ALGORITHMS, PROTOCOLS AND SYSTEMS FOR REMOTE OBSERVATION USING NETWORKED ROBOTIC CAMERAS

A Dissertation

by

NI QIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2008

Major Subject: Computer Science

ALGORITHMS, PROTOCOLS AND SYSTEMS FOR REMOTE OBSERVATION USING NETWORKED ROBOTIC CAMERAS

A Dissertation

by

NI QIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Dezhen Song
Committee Members,	Ricardo Gutierrez-Osuna
	Dmitri Loguinov
	Ergun Akleman
Head of Department,	Valerie E. Taylor

May 2008

Major Subject: Computer Science

ABSTRACT

Algorithms, Protocols and Systems for Remote Observation Using Networked Robotic Cameras. (May 2008)

Ni Qin,

B.S., Wuhan University; M.C.S., University of Mississippi Medical Center;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Dezhen Song

Emerging advances in robotic cameras, long-range wireless networking, and distributed sensors make feasible a new class of hybrid teleoperated/autonomous robotic remote "observatories" that can allow groups of peoples, via the Internet, to observe, record, and index detailed activity occurred in remote site. Equipped with robotic pan-tilt actuation mechanisms and a high-zoom lens, the camera can cover a large region with very high spatial resolution and allows for observation at a distance. High resolution motion panorama is the most nature data representation. We develop algorithms and protocols for high resolution motion panorama. We discover and prove the projection invariance and achieve real time image alignment. We propose a minimum variance based incremental frame alignment algorithm to minimize the accumulation of alignment error in incremental image alignment and ensure the quality of the panorama video over the long run. We propose a Frame Graph based panorama documentation algorithm to manage the large scale data involved in the online panorama video documentation. We propose a on-demand high resolution panorama video-streaming system that allows on-demand sharing of a high-resolution motion panorama and efficiently deals with multiple concurrent spatial-temporal user requests. In conclusion, our research work on high resolution motion panorama have

significantly improve the efficiency and accuracy of image alignment, panorama video quality, data organization, and data storage and retrieving in remote observation using networked robotic cameras.

To my lovely cat Maomao, my dear family Yu and Amy, and my parents

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Dezhen Song, and my committee members, Dr. Gutierrez-Osuna, Dr. Loguinov, and Dr. Akleman, for their guidance and support throughout the course of this research. Thanks to K. Goldberg for his insightful discussions.

Thanks to my colleagues Q. Hu, H. Lee, Y. Xu, Z. Goodwin, B. Green, and C. Kim for giving me a great time in Networked Robots Lab. Thanks also to my friends and the department faculty and staff for making my time at Texas A&M University a great experience. I also want to extend my gratitude to the National Education Foundation for providing funding support and Panasonic research for providing equipments.

Thanks to J. Fitzpatrick and R. Rohrbaugh of the Cornell Ornithology Lab, and D. Luneau of the University of Arkansas at Little Rock. Thanks to Richard Crosset, Billy Culbreath, and the support from Arkansas Electric Company, U.S. Fish and Wildlife Service. Thanks to Robert Johnson and the Arkansas Electric Company. Thanks to Patsy Arnett and the support from Brinkley Convention Center, and to Mary Harlin and her family for providing space for our wireless antenna in Arkansas.

Finally, special thanks to my lovely cat Maomao for accompanying me through my toughest time and to my husband Yu, my daughter Amy, and my parents.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Networked robotic camera become more and more popular in remote observation applications such as natural observation, surveillance, and distance learning. Consider the study of penguins in Antarctica, bears in Alaska, ants in redwood canopies, or lizards in Peruvian forests. Scientific study of animals in site which requires vigilant observation of detailed animal behavior over weeks or months. When animals live in remote and/or inhospitable locations, observation can be an arduous, expensive, dangerous, and lonely experience for scientists. Furthermore, human intervention can disturb animal behavior. Another example is the construction of large buildings and structures such as bridges, which involves a complex and highly precise sequence of operations. Small errors in alignment, reinforcement, or materials can result in extremely costly repairs or catastrophic failures. Regular inspection and documentation are well-established aspects of construction practice but may not be feasible when construction is performed in remote and dangerous environments. When animal observatories or construction site are far away from network infrastructure, they can only be accessed via long distance wireless communication with limited bandwidth. A low cost, low bandwidth, and energy-efficient solution is to use tele-operated robotic video cameras.

This dissertation follows the style of *IEEE Transactions on Automatic Control*.

Remote observation using camera systems have a long history. In 1950s, Gysel and Davis [1] built an early video camera based on remote wildlife observation system to study rodents. Biologists use remote photography systems to observe nest predation, feeding behavior, species presence, and population parameters [2–7]. Commercial remote camera systems such as Trialmaster [2] and DeerCam have been developed since 1986 and have been widely used in wildlife observation. The Internet enables webcam systems that allow the general public to access remote nature cameras. Thousands of webcams have been installed around the world, for example to observe elephants [8], tigers [9], bugs [10], birds/squirrels [11] [12], cranes [13], and swans [14]. Many other examples can be found at [15]. A more extreme remote observation system is the Virtual Planetary Exploration (VPE) project [16] operated in NASA in 1992. VPE project generated panorama view of Mars by mosaicing digital terrain data obtained from NASA's Viking orbiter satellites. It took about 10 minutes to generate 360 degree 6000x2000 panorama on Stardent GS2000 computer.

A remote observation system usually contains five main components – video acquisition system, data transmission system, data archiving and retrieve system, and intellectual data analysis system. Video acquisition system collects the live video data from the field. To fit the bandwidth constraint and satisfy the responsiveness requirement for remote observation, the video camera usually transmits a low resolution video (i.e. $\leq 640 \times 480$ pixels) with live frame rate (i.e. > 30 frames per second). Therefore, it suffers from a limited field of view, which becomes even worse when camera has zoom ability and operates at high zoom levels. An example is the Panasonic HCM 280. When set at a 22x zoom, this camera only covers 2.8° in its horizontal field of view, which looses context of the observed animal behavior. Presenting only $\leq 640 \times 480$ video stream can not provide enough context about the observed environment. Other video cameras such as wide angle camera [17, 18] and

polycameras [19] are able to provide large field of view. However they either suffer from high bandwidth consumption or low resolution. Due to bandwidth constraint and large volume of video data, efficient data representation and transmission protocol are the key factors for data transmission system. A remote observation system equipped with motion detection system can detect motion and transmit/record video data containing motion object only, which dramatically reduces the data volume. A intelligent observation system can do more than detecting moving object and is able to provide more understanding about the motion objects captured in the video.

Emerging advances in robotic cameras, long-range wireless networking, and distributed sensors make feasible a new class of hybrid teleoperated/autonomous robotic remote "observatories" that can allow groups of peoples, via the internet, to remotely observe, records, and index detailed activity occurred in remote site. Consider a high-resolution pan-tilt-zoom camera installed in a deep forest. Connected to the Internet through a long-range wireless network, the robotic camera allows scientists and/or the general public to observe nature remotely. Equipped with robotic pan-tilt actuation mechanisms and a high-zoom lens, the camera can cover a large region with very high spatial resolution and allows for observation at a distance. For example, a Panasonic HCM 280A pan-tilt-zoom camera has a 22x motorized optical zoom, a 350° pan range, and a 120° tilt range. It can reach a spatial resolution of 500 megapixel per steradian at its highest zoom level. The full coverage of the viewable region is more than 3 gigapixels if represented as a motion panorama.

In our remote observation system using networked robotic cameras as showing in figure 1, cameras are installed in remote observatory and connected to the Internet

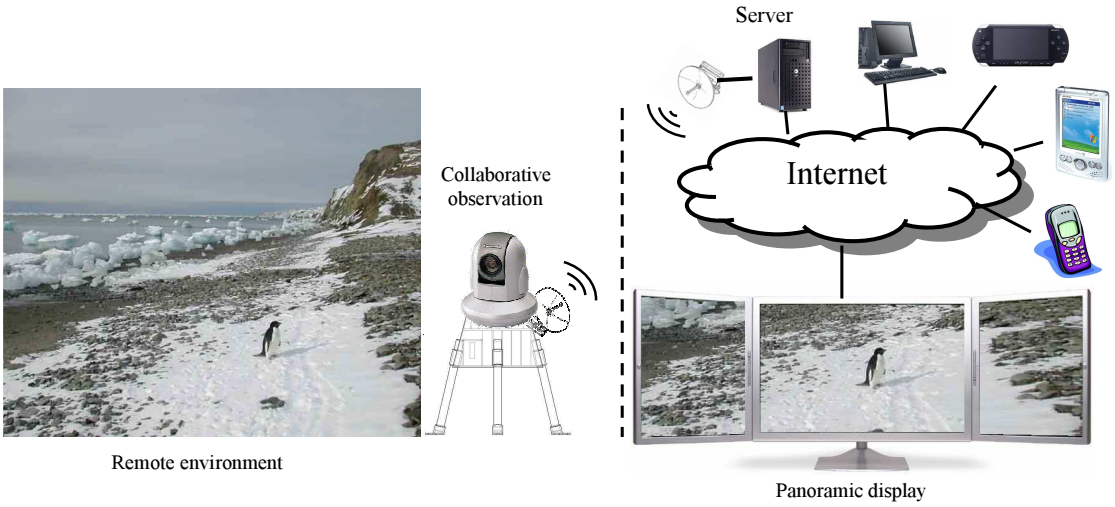


Fig. 1. Architecture and user interface for remote observation using networked robotic cameras.

through a long-range wireless network. Any user with Internet connection can access the system remotely. Users log on to a web host system and send their queries to a server. The server directly connects to the camera, controls the cameras and collects the video data. Since the camera cannot provide the concurrent coverage of the entire viewable region due to its limited field of view and the limited number of pixels in its CCD sensor, networked robotic camera is frequently steered to different pan/tilt position to inspect and document activities on site from the command inputs coming from preset command sets, human inspector commands, and on-site motion detectors.

As illustrated in Figure 2, the system architecture is consisted of three parts – camera control part, documentation part, and on-demand content delivery part. Camera is frequently steered to different pan/tilt position to inspect and document activities on site from the command inputs coming from preset command sets, human inspector commands, and on-site motion detectors. The programmable preset features ensure that the camera periodically patrols and searches for interesting regions. It includes two type of camera control commands: fixed locations and particular fea-

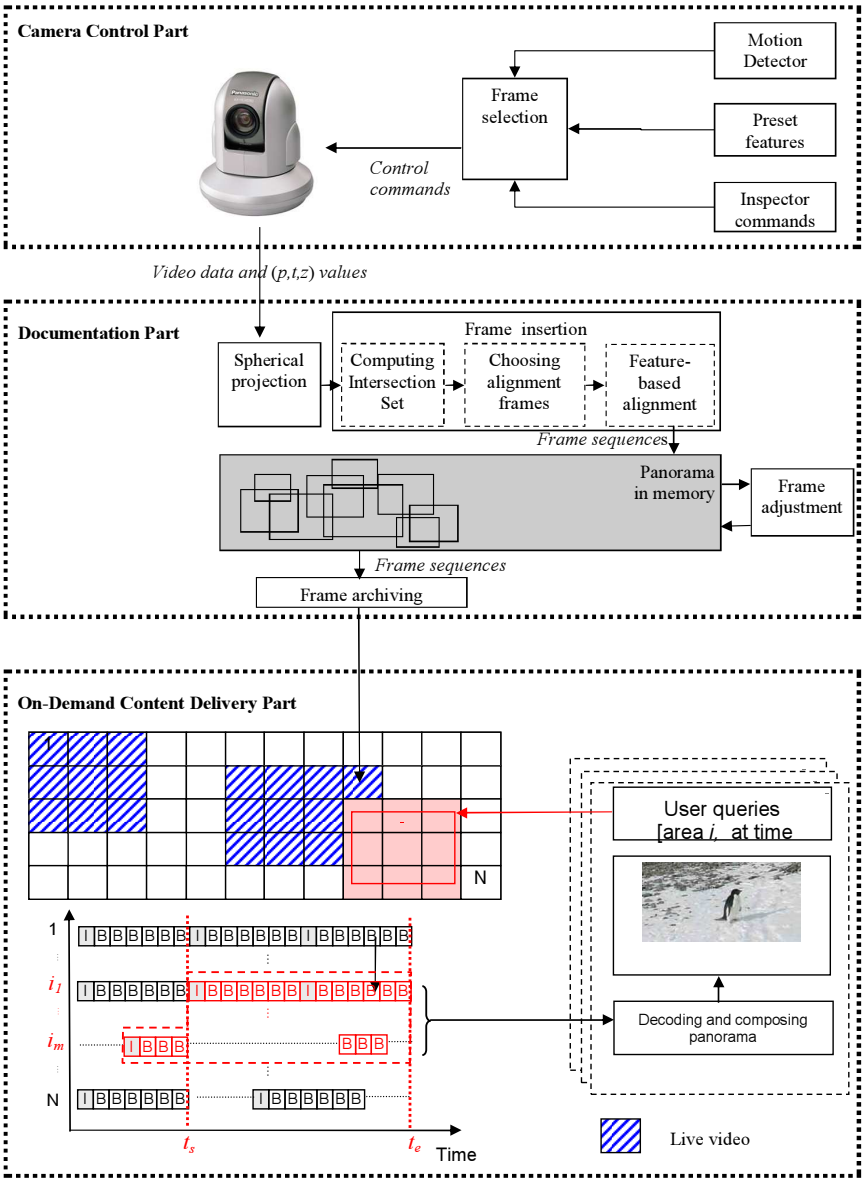


Fig. 2. System diagram. The system is consisted of three parts. The top part is camera control part. Camera motion is determined by a combination of preset points, human inspector commands, and motion detector inputs. The middle part is documentation part. The resulting video sequences are aligned and inserted into the evolving panorama video at real time. The lower part is on-demand content delivery part. The grid in the figure represents a patch-based high-resolution panorama video system that allows multiple users to query different part of video concurrently. I's and B's indicate the I-frame and the B-frame used in MPEG-2 compression.

tures. The former are good for a complete coverage of the known and fixed locations of the observation site whereas the later are good for the known and dynamic points of interest. Sporadic motions are captured by motion detectors, which also generate camera control commands. The motion detectors could be real pyroelectric sensors that are installed in the scene or just a motion detector built on image analysis [20]. Inspectors may also want to control the camera directly from time to time. With the highest priority, the inspector commands can always overrule autonomous commands from preset features and motion detectors. The priority sequence for the three types of commands is also configurable. Weighted by their priorities, commands are feeded into a frame selection module. Using the method in [21, 22], the frame selection module generates a single camera control command based on priority, geometric relationship between different commands, and previous camera visits.

With high field of view coverage capability offered by robotic cameras, high resolution panorama is the most nature data presentation. The "foveal" video images are aligned and inserted into a coherent panoramic display. Our user interface consists of two parts: a static background panorama that covers the user query region and a video segment superimposed on top of the background panorama if there are video data collected for the queried time duration. With high zoom and large pan/tilt capabilities, remote observation system equipped with robotic cameras is able to achieve giga-pixel resolution. Each user may want to observe a different sub region and time window of the panorama video. On the other hand, users might use low-power devices such as PDAs or cell phones, which do not have the computation power to perform expensive image alignment and panorama construction computation. The server should perform as much computation in generating and delivering panorama video as possible.

We want to seamlessly merge the live low resolution video frames into a high

resolution panoramic video as camera patrols in real time. Due to the errors introduced by camera potentiometer readings, merging video frames must be based on fast image registration under a fraction of a second. For example, an error of 0.5° in camera tilt position can cause a 41.67% error in coverage when a Panasonic HCM 280 camera operates at its highest zoom. Furthermore, camera mechanical errors may deteriorate with the possible influence of temperature, moisture, or other factors after initial setup. Existing image alignment algorithms take seconds to align a single image. Therefore a real time automatic image registration must be performed during the whole video acquisition process.

It is often the case that multiple users including nature scientists and the general public want to share the panorama video output at the same time. Transmitting the full-sized ever-changing giga-pixel panorama video to every user is unnecessary and expensive in bandwidth requirement. Each user may want to observe a different sub region and time window of the panorama video. For example, an ornithologist is often interested in bird video data early in the morning when the camera is aimed at the top of the forest. Therefore, depending on user queries and camera configurations, the server may transmit different contents to a user such as a pre-stored video segment, a high-resolution static image with the timestamp closest to the request time window, or a live video from the camera. A on-demand high resolution panorama video streaming protocol able to handle multiple concurrent spatial-temporal user requests is desired.

A sustained research effort is required to understand fundamental computational questions and develop the necessary IT infrastructure required for remote observation system. We focuses on exploring the fundamental computational questions, building prototype, and developing systems, protocols and algorithms for remote observation using networked robotic cameras.

CHAPTER II

HIGH RESOLUTION MOTION PANORAMA

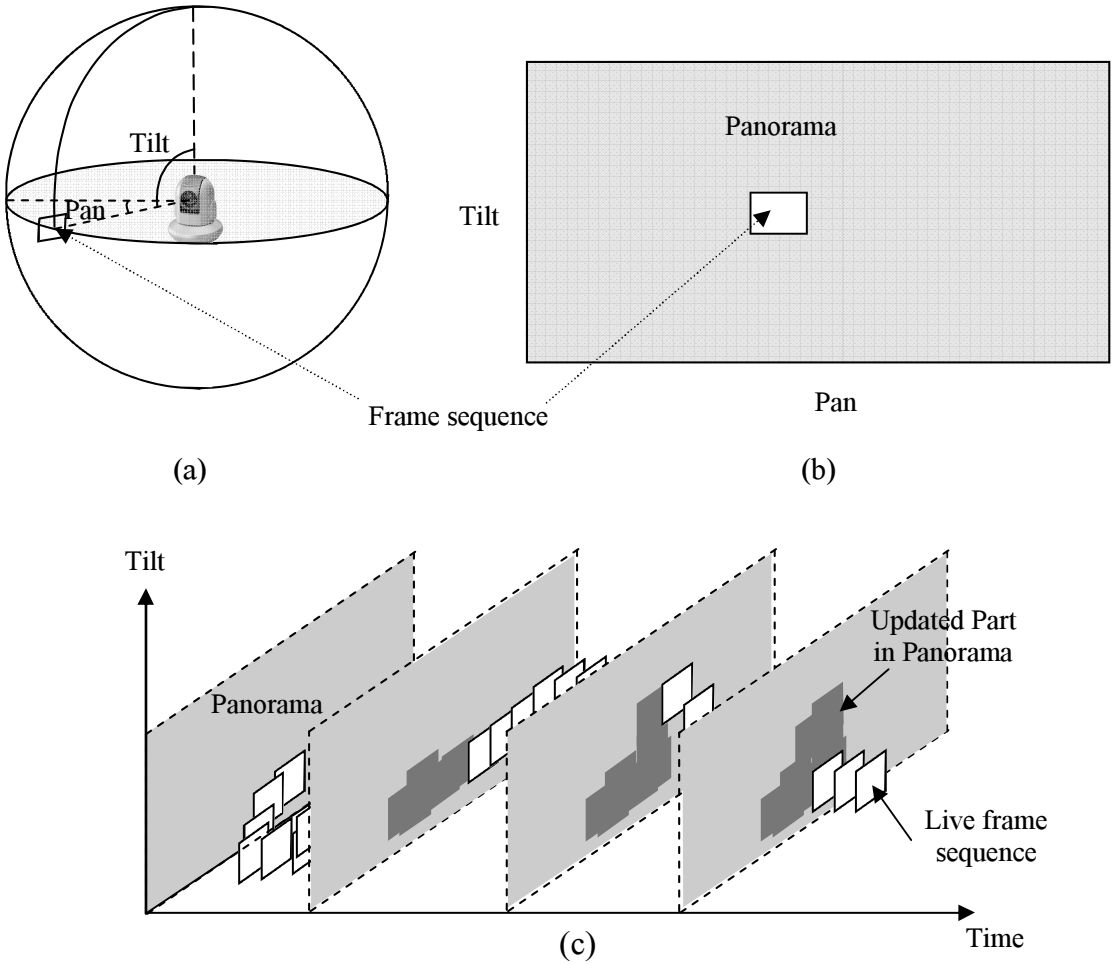


Fig. 3. A network robotic camera provides an evolving high-resolution panoramic display of the remote environment. (a) Camera and spherical field of view, (b) Current video image in context of planar panoramic display, (c) Time sequence of video images and evolving panoramic display.

In remote observation using networked robotic camera as illustrated in Figure 2, camera is frequently steered to different pan/tilt position to inspect and document

activities on site from the command inputs coming from preset command sets, human inspector commands, and on-site motion detectors.

With high field of view coverage capability offered by robotic cameras, high resolution panorama is most nature data presentation. The "foveal" video images are aligned and inserted into a coherent panoramic display. As in Figure 3, our user interface consists of two parts: a static background panorama that covers the user query region and a video segment superimposed on top of the background panorama if there are video data collected for the queried time duration. At the same time, the system updates panorama and documents frame sequences. Both video data and camera pan-tilt-zoom values are transmitted to our system. Frame sequences are generated by projecting video frames onto a spherical surface for alignment. The up-to-date part of the evolving panorama is stored in memory for real time display and image alignment and the historical part of the evolving panorama is stored in hard disk. We want to seamlessly merge the live low resolution video frames into a high resolution motion panorama as camera patrols in real time.

Merging video frames into the panoramic video must be based on image registration. Existing image registration algorithms take seconds to align a single image, which does not satisfy the system responsiveness requirement. In chapter III, we explore image re-projection problems in a spherical coordinate system and discover projection invariant properties on spherical surface and achieve real time image alignment to make it feasible to construction live high resolution motion panorama.

With camera frequently steered to different pan/titlt position, image alignment must be performed incrementally on the incoming video to generate motion panorama. Alignment errors can gets accumulated and propagated during motion panorama construction and update. In chapter IV, We proposed a minimum variance based incremental frame alignment algorithm that traces image location error variance density

to optimally estimate the extrinsic projection parameters for a newly arrived camera frame to ensure the quality of the panorama video over long run.

With a new video frame arrives, it is aligned with an optimal set of neighbor frames. As time goes by, we could accumulate large amount previous frames in the system. Keep all these frames in memory is impossible. Dumping everything to the disk will also slow the computation speed when the frames are needed for alignment. We need a efficient way to structure and organize the frame data. In chapter V, we propose a Frame Graph based panorama documentation algorithm including frame insertion, archiving and adjustment operations to manage the online panorama documentation.

When it comes to data retrieving, it is often the case that multiple users including nature scientists and general public want to share the panorama video output at the same time. Each user may want to observe a different sub region and time window of the panorama video. Transmitting the full size ever changing high resolution panorama video to every user is unnecessary and expensive in bandwidth requirement. In chapter VI, we proposed our on-demand high resolution panorama video-streaming system that allows on-demand sharing of a high-resolution panorama video and efficiently deal with multiple concurrent spatial-temporal user requests.

A. Related Work

1. Networked Robot System

Our system is designed to allow multiple online users to share access to robotic cameras. In the taxonomy proposed by Chong et al. [23], these are Multiple Operator Single Robot (MOSR) systems or Multiple Operator Multiple Robot (MOSR) systems. An Internet-based MOSR system is described by McDonald, Cannon, and their

colleagues [24, 25]. In their work, several users assist in waste cleanup using Point-and-Direct (PAD) commands. Users point to cleanup locations in a shared image and a robot excavates each location in turn. Recent developments in MOSR systems can be found in [26, 27]. In [27] Goldberg et al. propose the "Spatial Dynamic Voting" (SDV) interface. SDV collects, displays, and analyzes sets of spatial votes from multiple online operators at their Internet browsers using a Gaussian point clustering algorithm developed to guide the motion of a remote human "Tele-Actor". Existing work on MOSR and MOMR systems provides strategies to efficiently coordinate the control of the shared robot. Users are usually forced to share the same feedback from the robot. However, users may not be interested in the same event at the same time even when they access the system at the same time. This becomes more obvious when the shared robot is a robotic camera. Time and space of interests may vary for different online users.

2. Image Alignment

Panoramic display is a emerging new way of visualizing remote environments [28]. Panoramas can be classified as either cylindrical panoramas or spherical panoramas according to the number of axes involved in camera motion. A cylindrical panorama only involves pan motion, [29, 30] and its construction is relatively simple and fast. However, cylindrical panoramas cannot provide sufficient vertical field of view for natural environment observation.

Constructing a spherical panorama is much more complex because more parameters needs to be estimated in its nonlinear transformation model. It relies on image alignment techniques, which attempt to find the best set of transform parameters for images to compose the panorama. The transformation can be modeled by a projective projection model [31, 32]. After establishing the parameter model, the image

alignment problem searches for a optimized solution in parameter space.

Current image alignment techniques can be classified into three categories: direct method [32–37], frequency domain registration [38, 39], and feature-based image registration [40–49]. The direct method directly compares intensity values of pixels from the overlapping images and is sensitive to lighting conditions, while feature-based alignment works on a sparse set of feature points and is less sensitive to lighting conditions and needs less computation. Frequency domain registration works well for translation, but has problems with rotation.

Recent research on improving the speed of image alignment focuses on the feature-based method, which extracts features such as Harris corner point [40, 41, 43], Moravec's interest point [44], SUSAN corner point [47], vanishing point [49], and Scale Invariant Feature Transform (SIFT) [50]. Torr and Zisserman [41] outline the feature-based method: First, features are extracted automatically. An initial set of matches are computed based on proximity and similarity of their intensity neighborhood. These estimations inputs are then placed into a robust estimation algorithm such as the Least Median of Squares(LMedS) [42] or Random Sample Consensus(RANSAC) [49] to choose the solution with the largest number of inliers. Numerical minimization techniques such as the Levenberg-Marquardt algorithm are then applied to refine the estimation result from RANSAC. Since an SIFT feature point is invariant to projections, the combination of SIFT and RANSAC in [50] has been one of the most successfully image alignment method.

3. Projection Invariants

The development of projection invariants for PTZ cameras is inspired by invariant descriptors for 3D object recognition in pattern recognition [51]. For example, Euclidean distance is invariant to shift and rotation. Fourier Descriptor is invariant

to affine transformation and can be used to recognize objects from multiple view points [52]. The purpose of invariant descriptors in pattern recognition is to find object properties invariant to perspectives, lighting conditions, and lens parameters for object identification. This differs from our problem because we are looking at the shape-preserving property instead of an arbitrary object property.

Projection invariants can be used to improve image alignment efficiency. Due to their shape-preserving property, we know that there are no scaling difference or non-linear distortions among corresponding projection invariants. Therefore, we do not need to use sophisticated feature transformations in the image alignment algorithm. Instead, we can use simple feature transformation such as Zero-Crossing Edge Detector (ZCED) [53] to reduce computation cost. Furthermore, image alignment can be reduced to the problem that finds matching projection invariant pairs, which allows us to speed up the computation. To demonstrate the power of project invariants, we develop a new image alignment algorithm based on our projection invariants. An SIFT and RANSAC based algorithm is used for speed comparison in the paper.

4. Panoramic Video Systems

There are many methods to generate a panorama video. A panorama can be generated using a single fixed camera with a wide-angle lens or parabolic mirrors [17, 18, 54, 55]. However, due to the fact that it can not distribute pixels evenly in the space and the resolution limitation imposed by CCD sensors, it cannot generate high-quality video. A panorama video can also be generated by aligning videos from multiple cameras [56, 57]. Although the design can provide complete coverage with live video streams, those system require simultaneous transmission of multiple video streams and the bandwidth requirement is very high. Panorama video can also be built from registering a pre-recorded sequence of video frames [58–61] captured by a single

rotating camera. However, only portions of the panorama contain live video data at any moment. Our system fits into this category as well. Argarwala et al.'s panoramic video texture (PVT) [61] and Rav-Acha et al.'s dynamosaics [62] are representative work in this category that constructs pseudo-live panorama video out of a single video sequence by alternating time-space correspondence. Bartoli et al. [63] develop motion panoramas that extract moving objects first and then overlay the motion part on top of a static background panorama. We summarize the existing panoramic video systems in Table I. In existing systems, a panorama video is always transmitted and fully reconstructed at the user end because panorama resolution is not a concern. However, when the resolution of the panorama is very high, on-demand transmission is necessary. Our development is the first system that tackles this problem.

Table I. A comparison of existing panoramic video systems.

System	Camera	Bandwidth	Video Output	Sample Systems
Wide angle lens / mirrors	Single fixed	Low	Low quality live stream	[17, 18, 54, 55]
Multiple camera panorama video	Multiple fixed	High	Live panoramic video	[56, 57]
Panoramic video texture	Single pan	High	Pseudo-live panorama video by changing video temporal display	[61]
Dynamosaics	Single pan	High	Pseudo-live panorama video by changing space-time volume	[62]
Motion panorama	Single	Low	Static panorama background overlaid with live moving objects trajectory	[58, 63]
Our system	PTZ cameras	Low	Partial live panorama	This paper

Transmitting a panorama video is non-trivial. For a low resolution panorama video system, we can encode the whole panorama video and send it to clients. However

it consumes too much bandwidth when the resolution of the panorama increases. Furthermore, it cannot deal with random spatiotemporal accesses. Irani et al. [64,65] propose mosaic-based compression. A static panorama background is first constructed out of the video sequence and then each video frame is compressed using the static panorama background as a reference. Furthermore, it detects and indexes the motion objects and provides content-based video indexing. Although they do not deal with on-demand transmission, their work inspires our paper. Ng et al. [55] propose to partition the panorama into six vertical slices spatially and compress each sliced video sequence separately using MPEG-2. When a user requests for the video of a part of the panorama video, only sliced video sequences that intersect with the user's requested area are transmitted. This method is among the first to consider on-demand queries. However, its efficiency of encoding decreases as the camera tilt range increases. Also it repeats the data from previous frame when there is no live coverage; it is not efficient or a faithful representation of the remote environment. Our work advances the idea of partitioning panorama into 2-D patches and significantly reduces computation time and bandwidth by only encoding/transmitting updated patches.

CHAPTER III

PROJECTION INVARIANT IN REAL-TIME PANORAMA CONSTRUCTION

We know that computation speed of finding corresponding pixels can be greatly improved if we can find a subset of pixels that can maintain fixed relative positions. Those subset of pixels are referred to as projection invariants because of their shape-preserving properties. Note the fact that images from a PTZ camera can be treated with the same optical center and a spherical coordinate system is a natural coordinate system for organizing images with the same optical center. Therefore, we focus our research on search of projection invariants in spherical coordinate systems. The shape-preserving property of projection invariants can transfer the image re-projection process into a rigid body translation and rotation of projection-invariants. Experiment results from comparison study show that the projection invariant based image alignment algorithm outperforms the existing best image alignment method by at least an order of a magnitude.

A. Problem Definition

1. Assumptions

We assume that all images are taken from the same PTZ camera that is installed on a rigid base. No translational motion for the camera is allowed and mechanical vibrations are negligible. Since there are only PTZ motions, all images can be treated with the same optical center. Camera potentiometer readings give an estimation of camera pan/tilt position. These readings are inherently approximate with error (i.e. $\pm 1.0^{\circ}$) and cannot be directly used to assist re-projection, which requires a much higher angular resolution. For example, to accurately re-project images from

Panasonic HCM 280 camera requires an angular resolution of $< 0.0041^\circ$ at zoom = 21x and a resolution of 640x480.

We assume that the camera intrinsic parameters including lens distortion, skew factor, and CCD sensor size are pre-calibrated and known. The camera knows its zoom position (focal length) accurately based on pre-calibration. We also assume that the camera has a maximum Horizontal Field Of View (HFOV) less than or equal to 45 degrees.

2. Nomenclature

We use notations in format of $\{\cdot\}$ to refer to a coordinate system in the paper. We use left superscriptions to indicate the coordinate system of a point/set. Let us define,

- O as the camera optical center.
- $\{W\}$ as a 3D fixed Cartesian coordinate system with its origin at camera optical center point O. We refer to it as *world coordinate system*. A point in $\{W\}$ is denoted as ${}^{W}Q = {}^{W}[x\ y\ z]^T$.
- $\{C\}$ as a 3D Cartesian coordinate system with its origin at O, its Z axis overlapping with optical axis, its $X-Y$ plane parallel with CCD sensor plane and its X axis parallel to the horizontal direction of the image. In the paper we refer to it as the *camera coordinate system*. A point in $\{C\}$ is denoted as ${}^{C}Q = {}^{C}[x\ y\ z]^T$. Note that $\{C\}$ changes as the camera changes its PTZ settings.
- $\{C_A\}$ and $\{C_B\}$ as camera coordinate systems for images A and B, respectively.
- $\{I\}$ as a 2D image plane for image I. The origin of $\{I\}$ is the center of the image. We refer to it as the *image coordinate system*. A point in I is denoted

as ${}^{I}q = [u\ v\ 1]^T$. In the rest of the paper, we use Q notation to indicate a 3D Cartesian point and q to represent a 2D coordinate.

- $\{A\}$ and $\{B\}$ as a 2D image plane for images A and B, respectively. They follow the same definition of $\{I\}$ and are used during image alignment analysis.
- ${}^{A}q = [{}^{A}u, {}^{A}v, 1]^T$ as a point in $\{A\}$ and ${}^{B}q = [{}^{B}u, {}^{B}v, 1]^T$ as its corresponding position in $\{B\}$.
- f as camera focal length.
- (p_A, t_A) and (p_B, t_B) are the pan and tilt settings for images A and B, respectively.
- functions $s(\cdot)$ and $c(\cdot)$ as $\sin(\cdot)$ and $\cos(\cdot)$, respectively.

3. Perspective Projection and Re-Projection

Image acquisition in a perspective camera is a process that maps a 3D world onto a 2D image plane, which can be described by perspective projection model [66]. Therefore, a point in $\{W\}$ is converted to a point in $\{I\}$ by

$$ {}^{I}q = {}^{I}_{C}K\ {}^{C}_{W}R\ {}^{W}Q, \tag{3.1} $$

where rotation matrix ${}^{C}_{W}R$ maps a point from $\{W\}$ to $\{C\}$ and is determined by pan and tilt settings, which are camera extrinsic parameters. Intrinsic camera parameter matrix ${}^{I}_{C}K$ projects the points from $\{C\}$ to $\{I\}$, which is a function of focal length f and is determined by zoom level according to our assumptions.

According to Equation (3.1), 2D image points in two overlapping images A and

B can be mapped with each other using a 3×3 matrix M, [31, 37, 66] as,

$$ {}^{A}q = {}^{A}_{C_A}K \; {}^{C_A}_{C_B}R \; {}^{B}_{C_B}K^{-1} \; {}^{B}q = M \; {}^{B}q, \tag{3.2} $$

where ${}^{A}q$ and ${}^{B}q$ are corresponding points in $\{A\}$ and $\{B\}$, respectively, and rotation matrix ${}^{C_A}_{C_B}R$ characterizes the relationship between camera coordinate systems $\{C_A\}$ and $\{C_B\}$ for images A and B, respectively. Since Equation (3.2) just projects pixels in B to $\{A\}$, the process is referred to as the *re-projection* process and M as the *re-projection matrix* [67]. Matrices ${}^{A}_{C_A}K$ and ${}^{B}_{C_B}K$ are functions of focal lengths, which are known according to our assumptions. Rotation matrix ${}^{C_A}_{C_B}R$ is uniquely defined by camera pan and tilt values. Hence matrix M is a function of camera pan and tilt settings for images A and B,

$$ {}^{A}q = M(p_A, t_A, p_B, t_B) {}^{B}q. \tag{3.3} $$

If images A and B share the same focal length, then ${}^{A}_{C_A}K = {}^{B}_{C_B}K = K$ and Equation (3.2) can be simplified as,

$$ {}^{A}q = K \; {}^{C_A}_{C_B}R \; K^{-1} \; {}^{B}q = M \; {}^{B}q, \tag{3.4} $$

where M is just the similarity transformation of the rotation matrix ${}^{C_A}_{C_B}R$. Hence $|\det(M)| = 1$. With the knowledge of the re-projection, we are ready to introduce projection invariants.

4. Definition of Projection Invariants

The intuition behind projection invariants is the shape-preserving property. In other words, a projection invariant is a subset of pixels that maintain fixed relative positions with respect to each other under re-projection. Define ${}^{A}\mathbb{C} \subset A$ as a patch of pixels located at the overlapping region of images A and B. Therefore, it has a corresponding

position ${}^{B}\mathbb{C} \subset B$ in image B.

Definition 1 (Projection Invariant Definition)**.** *$\forall {}^{A}q_1, {}^{A}q_2 \in {}^{A}\mathbb{C}$ and their corresponding position ${}^{B}q_1, {}^{B}q_2 \in {}^{B}\mathbb{C}$, define $\Delta^{A}q = {}^{A}q_1 - {}^{A}q_2$ and $\Delta^{B}q = {}^{B}q_1 - {}^{B}q_2$, ${}^{A}\mathbb{C}$ and ${}^{B}\mathbb{C}$ are a pair of projection invariants if and only if the follow shape-preserving condition is satisfied,*

$$|\Delta^{A}q| = |\Delta^{B}q|, \tag{3.5}$$

where $|\cdot|$ is L2-norm.

Our objectives are to find/construct projection variants under re-projection in either planar coordinate systems such as M or its equivalence in other coordinate systems.

B. Projection Invariants

In this section, we first analyze the relationship between re-projection matrix M and projection invariants. We find that projection invariants do NOT exist in planar image coordinate systems. Hence we search nonlinear coordinates to pre-project images. We then prove that projection invariants exist and can be constructed in spherical coordinate systems. We now begin with the analysis of the relationship between re-projection matrix M and projection invariants.

1. Projection Invariants and Re-Projection

Plug Equation (3.2) into Equation (3.5), we get,

$$|M\Delta^{B}q| = |\Delta^{B}q|. \tag{3.6}$$

The re-projection matrix M can be expanded as $M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$. We have the following theorem,

Theorem 1 (Projection Invariant Condition)**.** *To meet the shape-preserving condition in Equation (3.5), if and only if the re-projection matrix M satisfies the following condition,*

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = R_{2\times 2} \quad and \quad m_{31} = m_{32} = 0 \tag{3.7}$$

over the projection invariant, where $R_{2\times 2}$ is a 2×2 rotation matrix.

Proof. (if): Plug Equation (3.6) into Equation (3.5). Equation (3.5) holds. This is trivial.

(only if): According to our nomenclature, we know that ${}^{A}q_1 = [{}^{A}u_1, {}^{A}v_1, 1]^T$ and ${}^{A}q_2 = [{}^{A}u_2, {}^{A}v_2, 1]^T$. Define $\Delta^{A}u = {}^{A}u_1 - {}^{A}u_2$ and $\Delta^{A}v = {}^{A}v_1 - {}^{A}v_2$. Then $\Delta^{A}q = [{}^{A}u, {}^{A}v, 0]^T$. Similarly, $\Delta^{B}q = [{}^{B}u, {}^{B}v, 0]^T$. From Equation (3.2), we know,

$$\begin{bmatrix} {}^{A}u \\ {}^{A}v \\ 0 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} {}^{B}u \\ {}^{B}v \\ 0 \end{bmatrix}$$

Take a close look at the third row of the equation above, we know that,

$$m_{31}{}^{B}u + m_{32}{}^{B}v = 0. \tag{3.8}$$

Since ${}^{B}u$ and ${}^{B}v$ can take arbitrary values, $m_{31} = m_{32} = 0$ has to be true in order to satisfy Equation (3.8). Therefore, the left hand side of Equation (3.6) is,

$$|M\Delta^{B}q| = \sqrt{(m_{11}{}^{B}u + m_{12}{}^{B}v)^2 + (m_{21}{}^{B}u + m_{22}{}^{B}v)^2}. \tag{3.9}$$

The right hand side of Equation (3.6) is,

$$|\Delta^B q| = \sqrt{{}^B u^2 + {}^B v^2}. \tag{3.10}$$

Plug Equations (3.9) and (3.10) into Equation (3.6), we have,

$$m_{11}^2 + m_{21}^2 = 1; \tag{3.11a}$$

$$m_{12}^2 + m_{22}^2 = 1; \tag{3.11b}$$

$$m_{11}m_{12} + m_{21}m_{22} = 0. \tag{3.11c}$$

Hence $R_{2\times 2}$ is a 2×2 rotation matrix. □

Remark Theorem 1 intuitively tells us that re-projection can be viewed as a rotation of the projection invariant if the projection invariant exists.

Unfortunately, the condition in Theorem 1 cannot always be satisfied by the re-projection matrix M. From Equation (3.2), we know that M is determined by camera parameters. Therefore, there is no guarantee that condition in Theorem 1 would be satisfied. In fact, it is not difficult to come with counter examples. However, this does provide the insight for searching for directions that lead to the discovery of projection invariants.

2. Spherical Wrapping

Theorem 1 reveals the fact that the deformation of the projected image in the re-projection process cannot be sensitive to camera parameters if projection invariants exist. The re-projection M defined by Equation (3.2) projects one planar image into another planar image space. It is not surprising that the amount of the deformation of the projected image is very sensitive to the relative positions of those two planes, which is determined by camera parameters.

Our immediate thinking is to try a different coordinate system. If we wrap the image around a spherical surface, then the re-projection between two spherical coordinate systems should introduce less deformation.

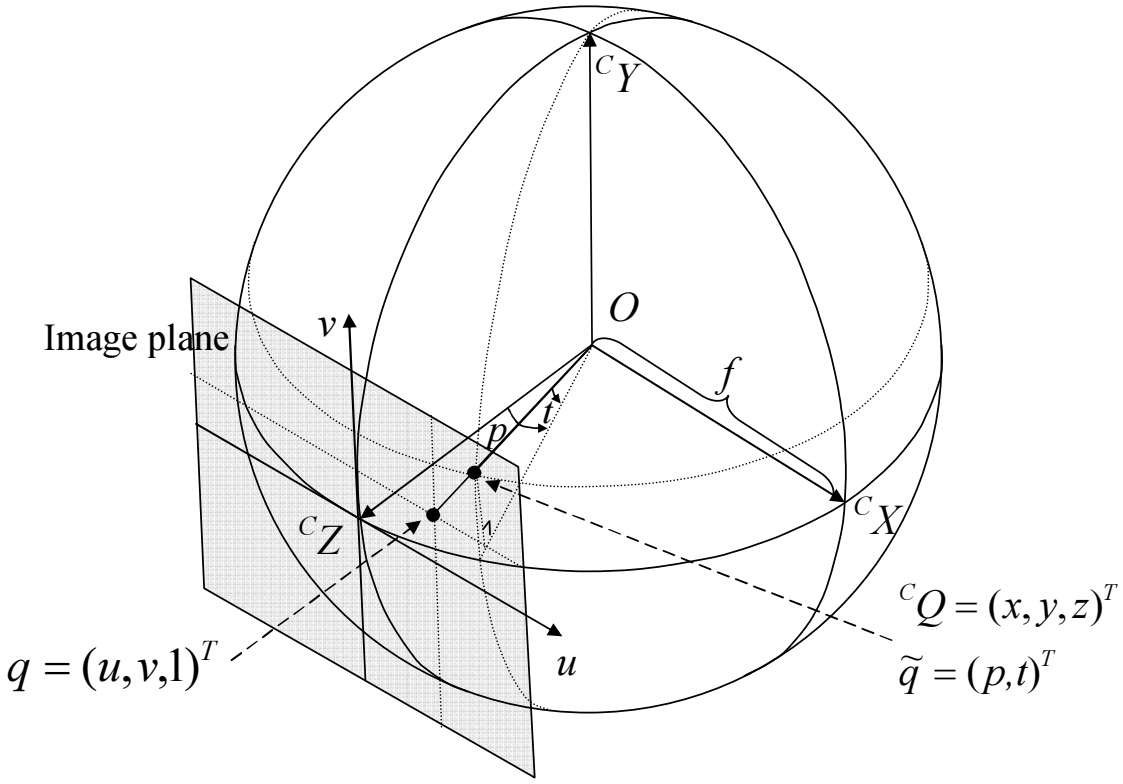


Fig. 4. An illustration of spherical wrapping and coordinate systems: q in image coordinate system, $\tilde{q}$ on the local spherical coordinate system, and ${}^C Q$ is the same point as $\tilde{q}$ but in the camera coordinate system.

The chosen sphere is centered at the camera optical center and has focal length f as its radius. Recall that I is the image captured by the camera. As illustrated in Figure 4, the projection generates a wrapped image $\tilde{I}$ based on a *local spherical coordinate system* $\{\tilde{I}\}$. Recall that ${}^I q = (u, v, 1)^T$ is a point in I. Define $\tilde{q} = (p, t)^T$ as the corresponding point in $\tilde{I}$, where (p, t) is the angular coordinate of the point.

The spherical wrapping that projects q to $\tilde{q}$ is,

$$p = \arctan(\frac{u}{f}), \tag{3.12a}$$

$$t = -\arctan(\frac{v}{\sqrt{u^2 + f^2}}). \tag{3.12b}$$

Each point in $\tilde{I}$ is defined using local pan and tilt spherical coordinates with units in radians. Spherical coordinate system $\{\tilde{I}\}$ usually consists of three elements including radius, pan, and tilt. Although images taken at different zoom levels have different radius, it is not difficult to scale them into the same spherical surface because $\{\tilde{I}\}$ is represented in angular coordinates instead of pixel coordinates. Therefore, we can treat f as the same and yield a 2D representation. Also, $\tilde{q} = (0, 0)^T$ overlaps with $q = (0, 0, 1)^T$. Note that $\{\tilde{I}\}$ has its origin centered at each image and is different from the global spherical coordinate defined by real camera pan and tilt settings. In fact, the p and t in $\tilde{q}$ only depend on its corresponding pixel coordinates in I. We use $\sim$ above I to indicate that $\tilde{I}$ is image I's spherical wrapping. We will use this convention in the rest of the paper. The spherical wrapping can be conducted without the knowledge of camera pan and tilt settings. This is an important feature that will be reiterated later.

3. Spherical Re-Projection (SRP)

Now the new re-projection can be performed between two local spherical coordinate systems, which is referred to as Spherical Re-Projection (SRP) to distinguish it from the planar re-projection in the rest of the paper.

Define $Q = {}^{C}Q = [x, y, z]^T$ as $\tilde{q}$ in $\{C\}$ as illustrated in Figure 4. Recall that $\cos(\theta)$ and $\sin(\theta)$ are denoted as $c(\theta)$ and $s(\theta)$, respectively. The relationship between

$\{\tilde{I}\}$ and $\{C\}$ can be described by function P and its inverse P^{-1},

$$\tilde{q} = \begin{bmatrix} p \\ t \end{bmatrix} = \begin{bmatrix} \arctan(x/z) \\ -\arctan(y/\sqrt{x^2+z^2}) \end{bmatrix} = P(Q), \tag{3.13}$$

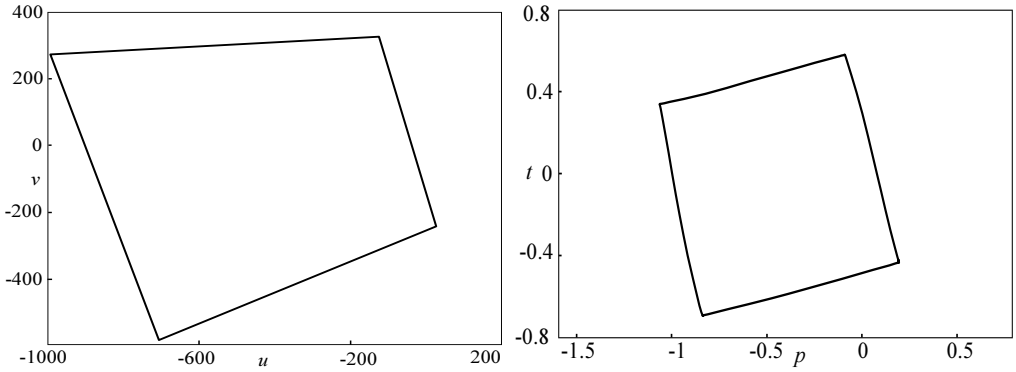
$$Q = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f \cdot c(t)s(p) \\ -f \cdot s(t) \\ f \cdot c(t)c(p) \end{bmatrix} = P^{-1}(\tilde{q}). \tag{3.14}$$

Let $\tilde{A}$ and $\tilde{B}$ be the resulting image from the spherical wrapping for image A and image B, respectively. Without loss of generality, we select image $\tilde{A}$ as the reference image. We shift image $\tilde{B}$ around $\tilde{A}$. To align the two images, we need to re-project $\tilde{B}$ into $\tilde{A}$'s space,

$$\begin{aligned} {}^{A}\tilde{q} &= P({}^{C_A}_{C_B}R\ {}^{B}Q) = P({}^{C_A}_{C_B}R\ P^{-1}({}^{B}\tilde{q})) \\ &= F({}^{C_A}_{C_B}R, {}^{B}\tilde{q}). \end{aligned} \tag{3.15}$$

where F is the SRP function, ${}^{A}\tilde{q} = ({}^{A}p, {}^{A}t)^T$ and ${}^{B}\tilde{q} = ({}^{B}p, {}^{B}t)^T$ are positions of the corresponding point in wrapped image $\tilde{A}$ and $\tilde{B}$, respectively.

We are interested in comparing the re-projection on the spherical surface with the original planar re-projection. The testing image is a square image with a resolution of 640×640. It is projected to another camera configuration that shares 30° tilt value and has 30° pan difference. Figure 5 suggests that the deformation on the spherical surface is significantly less than that in the original planar image space. Since the absolute distortion is an increasing function of image size, we conjecture that if we sample a very small square region on the spherical surface then the deformation for each square should be negligible after the spherical wrapping. If so, it possesses the property of a projection invariant.



(a) (b)

Fig. 5. Comparison of image deformation caused by the re-projection operation (a) in the original planar image space and (b) on the spherical surface. Note that the unit in (a) is pixel and the unit in (b) is radian.

4. Projection Invariants for SRP

Before we prove the conjecture, let us define a squared-shaped cell in image $\tilde{A}$ as,

$$^{A}\mathbb{C} = \{(^{A}p, ^{A}t)|^{A}p \in [^{A}p_o \pm p_c], ^{A}t \in [^{A}t_o \pm t_c]\}, \tag{3.16}$$

where $^{A}\tilde{q}_o = (^{A}p_o, ^{A}t_o)$ is the cell center coordinate, and (p_c, t_c) is the maximum cell span in pan and tilt directions. We define $^{B}\mathbb{C}$ as $^{A}\mathbb{C}$'s projection in image $\tilde{B}$ with its center at $^{B}\tilde{q}_o = (^{B}p_o, ^{B}t_o)$.

We need to adapt the Projection Invariant Condition in Theorem 1, which is constructed on planar re-projection, to the nonlinear SRP function F. Define $\Delta^{B}\tilde{q} = {}^{B}\tilde{q} - {}^{B}\tilde{q}_o$ and $\Delta^{A}\tilde{q} = {}^{A}\tilde{q} - {}^{A}\tilde{q}_o$. Equation (3.5) now becomes,

$$|\Delta^{A}\tilde{q}| = |\Delta^{B}\tilde{q}|. \tag{3.17}$$

We have the following corollary,

Corollary 1 (SRP Projection Invariant Condition)**.** *$^{A}\mathbb{C}$ and $^{B}\mathbb{C}$ are a pair of pro-*

jection invariants, if and only if the following condition is satisfied,

$$\mathbf{\Delta}^A \tilde{q} \approx R_{2\times 2} \mathbf{\Delta}^B \tilde{q}, \tag{3.18}$$

where the 2×2 *rotation matrix* $R_{2\times 2}$ *is not a function of* $\Delta^A \tilde{q}$ *or* $\Delta^B \tilde{q}$.

The proof of Corollary 1 is trivial. We can treat $R_{2\times 2}$ as the linearized approximation of F at center point of the cell because each cell is small and the linearized approximation is accurate enough. Then it follows the proof of Theorem 1. Note that we use '$\approx$' in Equation (3.18) instead of '='. This is acceptable because an image is a discretized representation of the real environment and any distortion that is less than half a pixel is negligible. In other words, Equation (3.18) tells us that the linearized nonlinear function F over the cell can be approximated by a same rotation matrix over the entire cell if the cell is projection invariant.

Now we are ready to prove the conjecture about SRP projection invariants.

Theorem 2. *If the corresponding spherical cells* ${}^A\mathbb{C}$ *and* ${}^B\mathbb{C}$ *are small,* $p_c \leq 5^\circ$ *and* $t_c \leq 5^\circ$, *and the camera has a vertical field of view* $\leq 34^\circ$, *then* ${}^A\mathbb{C}$ *and* ${}^B\mathbb{C}$ *are projection invariant under SRP.*

Proof. Recall that functions $s(\cdot)$ and $c(\cdot)$ as $\sin(\cdot)$ and $\cos(\cdot)$, respectively. From vector calculus, we know that

$$\nabla Q = \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} fc(t)c(p) & -fs(t)s(p) & c(t)s(p) \\ 0 & -fc(t) & -s(t) \\ -fc(t)s(p) & -fs(t)c(p) & c(t)c(p) \end{bmatrix} \begin{bmatrix} dp \\ dt \\ df \end{bmatrix}. \tag{3.19}$$

Define $[\Delta x, \Delta y, \Delta z]^T$ as the small displacement in $\{C\}$ and $[\Delta p, \Delta t, \Delta f]^T$ as the corresponding change in $\{\tilde{I}\}$. Since $p_c \leq 5^\circ$ and $t_c \leq 5^\circ$, $\Delta p < p_c/2 = 2.5^\circ$ and

$\Delta t < t_c/2 = 2.5°$. Hence we have

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = f \begin{bmatrix} c(t)c(p) & -s(t)s(p) & r_{13} \\ 0 & -c(t) & -r_{23} \\ -c(t)s(p) & -s(t)c(p) & r_{33} \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta t \\ \Delta f \end{bmatrix}, \tag{3.20}$$

where $r_{13} = c(t)c(p)/f$, $r_{23} = s(t)/f$, $r_{33} = c(t)c(p)/f$ corresponds to the last column of the Jacobian matrix in Equation (3.19). Since we have $\{\tilde{I}\}$ as part of a sphere, radius f remains constant. Therefore $\Delta f = 0$. To move the negative sign out of the second row of the matrix in Equation (3.20), we introduce coefficient matrix $H = \begin{bmatrix} f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & f \end{bmatrix}$. Then Equation (3.20) can be rewritten as,

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = H \begin{bmatrix} c(t)c(p) & -s(t)s(p) & r_{13} \\ 0 & c(t) & r_{23} \\ -c(t)s(p) & -s(t)c(p) & r_{33} \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta t \\ 0 \end{bmatrix}, \tag{3.21}$$

Recall that t are the tilt positions with respect to the image center inside an image. Recall that the camera has a maximum vertical field of view of 34°. To ensure that the existence of an overlapping region between the two images, the tilt overlap has to be larger than the tilt range of a cell t_c, the maximum value of t is $34/2 - t_c = 12°$ for $t_c = 5°$. Since $\cos(12°) = 0.995$, therefore, $0.995 \leq c(t) \leq 1$. If the camera has a resolution of 640×480, then pixel cell width is around $\frac{5}{34}480 = 68$ pixels for $p_c = t_c = 5°$. If we approximate $c(t) \approx 1$, the maximum distortion $(1 - 0.995) \times 68$ is less than half a pixel. If $t_c < 5°$, then the pixel cell width is also decreased. It is not difficult to show that $(1 - \cos(34/2 - t_c))\frac{t_c}{34}480 < 0.5$ for $0 < t_c < 5°$ because it is an increasing function of t_c for $0 < t_c < 5°$. Since the distortion is very small, instead

we drop $c(t)$ in the first column,

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \approx H \begin{bmatrix} c(p) & s(p)s(-t) & r_{13} \\ 0 & c(-t) & r_{23} \\ -s(p) & c(p)s(-t) & r_{33} \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta t \\ 0 \end{bmatrix}, \tag{3.22}$$

Since $\Delta f = 0$, we know that $[r_{13}, r_{23}, r_{33}]^T$ can take arbitrary values without affecting the equality in Equation (3.22). Let us choose $r_{13} = s(p)c(-t)$, $r_{23} = -s(-t)$, and $r_{33} = c(p)c(-t)$. Then we have,

$$\begin{bmatrix} c(p) & s(p)s(-t) & s(p)c(-t) \\ 0 & c(-t) & -s(-t) \\ -s(p) & c(p)s(-t) & c(p)c(-t) \end{bmatrix} = R_Y(p)R_X(-t), \tag{3.23}$$

where R_Y and R_X are rotation matrices along Y axis and X axis, respectively. Define $\Delta Q = [\Delta x, \Delta y, \Delta Z]^T$ and $\Delta\tilde{q} = [\Delta p, \Delta t, 0]^T$, Now Equation (3.22) is,

$$\Delta Q \approx H R_Y(p) R_X(-t) \Delta\tilde{q} \tag{3.24}$$

Hence, we have

$$\Delta^A Q \approx H R_Y({}^A p_o) R_X(-{}^A t_o) \begin{bmatrix} \Delta^A\tilde{q} \\ 0 \end{bmatrix}, \tag{3.25}$$

and,

$$\Delta^B Q \approx H R_Y({}^B p_o) R_X(-{}^B t_o) \begin{bmatrix} \Delta^B\tilde{q} \\ 0 \end{bmatrix}. \tag{3.26}$$

Since $\Delta^A Q = {}^A_B R \Delta^B Q$, we get,

$$\begin{bmatrix} \Delta^A \tilde{q} \\ 0 \end{bmatrix} \approx R_X({}^A t_o) R_Y(-{}^A p_o) H^{-1} \cdot {}^{C_A}_{C_B} R H R_Y({}^B p_o) R_X(-{}^B t_o) \begin{bmatrix} \Delta^B \tilde{q} \\ 0 \end{bmatrix}. \tag{3.27}$$

Since H and H^{-1} are diagonal matrices, we have $H^{-1} {}^{C_A}_{C_B} R H = {}^{C_A}_{C_B} R$. Equation (3.27) becomes,

$$\begin{bmatrix} \Delta^A \tilde{q} \\ 0 \end{bmatrix} \approx R_\Delta \begin{bmatrix} \Delta^B \tilde{q} \\ 0 \end{bmatrix}, \tag{3.28}$$

where

$$R_\Delta = R_X({}^A t_o) R_Y(-{}^A p_o) {}^{C_A}_{C_B} R R_Y({}^B p_o) R_X(-{}^B t_o), \tag{3.29}$$

is a rotation matrix because the multiplication of rotation matrices yields a rotation matrix. On the other hand, the last row has to satisfy $0 = 0$ no matter what value $\Delta^B \tilde{q}$ takes. This means R_Δ has to be in the following format,

$$R_\Delta = \begin{bmatrix} R_{2\times 2} & 0_{2\times 1} \\ 0_{1\times 2} & 1 \end{bmatrix}.$$

Hence it satisfies Corollary 1 and ${}^A\mathbb{C}$ and ${}^B\mathbb{C}$ are projection invariant under SRP. □

Remark Theorem 2 also tells us how to construct projection invariants under SRP and applicable cameras. Most PTZ cameras have vertical field of views less than 34°. When operated at high zooms, camera vertical field of views are even smaller. For example, a Panasonic HCM 280 camera has a 2.8° vertical field of view at zoom=21x. Even for a camera that has a vertical field of view larger than 34°, we can still construct projection invariants by sample cells that are within the 34° range.

Theorem 2 suggests that each cell can be treated as a rigid object in SRP, which

can lead to a significant computation savings. The next question is how to compute the rotation matrix $R_{2\times 2}$, which can be characterized by a single rotation angle θ. We have the following lemma,

Corollary 2. *Recall that (p_A, t_A) and (p_B, t_B) are the pan and tilt settings for images A and B, respectively. Rotation angle θ of rotation matrix $R_{2\times 2}$ can be approximated by,*

$$\begin{aligned}\theta \approx \arccos\big(&c({}^{A}p_o)c({}^{B}p_o)c(p_B - p_A) + s({}^{A}p_o)s({}^{B}p_o) * \alpha \\ &+ s(p_B - p_A)s({}^{A}p_o)c({}^{B}p_o)c(t_A) \\ &- s(p_B - p_A)c({}^{A}p_o)s({}^{B}p_o)c(t_B)\big).\end{aligned} \tag{3.30}$$

where α is a function of (p_A, t_A) and (p_B, t_B) only and can be pre-computed.

$$\alpha = c(t_A)c(t_B)c(p_B - p_A) + s(t_A)s(t_B). \tag{3.31}$$

(α is the dot product of Z axes of $\{C_A\}$ and $\{C_B\}$ in world coordinate system.)

Proof. Let us use the following vectors,

- $\begin{bmatrix} \Delta^{A}\tilde{q} \\ 0 \end{bmatrix} = [1/f, 0, 0]^T$,
- $\begin{bmatrix} \Delta^{B}\tilde{q} \\ 0 \end{bmatrix} = [1/f, 0, 0]^T$,
- ${}^{C_A}X_{0A} = HR_Y({}^{A}p_o)R_X(-{}^{A}t_o)\begin{bmatrix} \Delta^{A}\tilde{q} \\ 0 \end{bmatrix}$, and
- ${}^{C_B}X_{0B} = HR_Y({}^{B}p_o)R_X(-{}^{B}t_o)\begin{bmatrix} \Delta^{B}\tilde{q} \\ 0 \end{bmatrix}$.

It is clear that ${}^{C_A}X_{0A}$ and ${}^{C_B}X_{0B}$ are unit vectors. By defining ${}^{W}X_{0A}$ and ${}^{W}X_{0B}$ as

their corresponding coordinate in $\{W\}$, we know that

$$c(\theta) =< {}^{W}X_{0A}, {}^{W}X_{0B} >, \tag{3.32}$$

from the definition of vector inner product. From coordinate transform relationship, we know,

$$\begin{aligned}
{}^{W}X_{0A} &= {}^{W}_{C_A}R\ {}^{C_A}X_{0A} \\
&= R_Y(p_A)R_X(t_A)HR_Y({}^{A}p_o)R_X(-{}^{A}t_o)[1/f, 0, 0]^T \\
&= \begin{bmatrix} c(p_A)c({}^{A}p_o) - s(p_A)c(t_A)s({}^{A}p_o) \\ s(t_A)s({}^{A}p_o) \\ -s(p_A)c({}^{A}p_o) - c(p_A)c(t_A)s({}^{A}p_o) \end{bmatrix}.
\end{aligned}$$

Similarly, we can compute ${}^{W}X_{0B}$. Inserting them into Equation (3.32), we get Equation (3.30). □

Remark It is worth mentioning that if two images share similar pan positions (i.e. $|p_A - p_B| \leq 5^\circ$), then Equation (3.30) becomes

$$\theta \approx \arccos\big(c({}^{A}p_o)c({}^{B}p_o) + s({}^{A}p_o)s({}^{B}p_o)c(t_B - t_A)\big). \tag{3.33}$$

Recall that a standard camera has a maximum vertical field view of 34°. To guarantee the overlap between the two frames, the maximum value of $t_B - t_A$ has to be less than 17°. Therefore, $\cos(17^\circ) = 0.956 \leq c(t_B - t_A) \leq 1$ and $c(t_B - t_A)$ can be approximated by 1. Hence, we have,

$$\theta \approx {}^{B}p_o - {}^{A}p_o,$$

for this special case, which can further speed up the computation.

At the first glance, Equation (3.30) in Corollary 1 is very complex. It tells us that θ depends on ${}^{A}p_o$, ${}^{B}p_o$, $p_B - p_A$, t_A, and t_B. Since we choose the position for

${}^{A}\mathbb{C}$, we knew its center position ${}^{A}p_o$ in $\tilde{A}$. According to Equation (3.15), ${}^{B}\mathbb{C}$'s center position ${}^{B}p_o$ is uniquely defined by ${}^{A}p_o$, $p_B - p_A$, t_A, and t_B. Since ${}^{A}p_o$ is usually known in image $\tilde{A}$, θ uniquely depends on $p_B - p_A$, t_A, and t_B.

Therefore, the position and the orientation of ${}^{B}\mathbb{C}$ in image $\tilde{B}$ is uniquely defined by $p_B - p_A$, t_A, and t_B, which define the spatial relationship between the two intersecting images. The shape of ${}^{B}\mathbb{C}$ remains a square with the same side length as that of ${}^{A}\mathbb{C}$ in image $\tilde{A}$. This desirable shape-preserving property has many potential applications such as image alignment, panorama generation, real-time tracking of moving objects, and/or video encoding, where pixel correspondence dominates the computation. Since image alignment is a fundamental problem in computer vision, below we use it as a sample application to introduce how projection invariants can be used to accelerate the computation significantly for PTZ cameras.

C. Projection Invariant-based Image Alignment

1. Problem Description

A planar image aliment problem is to align two images by estimating M that minimizes the pixel/feature differences in the overlapping part of the two images. Among existing error metrics for pixel/feature differences, Sum of Squared Differences (SSD) is one of the most popular metrics, [36, 66],

$$SSD = \sum_{i \in A \cap B} \big(\text{Feature}_B({}^{B}q_i)) - \text{Feature}_A({}^{A}q_i)\big)^2,$$

where set $A \cap B$ is the overlapping pixel set between image A and image B, ${}^{A}q_i$ and ${}^{B}q_i$ are the ith overlapping pixel from image A and image B, respectively, and $\text{Feature}_A()$ and $\text{Feature}_B()$ are feature values for images A and B, respectively. Feature values can take different forms. Feature values can be pixel intensity values if direct methods

are used. Feature values can also be probability measure if posterior probability distribution is used to represent feature. On the other hand, the error measure is not necessarily limited to SSD. The analysis can be easily adapted to other non-negative difference metrics.

According to Equation (3.3), M can be determined by camera pan and tilt settings. When we align image B with respect to A, the pan and tilt settings (p_A, t_A) for image A are usually known as reference. Therefore, M can be determined by two unknown variables (p_B, t_B),

$$^{A}q = M(p_B, t_B)^{B}q. \tag{3.34}$$

Therefore, the image alignment problem for PTZ cameras is to solve the following optimization problem,

$$\arg\min_{(p_B,t_B)} \sum_{i \in A \cap B} \left(\text{Feature}_B(M(p_B, t_B)^{B}q_i)) - \text{Feature}_A(^{A}q_i)\right)^2. \tag{3.35}$$

There are two unavoidable problems if we solve the optimization problem in Equation (3.35) by directly evaluating candidate (p_B, t_B) pairs. The first problem is the speed. Define $m = |A \cap B|$ as the number of feature pixels in $A \cap B$ and let k be the number of candidate (p_B, t_B) pairs, it can easily take $O(km)$ re-projection operations. Since the re-projection computation involves extensive floating point computation, the dominating factor km is usually very large for high resolution images. The second problem, which is more of a concern, is the alignment accuracy. Since M is very sensitive to (p_B, t_B), a minor error in (p_B, t_B) would significantly change the shape of the feature pixel set $\{\text{Feature}_B(M(p_B, t_B)^{B}q_i) : i \in A \cap B\}$, which leads to inaccurate alignment.

Among all of the recently proposed methods, one of the most effective way to address this accuracy problem is to introduce a feature transformation that is not

sensitive to affine transformation. The re-projection process is an affine transformation. As one of the most popular feature transformation methods, Lowe's SIFT [68] is designed to be scaling and rotation invariant and fits the requirement. Combining SIFT with RANSAC [49] to choose the solution with the largest number of inliers, Brown and Lowe [50] have well-addressed the accuracy problem in image alignment. However, computing SIFT is expensive in time, because SIFT feature points have to be evaluated at different scaling levels and orientations. The long computation time limits its usage in time-critical applications.

Projection invariants can be used to improve image alignment efficiency. Due to their shape-preserving property, we know that there are no scaling difference or nonlinear distortions among corresponding projection invariants. Therefore, we do not need to use sophisticated feature transformations in the image alignment algorithm. Instead, we can use simple feature transformation such as Zero-Crossing Edge Detector (ZCED) [53] to reduce computation cost. Furthermore, image alignment can be reduced to the problem that finds matching projection invariant pairs, which allows us to speed up the computation. Building on the intuition, we can design a Projection Invariant-based Image Alignment Algorithm (PIIAA).

2. Projection Invariant-based Image Alignment Algorithm

As illustrated in Figure 6, our algorithm is based on a set of small square-shaped cells evenly scattered in the overlapping region. Defined in Equation (3.16), each cell is a projection invariant that satisfies the condition specified by Theorem 2. Define k_c as the number of cells, which is between 25 and 36 in most cases. Define ${}^{A}\mathbb{C}_j \subset \tilde{A}$, $1 \leq j \leq k_c$ as the jth cell. From potentiometer reading and its error range, we know

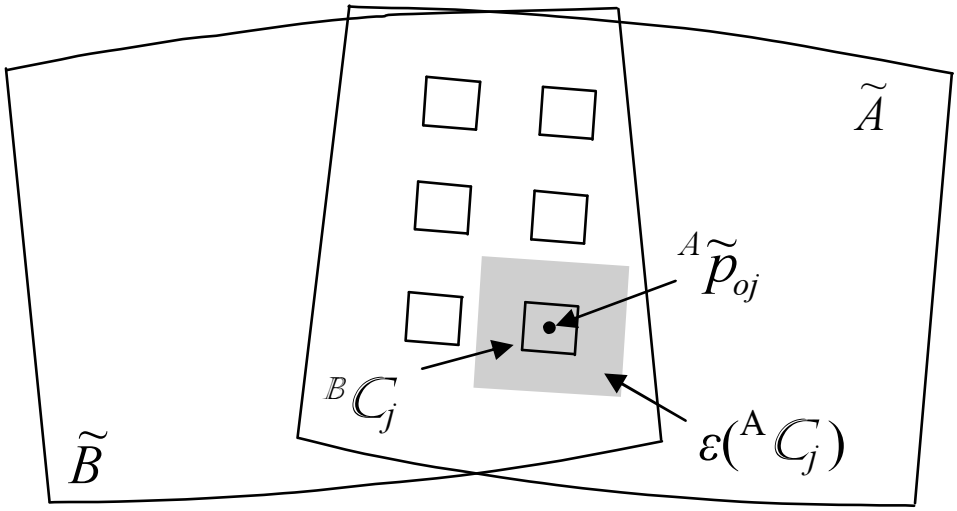


Fig. 6. An illustration of projection invariant-based image alignment algorithm. Image $\tilde{A}$ and image $\tilde{B}$'s barrel-like shape is due to spherical wrapping.

that the matching region of ${}^{B}\mathbb{C}_j \subset \tilde{B}$ will be found within region $\epsilon({}^{A}\mathbb{C}_j) \subset \tilde{A}$, which is the gray region in Figure 6,

$$\begin{aligned}\epsilon({}^{A}\mathbb{C}_j) = \{(p,t) \in \tilde{A} | p \in [{}^{A}p_{oj} \pm (p_c + .5p_{\max})] \\ t \in [{}^{A}t_{oj} \pm (t_c + .5t_{\max})]\},\end{aligned} \tag{3.36}$$

where $({}^{A}p_{oj}, {}^{A}t_{oj}) = {}^{A}\tilde{q}_{oj}$ is the center point of ${}^{A}\mathbb{C}_j$, (p_c, t_c) defines cell size, and $(p_{\max}, t_{\max})$ is the potentiometer error range. For example, for the images captured from a Canon VCC3 camera that has a 45° horizontal field of view, an image size of 640×480-pixels, and $\pm 1.5°$ potentiometer error, $\epsilon({}^{A}\mathbb{C}_j)$ is ± 20 pixels shifting range in $\tilde{A}$. Based on Corollary 2, we also know that the inverse rotation by $-\theta$ around cell center ${}^{B}q_{oj}$ defines the orientation of ${}^{B}\mathbb{C}_j$,

$${}^{B}\mathbb{C}_j = R_c(-\theta)\,{}^{A}\mathbb{C}_j.$$

Therefore, we transfer the optimization problem in Equation (3.35) to

$$\min_{(p_B,t_B)} \sum_{j=1}^{k_c} \left(\text{Feature}_B(R_c(-\theta)^A\mathbb{C}_j) - \text{Feature}_A({}^A\mathbb{C}_j)\right)^2, \tag{3.37}$$

subject to,

$${}^B\mathbb{C}_j \subset \epsilon({}^A\mathbb{C}_j). \tag{3.38}$$

Since ${}^B\mathbb{C}_j$ is considered as a solid square with only rotation and shifting, computing the solution becomes less costly. Each candidate solution will determine orientation and location of k_c cells.

Since the relative positions between cells are rigid and known, the search for a solution is to simultaneously shift all k_c rotated cells in $\tilde{A}$ and find the optimal solution with the pre-computed ${}^B\mathbb{C}_j$'s. Because k_c is a relatively small number (i.e. $25 \sim 36$) and each cell is very small (i.e. 10×10 pixels), the computation is very fast. Define $(\delta p, \delta t)$ as ${}^B\mathbb{C}_j$ shifting variable such that $\delta p \in [\pm 0.5 p_{\max}]$ and $\delta t \in [\pm 0.5 t_{\max}]$ to satisfy Equation (3.38). Because of the image resolution limit, there are only a constant number of $(\delta p, \delta t)$ pairs.

Another benefit is that feature detection and spherical wrapping do not need to be computed for the entire image. Only pixels in the selected cells and their neighboring search regions need to be computed. Define n as the number of total pixels in images A and B. We summary the analysis above as the Projection Invariant-based Image Alignment Algorithm below.

Algorithm 1: Projection Invariant-based Image Alignment Algorithm

input : Image A, Image B, Image A's pan and tilt setting (p_A, t_A)

output: Image B's pan and tilt setting (p_B, t_B)

Computing lookup table for spherical wrapping $\tilde{A}$ and $\tilde{B}$; $O(1)$

Select evenly scattered ${}^{A}\mathbb{C}_j$, $j = 1, ..., k_c$ in the overlapping region; $O(1)$

for *each* j, $0 \leq j \leq k_c$, **do**

- Compute ${}^{B}\mathbb{C}_j$ using initial readings from potentiometer; $O(1)$
- Compute Feature$_B({}^{B}\mathbb{C}_j)$ using ZCED; $O(1)$
- Compute Feature$_A(\epsilon({}^{A}\mathbb{C}_j))$ using ZCED; $O(1)$

for *each* $(\delta p, \delta t)$, **do**

- **for** *each* j, $0 \leq j \leq k_c$, **do**
  - Compute cell orientation θ; $O(1)$
  - Rotate Feature$_B({}^{B}\mathbb{C}_j)$ by $-\theta$; $O(1)$
  - Compute SSD for the cell j; $O(1)$
- Report the sum of SSD across all cells; $O(1)$

Report $(\delta p, \delta t)$ with the minimum SSD; $O(1)$

Add $(\delta p, \delta t)$ to initial potentiometer reading to get (p_B, t_B); $O(1)$

Since the property projection invariant of allows to avoid complex feature extractions, plus the fact that we can further limit the ZCED to the minimum number pixels, the PIIAA is actual a constant time algorithm if we do not consider image I/O time. This is very desirable in dealing with high resolution images.

D. Experiments and Results

We have implemented the algorithm and tested in a series of experiments. The computer we used for testing is a 3.2Ghz Desktop PC with 2GB RAM and a 120GB hard

disk. The C++ based source code is complied in Microsoft Visual Studio 2003.net under Windows XP Professional Edition.

We first compare the speed of our algorithm with the fastest method that is currently available [50]. It is a combination of SIFT and RANSAC with k-d tree support. We have used open source SIFT code[1] and k-d tree code[2] and implemented RANSAC according to [67]. Since this algorithm is used to construct panorama from aligning multiple image frames, it is referred to as Panorama Recognition Algorithm (PRA) in [50]. To ensure a fair comparison, we only compare image alignment time. Additional components in panorama construction such as image I/O, bundle adjustment, and blending/rendering are not counted in the time comparison.

We first investigate how well each algorithm scales up when image resolution increases. Images used in the test are taken by a Panasonic SD 360 camera with a maximum resolution of 2816×2112. Table II shows how much time each algorithm takes under different image resolutions. The input is a pair of overlapping images. The two algorithms are fed with the same input pair during the experiment. Both PRA and PIIAA are initialized with the same initial conditions (i.e. using the same inaccurate pan and tile potentiometer readings as their initial solutions). At each resolution level, we use 10 independent image pairs taken in the Texas A&M University campus. With each image pair as a trial, the time in the table is an average of 10 trials. Since the variance from trial to trial is small, it is not presented here. The factor column in the table indicates the speed improvement of PIIAA over PRA. It is clear that PIIAA is significantly faster than PRA for PTZ cameras. It is also desirable to see that factors get bigger as image resolution increases. Projection invariants clearly

[1]http://vision.ucla.edu/∼vedaldi/code/siftpp/siftpp.html

[2]http://ilab.usc.edu/toolkit/home.shtml

speed up the computation.

Table II. A comparison of algorithm speed versus image resolution.

Resolution	PRA Time (milisec.)	PIIAA Time (milisec.)	Factor
176×132	230.8	12.5	18.5x
352×264	1209.3	43.7	27.7x
704×528	5359.4	82.8	64.7x
1408×1056	24401.5	215.6	113.2x
2816×2112	113196.9	731.4	154.8x

CHAPTER IV

MINIMUM VARIANCE BASED INCREMENTAL FRAME ALIGNMENT

With camera frequently steered to different pan/tilt position, image alignment must be performed incrementally on the incoming video frames to generate panorama video. We define evolving panorama as a sequence of all video sequence inserted in temporal order. The incremental frame alignment is the sequential registration of large number of video frames into the panorama video during the panorama construction and update. When a new frame arrived, we compute its optimal position by aligning with a set of existing neighbor frames. We need to identify a subset of past frames that provide an optimal tradeoff between quality of the panorama and computation time. If we assume the alignment error is a random vector with zero mean, the magnitude of error variance determines the quality of alignment. We study how error variance gets accumulated and propagated in the incremental alignment process and propose a minimum variance based incremental frame alignment algorithm able to ensure the quality of the panorama video over long run. For k images, our algorithm runs in time $O(k \log k)$. Experiments show that our algorithm can reduce calibration error by 81% if compare with a method that simply selects frames with large overlapping regions.

A. Problem Description

1. Inputs and Assumptions

Definition of Frame Sequence: When the camera is moving, images are blurred and must be discarded. Once the camera has stopped, we define a *frame sequence* as a

sequence of camera frames from some fixed pan-tilt-zoom setting,

$$F = \{C(t_{\text{begin}}, t_{\text{end}}), p, t, z, X, \upsilon\}, \tag{4.1}$$

where C stands for the frame content data set, t_{begin} and t_{end} are the beginning time and ending time of the frame sequence respectively, (p, t, z) are the approximate pan, tilt, and zoom values obtained from the camera, X is a set of unknown image alignment parameters, and υ is a scalar that indicates how well the frame sequence is aligned with respect to its neighbors as discussed below.

Since the camera does not move for the duration of a frame sequence, we compute the alignment parameters using the first image of each frame sequence and use the same alignment parameters to transform the last image of the sequence to update the panorama. Below, we refer to the "frame" as the first image from a frame sequence.

Definition of an Evolving Panorama: The evolving panorama at time t includes all previous frame sequences,

$$P(t) = \{F | t_{\text{begin}} < t\}$$

inserted in temporal order.

Each panorama has a reference frame. The positional parameters X of other frame sequences are computed with respected to the reference frame. The reference frame is also the first frame of the panorama. Starting with reference frame, the panorama is initialized by commanding the camera to visit a sequence of preset coordinates that cover the field of view as we will show in Section 1. Actually, the panorama generation and maintenance need the same incremental frame alignment algorithm that will be introduced in Section 2.

Known Camera Intrinsic Parameters: Constructing the panorama requires projection and positional parameters. The projection parameters include image resolu-

tion, camera focus length, and CCD sensor size, all of which are known and fixed. We use these to project all images onto a fixed spherical surface. The set of positional parameters X from Equation 4.1 are unknown and must be computed.

Approximate Camera Pan, Tilt, Zoom Position: The tele-operator periodically sends a motion command to the camera, specified as a desired pan, tilt, and zoom (p, t, z) target. After the camera motors servo toward this target, they stop and the camera sends back an estimate of its resulting pan, tilt, and zoom position. As noted above, these estimates are inherently approximate. We use the approximate position for an initial estimate of how many pixels overlap between a pair of frames. Once the alignment parameter X is computed by the algorithm, we use it to refine the number of overlapped pixels.

Random Pair-wise Alignment Error: When computing the relative offset between two frames, the matching problem is a nonlinear minimization problem. Introduced by numerical methods for nonlinear optimization like Gaussian-Newton method, Simulated Annealing, or Genetic Algorithms, the error between true optimal and actual solution depends on initial point and truncation error. A good algorithm chooses its initial point randomly, which defines the alignment error to be a random vector. We assume the alignment error random vector has zero mean and variance σ^2, which usually is a function of truncation error and image characteristics and will be discussed in Section 1.

Errors in Pair-wise Alignment: We assume that the Average Matching Error (AME) A of each pixel (or feature point if using feature-based matching) can be approximated by a quadratic function in the vicinity of its optimal matching location. For the i^{th} pixel in a new frame with its location X_i, this is described by,

$$A(X_i) = a\|X_i - X_i^*\|_2^2 + b, \tag{4.2}$$

where X_i^* is optimal alignment location, a is a scaling factor, and b is the residual caused by noise. We assume that a and b are the same across all matching pixels.

2. Incremental Frame Alignment Problem

The incremental Frame Alignment problem is: *given a set of n existing frame sequences, find X for a newly arrived frame sequence.*

We solve it in two steps. The first step is to identify a subset of past frame sequences and decompose the alignment problem into multiple pair-wise alignment problems and give each an appropriate weight. In the second step, the pair-wise alignment problems are solved by applying standard image mosaicing methods. We use the direct matching method throughout the rest of the paper.

We focus on step one: identify a subset of past frames sequences that provide an optimal tradeoff between quality of the panorama and computation time.

B. Algorithms

We've assumed that error of X is a random vector with zero mean. Therefore, the magnitude of error variance determines the quality of alignment. To analyze the error variance, we first propose a quality metric to measure how sensitive an image alignment method is to errors. We study how error variance gets accumulated and propagated in the alignment process using a simple 1D example. Based on the analysis, we propose a minimum variance approach to select an optimal set of existing frames to register a newly arrived frame. We begin with definition of the quality metric.

1. Quality Metric for Image Alignment

We propose the following quality metric υ to quantify alignment error. The scalar υ measures average pixel-wise alignment variance and will be defined for each frame sequence.

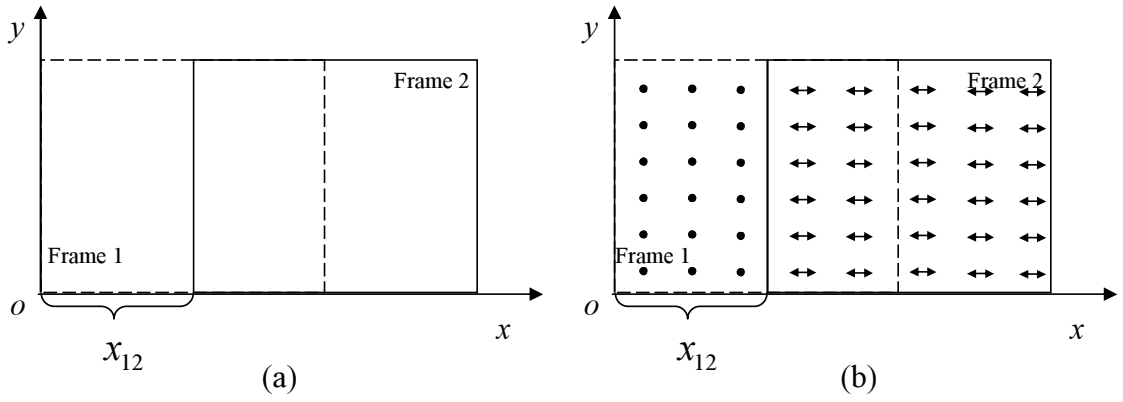


Fig. 7. An illustration of metric υ using a panorama composed by two equally sized frames with equal number of pixels. Frame 1 is the reference image in the alignment.

Since image alignment is not perfect due to round off errors and image characteristics, the displacement between the actual coordinate X_i of the i^{th} pixel and its ideal coordinate X_i^* is a random vector $D_i = X_i - X_i^*$. Let n_p be the number of pixels in panorama P. For P, metric υ is,

$$\upsilon(P) = \frac{1}{n_p} \sum_{i=1}^{n_p} Var(D_i) \tag{4.3}$$

Metric υ is defined for a frame sequence as the average alignment variance of all pixels in its first frame.

Figure 7 illustrates how to compute υ using a panorama with two equally sized frames. The displacement between the two frames is caused by camera pan motion

so that the only alignment parameter is the horizontal displacement, x_{12}, between the two frames. Frame 1 enters the system first, then Frame 2 is captured. Frame 2 will be put on the top of frame 1. Define x_{12}^* as the optimal displacement. Random displacement error is $d_{12} = x_{12} - x_{12}^*$. Since frame 1 is the reference frame, all its pixels have zero variance. Alignment variance of each pixel in frame 2 is σ^2. Figure 7(b) uses arrows to indicate variance amplitude. Let m, $m \leq n_p$, be number of pixels in each frame and m_{12}, $0 < m_{12} \leq m$, be number of overlapping pixels. Metric υ of the panorama can be computed as

$$\upsilon = \frac{1}{n_p}((m - m_{12}) \times 0 + m\sigma^2) = \frac{m}{n_p}\sigma^2, \tag{4.4}$$

where frame 1 contributes $m - m_{12}$ pixels to the panorama and frame 2 contributes m pixels to the panorama.

2. Analyzing Alignment Errors

In this section we use statistical metric υ to compare the quality of image alignment methods. We begin with the simplest pair-wise alignment operation.

Error Variance in Pair-wise Alignment Define O as the set of the overlapped pixels. According to the assumption in Section **??**, the Total Matching Error (TME) T over O becomes,

$$\begin{aligned} T &= \sum_{i \in O}(a\|X_i - X_i^*\|_2^2 + b) && (4.5)\\ &= |O|a\|X_i - X_i^*\|_2^2 + |O|b. && (4.6) \end{aligned}$$

The image alignment is an optimization problem,

$$\arg \min_{\{X_i, i \in O\}} T,$$

subject to image integrity constraint, which actually reduces the unknown set $\{X_i, i \in O\}$ to the single vector X defined in Equation 4.1. We must find X such that

$$T(X) \leq |O|b + \epsilon,$$

where ϵ is the truncation error from the minimization problem. Inserting it into Equation 4.5, all possible solutions must be inside the ball,

$$\|X - X^*\|_2 \leq \sqrt{\frac{\epsilon}{|O|a}}, \tag{4.7}$$

where X^* is the optimal solution. Recall that AME is an approximation of real matching function in the vicinity of the optimal. AME is unknown during the problem solving process. Therefore, we can not directly use X^* deducted from AME as the solution. Any point in the ball with radius $r = \sqrt{\frac{\epsilon}{|O|a}}$ is a possible solution. To solve the matching problem is just to sample a point from the ball with a unknown location. Any point in the ball is likely to be a solution if the matching algorithm chooses its initial point randomly. The dimensionality of the ball depends on the dimensionality of X.

For the simple 1D case in Figure 7, the ball degrades to a line segment. If we assume the solution is uniformly distributed, then its variance is

$$\sigma^2 = \frac{(2r)^2}{12} = \frac{r^2}{3} = \frac{\epsilon}{3|O|a}. \tag{4.8}$$

Inserting Equation 4.8 into Equation 4.4 and defining $\alpha = m_{12}/m$, we obtain the Metric υ for pair-wise image alignment:

$$\upsilon = \frac{\epsilon}{3n_p a\alpha}. \tag{4.9}$$

For the general $d-$dimension case $X = \{x_1, x_2, ..., x_d\}$, we have variances of the

marginal distributions along each dimension, $\{\sigma_{x_1}^2, \sigma_{x_2}^2, ..., \sigma_{x_d}^2\}$. We define

$$\sigma^2 = \max\{\sigma_{x_1}^2, \sigma_{x_2}^2, ..., \sigma_{x_d}^2\}.$$

Interestingly, though the distribution of the solution point in the ball is unknown, the $d-$dimension case has a similar format with the 1-dimensional case in Equation 4.8 with a different constant factor k_d, as summarized as the following theorem.

Theorem 3. Using AME approximation of image matching function in the vicinity of the optimal solution, the variance of alignment displacement error is

$$\sigma^2 = \frac{r^2}{k_d} = \frac{\epsilon}{k_d |O| a}, \tag{4.10}$$

where $k_d \geq 1$ and d is the problem dimensionality. The exact value of k_d depends on d and the joint probability distribution function of the solution distribution over the ball defined by Equation 4.7.

Proof. Define the joint probability density function as $f(x_1, x_2, ..., x_d)$, we have

$$\underbrace{\int_{-r}^{r} ... \int_{-r}^{r}}_{d} f(x_1, x_2, ..., x_d) dx_1 dx_2 ... dx_d = 1. \tag{4.11}$$

Without loss of generality, we assume $\sigma_{x_1}^2 = \sigma^2$. We compute $\sigma_{x_1}^2$ in the rest of the proof. Because x_1 has zero mean, we know

$$\sigma_{x_1}^2 = E(x_1^2) - E^2(x_1) = E(x_1^2).$$

We define,

$$f_1(x_1) = \underbrace{\int_{-r}^{r} ... \int_{-r}^{r}}_{d-1} f(x_1, x_2, ..., x_d) dx_2 ... dx_d, \tag{4.12}$$

and

$$F_1(y) = \int_{-r}^{y} f_1(x_1) dx_1, \tag{4.13}$$

as the marginal probability density function and the cumulative probability function for x_1 respectively. Now we are ready to compute σ^2,

$$\begin{aligned}
\sigma^2 &= \int_{-r}^{r} x_1^2 f_1(x_1) dx_1 \\
&= \int_{-r}^{r} x_1^2 dF_1(x_1) \\
&= x_1^2 F_1(x_1)|_{-r}^{r} - \int_{-r}^{r} 2x_1 F_1(x_1) dx_1 \\
&= r^2 - \int_{-r}^{r} 2x_1 F_1(x_1) dx_1 \\
&= r^2 - \int_{-r}^{0} 2x_1 F_1(x_1) dx_1 - \int_{0}^{r} 2x_1 F_1(x_1) dx_1 \\
&= r^2 + \int_{-r}^{0} (-2x_1) F_1(x_1) dx_1 - \int_{0}^{r} 2x_1 F_1(x_1) dx_1
\end{aligned}$$

Applying the Second Mean Value Theorem for Integrals, we know $\exists \xi \in [-r, 0], \exists \zeta \in [0, r]$ such that,

$$\int_{-r}^{0} (-2x_1) F_1(x_1) dx_1 = F_1(\xi) \int_{-r}^{0} (-2x_1) dx_1 = F_1(\xi) r^2,$$

and

$$\int_{0}^{r} (2x_1) F_1(x_1) dx_1 = F_1(\zeta) \int_{0}^{r} (2x_1) dx_1 = F_1(\zeta) r^2.$$

Therefore,

$$\sigma^2 = (1 + F_1(\xi) - F_1(\zeta)) r^2,$$

and

$$k_d = 1/(1 + F_1(\xi) - F_1(\zeta))$$

is the constant. □

As summarized in Theorem 3 the quality of the solution is determined by how many pixels are involved in the matching, $|O|$, and the image characteristics a.

Insertion Without Updating Panoramic Display A naive approach is to insert new frames using one panoramic image that is never updated. We can use use metric υ to analyze the resulting performance.

Consider inserting a new frame 3 with the same size into the panorama in Figure 7. Define m_{23}, $0 \leq m_{23} \leq m$, as number of overlapping pixels between frame 2 and frame 3. To simplify the notation, we also define $\beta = \frac{m_{23}}{m}$. Hence $m_{23} = \beta m$ as illustrated in Figure 8.

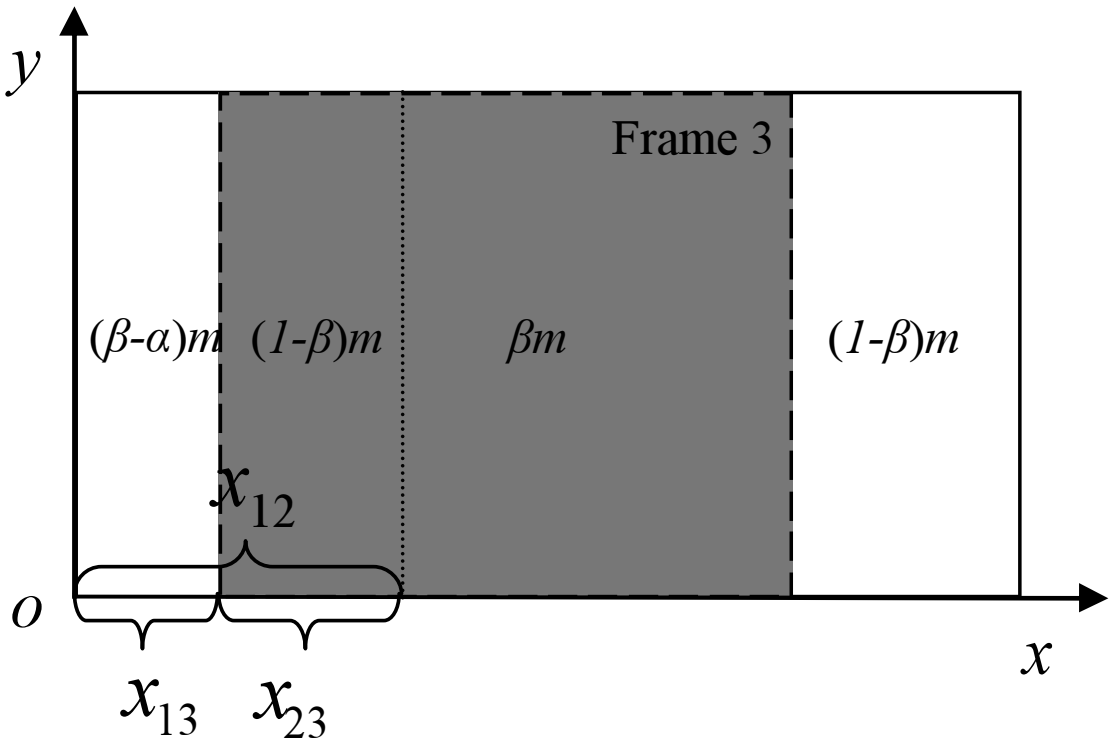


Fig. 8. Insertion of a new frame into the panorama generated by frame 1 and frame 2 in Figure 7.

Define x_{13} as the offset of frame 3 and x_{13}^* as the corresponding optimal offset. Recall that x_{12} is the offset of frame 2. Because frame 2 carries displacement error

$d_{12} = x_{12} - x^*_{12}$, the TME in Equation 4.5 becomes,

$$\begin{aligned} T &= (1-\beta)m\big(a(x_{13} - x^*_{13})^2 + b\big) \\ &+ \beta m\big(a(x_{13} - x^*_{13} + d_{12})^2 + b\big). \end{aligned}$$

This equation can be simplified as,

$$\begin{aligned} T &= ma(x_{13} - x^*_{13} + \beta d_{12})^2 \\ &+ m\big(ad^2_{12}(\beta - \beta^2) + b\big). \end{aligned} \tag{4.14}$$

It is not surprising that its residual $m\big(ad^2_{12}(\beta - \beta^2) + b\big)$ gets bigger because of the displacement error in frame 2. Using the result from Equation 4.7, the radius of the ball that covers possible solution is $\sqrt{\frac{\epsilon}{ma}}$. The variance of the solution for a given d_{12} is,

$$Var(x_{13}|d_{12}) = \frac{\epsilon}{3ma}.$$

Equation 4.14 also tells us the expected solution for a given d_{12} is,

$$E(x_{13}|d_{12}) = x^*_{13} - \beta d_{12}.$$

From knowledge of conditional variance, we know that

$$Var(x_{13}) = E(Var(x_{13}|d_{12})) + Var(E(x_{13}|d_{12})).$$

Therefore, we can get the variance of displacement for each pixel in frame 3,

$$Var(x_{13}) = \frac{\epsilon}{3ma}(1 + \frac{\beta^2}{\alpha}). \tag{4.15}$$

Now, we can compute metric υ for this case. Figure 8 also tells us that frame 1 contributes $(1-\alpha)m - (1-\beta)m = (\beta - \alpha)m$ pixels to the panorama, frame 2 contributes $(1-\beta)m$ to the panorama, and frame 3 contributes m pixels to the

panorama. Plug them in to Equation 4.3,

$$\begin{aligned} \upsilon &= \frac{1}{n_p}\big(m\frac{\epsilon}{3ma}(1+\frac{\beta^2}{\alpha})+(1-\beta)m\frac{\epsilon}{3\alpha ma}\big) \\ &= \frac{\epsilon}{3n_p a}(1+\frac{\beta^2}{\alpha}+\frac{1-\beta}{\alpha}). \end{aligned} \tag{4.16}$$

Comparing to υ from Equation 4.9, the result in Equation 4.16 may grow; the panoramic display deteriorates over time due to deterioration of the matching function, which decreases the subsequent alignment accuracy. This can also be seen in the increase of the residual in Equation 4.14, which indicates a decrease in the signal/noise ratio. Since the panorama is not updated, the deteriorating trend continues as new frames are inserted. To address this, we must update the panorama as frames are inserted. However, as shown in next section, this may suffer from error propagation if it is not designed properly.

Insertion With Updating Panoramic Display Instead of aligning frame 3 with respect to a fixed panorama, we can align it with respect to the existing frames including either frame 1 or frame 2 or both. The choice depends on a tradeoff between reducing

- variance, and
- computation time.

We use the example in Figure 8 to illustrate different outcomes for different approaches. As shown in the figure, there are three unknown variables: x_{12}, x_{13}, and x_{23}. The last variable x_{23} is defined as the offset between frame 2 and frame 3. We know that $x_{13} + x_{23} = x_{12}$ under ideal settings. Due to this relationship, we only need two out of three variables. Since x_{12} is known when the third frame enters the system, we first match frame 2 with frame 3.

Since there are βm pixels overlapped between the two images, the TME function T is,

$$T = \beta ma \|x_{23} - x_{23}^*\|_2^2 + \beta mb.$$

The corresponding variance is

$$Var(x_{23}) = \frac{\epsilon}{3\beta ma}.$$

However, we need to know $Var(x_{13})$, because frame 1 is the reference coordinate. We know that x_{12} and x_{23} are independent random variables. Therefore,

$$Var(x_{13}) = Var(x_{12}) + Var(x_{23}) = \frac{\epsilon}{3ma}(\frac{1}{\alpha} + \frac{1}{\beta}). \tag{4.17}$$

The variance from x_{12} propagates to x_{13} and can grow with each new insertion unless we choose the right images to align with as follows.

3. Image Alignment Methods

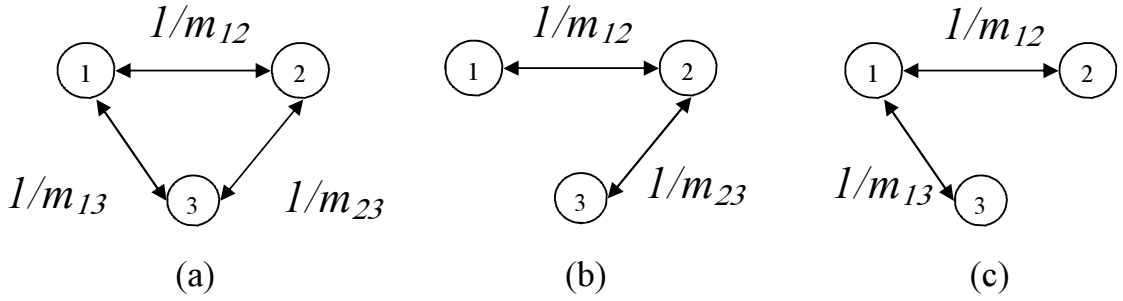


Fig. 9. Graphical representation of alternate methods. Each node represents a camera frame. Each edge represents an overlap between two frames. With edge length proportion to the inverse of the number of overlapping pixels, selective pair-wise matching finds the shortest path from node 3 to node 1 (the reference node).

Selective Pair-wise Matching (SPM) An alternative is to align frame 3 with frame 1. Define m_{13}, $0 \leq m_{13} \leq m$, as number of pixels between frame 1 and frame 3. To simplify the notation, we define $\gamma = m_{13}/m$. Following a similar derivation, we obtain

$$Var(x_{13}) = \frac{\epsilon}{3ma\gamma}. \tag{4.18}$$

Although Equation 4.18 does not contain variance from frame 2, $Var(x_{13})$ is not necessarily smaller than that of Equation 4.17. If we limit ourselves to pair-wise matching, the choice of matching depends on which pair yields smaller variance,

$$\begin{aligned} Var(x_{13}) &= \frac{\epsilon}{3ma} \min\{\frac{1}{\gamma}, \frac{1}{\alpha} + \frac{1}{\beta}\} \\ &= \frac{\epsilon}{3a} \min\{\frac{1}{m_{13}}, \frac{1}{m_{12}} + \frac{1}{m_{23}}\}. \end{aligned}$$

Figure 9 uses a graph to illustrate the selective pair-wise matching process. With each node represents a frame and each edge represents the overlapping relationship between frames, the choice of the least variance matching is to find the shortest path from the new node to the reference node.

Minimum Variance Matching (MVM) In Figure 8, another possible method is to simultaneously align the third frame with both frame 1 and frame 2. This is different from the result in Equation 4.15, because more pixels are involved in the matching process. In Equation 4.15, part of frame 1 has been covered by frame 2 in the fixed panorama and hence can not participate the alignment process. Equation 4.10 shows that variance declines as more pixels are involved in the matching. However, it also could increase the chance of error propagation and increase the variance. The minimum variance matching approach is to find the best set of matching images so that the variance of matching is the smallest.

Let us consider a general case. Assume that the j^{th} frame enters the system, it intersects with a set of existing frames M_j. For the l^{th} frame in M_j, we also know that the number of pixels in frame j intersecting with frame l is m_{jl}. Define X_j and X_l as the vectors that describe the location of image j and image l with respect to the reference image respectively.

Define X_{jl} and X_{jl}^* as the relative offset and the optimal relative offset between frame j and frame l. Then the TME formulation of the matching between frame j and all images in set M_j is,

$$T = \sum_{l \in M_j} \big(a m_{jl} \| X_{jl} - X_{jl}^* \|_2^2 + b m_{jl} \big).$$

Since we are looking for the absolution location $X_j = X_l + X_{jl}$, we change the equation above to,

$$T = \sum_{l \in M_j} \big(a m_{jl} \| X_j - X_l - X_{jl}^* \|_2^2 + b m_{jl} \big).$$

Apply the same approach we did for Equation 4.14, we get

$$E(X_j | \{X_l, l \in M_j\}) = \frac{\sum_{l \in M_j} \big(m_{jl} (X_l + X_{jl}^*) \big)}{\sum_{l \in M_j} m_{jl}} \tag{4.19}$$

and

$$Var(X_j | \{X_l, l \in M_j\}) = \frac{\epsilon}{k_d a \sum_{l \in M_j} m_{jl}}.$$

Therefore,

$$\begin{aligned} Var(X_j) &= Var(E(X_j | \{X_l, l \in M_j\})) \\ &+ E(Var(X_j | \{X_l, l \in M_j\})) \\ &= \frac{\sum_{l \in M_j} m_{jl}^2 Var(X_l)}{(\sum_{l \in M_j} m_{jl})^2} \\ &+ \frac{\epsilon}{k_d a \sum_{l \in M_j} m_{jl}}. \end{aligned}$$

From Theorem 3, we know that $Var(X_l) = \frac{\epsilon}{k_d a} w_l$, where w_l has been computed when

the l^{th} image entered the system. Inserting them into $Var(X_j)$, we get

$$Var(X_j) = \frac{\epsilon}{k_d a}\Big(\frac{1}{\sum_{l\in M_j} m_{jl}} + \frac{\sum_{l\in M_j} m_{jl}^2 w_l}{(\sum_{l\in M_j} m_{jl})^2}\Big). \tag{4.20}$$

Matching over all overlapping frames may not provide us with the smallest variance. What we want is an optimal set of overlapping frames. If the l^{th} image is not used in the matching, we can simply set $m_{jl} = 0$ in Equation 4.20 to get the new variance. This defines a minimization problem. Define $I_l, l \in M_j$ as the image choice variable, we get the following optimization problem,

$$\min F(\{I_l, l \in M_j\}) = \frac{1}{\sum_{l\in M_j} I_l} + \frac{\sum_{l\in M_j} I_l^2 w_l}{(\sum_{l\in M_j} I_l)^2} \tag{4.21}$$

subject to

$$\sum_{l\in M_j} I_l \leq \bar{m}_j, \tag{4.22}$$

$$I_l = \{0, m_{jl}\}, \forall l \in M_j \tag{4.23}$$

where $\bar{m}_j$ is the maximum limit for number of pixels involved in the matching problem. The constraint in Equation 4.22 controls the size of the subsequent matching problem to limit computation time. We solve this optimization problem to derive the optimal set of matching images.

Minimum Variance Based Incremental Frame Alignment Algorithm (MVIFA) The optimal solution of Equation 4.21 yields the minimum variance. However, this is a nonlinear combinatorial problem, which could be very computationally expensive. Though the number of overlapping images $k = |M_j|$ is usually a small number, solving it exhaustively requires time exponential in k.

Looking closer, we observe that when the constraint in Equation 4.22 is binding,

$$\sum_{l \in M_j} I_l = \bar{m}_j,$$

the objective function in Equation 4.21 becomes

$$F(\{I_l, l \in M_j\}) = \frac{1}{\bar{m}_j} + \frac{\sum_{l \in M_j} I_l^2 w_l}{(\bar{m}_j)^2}.$$

Then the minimization problem is simplified as,

$$F' = \min_{\{I_l, l \in M_j\}} \sum_{l \in M_j} I_l^2 w_l \tag{4.24}$$

subject to the constraint in Equation 4.23. The l^{th} candidate matching image takes m_{jl}-pixel space in total $\bar{m}_j$ pixels and contributes $m_{jl}^2 w_l$ to variance if it is selected. The variance per pixel is $m_{jl}^2 w_l / m_{jl} = m_{jl} w_l$. Define candidate solution set as $\hat{M}_j \subseteq M_j$, sum of pixels in $\hat{M}_j$ as $s_1 = \sum_{l \in \hat{M}_j} m_{jl}$, and partial variance sum as $s_2 = \sum_{l \in \hat{M}_j} I_l^2 w_l$. We propose an approach that is based on the order of the variance density and solves the problem for the case that the constraint in Equation 4.22 is binding. This algorithm takes the images that contribute less variance first and gradually expands the set until it reaches the constraint.

The algorithm above does not directly offer a solution when $\sum_{l \in M_j} m_{jl} < \bar{m}_j$. This is not a problem, because we can treat $\bar{m}_j$ as a variable to perform a search over it. Recall the F' defined in Equation 4.24, this new optimization problem is,

$$\min_{\bar{m}_j} \frac{1}{\bar{m}_j} + \frac{F'}{\bar{m}_j^2}, \tag{4.25}$$

which can be solved straightforwardly by keeping tracking of F value in the for loop of the MVIFA algorithm. Instead of using the final $F(\hat{M}_j)$, we output the smallest F and its corresponding set of frames. With this modification, we have

Algorithm 2: MVIFA Algorithm

$\hat{M}_j = \emptyset$, $s_1 = 0$, $s_2 = 0$;	$O(1)$
Compute $m_{jl}w_l$, $l \in M_j$;	$O(k)$
Sort $\{m_{jl}w_l, l \in M_j\}$ in ascending order;	$O(k \log k)$
for *each l in the ascending sequence of $m_{jl}w_l$;*	$O(k)$
do	
if $s_1 + m_{jl} \leq \bar{m}_j$ **then**	
$s_1 = s_1 + m_{jl}$, $s_2 = s_2 + m_{jl}^2 w_l$, $\hat{M}_j = \hat{M}_j \cup \{l\}$	
else	
Break for loop	
$F(\hat{M}_j) = \frac{1}{s_1} + \frac{s_2}{s_1^2}$;	$O(1)$
Output $\hat{M}_j$ and $F(\hat{M}_j)$;	$O(1)$

Theorem 4. The MVIFA algorithm finds the optimal set of overlapping frames in $O(k \log k)$ time for a image with k overlapping frames.

4. Pair-wise Matching

As stated in Section 2, with an optimal set of existing frames, the resulting pair-wise alignment sub problems can be solved using any image mosaicing methods. Equation 4.19 also tells us that the optimal alignment parameter, X, is a weighted average of the pair-wise matching results using the number of overlapping pixels as the weight.

C. Experiments and Results

We have installed a Canon VCC3 Pan-Tilt-Zoom camera at the UC Berkeley campus. The camera has a pan range of 180° and a tilt range of 55°. It features an 1/4-inch CCD sensor with a maximum resolution of 768 × 576. Its horizontal field of view

ranges from 4° to 46°. Our processor is a 2.53Ghz Intel Pentium 4 PC with 1GB RAM and an 80GB hard drive.

1. Construction Phase

In construction phase, we construct a panorama by directing the camera to visit a set of predefined coordinates, each of which defines a composing frame of the panorama. We have taken 21 320×240-pixel frames. During the construction process, we combine our MVIFA Algorithm with Breadth First Search (BFS) to generate a panorama. The BFS starts with camera home position frame, which also our reference frame. It is node 0 in Figure 10. The BFS incrementally covers all 21 points represented by the 21 nodes in the graph illustrated in Figure 10. The pair-wise matching algorithm is a feature-based algorithm. The overall computation time to generate such a panorama is 9.7 seconds, which is even less than the camera travel time. The VCC3 camera can only travel with a maximum speed of 70° per second. To cover all 21 points, it takes about 30 seconds because of frequent stops. Since our algorithm generates the panorama incrementally, it can compute the panorama as the camera travels around. It outputs the full panorama 331 milliseconds after the camera completes its travel. The 21 nodes in Figure 10 are numbered according to the order of arrival. Note that nodes 5, 10, 11, 13, 16, and 18 only align with a subset of their neighbors, which confirms our analysis that to align with as many frames as possible does not necessarily minimize the variance.

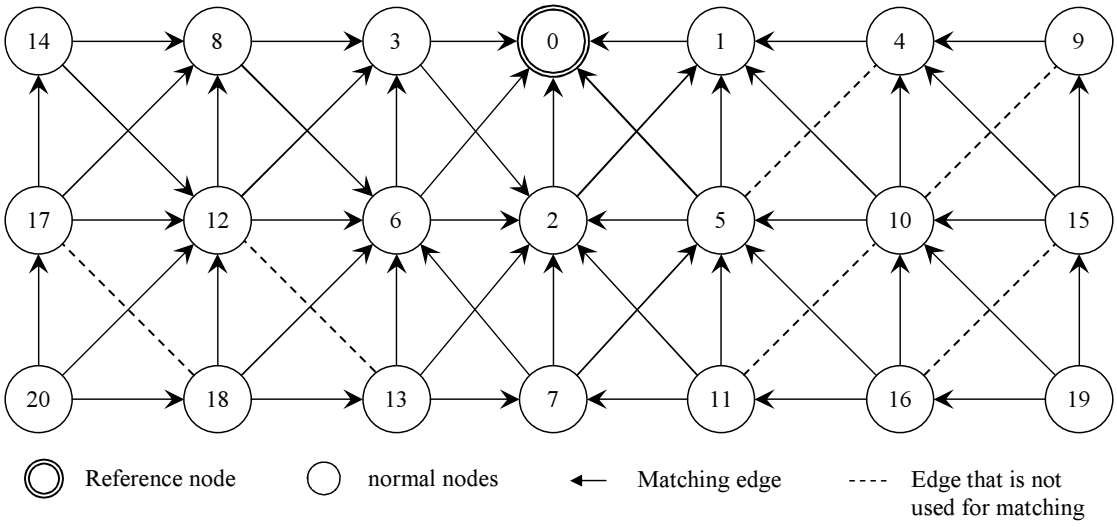


Fig. 10. Resulting matching sequence from MVIFA-BFS using the 21 frames. Each node represents a frame and node numbers are corresponding to BFS frame capturing order. The distribution of matching edges is determined by image alignment mechanisms. The alignment edges are directional: node a → node b means frame a is captured later and uses the existing frame b for alignment.

2. Update Phase

We next test how long it takes to update an existing panoramic display. Based on results of 1000 test runs, the algorithm required an average of 331 milliseconds to update the panorama. The parameter $\bar{m}_j$ in Equation 4.22 determines the trade-off between panorama quality and computation time. In our settings, $\bar{m}_j = 90000$ offers the best trade-off. The update operation is activated when the camera leaves for a new pan-tilt-zoom setting. Since camera travel and stabilization time usually requires more than 331 milliseconds, image alignment can be computed as fast as the camera can be tele-operated.

3. Performance Comparison

We compare the calibration accuracy of our MVIFA algorithm with that of two other options. The first option is to simply align a newly-captured frame with its recent neighbors, which is called Time-Based Incremental Frame Alignment (TBIFA). The rationale behind it is that recent neighbors are less vulnerable to the change of environment. The second option is to align the newly-captured frame with the frames with large overlapping regions, which is called Location-Based Incremental Frame Alignment (LBIFA). The rationale behind it is that large overlaps tend to produce less variance. To ensure a fair comparison, we set the same constraint in Equation 4.22 across all three options. We select the total number of feature pixels involved as $\bar{m}_j = 5000$. For the TBIFA, we rank neighbors according to their arrival time. We add the most recent images to the alignment set until the constraint in Equation 4.22 is binding. For the LBIFA, the only difference is that we rank all neighbors of the new frame according to the size of the overlapping area. For each alignment method, we insert 500 frames into the system as a trial. We repeat each trial 50 times. The data shown in Figure 11 is an average of 50 trials.

Recall that our algorithm selects a subset of frames to align a new frame to minimize the variance of the measured pan and tilt position of the new frame. Therefore, the alignment accuracy is measured using the average variance of the pan and tilt positions of the last 20 frames after the new frame is inserted. Because the variance of a single frame heavily depends on its distance to the reference frame, we use the average of 20 to smooth the location variation in comparison. Since each frame is uniformly, independently, and identically distributed in the camera pan and tilt space, the mean location of the 20 frames is about the same according to the Strong Law

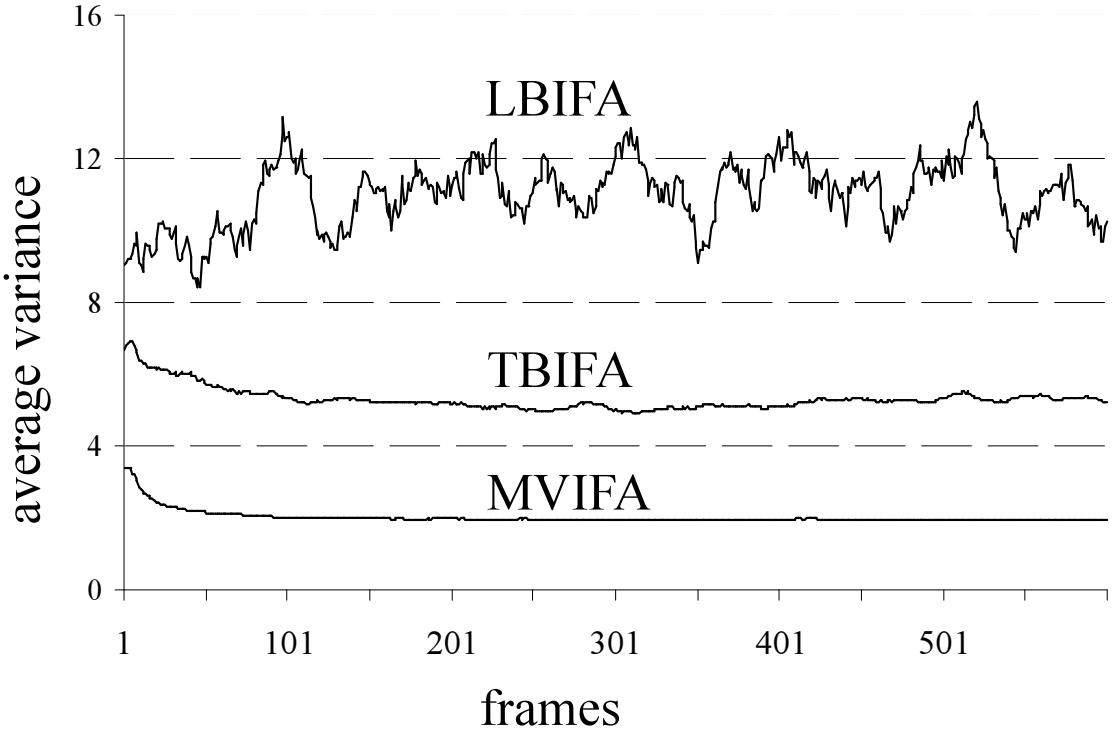


Fig. 11. A comparison of alignment results using the MVIFA algorithm, Location-Based Extrinsic Calibration (LBIFA), and Time-Based Extrinsic Calibration (TBIFA). The variance unit is $\frac{\epsilon}{k_d a} \times 10^{-3}$.

of Large Numbers. Although variance usually does not have a unit, Equation 4.20 suggests that the variance in our system can be measured by constant $\frac{\epsilon}{k_d a}$.

Figure 11 illustrates some interesting results. Both the TBIFA algorithm and our MVIFA algorithm show a trend of convergence. This is due to the fact that there are not enough pixels to bind the constraint in Equation 4.22 at the beginning. As more and more frames enter the system, the constraint binds and the average variance converges to a fixed value. It is clear that the MVIFA algorithm is more effective in variance reduction. Our data shows that it reduces the variance by 65% on average if compared with TBIFA. What surprises us is that the LBIFA is actually the worst among the three methods. One big problem is that variance does not converge for

the 500 frames inserted. This is because the selection of candidate frames is solely based on the size of the overlapping area, which does not consider the variance of the selected frame. Even after the constraint is binding, a single frame with very large variance can dominate the solution. We know that the variances of initial frames are large. A good method should avoid those frames whenever possible. The TBIFA can avoid them over time, but the LBIFA fails and hence cannot converge. Our MVIFA algorithm reduces variance by 81% on average in comparison to LBIFA.

CHAPTER V

FRAME GRAPH BASED PANORAMA DOCUMENTATION ALGORITHM

As illustrated in Figure 12, our system automatically steers a networked pan-tilt-zoom camera to inspect and document construction activities. The input is a set of preset image features, human inspector commands, and on-site motion detectors. The resulting "foveal" video images are aligned and inserted into a coherent panoramic display. Figure 12(b) illustrates the evolving panorama interface.

The evolving panorama can structure and organize the documented video frames. If stored naively, the evolving panorama can consume a large amount of memory. For example, the evolving panorama in Figure 12(b) could have a maximum resolution of 28800 $\times$ 9600 at zoom=10x. We propose a Frame Graph based Panorama Documentation algorithm (FGPD) including frame insertion, archiving and adjustment operations to manage the online panorama documentation. For a panorama with n frames and a new frame with k neighbors and p overlapping pixels, our frame insertion algorithm registers the new frame at $O(\log^2 n + k \log k + p)$ time, our frame archiving algorithm moves outdated neighboring frames to hard disk at $O(k^2 + k \log^2 n)$ time, and our frame adjustment algorithm improves the alignment quality of the panorama at time linear to n and the number of overlapping pixels. Our panorama documentation algorithm combines the PIIAA and MVIFA algorithm and achieve real time panorama video construction.

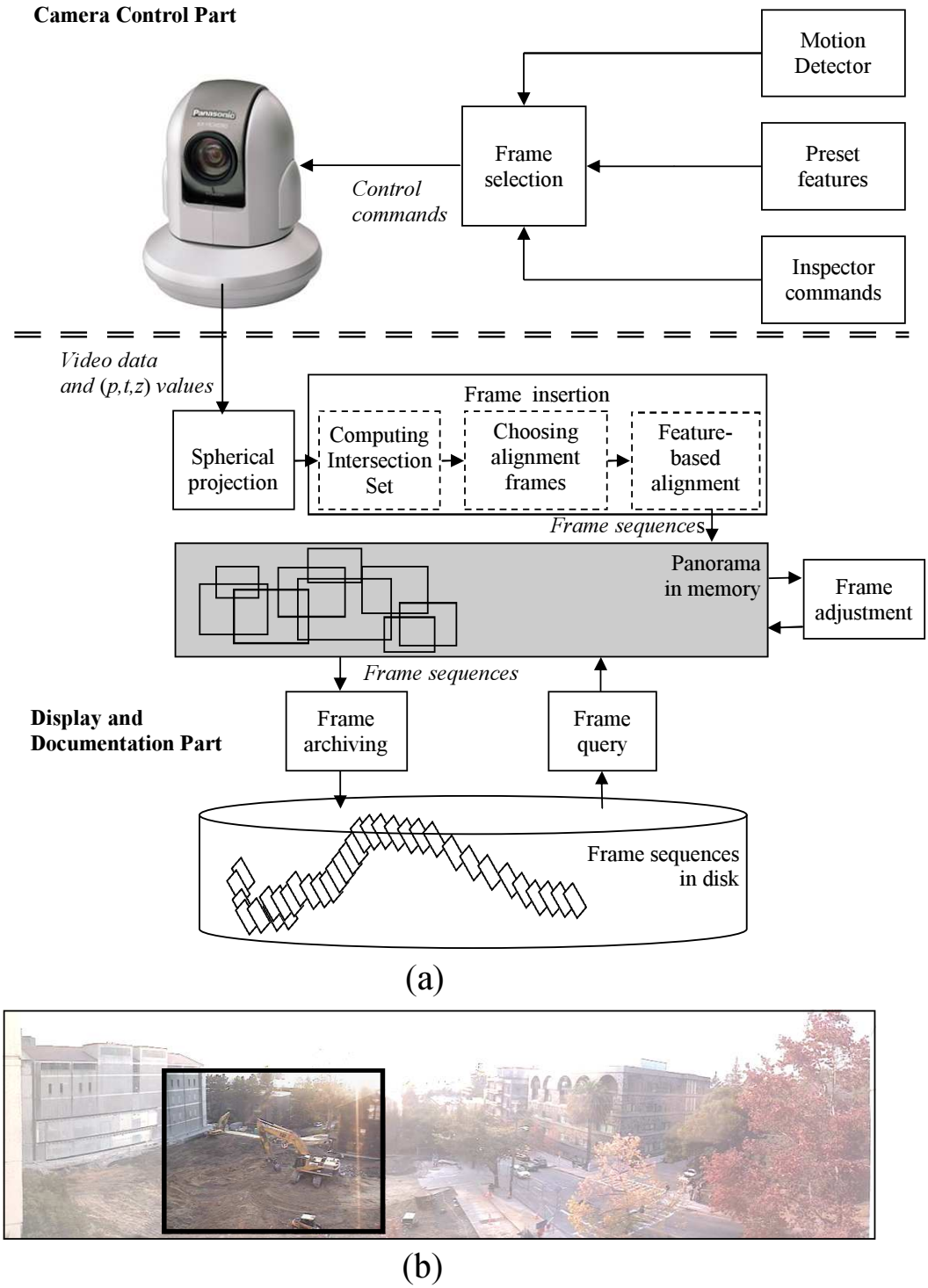


Fig. 12. Top figure (a) camera motion is determined by a combination of preset points, human inspector commands, and motion detector inputs. The resulting video sequences are aligned and inserted into the evolving panorama. Lower figure (b) illustrates panoramic interface, the inset frame is a sample detail captured by the robotic camera and insertion algorithms.

A. Algorithms

1. Frame Graph

Our evolving panorama is a collection of parameterized frame sequences stored in Frame Graph (FG), which is a variation of planar 2D graph. In an FG, node j contains,

- *node ID* j,
- *frame sequence* F_j,
- *rectangle* R_j *that describes the image coverage area, and*
- *total number of pixels of image* m_j.

Edge e_{jl} links node j and node l, which contains,

- *edge ID in format of* jl,
- *indicator variable* I_{jl} *to show if the edge has been used for alignment, where*

$$I_{jl} = \begin{cases} 0 & \textit{no alignment} \\ 1 & \textit{frame } j \textit{ is aligned to frame } l \\ -1 & \textit{frame } l \textit{ is aligned to frame } j \end{cases}$$

- *relative offset* X_{jl} *between node* j *and node* l *if* $I_{jl} \neq 0$,
- *number of overlapping pixels* m_{jl}, *and*
- *rectangle that describes the overlapped area* R_{jl}.

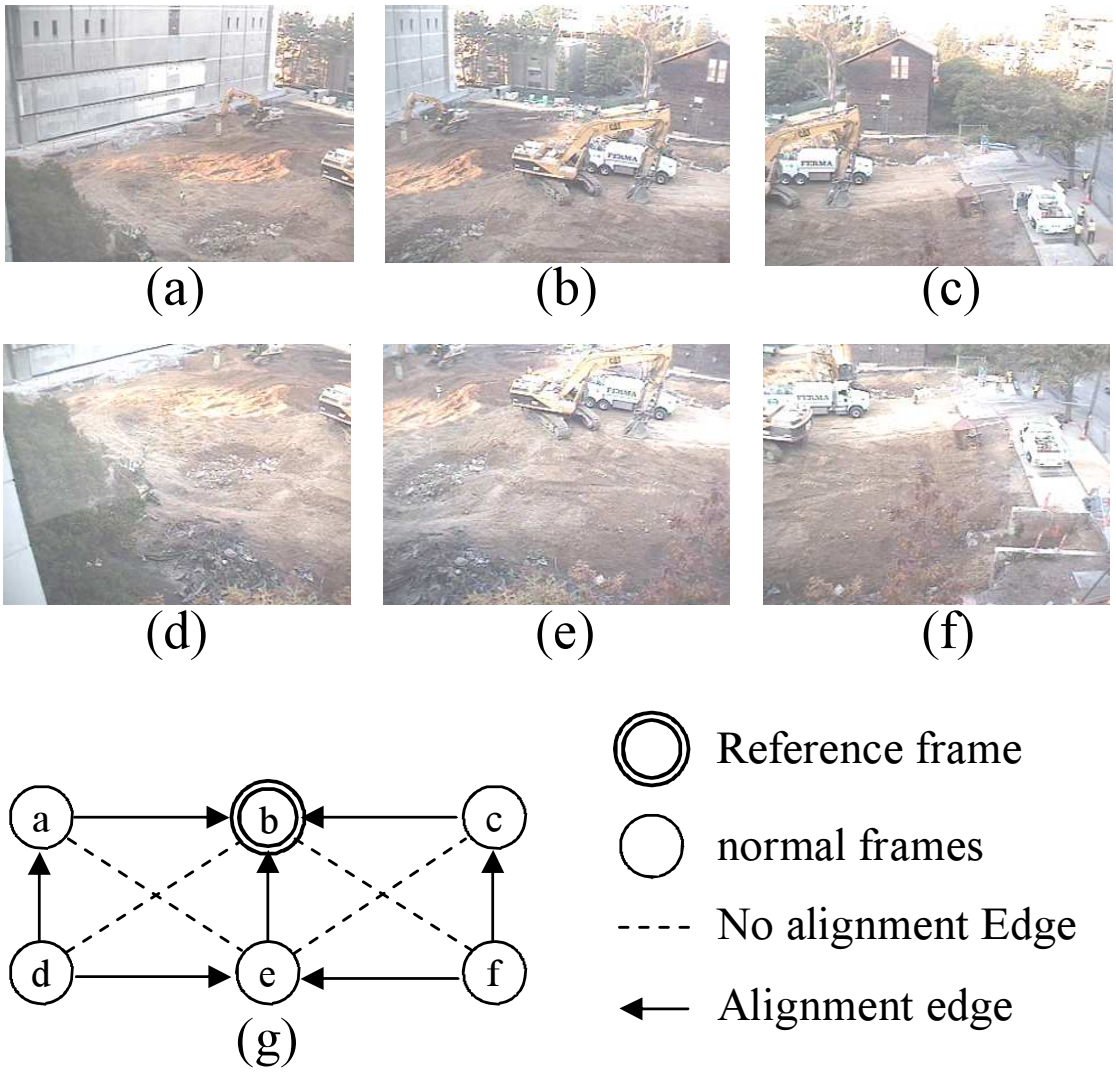


Fig. 13. An example of frame graph with six frames. Figures (a-e) are frames and figure (b) is the corresponding FG.

Figure 13 illustrates a sample FG with six frame sequences. For frame sequence j, its M_j is just its edge set and $\hat{M}_j$ is just the set of edges that have $I_{jl} = 1$. Alignment edges and nodes formulate a Directional Acyclic Graph (DAG) with its only sink located at the reference frame. As a data structure, FG also has a set of maintenance algorithms including frame insertion, archiving, and adjustment.

2. Frame Insertion Algorithm

Each time after the camera changes its pan-tilt-zoom settings, a new frame sequence will be generated and needs to be inserted into the FG. As illustrated in Figure 12, frame insertion algorithm contains three parts: computing intersection frames, choosing the optimal alignment frames, and performing pair-wise alignment.

On the other hand, according to the $p-$ pixel limit imposed by the MVIFA algorithm and the complexity bound of the pair-wise algorithm, the overall pair-wise alignment time is $O(p)$. The remaining part is to find the existing frames that intersect the new frame, which is to find M_j for new frame j.

Assume there are n nodes in the FG at the moment. If n is small, an $O(n)$ linear brute-force search can identify the set. However, n grows as the number of frame sequences accumulates. Computing M_j efficiently requires an indexing data structure. Since we want to find out all overlapping frames, each of which is represented by a rectangle, this formulates a range search problem with the query window defined by the new frame. However, a regular 2D range searching problem [69] only reports points that intersect a query rectangle whereas the queried objects are also rectangles in our problem. A simple solution is to store center points of all existing frames and enlarge the query rectangle, which is similar to compute Minkowski Sums [70] for each queried rectangle. Therefore, we can identify set M_j in $O(\log^2 n + k)$ for $k = |M_j|$. With M_j, we can establish the edges between the new node and the existing nodes.

The complete frame insertion algorithm is described as follows,

Algorithm 3: Frame Insertion Algorithm

Compute M_j using range search;	$O(\log^2 n + k)$
Add edges to FG ;	$O(k)$
Run MVIFA Algorithm to get $\hat{M}_j$;	$O(k \log k)$
Run pair-wise alignment algorithm for each edge in $\hat{M}_j$;	$O(p)$
Update alignment edges ;	$O(k)$
Insert the center point of the new frame to the range tree;	$O(\log^2 n)$

Theorem 5. If a range tree is used as indexing data structure, it takes $O(\log^2 n + k \log k + p)$ time to insert a new frame to a Frame Graph.

3. Frame Archiving Algorithm

A new frame may cover an old frame. If a frame has been mostly covered by its later neighboring frames, we should archive the frame to hard disk to reduce the number of nodes in the FG. Define $p_t \geq 1$ be the minimum number of pixels a frame has to contribute to the panorama, frame archiving algorithm is performed right after new frame j has been inserted,

Algorithm 4: Frame Archiving Algorithm

for *each node* $l \in M_j$ **do**	
Compute region $\bar{R}_l = \{\cup_i R_{li}, i \in M_l, t_i > t_l\}$;	$O(k)$
if $pixel\_number(R_l - \bar{R}_l) < p_t$ **then**	
archive node l and its edges;	$O(k)$
delete l from the range tree ;	$O(\log^2 n)$

Theorem 6. It takes $O(k^2 + k \log^2 n)$ time to find and archive the old frames that are covered by a new frame.

4. Frame Adjustment Algorithm

On the other hand, a new frame may provide better alignment choice to its overlapping frames which leads to frame adjustment algorithm. After frame j enters the system, there is a subset of overlapping images $M_j - \hat{M}_j$ that are not used to align frame j. We know that the frames with big alignment errors are located in the subset. The frame adjustment algorithm is targeted at the worst aligned frame l in set $M_j - \hat{M}_j$. Define M_l and $\hat{M}_l$ be the set of overlapping frames and the set of alignment frames for frame l respectively. Let m_{jl} be the number of overlapping pixels between frame l and frame j.

Algorithm 5: Frame Adjustment Algorithm

Find the node $l \in M_j - \hat{M}_j$;	$O(k)$
Update $\hat{M}_l$ using the MVIFA Algorithm;	$O(1)$
if $j \in \hat{M}_l$ **then**	
Run pair-wise alignment algorithm between frame l and frame j;	$O(m_{jl})$
Update alignment edges for frame l;	$O(k)$
Recursively adjust frames that aligned to frame l;	$O(n)$

As illustrated in the algorithm, for the adjusted frame l, we only need to perform one pair-wise alignment between frame l and frame j, which yields X_{jl}. X_l can be refined incrementally because of the weighted sum format. Changing of X_l leads to the adjustment of all other frames that either directly or indirectly aligned to frame l. Since $n > k$, the total complexity of the frame adjustment algorithm is,

Theorem 7. It takes $O(n + m_{jl})$ time to adjust the alignment parameters of frame l and other effected frames after frame j enters the system.

B. Experiments and Results

As shown in Table III and Figure 14, images from 4 different cameras are used in experiments. Cameras VCC3, VCC4, and HCM 280 are PTZ cameras. Camera SD 630 is a regular digital camera mounted on a tripod, which provides high resolution images for comparing algorithms.

Table III. A comparison of technical specifications of cameras tested in our experiments. VCC3, VCC4, and SD 630 are from Canon. HCM 280 is from Panasonic.

Camera	pan	tilt	zoom	focal length
VCC3	$-90° \sim +90°$	$-30° \sim +25°$	10x	$4.2 \sim 42mm$
VCC4	$-100° \sim +100°$	$-30° \sim +90°$	16x	$4 \sim 64mm$
HCM 280	$-175° \sim +175°$	$0° \sim -120°$	21x	$3.8 \sim 79.8mm$
SD 630	N/A	N/A	3x	$5.8 \sim 17.4mm$

(a) VCC3 (b) VCC4 (c) HCM 280 (d) SD 630

Fig. 14. Cameras tested in the experiments.

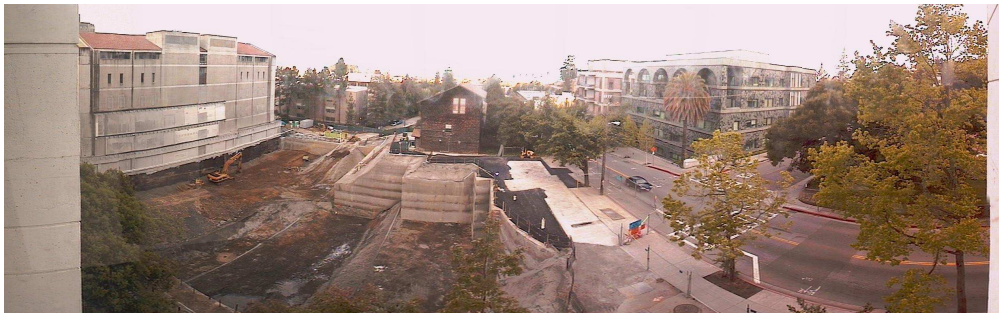
Our real time panorama video construction combines the PIIAA, MVIFA and FGPD algorithm. Constructing panorama requires to perform a large number of image alignments at various camera PTZ settings. With applications range from natural observation and building construction documentation, our algorithm has been tested in four different sites as illustrated in Figure 15.

We have designed and implemented several versions of this system. We deployed our first construction camera system in June 2003 to monitor the Stanley Hall building construction at UC Berkeley. At 285,000 square feet and 11 floors, the new Stanley building is the largest campus construction project in 20 years. This $162 million project is a research and teaching building scheduled to open in 2006. We used a Canon VCC3 robotic camera. Figure 15(a) describes the site. We initially focused on camera control. Over 93060 frames were recorded in the subsequent 2 years. The most frequent users were construction project managers. From their feedback, we noticed that there is a great interest for high-resolution panoramic video inspection and documentation system.

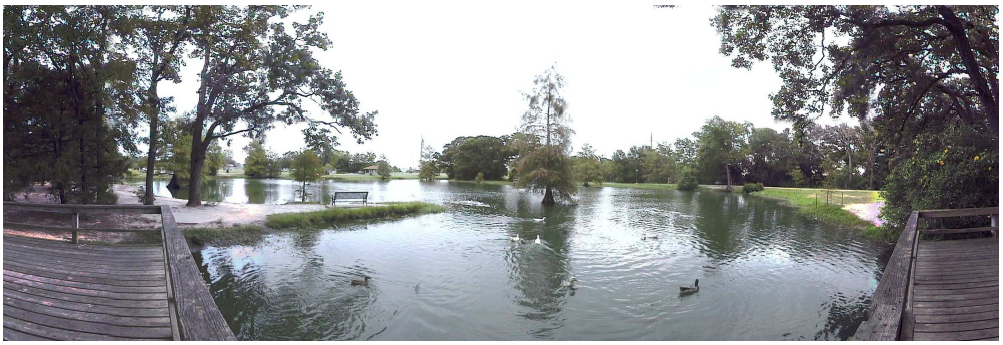
Development of the evolving panorama began in the summer of 2003. We began experiments with an improved Canon VCC4 camera our laboratory. As a test of concept, we built small panorama consisting of 8 frames and superimposed live video into the panorama to provide context and focus in the interface. We discovered that (1) nominal pan-tilt-zoom values do not provide adequate accuracy for frame registration and (2) traditional static panorama generation methods are either too slow to fit speed requirement or limited to simple small scale cylindrical panoramas.

In September of 2004, we moved our Canon VCC3 camera from the Stanley Hall construction site to the CITRIS Hall construction site approx 1/4 mile away on the UC Berkeley campus. This $120 million project will add 150,000 square feet of research and teaching space when it is completed in 2007. Figure 15(b) illustrates

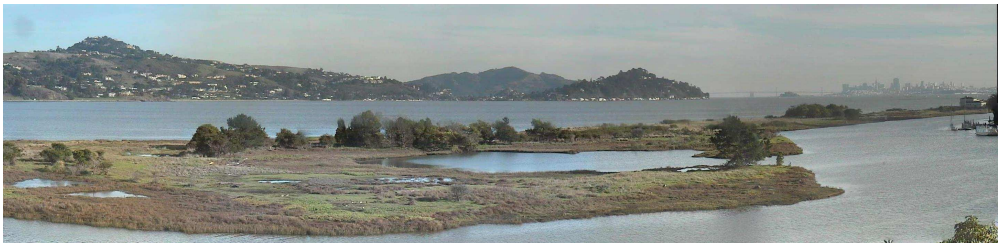
(a) Construction documentation of Stanley Hall building construction at UC Berkeley.



(b) Construction documentation of CITRIS II building at UC Berkeley.



(c) Pilot test of natural observation at Central Park, College Station, TX



(d) Natural Observation at Richardson Bay Audubon Sanctuary, San Francisco Bay.

Fig. 15. A snapshot of panoramic video created for bird watching.

the camera view at a resolution of $2600 * 900$ pixels. Our new panorama generation algorithms significantly reduce the panorama construction and update time from over 180 seconds to construct the 8-frame panorama in October of 2003 to only 9.7 seconds to construct the 21-frame panorama in Figure 15(b).

After the panorama is constructed, it only takes 331 milliseconds to update it, which allows it to be updated in real time and allows user to steer the Canon VCC4 PTZ camera to patrol the entire observation site to generate panorama on the fly. Other than minor interruptions caused by hardware failure and network upgrade, the system has been online stably and continuously for approximately 9 months. We have archived more than 3150 frame sequences and generated 149 panoramic images of the construction progress. The CITRIS Hall contruction management team uses our system daily to track progress.

We select some panoramas collected to create time-elapsing motion panorama for building construction progress documentation. The resulting motion panorama contains 103 invidual panoramas from Feb 10, 2005 to June 2, 2005. Some panoramas are not selected in the final motion panroama because of bad weather and lack of construction progress during holidays. At a resultion of 2600×900 pixels, Figure 15(b) is a snapshot of the motion panorama.

We also apply our algorithm to natural observation. As a pilot test, Figure 15(c) illustrates a snapshot of motion panorama generated during bird watching. Experiments were conducted from Aug 24, 2005 to Aug 31, 2005. We have collected 2186 frames and the original panorama has a resolution of 4000×1000 with a 240° horizontal field of view and 60° vertical field of view. The camera used is a Panasonic HCM 280 networked pan-tilt-zoom camera. Under the same setup, Figure 15(d) illustrates a panorama generated by PIIAA at an installation site for natural observation.

CHAPTER VI

ON-DEMAND HIGH RESOLUTION PANORAMA VIDEO STEAMING

With high zoom and large pan/tilt capabilities, remote observation system equipped with robotic cameras is able to achieve giga-pixel resolution. It is often the case that multiple users including nature scientists and the general public want to share the panorama video output at the same time. Each user may want to observe a different sub region and time window of the panorama video. Transmitting the full-sized ever-changing giga-pixel panorama video to every user is unnecessary and expensive in bandwidth requirement. In this section, we present systems and algorithms that allow on-demand sharing of a high-resolution panorama video. It is the first panorama video system that is designed to efficiently deal with multiple different spatiotemporal requests. We propose a patch-based approach in a spherical coordinate system to organize data captured by cameras at the server end. Built on an existing video-streaming protocol, the patch-based approach allows efficient on-demand transmission of the request regions.

A. Inputs and Assumptions

1. Evolving Panorama Video

An evolving panorama video is the data representation we design to deal with spatiotemporal camera frame inputs and user requests. The evolving panorama video is not a panorama but a collection of individual frames with timestamped registration parameters. The registration parameters allow the frame to be registered as part of a virtual spherical panorama.

A panorama is usually constructed by projecting frames taken at different camera

configurations into a common coordinate system, which is referred to as a composite panorama coordinate system. We choose a spherical coordinate system as the composite panorama coordinate system due to its relative small distortion if compared to a planar panorama composite coordinate system and large tilt coverage if compared to a cylindrical panorama composite coordinate system. In section III, we have shown that image alignment on the same spherical surface can be performed very efficiently because there exist projection invariants to allow the quick computation of registration parameters. Using a pre-calibrated camera, a point $q = (u, v)^T$ in a newly-arrived video frame F is projected to the point $\tilde{q} = (\tilde{u}, \tilde{v})^T$ in $\tilde{F}$ in the spherical coordinate system. The spherical coordinate system is centered at the lens optical center and has its radius equal to focal length f of the lens. The spherical pre-projection that projects q to $\tilde{q}$ is,

$$\tilde{u} = \arctan\left(\frac{u}{f}\right), \tag{6.1a}$$

$$\tilde{v} = -\arctan\left(\frac{v}{\sqrt{u^2 + f^2}}\right). \tag{6.1b}$$

Each point $(\tilde{u}, \tilde{v})^T$ in $\tilde{F}$ is defined using local pan and tilt spherical coordinates with units of radians. This is a local spherical coordinate because it forces the camera's optical axis to overlap with vector $(\tilde{u} = 0, \tilde{v} = 0)$. The next step is to re-project the local spherical coordinate to a global spherical coordinate to obtain image registration parameters using image alignment. The concept of an evolving panorama video builds on the fact that the panorama is continuously updated by the incoming camera frames. In fact, we do not store and build the whole panorama in order to avoid expensive computation.

Different clients might have different spatiotemporal requests. It is important to understand the relationship between the evolving panorama video and user requests.

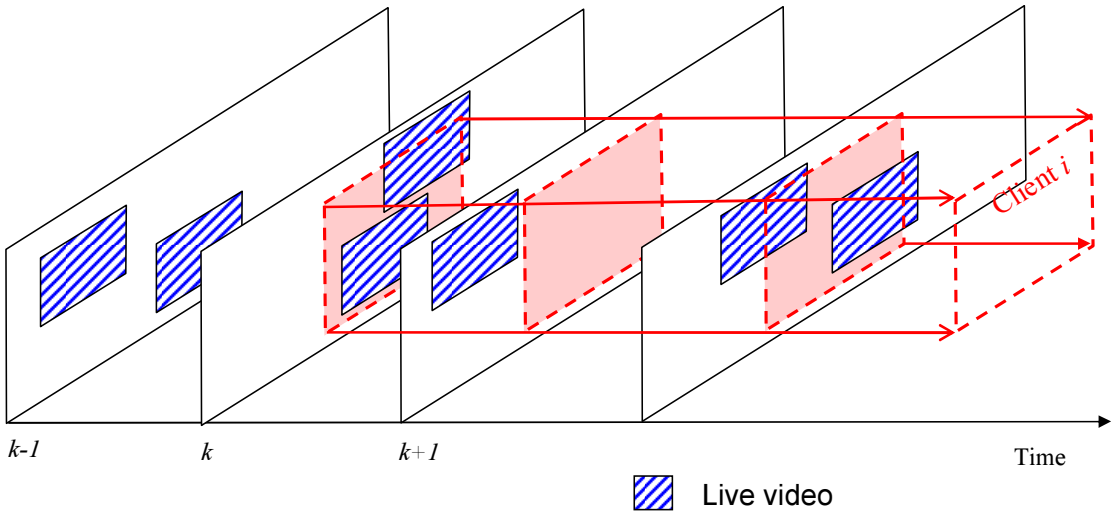


Fig. 16. The relationship between the evolving panorama and a user request. The striped regions indicate how the evolving panorama updates as camera frames arrive. The shaded box indicates the part of the data the user queries.

For a giga-pixel panorama video, it is impractical to transmit the entire video sequence due to bandwidth limitations. The screen resolution of the display device also limits the resolution of the video. Additionally, a user might not be interested in the entire viewable region. As illustrated in Figure 16, a typical user request can be viewed as a 3D rectangular query box in space and time. Define r_i as the ith request,

$$r_i = [u, v, w, h, t_s, t_e], \tag{6.2}$$

where (u, v) defines the center position of the requested rectangle on the panorama, w and h are width and height of the rectangle, and time interval $[t_s, t_e]$ defines the time window of the request. Figure 16 only illustrates a single user request. At any time k, there may be many different concurrent requests. Addressing the need of different

and concurrent requests is the requirement for our system.

With the concept of the evolving panorama and user requests, we are ready to introduce the data representation and algorithms for the system.

B. Data Representation and Algorithms

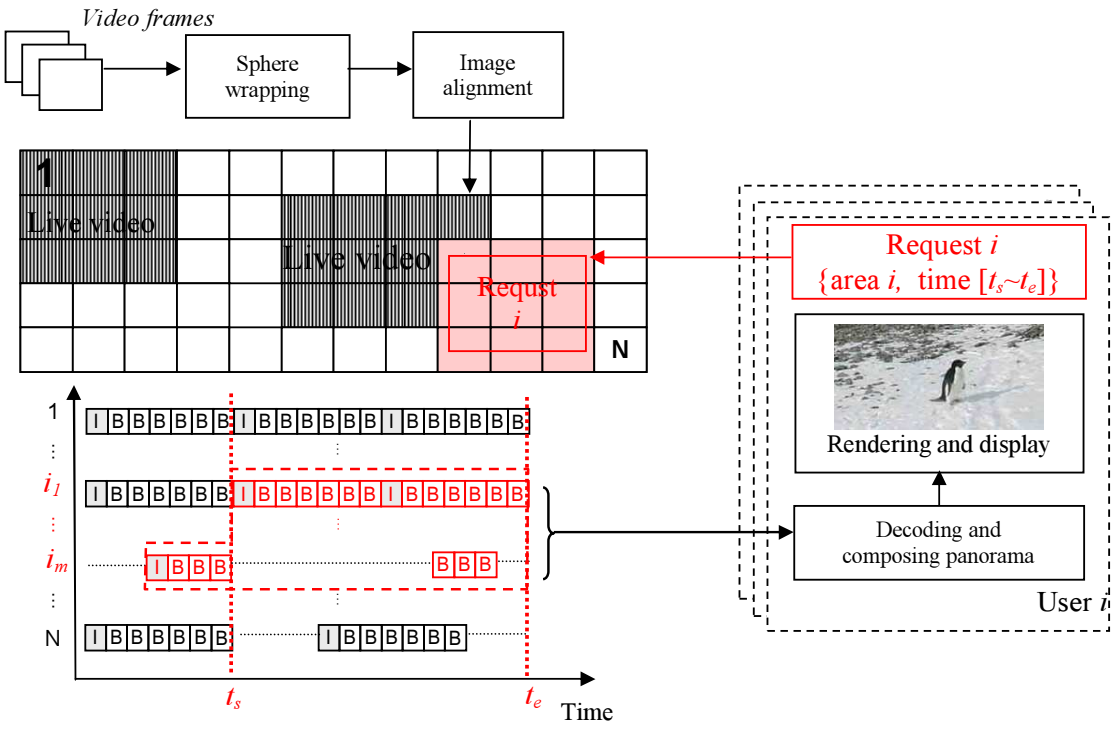


Fig. 17. Evolving panorama video system diagram. The left hand side illustrates the server side. The right hand side is a user at the client side. The grid at server represents a patch-based high-resolution panorama video system that allows multiple users to query different parts of the video concurrently. I's and B's indicate the I-frame and the B-frame used in MPEG-2 compression. A user sends a spatiotemporal request to server side and to retrieve the part of his/her interests in the panorama.

We propose a patch-based panorama video data representation. This data representation allows us to partition the image space and allows partial update and partial

retrieval. Built on the data representation, we then present a frame insertion algorithm and a user query algorithm. To illustrate the idea, we build our algorithms based on the MPEG-2 streaming protocol, which is the most popular protocol that can be decoded by a majority of client devices. However, the design can be easily extended to more recent protocols such as the MPEG-4 family for better compression and performance.

1. Patch-based Evolving Panorama Video Representation

We partition the panorama video into patches and encode each patch individually using MPEG-2 algorithms. The grid in Figure 17 shows a snapshot of the patch-based panorama at a given time. Only a subset of patches contain live video data because cameras cannot provide full coverage of the entire viewable region at a high-zoom setting. The panorama snapshot is a mixture of live patches and static patches. Let us define the jth patch as p_j, $j = 1, ..., N$ for a total of N patches. Each patch contains a set of video data $p_j = \{p_{jk} | k = 1, ..., \infty\}$ across the time dimension. Define F_k as the camera coverage in the viewable region at time k. If p_j intersects with F_k, p_{jk} contains live video data at time k. Otherwise, p_{jk} is empty and does not need to be stored. To summarize this, the whole patch-based evolving panorama video $\mathbb{P}_t$ at time t is a collection of live patches p_{jk}s,

$$\mathbb{P}_t = \{p_{jk} | j = 1, ..., N, k = 1, ..., t, p_{jk} \cap F_k \neq \emptyset\}. \tag{6.3}$$

2. Frame Insertion Algorithm

When a new video frame F_t arrives at time t, we need to update $\mathbb{P}_{t-1}$ to get $\mathbb{P}_t$,

$$\mathbb{P}_t = \mathbb{P}_{t-1} \cup \{p_{jt} | j \in \{1, ..., N\}, p_{jt} \cap F_t \neq \emptyset\}. \tag{6.4}$$

Implementing Equation (6.4) on the streaming server is nontrivial. As illustrated in Figure 17, for raw video frame F_t, its extrinsic camera parameters are first estimated by aligning with previous frames. The alignment process is performed on the spherical surface coordinate system. Next, we project the frame F_t onto the composite panorama spherical coordinate system. For each patch p_j intersecting with F_t, we encode it individually. We use an MPEG-2 encoder for patch encoding in our implementation. As with any MPEG-2 encoders, the size boundary for the number of frames inside one group of pictures (GOP) is predefined. Each GOP contains one I frame and the rest of the frames are either P frames or B frames. The size of the GOP should not be too large for quick random temporal video retrieval. Each patch holds its own GOP buffer. If the patch p_j intersects the current frame F_t, the updated patch data are inserted into patch video sequence P_j's GOP buffer. Whenever the GOP buffer reaches its size limit, we encode it using the standard MPEG-2. Since only a partial area of the panorama contains live video data at a certain time range and the number of the frames inside the GOP is predefined, the patch video data p_{jk} inside one patch video segment are not necessarily continuous in the time dimension. We summarize the patch-based evolving panorama video encoding algorithm below.

Algorithm 6: Frame Insertion Algorithm

input : F_t

output: Updated evolving panorama video

wrap F_t onto the spherical surface;

estimate F_t's registration parameters by aligning it with previous frames;

project F_t onto the sphere panorama surface;

for *each* p_j *and* $p_j \cap F_t \neq \emptyset$ **do**
    insert p_{jt} into p_j's GOP buffer;

for *each* p_j, $j = 1, ..., N$ **do**
    if p_j*'s GOP buffer is full* **then**
        encode patch video segment;
        store patch video segment start position and time data into lookup table;
        reset GOP buffer for incoming data;

3. User Query Algorithm

At time t, the system receives the ith user request $r_i = [u, v, w, h, t_s, t_e]$. To satisfy the request, we need to send the following data to the user at time t,

$$r_i \cap \mathbb{P}_t = \{p_{jk} | j \in \{1, ..., N\}, k \in [t_s, t_e], \\ p_{jk} \cap r_i \neq \emptyset, p_{jk} \neq \emptyset\}. \tag{6.5}$$

We implement this query as follows: for each p_j we keep track of its start position and the timestamp of I frames in a lookup table, which is used for random spatiotemporal video access. After receiving r_i, the streaming server first locates the nearest I frame with respect to t_s and t_e. If the streaming server identifies there is no live data in patch p_j in the requested time range, no additional video data is transmitted for patch

p_j. This procedure can be summarized as the following algorithm.

Algorithm 7: User Query Algorithm

input : r_i

output: $r_i \cap \mathbb{P}$ in MPEG-2 format

Identify patch set $S = \{p_j | j \in \{1, ..., N\}, p_j \cap r_i \neq \emptyset\}$;

for *each* $p_j \in S$ **do**

- find the nearest I frame p_{jb} earlier or equal to t_s;
- find the nearest I frame p_{jc} later or equal to t_e;
- transmit the patch segments between p_{jb} and p_{jc};

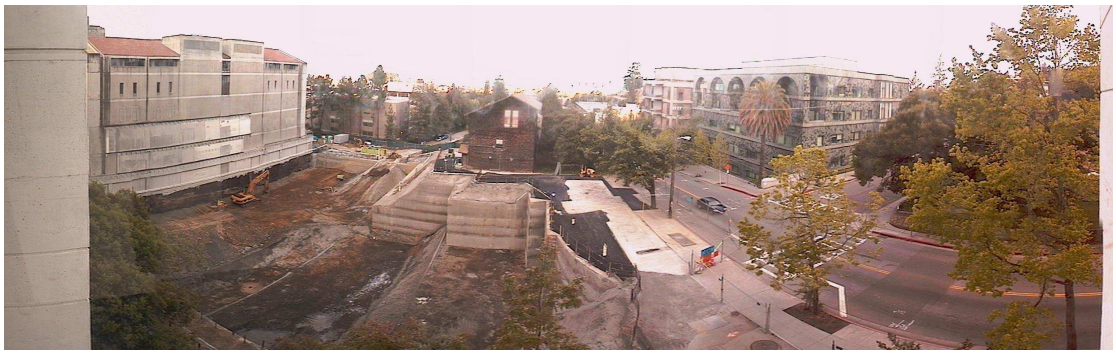
The decoding procedure at the client side is the standard MPEG-2 decoding. It is worth mentioning that the output of the system is not always a video segment. As illustrated in Figure 16, a user-requested region does not overlap with camera coverage at time $k+1$. It is possible that a user request might not intersect with any camera frames for the entire query time window $[t_s, t_e]$. For this situation, this algorithm will output an I-frame that is closest to $[t_s, t_e]$. Therefore, it sends a static image closest to the request. If the user request happens to be overlapped with current live camera coverage, the user receives live video. This algorithm allows three types of outputs: a pre-stored video, a live video, and a static image.

C. Experiments and Results

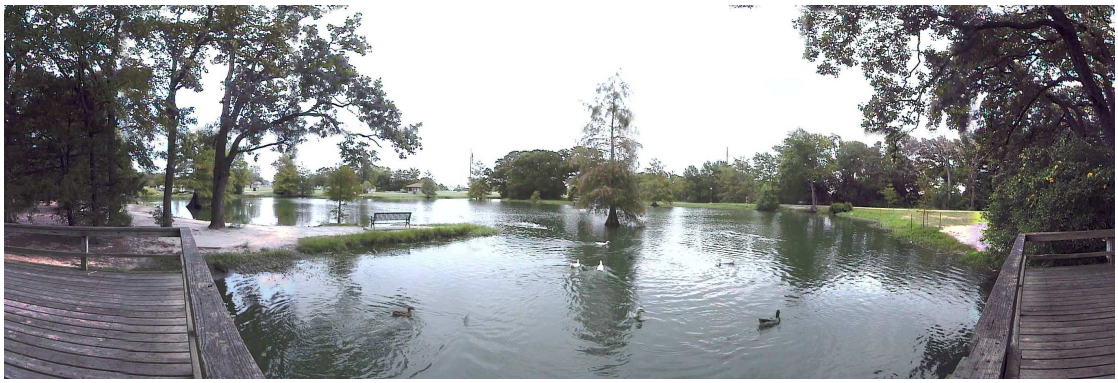
We test our algorithms using a Dell Dimension DX with a 3.2Ghz Pentium dual-core processor and 2GB RAM. The video camera is a Panasonic HCM 280a. It has a $2.8^\circ - 51^\circ$ horizontal field of view. We have implemented our algorithms using Visual C++ in Microsoft Visual Studio 2003.NET and adopted the MPEG-2 encoder and decoder source code developed by the MPEG Software Simulation Group.

We have conducted experiments using the data from field tests. As illustrated in

Figure 18, we have deployed our camera in two testing fields including a construction site at UC Berkeley and a pond in Central Park, College Station, Texas. We have collected data at both sites. For the construction site, data cover a duration from Feb. 10, 2005 to Jun. 2, 2005. The camera has been controlled by both online users and a pre-programmed patrolling sequence. Data collected in the park cover the experiment duration of Aug. 24, 2005 to Aug. 31, 2005. The construction site provides an urban environment setting while tests in the park provide a natural environment setting.



(a) Construction site of the CITRIS II building at UC Berkeley.



(b) Central Park, College Station, TX

Fig. 18. Experiment sites.

The data for each trial consist of 609 image frames captured at a resolution of

640×480. For a frame rate of 25 frames per second, the data represent 24 seconds of recording by the HCM 280a. The overall raw RGB data file size is 536 megabytes for the 24-bit color depth used in the experiment. The constructed panorama has an overall resolution of 2742×909 after cropping the uneven edges. The panorama size is much smaller than what the camera can provide (i.e. giga-pixel level). Since our tests involve speed tests, a large image file will involve an excessive mixture of RAM and disk operations, which could bias the speed test results. Using a smaller data set can minimize disk-seeking operations and reveal the real difference in computation speed.

In the first test, we are interested in testing how much storage savings we can gain from the design and how much computation time is needed to achieve the gain. During all the tests, we set the MPEG-2 quantization level to 50 without a rate limit. Therefore, we can compare the size of the video file data at the same video quality at different patch size settings.

Table IV. Storage and computation speed versus different patch sizes.

	Patch size	#Patches	File size (kb)	Speed
1	96×96	290	8044	6.9x
2	128×96	220	8191	6.4x
3	256×192	55	8871	5.0x
4	320×240	36	9965	3.8x
5	480×320	18	11099	3.1x
6	2742×909	1	22163	1x

The last row in Table IV actually encodes the entire panorama video at once without using patches, which is used as the benchmarking case. In this case, we update and generate a full panorama for each arriving camera frame. Then the full panorama is added into the GOP for encoding (same as [55]). The file size in Table IV is displayed in units of kilobytes. Smaller file size means less storage and is

preferable. It is interesting to see that patch-based approach has significant savings in storage. This is expected because our system does not encode the un-updated part of the panorama as opposed to the benchmarking case which repeatedly encodes the un-updated regions. The speed column compares the computation speed under the various patch size settings with the benchmarking case. As shown in the Table IV, encoding the entire panorama in the benchmarking case takes more time than that of the patch-based approach. The computation speed gets faster as the patch size reduces. This can be explained by two reasons 1) less data: we do not repeatedly encode the un-updated region and 2) smaller problem space: the block matching problem space is much smaller for a smaller patch size in the MPEG-2 encoding.

In the second test, we are interested in studying how much bandwidth is needed for a normal user query. We assume that user has a screen resolution of 800×600. Therefore, the request follows the same size. We know that the bandwidth requirement depends on how many patches the request intersects with. We study two cases including the best-case scenario and the worst-case scenario. The best-case scenario refers to the case that the request intersects with the least number of patches. The worst-case scenario is the opposite. Again, the last row of the table is the benchmark case. Table V summarizes the test results. As expected, a smaller patch size is preferred because it requires less bandwidth.

Table V. Bandwidth for a user query versus different patch sizes.

	Patch size	Worst case (kbps)	Best case (kbps)
1	96×96	739.7	582.5
2	128×96	794.3	608.1
3	256×192	1344.1	860.2
4	320×240	1476.3	830.4
5	480×320	1849.8	822.1
6	2742×909	7387.7	7387.7

CHAPTER VII

CONCLUSION AND FUTURE WORK

We present system architecture, data representation, algorithms and protocols of high resolution motion panorama for remote observation using networked robotic cameras. We have analyzed, derived, and proved that projection invariants under spherical coordinate systems. We present a projection invariant-based image alignment algorithm, which outperformed the best algorithm available by at least an order of a magnitude.

We propose a variance-based quality metric to analyze how errors get accumulated and use it to show that arbitrarily selecting a set of existing frames to register new frames can cause registration errors to grow out of control in the incremental frame registration process. We then propose a minimum variance alignment algorithm to guarantee the quality of motion panorama over the long run. Our algorithm can register a new frame in $O(k \log k)$ time for a panorama with k overlapping frames.

We propose a Frame Graph based panorama documentation algorithm including frame insertion, archiving and adjustment operations to efficiently manage the online panorama documentation.

We propose a patch-based panorama video encoding/decoding system that allows multiple online users to share access to pan-tilt-zoom cameras with various spatiotemporal requests. We have implemented the system and conducted field tests. The experiments have shown that our system can significantly reduce the storage needs and bandwidth requirements of online users.

A intelligent remote observation systems can provide the content understanding ability. In Fall 2005, we join the search effort for legendary Ivory-Billed Wookpecker (IBWO). We develop a autonomous nature observation system equipped with high

resolution robotic video cameras. Our system has been installed in Brinkley, Arkansas since Oct. 26, 2006. In the future, we will develop biometric filter algorithm to compute the probability of matching a moving object with a known species from image sequences. The biometric filter algorithm will extract both bird first order information such as shape and size and second order bird data velocity by applying Extended Kalman Filter (EKF) in velocity filter.

REFERENCES

[1] L. W. Gysel and E. M. J. Davis, "A simple automatic photographic unit for wildlife research," *Journal of Wildlife Management*, vol. 20, pp. 451–453, 1956.

[2] T. E. Kucera and R. H. Barrett, "The trailmaster camera system for detecting wildlife," *Wildlife Society Bulletin*, vol. 21, pp. 505–508, 1993.

[3] K. Iida, R. Takahashi, Y. Tang, T. Mukai, and M. Sato, "Observation of marine animals using underwater acoustic camera," *Japanese Journal of Applied Physics*, vol. 45, no. 5B, pp. 4875–4881, 2006.

[4] T. L. Cutler and D. E. Swann, "Using remote photography in wildlife ecology: A review," *Wildlife Society Bulletin*, vol. 27, no. 3, pp. 571–581, 1999.

[5] M. D. Sanders and R. F. Maloney, "Causes of mortality at nests of ground-nesting birds in the Upper Waitaki Basin, South Island, New Zealand: a 5-year video study," *Biological Conservation*, vol. 106, no. 2, pp. 225–236, 2002.

[6] M. M. Stake and D. A. Cimprich, "Using video to monitor predation at black-capped vireo nests," *BioOne*, vol. 105, no. 2, pp. 348–357, 2003.

[7] S. B. Lewis, P. DeSlmone, K. Titus, and M. R. Fuller, "A video surveillance system for monitoring raptor nests in a temperate rainforest environment," *Northwest Science*, vol. 78, no. 1, pp. 70–74, 2004.

[8] Africa WebCams, "http://www.zulucam.org/," 2005.

[9] Tigerhomes.org, "http://www.tigerhomes.org/ animal/ web-cams.cfm," 2005.

[10] D. E. Weber, B. Grosser, and G. Fried, "http://bugscope.beckman.uiuc.edu/," 2005.

[11] New York Wild, "http://www.newyorkwild.org/ webcams/ webcams.htm," 2005.

[12] James Reserve, "http://www.jamesreserve.edu/webcamsphp.lasso," 2005.

[13] National Geographic, "http://magma.nationalgeographic.com/ ngm/ cranecam/cam.html," 2005.

[14] Swan Cam, "http://www.osage.net/ mccb/trumpeterswan.html," 2005.

[15] WildCam.com, "http://www.wildcam.com/," 2005.

[16] L. Hitchner, "Virtual planetary exploration: A very large virtual environment," in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, Chicago, IL, July 1992, vol. 6, pp. 1–16.

[17] S. K. Nayar, "Catadioptric omnidirectional camera," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997, pp. 482–488.

[18] Y. Xiong and K. Turkowski, "Creating image-based VR using a self-calibrating fisheye lens," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997, pp. 237–243.

[19] R. Swaminathan and S. K. Nayar, "Polycameras: Camera clusters for wide angle imaging," Columbia University:TR CUCS-013-99, 1999.

[20] D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson, "Design and use of linear models for image motion analysis," *International Journal of Computer Vision*, vol. 36, no. 3, pp. 171-93, Feb.-Mar. 2000.

[21] D. Song and K. Goldberg, "Sharecam part I: Interface, system architecture, and implementation of a collaboratively controlled robotic webcam," in *IEEE/RSJ International Conference on Intelligent Robots*, Las Vegas, NV, Oct. 2003, vol. 2, pp. 1080 – 1086.

[22] D. Song, A. Pashkevich, and K. Goldberg, "Sharecam part II: Approximate and distributed algorithms for a collaboratively controlled robotic webcam," in *IEEE/RSJ International Conference on Intelligent Robots*, Las Vegas, NV, Oct. 2003, vol. 2, pp. 1087 – 1093.

[23] N. Chong, T. Kotoku, K. Ohba, K. Komoriya, N. Matsuhira, and K. Tanie, "Remote coordinated controls in multiple telerobot cooperation," in *IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000, vol. 4, pp. 3138–3343.

[24] D. J. Cannon, "Point-and-direct telerobotics: Object level strategic supervisory control in unstructured interactive human-machine system environments," Ph.D. dissertation, Stanford University, Palo Alto, CA, June 1992.

[25] M. McDonald, D. Small, C. Graves, and D. Cannon, "Virtual collaborative control to improve intelligent robotic system efficiency and quality," in *IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997, vol. 1, pp. 418–424.

[26] K. Goldberg and B. Chen, "Collaborative control of robot motion: Robustness to error," in *IEEE International Conference on Intelligent Robots and Systems*, Maui, HI, Oct. 2001, vol. 2, pp. 655–660.

[27] K. Goldberg, D. Song, and A. Levandowski, "Collaborative teleoperation using

networked spatial dynamic voting," *The Proceedings of the IEEE*, vol. 91, no. 3, pp. 430–439, Mar. 2003.

[28] R. Benosman and S. B. Kang, *Panoramic Vision*, Springer, New York, 2001.

[29] S. E. Chen, "QuickTime VR — an image-based approach to virtual environment navigation," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, Los Angeles, CA, Aug. 1995, vol. 29, pp. 29–38.

[30] B. Y. Kim, K. H. Jang, and S. K. Jung, "Adaptive strip compression for panorama video streaming," in *Computer Graphics International*, Crete, Greece, June 2004, pp. 470–475.

[31] R. Hartley, "Self-calibration of stationary cameras," in *International Journal of Computer Vision*, 1997, vol. 22, pp. 5–23.

[32] H. Shum and R. Szeliski, "Construction and refinement of panoramic mosaics with global and local alignment," in *Sixth International Conference on Computer Vision*, Mumbai, India, Jan. 1998, pp. 953–956.

[33] S. Coorg and S. Teller, "Spherical mosaics with quaternions and dense correlation," *International Journal of Computer Vision*, vol. 37, no. 3, pp. 259–273, 2000.

[34] S. B. Kang and R Weiss, "Characterization of errors in compositing panoramic images," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997, pp. 103–109.

[35] R. Szeliski, "Image mosaicing for tele-reality applications," in *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, Sarasota, FL, Dec. 1994, pp. 44–53.

[36] R. Szeliski, "Video mosaics for virtual environments," in *Proceedings of IEEE Computer Graphics and Applications*, Mar. 1996, vol. 16, pp. 22–30.

[37] R. Szeliski and H. Shum, "Creating full view panoramic image mosaics and environment maps," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, Los Angeles, CA, Aug. 1997, vol. 31, pp. 251–258.

[38] E. D. Castro and C. Morandi, "Registration of translated and rotated images using finite fourier transform," in *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1987, vol. 9, pp. 700–703.

[39] B. S. Reddy and B. N. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," in *Proceedings of IEEE Transactions on Image Processing*, Aug. 1996, vol. 5, pp. 1266–1271.

[40] C. J. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings 4th Alvey Vision Conference*, Manchester, UK, 1988, pp. 147–151.

[41] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Proceedings of the International Workshop on Vision Algorithm: Theory and Practice*, Corfu, Greece, Aug. 1999, pp. 278–294.

[42] Z. Zhang, R. Deriche, O. Faugeras, and Q. T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial Intelligence*, vol. 78, pp. 87–119, 1995.

[43] I. Zoghlami, O. Faugeras, and R. Deriche, "Using geometric corners to build a 2D mosaic from a set of images," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997, pp. 420–425.

[44] B. Hu, C. Brown, and A. Choi, "Acquiring an environment map through image mosaicking," in *University of Rochester:TR-786*, Nov. 2001.

[45] G. Borgefors, "Hierarchical chamfer matching: A parametric edge matching algorithm," in *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence*, Nov. 1988, vol. 10, pp. 849–865.

[46] H. Li, B. S. Manjunath, and S. K. Mitra, "A contour-based approach to multisensor image registration," in *Proceedings of IEEE Transactions on Image Processing*, Mar. 1995, vol. 4, pp. 320–334.

[47] S. H. Cho, Y. K. Chung, and J. Y. Lee, "Automatic image mosaic system using image feature detection and taylor series," in *Proceedings of VIIth Digital Image Computing: Techniques and Applications*, Sydney, Australia, Dec. 2003, pp. 549–556.

[48] Y. Kanazawa and K. Kanatani, "Image mosaicing by stratified matching," in *Proceedings of Statistical Methods in Video Processing Workshop*, Denmark, Jan. 2002, pp. 31–36.

[49] W. Zhang, J. Kosecka, and F. Li, "Mosaics construction from a sparse set of views," in *Proceedings of First International Symposium on 3D Data Processing Visualization and Transmission*, Padova, Italy, June 2002, pp. 177–180.

[50] M. Brown and D. Lowe, "Recognising panoramas," in *Proceedings of IEEE*

International Conference on Computer Vision, Nice, France, Oct. 2003, vol. 2, pp. 1218–1225.

[51] D. Forsyth, J. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell, "Invariant descriptors for 3D object recognition and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 971–991, Oct. 1991.

[52] K. Arbter, W. Snyder, H. Burkhardt, and G. Hirzinger, "Application of affline-invariant fourier descriptors to recognition of 3D objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 640–646, July 1990.

[53] R. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company, Reading, MA, 1992.

[54] S. Baker and S. K. Nayar, "A theory of single-viewpoint catadioptric image formation," *International Journal of Computer Vision*, vol. 35, no. 2, pp. 175 – 196, Nov. 1999.

[55] K. Ng, S. C. Chan, and H. Y. Shum, "Data compression and transmission aspects of panoramic videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 82–95, Jan. 2005.

[56] R. Swaminathan and S. K. Nayar, "Nonmetric calibration of wide-angle lenses and polycameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1172–1178, Oct. 2000.

[57] J. Foote and D. Kimber, "Enhancing distance learning with panoramic video," in *Proceedings of the 34th Hawaii International Conference on System Sciences*,

2001, pp. 1–7.

[58] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu, "Efficient representations of video sequences and their applications," *Signal Processing : Image Communication*, vol. 8, pp. 327–351, Nov. 1996.

[59] E. Trucco, A. Doull, F. Odone, A. Fusiello, and D. Lane, "Dynamic video mosaicing and augmented reality for subsea inspection and monitoring," in *Proceedings of the Oceanology International Conference*, Brighton, UK, Mar. 2000, pp. 297–306.

[60] Z. Zhu, G. Xu, E. M. Riseman, and A. R. Hanson, "Fast generation of dynamic and multi-resolution 360-degree panorama from video sequences," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, June 1999, vol. 1, pp. 9400–9406.

[61] A. Agarwala, C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, "Panoramic video textures," in *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, Los Angeles, CA, July 2005, vol. 24, pp. 821–827.

[62] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, "Dynamosaics: Video mosaics with non-chronological time," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2005, vol. 1, pp. 58–65.

[63] A. Bartoli, N. Dalal, and R. Horaud, "Motion panoramas," *Computer Animation and Virtual Worlds*, vol. 15, pp. 501–517, 2004.

[64] M. Irani, S. Hsu, and P. Anandan, "Video compression using mosaic representations," in *Signal Processing: Image Communication*, 1995, vol. 7, pp. 529–552.

[65] M. Irani and P. Anandan, "Video indexing based on mosaic representations," in *Proceedings of the IEEE*, May 1998, vol. 86, pp. 905–921.

[66] Richard Szeliski, "Image alignment and stitching, Microsoft Research, Technical Report MSR-TR-2004-92," 2004.

[67] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision, 2nd Edition*, Cambridge University Press, New York, NY, 2004.

[68] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 4, pp. 91–110, Nov. 2004.

[69] P. Agarwal and J. Erickson, *Geometric range searching and its relatives*, pp. 1–56, American Mathematical Society Press, Providence, RI, 1999.

[70] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer, New York, NY, 1991.

VITA

Name Ni Qin

Address Department of Computer Science, Texas A&M University, College Station, TX, 77843-3112

Email Address nqin@cs.tamu.edu

Education B.S., Biology, Wuhan University, 1997

M.C.S., Medical Biochemistry, University of Mississippi Medical Center, 2000

M.S., Computer Science, Texas A&M University, 2003

Ph.D., Computer Science, Texas A&M University, 2008