an HFR requires a heavy computational load.

We introduce a real-time video mosaicing algorithm that can resolve this trade-off between tracking accuracy and computational load in video mosaicing for rapid camera motion. In our algorithm, steps (1)–(3) are executed at an HFR on an HFR vision platform, and step (4) is executed at dozens of frames per second to enable the human eye to monitor the panoramic image. In this study, we focus on acceleration of steps (1)–(3) for HFR feature point tracking

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

at hundreds of frames per second or more on the basis of the following concepts:

(a) Feature point detection accelerated by hardware logic

Generally, feature point detection requires pixel-level computation of the order $O(N^2)$, where $N^2$ is the image size. Gradient-based feature point detection is suitable for acceleration by hardware logic because the local calculation of brightness gradients can be easily parallelized as pixel-level computation. In this study, we accelerate gradient-based feature point detection by implementing its parallel circuit logic on an FPGA-based HFR vision platform.

(b) Reduction in the number of feature points

Thousands of feature points are not always required for estimating transform parameters between images in video mosaicing. The computational load can be reduced by selecting a smaller number of feature points because feature-level computation of the order $O(M^2)$ is generally required in feature point matching; $M$ is the number of selected feature points. In this study, the number of feature points used in feature point matching is reduced by excluding closely crowded feature points as they often generate corresponding errors between images in feature point matching.

(c) Narrowed search range by assuming HFR video

Feature point matching still requires heavy computation of the order $O(M^2)$ if it is required that all the feature points correspond to each other between frames, even when a reduced number of feature points are used. In an HFR video, it can be assumed that frame-to-frame image displacement grows considerably smaller, which allows a smaller search range to be used for points corresponding to points in the preceding frame. This narrowed search range reduces the computational load of feature point matching to corresponding feature points between frames on the order of $O(M)$.

Our algorithm for real-time HFR video mosaicing consists of the following processes:

(1) Feature point detection

(1-a) Feature extraction

To extract feature points such as upper-left vertexes, we use the following brightness gradient matrix $C(\boldsymbol{x},t)$:

$$C(\boldsymbol{x},t)=\sum_{\boldsymbol{x}\in N_a(\boldsymbol{x})}\begin{bmatrix} I_x'^2(\boldsymbol{x},t) & I_x'(\boldsymbol{x},t)I_y'(\boldsymbol{x},t) \\ I_x'(\boldsymbol{x},t)I_y'(\boldsymbol{x},t) & I_y'^2(\boldsymbol{x},t)\end{bmatrix}, \quad (1)$$

where $N_a(\boldsymbol{x})$ is the $a\times a$ adjacent area of pixel $\boldsymbol{x}=(x,y)$, and $I_x'(\boldsymbol{x},t)$ and $I_y'(\boldsymbol{x},t)$ indicate the following positive values of $I_x(\boldsymbol{x},t)$ and $I_y(\boldsymbol{x},t)$, respectively:

$$I_\xi'(\boldsymbol{x},t)=\begin{cases} I_\xi(\boldsymbol{x},t) & (I_\xi(\boldsymbol{x},t)>0) \\ 0 & (\text{otherwise})\end{cases} \quad (\xi=x,y), \quad (2)$$

where $I_x(\boldsymbol{x},t)$ and $I_y(\boldsymbol{x},t)$ are $x$ and $y$ differentials of the input image $I(\boldsymbol{x},t)$ at pixel $\boldsymbol{x}$ at time $t$.

$\lambda(\boldsymbol{x},t)$ is defined as a feature for feature tracking using the Harris corner detection [11] as follows:

$$\lambda(\boldsymbol{x},t)=\det\, C(\boldsymbol{x},t)-\kappa(\mathrm{Tr}\; C(\boldsymbol{x},t))^2. \quad (3)$$

Here, $\kappa$ is a tunable sensitive parameter, and values in the range 0.04–0.15 have been reported as feasible.

(1-b) Feature point detection

Thresholding is conducted for $\lambda(\boldsymbol{x},t)$ with a threshold $\lambda_T$ to obtain a map of feature points, $R(\boldsymbol{x},t)$, as follows:

$$R(\boldsymbol{x},t)=\begin{cases} 1 & (\lambda(\boldsymbol{x},t)>\lambda_T) \\ 0 & (\text{otherwise})\end{cases}. \quad (4)$$

The number of feature points when $R(\boldsymbol{x},t)=1$ is counted in the $p\times p$ adjacent area of $\boldsymbol{x}$ as follows:

$$P(\boldsymbol{x},t)=\sum_{\boldsymbol{x}'\in N_p(\boldsymbol{x})}R(\boldsymbol{x}',t), \quad (5)$$

where $P(\boldsymbol{x},t)$ indicates the density of the feature points.

(1-c) Selection of feature points

To reduce the number of feature points, closely crowded feature points are excluded by counting the number of feature points in the neighborhood. The reduced set of feature points $R'(t)$ is calculated at time $t$ as follows:

$$R'(t)=\{\boldsymbol{x}\mid P(\boldsymbol{x},t)\le P_0\}, \quad (6)$$

where $P_0$ is a threshold used to sparsely select feature points. We assume that the number of feature points is less than $M$.

(2) Feature point matching

(2-a) Template matching

To enable correspondence between feature points at the current time $t$ and those at the previous time $t_p=t-n\Delta t$, template matching is conducted for all the selected feature points in an image. $\Delta t$ is the shortest frame interval of a vision system, and $n$ is an integer. For template matching to enable the correspondence of the $i$-th feature point at time $t_p$ belonging to $R'(t_p)$, $\boldsymbol{x}_i(t_p)$ $(1\le i\le M)$, to the $i'$-th feature point at time $t$ belonging to $R'(t)$, $\boldsymbol{x}_{i'}(t)$ $(1\le i'\le M)$, the sum of absolute difference (SAD) is calculated as follows:

$$E(i',i;t,t_p)=\sum_{\boldsymbol{\xi}=(\xi,\eta)\in W_m}|I(\boldsymbol{x}_{i'}(t)+\boldsymbol{\xi},t)-I(\boldsymbol{x}_i(t_p)+\boldsymbol{\xi},t_p)|, \quad (7)$$

where $W_m$ is an $m\times m$ window of template matching.

To reduce the number of mismatched points, $\hat{\boldsymbol{x}}(\boldsymbol{x}_i(t_p);t)$, which indicates the feature point at time $t$ corresponding to the $i$-th feature point $\boldsymbol{x}_i(t_p)$ at time $t_p$, and $\hat{\boldsymbol{x}}(\boldsymbol{x}_{i'}(t);t_p)$, which indicates the feature point at time $t_p$ corresponding to the $i'$-th feature point $\boldsymbol{x}_{i'}(t)$ at time $t$, are bidirectionally searched by selecting the feature points when $E(i',i;t,t_p)$ is at a minimum in their adjacent areas as follows:

$$\hat{\boldsymbol{x}}(\boldsymbol{x}_i(t_p);t)=\boldsymbol{x}_{i'(i)}(t)=\underset{\boldsymbol{x}_{i'}(t)\in N_b(\boldsymbol{x}_i(t_p))}{\arg\min}\; E(i',i;t,t_p), \quad (8)$$

$$\hat{\boldsymbol{x}}(\boldsymbol{x}_{i'}(t);t_p)=\boldsymbol{x}_{i(i')}(t_p)=\underset{\boldsymbol{x}_i(t_p)\in N_b(\boldsymbol{x}_{i'}(t))}{\arg\min}\; E(i',i;t,t_p), \quad (9)$$

where $i'(i)$ and $i(i')$ are determined as the index numbers of the feature point at time $t$, corresponding to $\boldsymbol{x}_i(t_p)$, and the feature point at time $t_p$, corresponding to $\boldsymbol{x}_{i'}(t)$, respectively. The pair of feature points between time $t$ and $t_p$ are selected when the corresponding feature points are mutually selected as the same points as follows:

$$\tilde{\boldsymbol{x}}(\boldsymbol{x}_i(t_p);t)=\begin{cases}\hat{\boldsymbol{x}}(\boldsymbol{x}_i(t_p);t) & (i=i(i'(i))) \\ \emptyset & (\text{otherwise})\end{cases}, \quad (10)$$

where $\emptyset$ indicates that no feature point at time $t$ corresponds to the $i$-th feature point $\boldsymbol{x}_i(t_p)$ at time $t_p$, and the feature points at time $t$ that have no corresponding point at time $t_p$, are adjudged to be newly appearing feature points.

We assume that the image displacement between times $t$ and $t_p$ is small, and the feature point $\boldsymbol{x}_i(t)$ at time $t$ can be matched with a feature point at time $t_p$ in the $b \times b$ adjacent area of $\boldsymbol{x}_i(t)$. The processes described in Eqs. (8)–(10) are conducted for all the feature points belonging to $R'(t_p)$ and $R'(t)$, and we can reduce the computational load of feature point matching in the order of $O(M)$ by setting a narrowed search range with a small value for $b$.

(2-b) Frame interval selection

To avoid accumulated registration errors caused by small frame-to-frame displacements, the frame interval of feature point matching is adjusted for feature point matching at time $t + \Delta t$ using the following averaged distance among the corresponding feature points at times $t$ and $t_p$:

$$\bar{d}(t;t_p) = \frac{\sum_{\boldsymbol{x}_i(t_p) \in Q(t;t_p)} |\tilde{\boldsymbol{x}}(\boldsymbol{x}_i(t_p);t) - \boldsymbol{x}_i(t_p)|}{S(Q(t;t_p))}, \tag{11}$$

where $Q(t;t_p)$ is a set of the feature points $\boldsymbol{x}_i(t_p)$ at time $t_p$ that satisfy $\tilde{\boldsymbol{x}}(\boldsymbol{x}_i(t_p);t) \neq \emptyset$ $(i = 1, \cdots, M)$. $S(Q(t;t_p))$ is the number of elements belonging to $Q(t;t_p)$.

Depending on whether $\bar{d}(t;t_p)$ is larger than $d_0$, $n = n(t)$ is determined for feature point matching at time $t + \Delta t$; the frame interval of feature point matching is initially set to $\Delta t$.

(2-b-i) $\bar{d}(t;t_p) \leq d_0$,

The image displacement between times $t$ and $t_p = t_p(t)$ is too small for feature point matching. To increase the frame-to-frame displacement at the next time $t+\Delta t$, the parameter $n(t + \Delta t)$ is incremented by one as follows:

$$n(t + \Delta t) = n(t) + 1. \tag{12}$$

This indicates that the feature points at time $t + \Delta t$ are matched with those at time $t_p(t+\Delta t) = t-n(t)\Delta t$. Without conducting any process in steps (3) and (4) below, return to step (1) for the next time interval $t + \Delta t$.

(2-b-ii) $\bar{d}(t;t_p) > d_0$

To reset the frame-to-frame displacement at time $t + \Delta t$, $n(t + \Delta t)$ is set to one as follows:

$$n(t + \Delta t) = 1. \tag{13}$$

This indicates that the feature points at time $t + \Delta t$ are matched with those at time $t_p(t+\Delta t) = t$. Thereafter, go to steps (3) and (4); the selected pairs of feature points, $\boldsymbol{x}_i(t)$ $(\in Q(t;t_p))$ and $\tilde{\boldsymbol{x}}(\boldsymbol{x}_i(t);t_p)$, are used in steps (3) and (4).

(3) Transform estimation

Using the selected pairs of feature points, affine parameters between the two images at times $t$ and $t_p$ are estimated as follows:

$$\boldsymbol{x}_i^T(t) = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \tilde{\boldsymbol{x}}^T(\boldsymbol{x}_i(t);t_p) + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix}, \tag{14}$$

$$= \boldsymbol{A}(t;t_p)\tilde{\boldsymbol{x}}^T(\boldsymbol{x}_i(t);t_p) + \boldsymbol{b}^T(t;t_p), \tag{15}$$

where $a_j$ $(j = 1, \cdots, 6)$ are components of the transform matrix $\boldsymbol{A}(t;t_p)$ and translation vector $\boldsymbol{b}(t;t_p)$ that express the affine transform relationship between the two images at times $t$ and $t_p$. In this study, they are estimated by minimizing the following Tukey's biweight evaluation functions, $E_1$ and $E_2$:

$$E_l = \sum_{\boldsymbol{x}_i(t_p) \in Q(t;t_p)} w_{li} \cdot d_{li}^2 \quad (l = 1, 2). \tag{16}$$

Deviations $d_{1i}$ and $d_{2i}$ are given as follows:

$$d_{1i} = |x_i'' - (a_1 x_i' + a_2 y_i' + a_5)|, \tag{17}$$

$$d_{2i} = |y_i'' - (a_3 x_i' + a_4 y_i' + a_6)|, \tag{18}$$

where $\tilde{\boldsymbol{x}}(\boldsymbol{x}_i(t);t_p) = (x_i', y_i')$ and $\boldsymbol{x}_i''(t) = (x_i'', y_i'')$ indicate the temporally estimated location of the $i$-th feature point at time $t$ in Tukey's biweight method. The following processes are iteratively executed to reduce estimation errors caused by distantly mismatched pairs of feature points. Here, the weights $w_{1i}$ and $w_{2i}$ are set to one and $\boldsymbol{x}_i''(t)$ set to $\boldsymbol{x}_i(t)$ as their initial values.

(3-a) Affine parameter estimation

On the basis of the least squares method to minimize $E_1$ and $E_2$, affine parameters $a_j$ $(j = 1, \cdots, 6)$ are estimated by calculating the weighted product sums of the $xy$ coordinates of the selected feature points as follows:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_5 \end{bmatrix} = \begin{bmatrix} \sum_i w_{1i} x_i'^2 & \sum_i w_{1i} x_i' y_i' & \sum_i w_{1i} x_i' \\ \sum_i w_{1i} x_i' y_i' & \sum_i w_{1i} y_i'^2 & \sum_i w_{1i} y_i' \\ \sum_i w_{1i} x_i' & \sum_i w_{1i} y_i' & \sum_i w_{1i} \end{bmatrix}^{-1} \begin{bmatrix} \sum_i w_{1i} x_i'' x_i' \\ \sum_i w_{1i} x_i'' y_i' \\ \sum_i w_{1i} x_i'' \end{bmatrix}, \tag{19}$$

$$\begin{bmatrix} a_3 \\ a_4 \\ a_6 \end{bmatrix} = \begin{bmatrix} \sum_i w_{2i} x_i'^2 & \sum_i w_{2i} x_i' y_i' & \sum_i w_{2i} x_i' \\ \sum_i w_{2i} x_i' y_i' & \sum_i w_{2i} y_i'^2 & \sum_i w_{2i} y_i' \\ \sum_i w_{2i} x_i' & \sum_i w_{2i} y_i' & \sum_i w_{2i} \end{bmatrix}^{-1} \begin{bmatrix} \sum_i w_{2i} y_i'' x_i' \\ \sum_i w_{2i} y_i'' y_i' \\ \sum_i w_{2i} y_i'' \end{bmatrix}. \tag{20}$$

Using the affine parameters estimated in Eqs. (19) and (20), the temporally estimated locations of feature points can be updated as follows:

$$\boldsymbol{x}_i''^T(t) = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \tilde{\boldsymbol{x}}^T(\boldsymbol{x}_i(t);t_p) + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix}. \tag{21}$$

(3-b) Updating weights

To reduce the errors caused by distantly mismatched pairs of feature points, $w_{1i}$ and $w_{2i}$ are updated for the $i$-th pair of feature points using the deviations $d_{1i}$ and $d_{2i}$ as follows:

$$\begin{bmatrix} w_{1i} \\ w_{2i} \end{bmatrix} = \begin{cases} \begin{bmatrix} \left(1 - \frac{d_{1i}^2}{W_u^2}\right)^2 \\ \left(1 - \frac{d_{2i}^2}{W_u^2}\right)^2 \end{bmatrix} & (d_{1i} \leq W_u, d_{2i} \leq W_u) \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{(otherwise)} \end{cases}. \tag{22}$$

$W_u$ is a parameter used to determine the cut point for evaluation functions at $u$ time iteration. As the number of

iterations $u$ is larger, $W_u$ is set to a smaller value. Increment $u$ by one and return to step (3-a) during $u \leq U$.

After $U$ time iteration, the affine parameters between two input images at times $t$ and $t_p$ are determined, and the affine transform matrix and vector are obtained. $\boldsymbol{A}(t;t_p)$ and $\boldsymbol{b}(t;t_p)$ are accumulated with the affine parameters estimated at time $t_p$ as follows:

$$\boldsymbol{A}(t) = \boldsymbol{A}(t;t_p)\boldsymbol{A}(t_p), \tag{23}$$

$$\boldsymbol{b}^T(t) = \boldsymbol{A}(t_p)\boldsymbol{b}^T(t;t_p) + \boldsymbol{b}^T(t_p), \tag{24}$$

where $\boldsymbol{A}(t)$ and $\boldsymbol{b}(t)$ indicate the affine parameters of the input image at time $t$, compared with the input image at time 0; $\boldsymbol{A}(0)$ and $\boldsymbol{b}(0)$ are initially given as the unit matrix and zero vector, respectively.

(4) Image composition

To monitor the panoramic image at intervals of $\Delta t'$, the panoramic image $G(\boldsymbol{x},t)$ is updated by attaching an affine-transformed input image at time $t$ to the panoramic image $G(\boldsymbol{x},t-\Delta t')$ at time $t-\Delta t'$ as follows:

$$G(\boldsymbol{x},t) = \begin{cases} I(\boldsymbol{x}',t) & (\text{if } I(\boldsymbol{x}',t) \neq \emptyset) \\ G(\boldsymbol{x},t-\Delta t') & (\text{otherwise}) \end{cases}, \tag{25}$$

where the panoramic image $G(\boldsymbol{x},t)$ is initially given as an empty image at time 0; $\boldsymbol{x}'$ indicates the affine-transformed coordinate vector at time $t$ as follows:

$$\boldsymbol{x}'^T = \boldsymbol{A}(t)\boldsymbol{x}^T + \boldsymbol{b}^T(t). \tag{26}$$

Our video mosaicing algorithm can be efficiently executed as a multi-rate video processing method. The interval in steps (1) and (2) can be set to the shortest frame interval $\Delta t$ for accurate feature point tracking in HFR vision, while that in step (3) depends on the frame-to-frame image displacement; it is the shortest frame interval when there is significant camera motion, and it becomes larger when the camera motion becomes smaller. On the other hand, the interval in step (4), $\Delta t'$, can be independently set to dozens of millisecond to enable the human eye to monitor the panoramic image.

## III. System Implementation

### A. High-speed vision platform: IDP Express

To realize real-time video mosaicing at an HFR, our video-mosaicing method was implemented on the high-speed vision platform IDP Express [10]. The implemented system consisted of a camera head, a dedicated FPGA board (IDP Express board), and a PC. The camera head was highly compact and could easily be held by a human hand. Its dimensions and weight were 23×23×77 mm and 145 g, respectively. On the camera head, eight-bit RGB images of 512×512 pixels could be captured at 2000 fps with a Bayer filter on its CMOS image sensor. The IDP Express board is designed for high-speed processing and recording of 512×512 images transferred from the camera head at 2000 fps. On the IDP Express board, hardware logic can be implemented on a user-specified FPGA (Xilinx XC3S5000-4FG900) for acceleration of image processing algorithms. The 512×512 input images, and the processed results on the IDP Express board, can be memory-mapped via 16-lane PCI-e buses in real time at 2000 fps onto the allocated memories in the PC, and arbitrary processing for the execution of software on the PC can be programmed with an ASUSTek P5E mainboard, Core 2 Quad9300 bulk 2.50 GHz CPU, 4 GB memory, and a Windows XP Professional 32-bit OS.

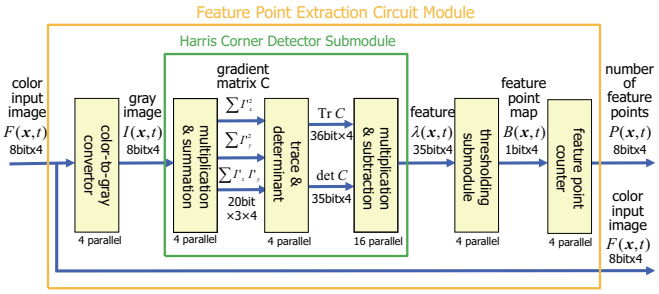


Fig. 1. Schematic data flow of a feature point extraction circuit module.

### B. Implemented hardware logic

To accelerate steps (1-a) and (1-b) in our video mosaicing method, we designed a feature point extraction circuit module in the user-specific FPGA on the IDP Express; steps (1-a) and (1-b) have a computational complexity of $O(N^2)$. The other processes (steps (1-c), (2), (3), and (4)) were implemented in software on the PC because, with their computational complexity of $O(M)$, they can be accelerated by reducing the number of feature points.

Figure 1 is a schematic data flow of the feature point extraction circuit module implemented in the FPGA on the IDP Express board. It consists of a color-to-gray converter submodule, a Harris feature extractor submodule, and a feature point counter submodule. Raw eight-bit 512×512 input color images with a Bayer filter, $F(\boldsymbol{x},t)$, are scanned in units of four pixels from the upper left to the lower right using $X$ and $Y$ signals at 151.2 MHz. The color-to-gray converter submodule converts RGB images into eight-bit gray-level images $I(\boldsymbol{x},t)$ in parallel with the four pixels after RGB conversion for input images with a Bayer filter.

In the Harris feature detector submodule, $I'_x$ and $I'_y$ are calculated using 3×3 Prewitt operators, and $\sum I'^2_x$, $\sum I'_x I'_y$, and $\sum I'^2_y$ in the adjacent 3×3 pixels ($a = 3$) are calculated as 20-bit data using 160 adders and 12 multipliers in parallel in units of four pixels at 151.2 MHz. The feature $\lambda(\boldsymbol{x},t)$ is calculated as 35-bit data by subtracting det $C(\boldsymbol{x},t)$ with a three-bit shift value of $(\text{Tr } C(\boldsymbol{x},t))^2$ in units of four pixels when $\kappa$ = 0.0625; 16 multipliers, 24 adders, and four three-bit shifters are implemented for calculating $\lambda(\boldsymbol{x},t)$.

The feature point counter submodule can obtain the positions of feature points as a 512×512 binary map $B(\boldsymbol{x},t)$ by thresholding $\lambda(\boldsymbol{x},t)$ with a threshold $\lambda_T$ in parallel in units of four pixels at 151.2 MHz; $B(\boldsymbol{x},t)$ indicates whether a feature point is located at pixel $\boldsymbol{x}$ at time $t$. The number of feature points in the 5×5 adjacent area ($p = 5$) is counted for all the feature points as a 512×512 map $P(\boldsymbol{x},t)$ using 96 adders in parallel in units of four pixels; $P(\boldsymbol{x},t)$ is used for checking closely crowded feature points in step (1-c).

The color input image $F(\boldsymbol{x}, t)$ and the number of feature points $P(\boldsymbol{x}, t)$ are outputted to FIFO memory for an external PC with $X$ and $Y$ signals. The delay time in outputting $F(\boldsymbol{x}, t)$ and $P(\boldsymbol{x}, t)$ using the feature point extraction circuit module is 45 clocks (1 clock = 13.2 ns) after the raster scanning has been performed, while the delay time in outputting them to the PC is one frame (0.5 ms).

### C. Specification

We implemented steps (1-a) and (1-b) of the above circuit module in the FPGA (Xilinx XC3S5000-4FGG900) on the IDP Express board. The resources consumed by the FPGA are listed in Table I. The extracted feature points can be outputted to the PC for 512×512 images at 2000 fps.

Steps (1-c), (2), (3), and (4) were implemented in software on the PC, and the following parameters were used in this study. The threshold $P_0$ in step (1-c) was determined in order to reduce the number of feature points ($M \leq 300$), depending on the experimental scene. Step (2-a) executed $5 \times 5$ ($m = 5$) template matching with bidirectional search in the $31 \times 31$ adjacent area ($b = 31$). In step (2-b), the frame interval of feature point matching was determined with $d_0 = 7$. Step (3) executed Tukey's biweight method with $L = 10$ iteration; the parameters for the biweight evaluation functions were set at $W_l = 11 - l$ ($l = 1, \cdots, 10$).

Table II summarizes the execution time taken for video mosaicing. The execution times of steps (1-a) and (1-b) include the image acquisition time for a 512×512 image on the FPGA board. The total execution time of steps (1), (2), and (3) is less than $\Delta t = 2$ ms; it was worst when step (3) was executed at all the frames. The execution time of step (4) was much larger than that of the other steps. Step (4) was implemented as a multi-threaded process that was not able to disturb the real-time processes of the other steps when its interval was set to $\Delta t' = 60$ ms for monitoring by the human eye. We confirmed that video mosaicing can be executed for 512×512 images in real time at 500 fps, including feature point extraction and tracking for $M \leq 300$.

TABLE I

FPGA RESOURCE CONSUMPTION.

| Device Type | Xilinx XC3S5000 |
|---|---|
| Slice | 5,864/33,280 (17%) |
| Slice Flip Flop | 9,015/66,560 (13%) |
| 4 input LUT | 6,038/66,560 (9%) |
| Bounded IOB | 195/633 (30%) |
| Block RAM | 15/104 (14%) |
| MULT18X18s | 28/104 (26%) |
| GCLK | 2/8 (25%) |

TABLE II

EXECUTION TIMES.

| | time [ms] |
|---|---|
| (1-a,1-b) Feature extraction / Thresholding@ | 0.86 |
| (1-c) Selection of feature points | 0.05 |
| (2-a) Template matching | 0.97 |
| (2-b) Frame interval selection | 0.01 |
| (3) Affine transform estimation | 0.09 |
| (4) Image composition | 51.43 |
| Total ((1)–(3)) | 1.98 |

Fig. 2. Experimental scene.

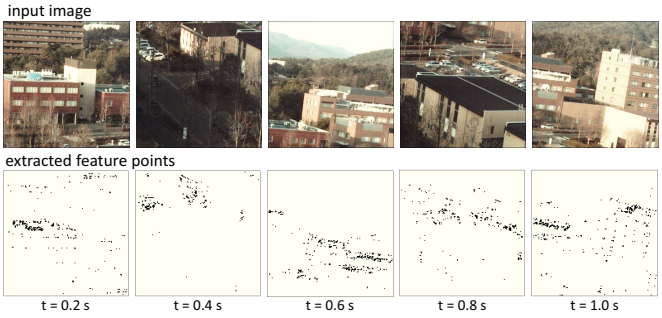


Fig. 3. Input images and extracted feature points.

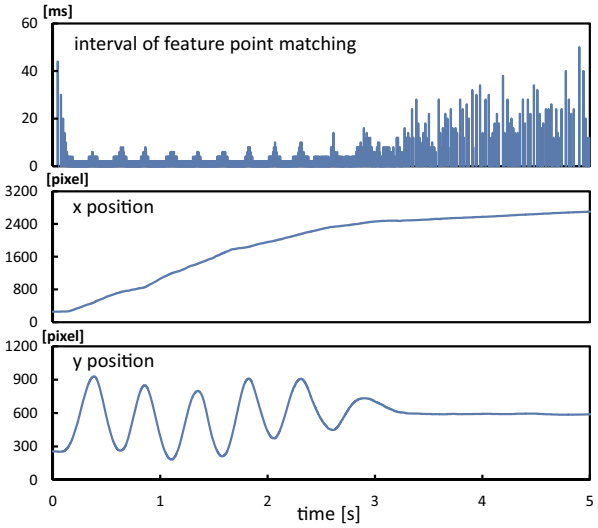


Fig. 4. Frame interval of feature point matching, and $xy$ coordinates of translation vector.

## IV. EXPERIMENT

In this section, we present the experimental results of real-time HFR video mosaicing for an outdoor scene captured using a camera head moved quickly and periodically by a human hand. Figure 2 shows the experimental outdoor scene. It was captured from the top of an eight-story building in Hiroshima University by an operator who manually moved the hand-held camera head with his hand from left to right with periodic up-and-down motions approximately five times per three second, and then decelerated the camera's motion. The frame rate and exposure time of the IDP Express system were set to 500 fps and 0.25 ms, respectively. To select 300 feature points or less from a 512×512 image, the threshold parameters were set to $\lambda_T = 5 \times 10^7$ and $P_0 = 25$.

Figure 3 shows the five-input image sequence and the extracted feature points, taken at intervals of 0.2 s. The affine

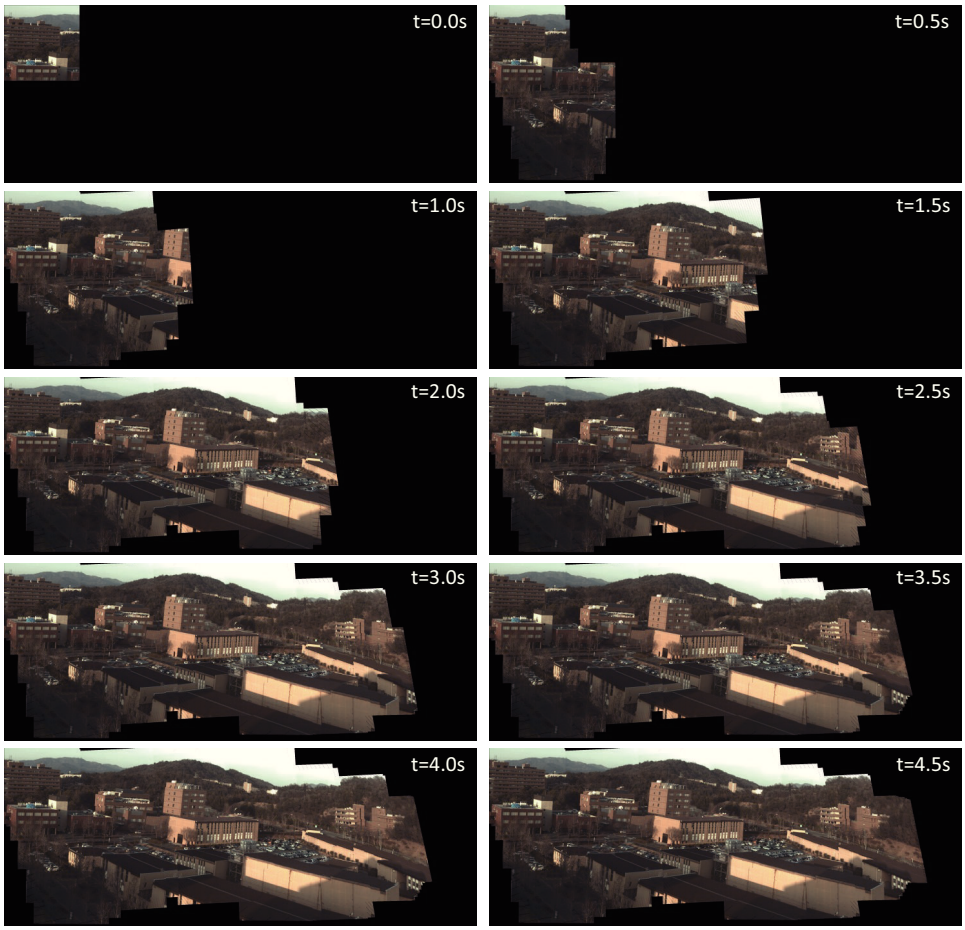


Fig. 5. Synthesized panoramic images

matrix and translation vector were set as unit matrix and zero vector, respectively, at $t = 0$. It can be seen that most of the feature points in the input images were correctly extracted. Figure 4 shows the frame interval of feature point matching, and the $x$- and $y$-coordinates for translation vector $\boldsymbol{b}(t)$ for $t = 0.0$–$5.0$ s. Corresponding to the camera's motion, it can be seen that a short frame interval was set for the high-speed scene, and a long frame interval for the low-speed scene.

Figure 5 shows the synthesized panoramic images of $3200\times1200$ pixels, taken at intervals of 0.5 s. With time, the panoramic image was extended by accurately stitching affine transformed input images over the panoramic image at the previous frame, and large buildings and background forests on the far side were observed in a single panoramic image at $t = 4.5$ s. These figures indicate that our system can accurately generate a single synthesized image in real time by stitching $512\times512$ input images at 500 fps when the camera head is quickly moved by a human hand.

## V. Conclusion

We developed a high-speed video mosaicing system that can stitch $512\times512$ images at 500 fps to give a single synthesized panoramic image in real time. On the basis of the experimental results, we plan to improve our video mosaicing system for more robust and long-time video mosaicing, and to extend the application of this system to video-based SLAM and other surveillance technologies in the real world.

## References

[1] B. Zitova and J. Flusser, "Image registration methods: A survey," *Image Vis. Comput.*, **21**, 977–1000, 2003.
[2] J. Shi and C. Tomasi, "Good features to track," *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 593-600, 1994.
[3] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, **60**, 91–110, 2004.
[4] M. Kourogi, T. Kurata, J. Hoshino, and Y. Muraoka, "Real-time image mosaicing from a video sequence," *Proc. Int. Conf. Image Proc.*, 133–137, 1999.
[5] J. Civera, A.J. Davison, J.A. Magallon, and J.M.M. Montiel, "Drift-free real-time sequential mosaicing," *Int. J. Comput. Vis.*, **81**, 128–137, 2009.
[6] R.H.C. de Souza, M. Okutomi, and A. Torii, "Real-time image mosaicing using non-rigid registration," *Proc. 5th Pacific Rim Conf. on Advances in Image and Video Technology*, 311–322, 2011.
[7] Y. Watanabe, T. Komuro, and M. Ishikawa, "955-fps real-time shape measurement of a moving/deforming object using high-speed vision for numerous-point analysis," *Proc. IEEE Int. Conf. Robot. Autom.*, 3192–3197, 2007.
[8] S. Hirai, M. Zakoji, A. Masubuchi, and T. Tsuboi, "Realtime FPGA-based vision system," *J. Robot. Mechat.*, **17**, 401–409, 2005.
[9] I. Ishii, R. Sukenobe, T. Taniguchi, and K. Yamamoto, "Development of high-speed and real-time vision platform, $H^3$ Vision," *Proc. IEEE/RSJ Int. Conf. Intelli. Rob. Sys.*, 3671–3678, 2009.
[10] I. Ishii, T. Tatebe, Q. Gu, Y. Moriue, T. Takaki, and K. Tajima, "2000 fps real-time vision system with high-frame-rate video recording," *Proc. IEEE Int. Conf. Robot. Autom.*, 1536–1541, 2010.
[11] C. Harris and M. Stephens, "A combined corner and edge detector," *Proc. the 4th Alvey Vis. Conf.*, 147–151, 1988.