AN ABSTRACT OF THE PROJECT REPORT OF

Zhongyuan Feng for the degree of Master of Science in Computer Science presented on 05/22/2015.

Title: Efficient Incremental Panorama Reconstruction from Multiple Videos

Abstract approved: _____

Sinisa Todorovic

Constructing a panorama from a set of videos is a long-standing problem in computer vision. A panorama represents an enhanced still-image representation of an entire scene captured in a set of videos, where each video shows only a part of the scene. Importantly, a panorama shows only the scene background, whereas any foreground objects appearing in the videos are not of interest. This report presents a new framework for an efficient panorama extraction from a large set of 140-150 videos, each showing a play in a given football game. The resulting panorama is supposed to show the entire football field seen in the videos, without foreground players. Prior work typically processes all video frames for panorama extraction. This is computationally expensive. In contrast, we design an efficient approach which incrementally builds the panorama by intelligently selecting a sparse set of video frames to process. Our approach consists of the following steps. We first identify the moment of snap (MOS) in each play, because these video frames are usually captured without camera motion, and thus provide good-quality snap shots of the field. All detected MOS frames are then used to build an initial panorama using the standard Bundle Adjustment algorithm. The initial panorama is used in our subsequent steps to search for new video frames that will maximally cover yet unreconstructed parts the football field. For this, we compute a homographic projection of the MOS video frames onto the current panorama, and estimate an area on the reconstructed football field with the lowest confidence score. The score of an area is made directly proportional to the number of video frames covering that area. Then, we identify a new sparse set of video frames that maximally cover the lowest-confidence area in the current panorama.

Finally, this new set of frames and the current panorama are jointly input to the Bundle Adjustment algorithm to produce a new panorama estimate. This process is iterated until the confidence scores associated with different parts of the panorama stop changing, or when there are no new video frames that would extend the current panorama. In each iteration, we also adaptively build a background model of the football field, and in this way remove foreground from the panorama. We test our approach on a large dataset of videos showing 10 football games. The results demonstrate that our approach is more efficient than state-of-the-art methods, while yielding competitive results. To the best of our knowledge, prior work has not considered efficiency, but only accuracy of panorama extraction. Our approach has an advantage of being flexible to various time budgets set by real-world applications.

Efficient Incremental Panorama Reconstruction from Multiple Videos

by

Zhongyuan Feng

A PROJECT REPORT

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented 05/22/2015
Commencement June 2015

Master of Science project report of Zhongyuan Feng presented on 05/22/2015.

APPROVED:

Major Professor, representing Computer Science

Director of the Oregon State University of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my project report will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my project report to any reader upon request.

Zhongyuan Feng, Author

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Motivation

1.1 Background

A panorama represents an enhanced still-image representation of an entire scene captured in image sequences or a set of videos, where each image or video shows only a part of the scene. In particular, the panorama in this report is composed of video clips captured by hand-held cameras from American football play fields. Panorama construction has been widely used for many applications since it was applied to track human bodies from static cameras at 1997 [33]. Other applications include surveillance video analysis [13], sport videos analysis [16], scene segmentation [11], content-based video genre classification [27], video summarization [32], video compression [20] and multimedia [9].

Panorama reconstruction is a long-standing problem. Traditionally, background panorama is generated either by direct methods that iteratively minimize errors with all available data within a certain overlapped regions, or by feature based methods that apply feature extraction from images, e.g., Harris corners or SIFT features, for geometrical relationship estimation, that is, homography matrices. Direct methods have the advantage of precise representation of pixel-wise background information, and feature based matching methods is robust to image scaling or rotation issues. However, both methods have been criticized for various reasons. Direct methods typically vulnerable to dynamic environment, such as light illumination, change of time, motions caused by tree leaves or bushes. While processing feature extraction and matching is computational expensive, thus it is only applicable in limited situations.

To this end, we extend existing technologies in this report, and present a new framework to efficiently reconstruct background panorama from large dataset. Figure 1.1 shows an overview of out framework. Specifically, the framework consists of 3 steps: model initialization, key-frame selection and incremental update. At the first step, a frame subset is selected from our dataset, and used to initialize panorama as well as *Heat Map* for subsequent key-frame selection process. Intuitively, heat map indicates the pixel-wise confidence of reconstructed panorama at intermediate steps. That is, pixels with higher

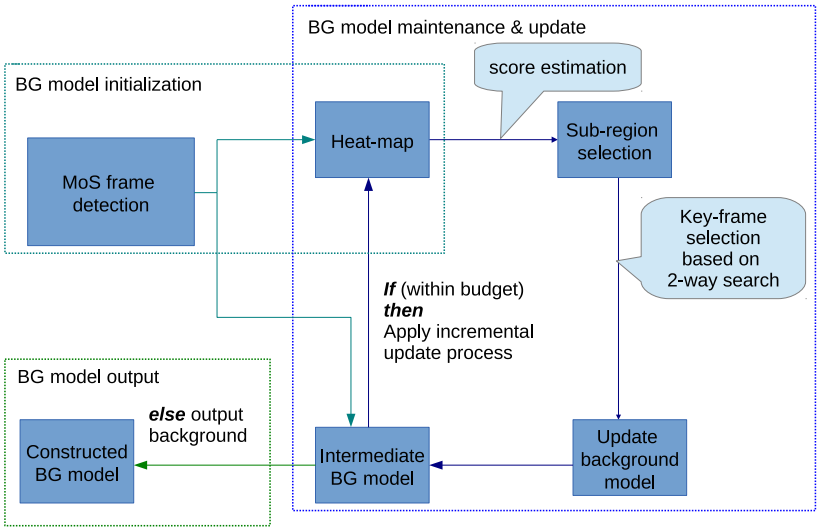

Figure 1.1: Our incremental panorama construction framework from multiple videos.

values indicate lower confidence or correctness probability of current areas. To perform update incrementally, the heat map is divided into different areas based on overlaps of projected frame, and sub-region selection is then accomplished based on estimated utilities or scores. A sub-region with the largest heat value is then selected, and key-frame selection algorithm is used to update subset as well as background panorama. During the whole process, an efficient feature filter is also applied to improve feature matching efficiency. This process repeats until computation budget limitation is reached. This idea is inspired by followed analysis. For one thing, among almost given videos, we observed that most neighboring frames can be overlapped with high proportional identical fields when the motion of camera is very small or equals zero. To avoid redundant information as much as possible, such frames should be used as fewer as possible. For another, the probability, or the certainty of panorama construction will decrease as long as the number of available matched features belonging to background field decreases sharply. This often happens when camera moves fast or zooms in/out quickly to emphasize particular players/events. To minimize the construction error, more frames should be used to project and cover such ill-featured areas with lower construction confidence.

For another, model initialization is an essential step. The selected subset should be able to cover background scene as much as possible so that main-structure of the scene can be captured so as to minimize exploration steps during update and overall runtime as well. Another important issue is that this is also important to connect those isolated potential sub-regions, which can be found from either initial images, or explored in intermediate steps.

In summary, our contribution in this report includes: firstly, we apply a new strategy for electing a subset of images from the large input set, referred to as key-frame selection algorithm. Secondly, we present a new method for removing low-quality features as a pre-process step for feature matching. Thirdly, we design a new strategy for iteratively selecting which new images to process for updating the panorama.

1.2 Outline

Below is the outline organized for the left parts.

In chapter 2 we gives literature overview about panorama reconstruction problem as well as their limitations within our experiment settings. In chapter 3 we present detailed introduction to those techniques to be used for our framework. We explain our framework with more details in chapter 4, and gives our experimental results as well as discussions in chapter 5. In chapter 6 we discuss our plan for works to do in the future.

Chapter 2: Literature Review

Panorama reconstruction plays an important role in computer vision. A panorama represents an enhanced still-image representation of an entire scene captured in a set of videos, where each video shows only a part of the scene. Importantly, a panorama shows only the scene background, whereas any foreground objects appearing in the videos are not of interest. In recent years, it has been used in many applications, such as video surveillance [6, 5], multimedia [17, 9], action recognition [30] and sport video analysis [26, 1, 16]. Generally speaking, most panorama reconstruction works can be categorized into two types: direct method and feature based method.

2.1 Direct Method

Direct method, also referred to as pixel-wise based method, tries to minimize a predefined cost function by repeatedly updating parameters in an overlapped window, with all available input data. Usually sum squared errors (SSE) or weighted version (WSSE) are defined as cost evaluation for this purpose, and the error is defined as identity differences [18]. Direct method has the advantage of providing precise solution given available data, meanwhile, it has an intrinsic weakness that cannot well deal with data with dynamic environment. Additionally, due to search process only depends on fixed steps, each pixel would be processed in multiple times, which make it computationally expensive. To accelerate the search step, other alternative approaches were proposed like hierarchical method, which tries to narrow search space via image pyramid; or Fourier based method, which has the advantage of narrowing search space, at the same time can still keep signal magnitude when those signals are blurred away by hierarchial methods [29].

2.2 Feature Based Method

Anther method to reconstruct the panorama, which is more popular, is to extract interested features, then do feature matching as well as image matching for stitching, under

the assumption that all contained background areas (or point correspondences) belong to (or lie on) the same planar surface [4]. For example, Scale-Invariant Feature Transform (SIFT) [23] are often used since it is rotation and partial affine transformation invariant. In addition, some mid-level features such as lines [35] or ellipses [19] are also used to capture more contextual constraints. Recently, other features like PCA-SIFT, SURF or FAST are also used for feature representation.

However, in real world applications, input data are not always contain sufficient features that are qualified for homography evaluation. For example, in those videos capturing from sport matches, such as football match or soccer match, camera often zooms in or out to capture, emphasize or repeat certain significant events, players or event trends. Consequently, most mid-level features would not be available, meanwhile a large proportion of features would be extracted from players or other moving foreground objects. As the result, the estimated homographies will not correctly reflect the actual geometrical relationships between frames and field model. For the aforementioned problem, one can simply add more frames, exploiting more global distinctive features belonging to such ill-featured areas, from which more accurate homographies can be provided. Generally, given that the allowed computational budget is large enough, such areas would diminish gradually by combining more correctly projected frames.

Instead of trying to find global features, [16] managed to find more local non-distinctive image feature correspondences, with the aid of tracked features from previous steps and currently detected global distinctive features from frame sequences of a video. A "seed" frame is selected for robust initialization, then all features are validated and propagated to the next frame via local non-distinctive feature tracking, which will in return helps tracking more matched local features for the next frames. A stability test is used for selecting the seed frame, from which the features would result in the most stable registration transform. [16] experimentally show that the proposed method outperforms other methods which are only capable of capturing global feature correspondences.

2.3 Sparse Kernel Method

Alternatively, [10, 1, 12] proposed a new framework by treating the background image as low-rank matrix, and foreground as sparse error given an observation, under the assumption that the foreground is sufficiently sparse. Solving this rank minimization problem

via the technology called Inexact Augmented Lagrangian Multiplier [21], by solving its relaxed L1-norm problem, the author showed that this can handle more complex circumstances with the presence of camera motions. Moreover, if the sparsity constraint holds everywhere, applying random sub-sampling method would make it more efficient, which is theoretically guaranteed to converge to the same solution.

Strictly speaking, those new approaches, as mentioned above, are general frameworks for modeling background, still, with competitive results, they provide a promising way to solve general panorama reconstruction problem. However, although they are robust to dynamic environment, it is not suitable for American football video dataset. Since update of low-rank approach depends on the solution from previous steps, the sparsity constraint is easily violated due to the "zoom-in/out" situation or with fast camera motions, which often happens in most real-world sport videos.

Chapter 3: Introduction

3.1 Feature Extraction & Matching

As mentioned in previous section, to reconstruct field panorama, the first step is model initialization via selected frame subset, followed by feature extraction and pairwise matching for homography estimation between image pairs.

Similar to previous work of [4], SIFT features are used for our approach and used later to estimate 3×3 homography matrices. As pointed out in [23], SIFT feature is defined as scale-space extreme point, and described as normalized orientation histograms. This gives SIFT features the property of more robust to affine transformation compared to other features, like Harris Corners. Therefore, SIFT feature has the advantage of invariant to image rotation or distortion caused by affine transformation, thus large portion of motions caused by either cameras jitters, or moving objects can be eliminated, this provides better image stabilization and registration results.

Modern methods often apply Approximate Nearest Neighbor to accelerate the matching step. A widely used method is mentioned by [23], where two best matched feature candidates are founded firstly, and only those candidates with larger ratios (ratio of distance between the second best matched feature and the first best matched feature) than a certain threshold are kept.

To accelerate feature matching process, KD-Tree is constructed for fast feature indexing. The general idea is to construct a balanced tree in high dimensional space F^m (feature space, where $m = 128$ for SIFT features). By selecting a certain dimension (such as median), the feature set is then split in two halves with nearly equal size. Thus the look-up process can be performed within $O(\log_2 N)$ instead of $O(N^2)$, where N is the number of features.

The next step is to select two nearest candidates to current feature. Suppose their distances are denoted as D_{f,f_1}, D_{f,f_2}. Only the nearest feature satisfying $r_{dist} = \frac{D_{f,f_1}}{D_{f,f_2}} \leq D_{thres}$ will be accepted and labeled as the best match, where D_{thres} is a predefined constant. We recommend [2] with more details.

3.2 Feature Matchability Estimation

Feature matching is a fundamental problem in many computer vision applications, and plays an essential role in computation efficiency with large dataset. For example, a frame with 800×480 resolution normally contains about 500 qualified SIFT features. Additionally, feature matching is also time-consuming. Alternatively, one can also accelerate matching processing via either decreasing the size of dataset, with the risk of losing partial field areas, or adjusting threshold to higher value and only keeping smaller matched candidate set, with the risk of inaccurate estimated homographies due to insufficient matched features.

Instead we predict which features are "good" or not for homography estimation before feature matching step. By doing this we have the following advantages: firstly, more features extracted from background field can be kept, thus filtering out others like those extracted from foreground objects, even they have higher match potentials. Take an example of American football video clips, cameras often focus on players when the play starts or an significant event happens such as kickoff, thus most image or frame contents are composed of foreground areas. With feature filters, ambiguous features can be safely removed without jeopardizing robustness of homographies. Additionally, it will give us much smaller number of features that will perform matching at the next step.

Inspired by [15], a Random Forest classification framework is employed for this task. Random forest has the advantage of efficient training process, even with large training dataset, while still be capable of keeping high accuracy performance compared to other complicated models, such as SVM or Neural Network. Additionally, due to the internal nature of the framework, it is no need to do cross-validation during training process for unbiased models, which is often done by repeatedly doing bootstrap sampling in multiple times.

3.3 Homography Estimation

Given an image pair I_i, I_j, as well as matched features $\boldsymbol{f_i^m}, \boldsymbol{f_j^n}$ indexed by m, n respectively, the goal is to estimate homography $\boldsymbol{H_{ij}}$ [1] such that $\boldsymbol{H_{ij}}\boldsymbol{f_i^m} = \boldsymbol{f_j^n}$. Here 2D coordinate features f usually use homogeneous representation $(x, y) \rightarrow (x, y, z), z = 1$,

[1] $\boldsymbol{H_{ij}}$ denotes geometric transformation from image I_i to I_j.

and (x, y) represents feature position of an image in Cartesian space. Unfold equation, we have[2]

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_i^m \\ y_i^m \\ z_i^m \end{bmatrix} = \begin{bmatrix} x_j^n \\ y_j^n \\ z_j^n \end{bmatrix} \tag{3.1}$$

Transformation between the two representation $(x^{'}, y^{'}) \leftrightarrow (x, y, z)$ can be done by the following equations,

$$\begin{aligned} x_j^{n^{'}} &= \frac{H_{11}x_i^m + H_{12}y_i^m + H_{13}z_i^m}{H_{31}x_i^m + H_{32}y_i^m + H_{33}z_i^m} \\ y_j^{n^{'}} &= \frac{H_{21}x_i^m + H_{22}y_i^m + H_{23}z_i^m}{H_{31}x_i^m + H_{32}y_i^m + H_{33}z_i^m} \\ s.t.\ x_j^{n^{'}} &= \frac{x_j^n}{z_j^n}, y_j^{n^{'}} = \frac{y_j^n}{z_j^n} \end{aligned} \tag{3.2}$$

With the fact that factor $z = 1$,

$$\begin{aligned} x_j^{n^{'}}(H_{31}x_i^m + H_{32}y_i^m + H_{33}) &= H_{11}x_i^m + H_{12}y_i^m + H_{13} \\ y_j^{n^{'}}(H_{31}x_i^m + H_{32}y_i^m + H_{33}) &= H_{21}x_i^m + H_{22}y_i^m + H_{23} \end{aligned} \tag{3.3}$$

Re-writing the above equations,[3]

$$\begin{aligned} \boldsymbol{a_x h} &= \mathbf{0} \\ \boldsymbol{a_y h} &= \mathbf{0} \\ \boldsymbol{h} &= (\boldsymbol{h^1}, \boldsymbol{h^2}, \boldsymbol{h^3})^T \end{aligned} \tag{3.4}$$

[2] We use notation $\boldsymbol{H}$ here instead of $\boldsymbol{H_{ij}}$ for clearance.

[3] $\boldsymbol{h}$ indicates decomposition of $\boldsymbol{H}$ row by row.

where

$$\begin{aligned} \boldsymbol{a_x} &= (-x_i^m, -y_i^m, -1, 0, 0, 0, x_j^n x_i^m, x_j^n y_i^m, x_j^n)^T \\ \boldsymbol{a_y} &= (0, 0, 0, -x_i^m, -y_i^m, -1, y_j^n x_i^m, y_j^n y_i^m, y_j^n)^T \end{aligned} \tag{3.5}$$

With N matched feature pairs, we have the form $\boldsymbol{Ah} = \boldsymbol{0}$ by combing all pairs, where A has the following form,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_{\boldsymbol{x_1}}^T \\ \boldsymbol{a}_{\boldsymbol{y_1}}^T \\ \vdots \\ \boldsymbol{a}_{\boldsymbol{x_N}}^T \\ \boldsymbol{a}_{\boldsymbol{y_N}}^T \end{bmatrix} \tag{3.6}$$

This can be solved with Direct linear transformation (DLT) via SVD decomposition $\boldsymbol{A} = U\Sigma V^T$, where the right singular vector is the solution to this problem, where $\boldsymbol{H}_{ij}$ can be restored by reshaping it back to 3×3 matrix. Since there are 8 unknown parameters of $\boldsymbol{H}$, at least 4 matched point pairs are needed for this problem. Furthermore, RANSAC method is also applied as previous step to select the best candidates for robust estimation. The idea is to find the optimal solution with the largest consistent number of inliers (inliers are defined as candidates with lower errors than a certain threshold), by repeatedly solving the problem with randomly selected subset (4 points for homography estimation).

3.4 Bundle Method

The goal of Bundle method is to refine the reconstruction model and pursue the best parameters, with minimum value of predefined cost function via optimizing parameters jointly once at a time. During stitching step, pairwise homographies are used to project all images to a reference plane. Cumulative projection accomplish this process for those images don't have direct geometrical relationships due to insufficient matched features. Therefore, it is an essential step to jointly find the best solution with minimized error, since errors could be accumulated via consecutive projection.

Following the annotation and discussion about bundler method in [4], each camera can be parameterised by rotation vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$ and its focal length f; homography $\boldsymbol{H}_{ij}$ describes relationship between the two cameras[4]:

$$\begin{aligned} \boldsymbol{H}_{ij} &= \boldsymbol{K}_i \boldsymbol{R}_i \boldsymbol{R}_j^T \boldsymbol{K}_j^{-1} \\ where \quad \boldsymbol{K_i} &= \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \quad [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix} \end{aligned} \tag{3.7}$$

The cost function is defined such that $\boldsymbol{C} = \sum_{ij} \sum_k (\boldsymbol{r}_{ij}^k)^2$, where $\boldsymbol{r}_{ij}^k = \boldsymbol{p}_{ij}^k - \boldsymbol{f}_j^n, \boldsymbol{p}_{ij}^k = \boldsymbol{H}_{ij} \boldsymbol{f}_i^m$. Parameters are then updated via Levenberg-Marquardt method at each iteration, by pursing derivative for each projection correspondence, until no improvement is gained.

3.5 Online K-Means Clustering

The online K-Means clustering algorithm proposed by [28] is designed to solve the clustering problem with only sequentially readable large data, where normal K-Means algorithms will not work due to limitation of either computation time or memory issue. We employ the algorithm for incremental update, given the candidates searched either determined by MoS detection, or by key-frame selection algorithm from videos in intermediate steps. Similarly, the framework applies clustering methods in pixel-wise manner when a new frame comes. Essentially, this approach deals with the same problems with us, since the given video collections are too big to be loaded all at one time into memory.

[4] For more details about exponential expression $e^{[\boldsymbol{\theta}]_\times}$ of rotation matrix, please refer to [14], Appendix 4.3.

Chapter 4: Problem Statement

Given video collections with size N, $V = \cup_k V_k,\ k \in \{1, \ldots, N\}$, each video clip only contains a part of field scene, the task is to reconstruct panorama $\mathbb{P}$ representation of entire scene, with a much smaller subset $\Theta = \cup_{k=1}^{N} \Theta_k$, where for each $\Theta_k = \cup F_l^k, l \in \{1, \ldots, |\Theta_k|\}, l \ll |V_k|$, and any foreground objects appearing are removed.
Our framework includes 3 steps: model initialization; key-frame selection and incremental update. We set $\Theta^0 = Moment\ of\ Snap\ (MoS)$ for model initialization, which is sub-sequentially dynamically determined by key-frame selection based on *Heat Map* $\mathbb{H}$. Given selected Θ and current $\mathbb{P}, \mathbb{H}$, we perform *Online K-Means Clustering* for incremental update. During the whole process, homography h_l^k is used to project frame F_l^k onto panorama plane.

4.1 Model Initialization

4.1.1 MoS Detection

Given a video clip, [24] is utilized to detect MoS frames for initialization. A $Moment\ of\ Snap\ (MoS)$ is defined as the moment where both team players are prepared to start the game to snap the ball before any player motion is shown in a given American football video clip. The idea behind this step is that compared to other frames, MoS frame normally contains much less motions caused by either camera jitters, at which moment camera is more focused on, or by players due to the definition. At the same time, empirical experiment results with our given dataset show that when MoS happens, a sufficient background sub-region is usually contained for feature extraction and pairwise image matching.

4.1.2 Initialization

During initialization step, all pixels of projected key-frames $\widetilde{f}_j$ sharing the same location $p(i, j)$ on $\mathbb{P}$ are stacked together, which are denoted as $\boldsymbol{p}_{(i,j)}$, and a feature vector $\boldsymbol{f}_{p(i,j)}$

is constructed, which is denoted as $\hat{\boldsymbol{f}}_{(i,j)} = (\boldsymbol{f}_{p(i,j)^0}, \boldsymbol{f}_{p(i,j)^1}, \ldots, \boldsymbol{f}_{p(i,j)^{|\boldsymbol{p}_{(i,j)}-1|}})^T$. The stacked feature vectors are sub-sequentially fed to $Online\ K-Means\ Clustering$ method (k=2) with any $\boldsymbol{p}_{(i,j)}$ such that $|\boldsymbol{p}_{(i,j)}| > 1$. For other positions with only one overlapped pixel, that is $|\boldsymbol{p}_{(i,j)}| = 1$, the maximum value (1 after normalization) is set.
We also denote $\mathbb{C}_p^0$ and $\mathbb{C}_p^1$ as the background and foreground cluster center, respectively, where we set $\mathbb{P} = \mathbb{C}^0$. Given a frame F and its projection $\widetilde{F}$ on $\mathbb{P}$ via $\boldsymbol{H}$, let $\mathbb{P}(\widetilde{F})$ be the sub-region on $\mathbb{P}$ covered by $\widetilde{F}$, and $\mathbb{C}_p(\widetilde{F})$ the subset of $\mathbb{C}$. Cluster centers are then estimated via the following equations,

$$\begin{aligned} \hat{\mathbb{C}}_p^i(\widetilde{F}) &= \mathscr{F}_{\mathbb{C}}(\mathbb{C}_{\boldsymbol{f}}^i, \widetilde{F}_{\boldsymbol{f}}) \\ \hat{\mathbb{H}} &= \mathscr{F}_{\mathbb{H}}(\mathbb{H}, \hat{\mathbb{C}}^i) \\ s.t.\ p \in \mathbb{P}(\widetilde{F}),\ i &\in \{0, 1\} \end{aligned} \tag{4.1}$$

where $\mathscr{F}_{\mathbb{C}},\ \mathscr{F}_{\mathbb{H}}$ are streaming k-means related functions for updating cluster centers $\mathbb{C}$ and heat map $\mathbb{H}$, via the vectorized features $\mathbb{C}_{\boldsymbol{f}},\ \widetilde{F}_{\boldsymbol{f}}$ extracted from $\mathbb{C}$ and $\widetilde{F}$, and the updated cluster center $\hat{\mathbb{C}}$, respectively.
In general, the designed features contains both local and contextual information for robustness. In particular, each $\boldsymbol{f}_{\boldsymbol{p}_{(i,j)}}$ is the concatenation of the following information: the number of edges, evaluated from a 3×3 patch to which the current pixel is centralized; the average 3-channel color values of the patch estimated within converted HSV space; and the RGB value of the pixel, all of which composing a 7×1 feature vector in total.

4.2 Key-frame Selection

Given an initialized panorama $\mathbb{P}$, along with the corresponded heat map $\mathbb{H}$ of videos $V_k,\ k \in \{1, \ldots, N\}$, the task of key-frame selection is to select a set of frames from video collections and update $\mathbb{P}$ and $\mathbb{H}$ incrementally. The designed key-frame selection algorithm is based on the following analysis.
Firstly, among the most given videos, in our case the American Football match videos, highly overlapped field often exists among the temporally-neighbored frames when camera moves slightly or keeps still, by which reflected the inference that a large proportion of frames contains much redundant information. We should use such number of frames as small as possible. Second, the probability, or panorama construction certainty will

decrease as long as the number of available features belonging to background field decreases sharply. This would happen when camera moves fast or zooms in/out to emphasize particular players/events. To minimize the construction error, more frames should be employed, of which the projected frames lie in such field with lower construction confidence. If only considering the first case, a simple conclusion is given that one can select such frame F^* that maximize the average constructed panorama field via available frames.

Nonetheless, the existence of the second case make it more difficult to be handled. One particular example is that for every frame F of a given video V_k with relatively large camera motion, its following frame have the same overlapped area ratio as 0.5. Setting the first frame as reference plane, we have $\Theta_k = \cup F_j$, such that $j \in \{1, \ldots, |V_k|\}, j \bmod 2 = 1$. However, Θ_k would fail the panorama construction pipeline, simply because of every key-frame is independent of the others with none overlapped area, therefore no actual matched features between frames exist. Consequently, the homographies to be pursued would either failed to be generated or totally wrong.

In case of the aforementioned situation happens, we instead use normalized constructed panorama areas as estimated score (or utility) proposed by [22]. The idea is that for every selected key-frame $F_{j'}$ with its projection $\widetilde{F}_{j'}$ sharing no field with current panorama, the estimated score (or utility U) is initially defined as:

$$U(F, h_j) = \cup_{j=1}^{|\Theta_k|} F_j \circ h_j + \lambda Y_F = \cup_{j=1}^{|\Theta_k|} \widetilde{F_j} + \lambda Y_F \tag{4.2}$$

Where Y_F is the number of such frames $F_{j'}$ within Θ_k. Applying equation 4.2, a key-frame candidate F_c^t would be added into Θ_k at step t, if the following condition hold,

$$\begin{aligned} P_k^{t-1} \cup \widetilde{F}_c^t + \lambda Y_F \geq P_k^{t-1} + \mathbb{A}(F_c^t) + \lambda(Y_F + 1) \\ w.r.t.\ P_k^{t-1} := \mathbb{A}_{j=1}^{|\Theta_k^{t-1}|}(\widetilde{F}_j) \end{aligned} \tag{4.3}$$

$$\widetilde{F}_c^t - (P_k^{t-1} \cap \widetilde{F}_c^t) \geq \mathbb{A}(F_c^t) + \lambda \tag{4.4}$$

where $\mathbb{A}(\cdot)$ is the union operation on all elements it contains. Furthermore, to make λ more meaningful for projected frames falling into different projection spaces, the terms $\mathbb{A}(\cdot)$ in equation 4.2 is normalized via the mean value of $P_k^{t'} \cap \widetilde{F}_j$, where t' denotes the

time stamp of P_k when F_j was added into Θ_k, e.g., for valid $F_c^{t'}$, $t' = t-1$. Then equation 4.4 becomes,

$$\begin{aligned} \bar{P_k^t} &= \frac{P_k^{t-1} \cup \widetilde{F}_c^t}{\frac{1}{|\Theta_k^{t-1}|+1}(\sum_{j=1}^{|\Theta_k^{t-1}|}(P_k^{t-1} \cap \widetilde{F}_j) + P_k^{t-1} \cap \widetilde{F_c^t})} \\ &\geq \bar{P_k}^{t-1} + 1 + \lambda \end{aligned} \tag{4.5}$$

Consequently, taking a certain value for $\lambda < 0$, we say $F_c^t \in \Theta_k$ holds if the average number of frames composing P_k is increased by at least $1+\lambda$, e.g., $\lambda = -0.5$. In other words, utility function can be defined as $U(\cdot) = \bar{P_k^t} \geq \bar{P_k^{t-1}} + 1 + \lambda$.

Algorithm 1 summarizes the key-frame selection process. For other parameters, e is the exponential parameter with base 2, determining the index j of frame to be analyzed F_j, and τ the minimum value of e. $F_j^{'}$ records the most recently accessed frame for estimating with current frame F_j the homography, $h_{<j',j>}$ (line 4), which is used to update cumulative homography $\boldsymbol{H}$ (initialized by a 3×3 identity matrix I) at the next step (line 5). Function $sign(D) \in \{+1, -1\}$ indicates the scan direction, taking value $+1$ for forward scan, and -1 for backward scan to control the increase direction of j.

Remind that the key idea of the selection procedure (start with line 2) is that we want to use as small as possible total number of steps to find out the most representative frames, i.e., the maximized utilities. The step size increases exponentially if current homography is successfully founded at current step, while restarting with minimum step size from the last successful frame if failed. The function *matched* in line 4 returns estimated homography $h_{<j',j>}$ if succeed, more details about its definition is given in the experiment section.

After key-frame selection step is done, update current model based on those new coming frames. Let $\mathbb{H}_{(i,j)}$ denotes pixel value located at location (i,j), for each (i,j), $\mathbb{H}_{(i,j)}$ is determined by the following 3 factors. ① normalized vector length $\frac{|\hat{\boldsymbol{f}}_{(i,j)}|}{|\hat{\boldsymbol{f}}|}$, where $|\hat{\boldsymbol{f}}| = \max_{(i,j)} |\hat{\boldsymbol{f}}_{(i,j)}|$. ② distance separation $d_{(i,j)}$, defined by equation 4.6. Where the covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Sigma_0} + \boldsymbol{\Sigma_1}$ is the cluster covariance matrix, of which the two terms on the right side are of background and foreground cluster centers, respectively; and ρ the empirical coefficient parameter predefined as 0.6 in practice. ③ similarity distance $s_{(i,j)}$, indicating how similar $\mathbb{H}_{(i,j)}$ is to its neighbor based background, defined

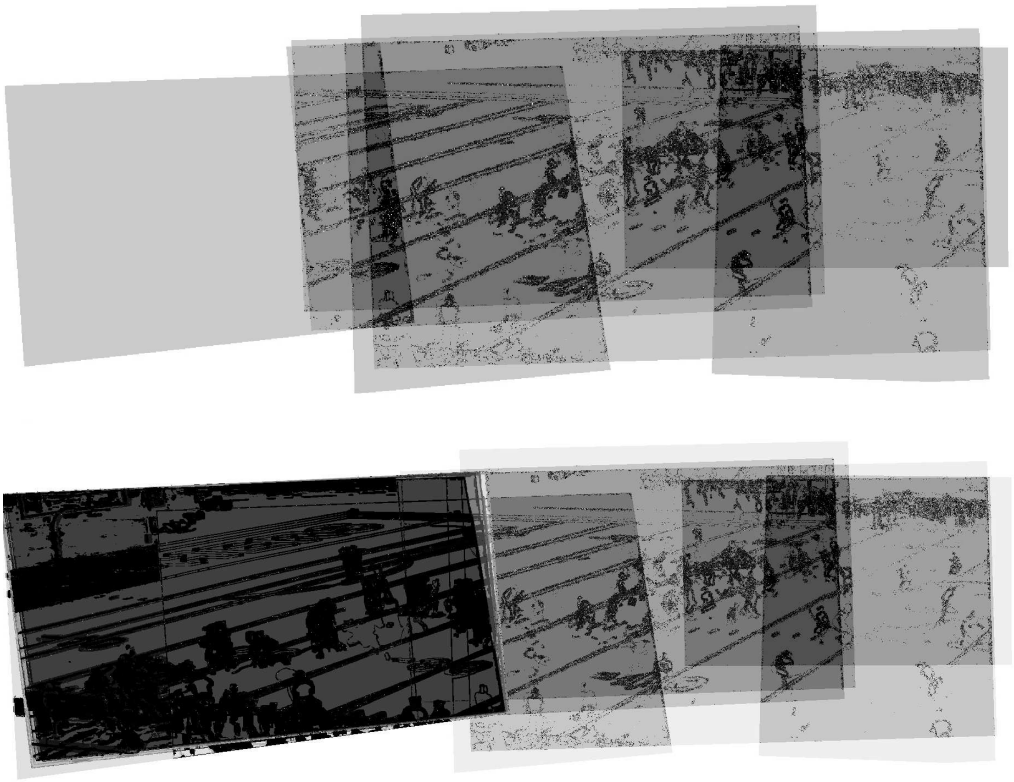

Figure 4.1: A toy demonstrating how key-frame selection algorithm works. Top: heat map is constructed with all pixels set as "all inconfident", shown as white pixels with value 255, and initialized by 10 frames, where darker regions indicates larger number of pixels sharing the same position after projection with estimated homographies are stacked. Down: each projected frame is assigned by a score measured by sum pixel value within projection area of the frame. A frame with the maximum score is then selected, and new frames are brought in by 2-way search algorithm described in Alg. 1.

by equation 4.7. Where $\Phi_{(i,j)}$ contains all neighbors of the pixel located at (i, j), which is also defined by a 3×3 size patch. $\bar{\boldsymbol{\mu}}_{\Phi_{(i,j)}}$ the average cluster centers over all its neighbors, the similar definition with $\bar{\boldsymbol{\Sigma}}_{\Phi_{(i,j)}}$. With those designed features, we show that in our experiments this method performs quite well and outperforms other methods with even more complicated strategy as post-process steps.

Algorithm 1 Key-frame selection

Require: Video $V_k = \cup_m F_m^k$,video length $m = |V_k|$;
1: $P_k = \Theta_k = \emptyset, e = \tau, F_{j'} = F_m^k, j := j^{'} + sign(D) \times 2^e$; ▷ forward scan;
2: **while** *true* **do**
3: | **break** if $j > |V_k|$;
4: | **if** $matched(F_{j'}^k, F_j^k)$ **then**
5: | | $\hat{H} = H \circ h_{<j',j>}, e = e + 1, F_{j'} = F_j$;
6: | **else**
7: | | $e^{'} = \max(e - 1, \tau), F_{j'}^k = F_{e'}^k, e = \tau$, **continue**;
8: | **end if**
9: | **if** $\mathbb{1}_{U(F_j^k, \hat{H})}$ **then**
10: | | $P_k = P_k \cup \widetilde{F_j}^k, \Theta_k = \Theta_k \cup F_j^k$;
11: | **end if**
12: **end while**
13: $e = \tau, F_{j'}^k = F_m^k$; ▷ backward scan;reset value
14: **while** *true* **do**
15: | **break** if $j < 0$;
16: | Repeat: line 4 $\sim$ 10;
17: **end while**
18: **return** Θ_k;

4.3 Incremental Update

Given updated model $\mathbb{P}, \mathbb{H}$ at current step, we apply Algorithm 2 to update $\mathbb{P}$ and $\mathbb{H}$ incrementally. More specifically, we use the following equations 4.6, 4.7 to update our model,

$$\begin{aligned} d_{(i,j)}^w = \exp(-\rho(\boldsymbol{f}_{\boldsymbol{p}_{(i,j)}} - \mathbb{C}_{(i,j)}^w)^T \boldsymbol{\Sigma}_{(i,j)}^w (\boldsymbol{f}_{\boldsymbol{p}_{(i,j)}} - \mathbb{C}_{(i,j)}^w)), w \in \{0,1\} \\ \mathbb{C}_{(i,j)}^0 = \mathbb{1}(\cdot)\mathbb{C}_{(i,j)}^0 + (1 - \mathbb{1}(\cdot))\mathbb{C}_{(i,j)}^1 \\ \mathbb{1}(\cdot) = d_{(i,j)}^0 - \sigma_0 > d_{(i,j)}^1 + \sigma_1 \\ \mathbb{P}_{(i,j)} = \mathbb{C}_{(i,j)}^0 \end{aligned} \tag{4.6}$$

$$
\begin{aligned}
d^{'}_{(i,j)} &= \exp(-\rho(\mathbb{C}^0_{(i,j)} - \mathbb{C}^1_{(i,j)})^T \mathbf{\Sigma}_{(i,j)} (\mathbb{C}^0_{(i,j)} - \mathbb{C}^1_{(i,j)})) \\
s_{(i,j)} &= \frac{1}{2} \hat{\boldsymbol{\mu}}^T_{(i,j)} (\mathbf{\Sigma}_{(i,j)} + \bar{\mathbf{\Sigma}}_{\Phi_{(i,j)}})^{-1} \hat{\boldsymbol{\mu}}_{(i,j)} \\
\hat{\boldsymbol{\mu}}_{(i,j)} &= \mathbb{C}^0_{(i,j)} - \bar{\boldsymbol{\mu}}_{\Phi_{(i,j)}} \\
\bar{\boldsymbol{\mu}}_{\Phi_{(i,j)}} &= \frac{1}{|\Phi_{(i,j)}|} \Sigma_{(i',j') \in \Phi_{(i,j)}} \mathbb{C}^{\mathbf{0}}_{(i',j')}
\end{aligned}
\tag{4.7}
$$

where function $\mathscr{F}_{\mathbb{H}}$ and $\mathscr{F}_{\mathbb{P}}$ in line 11 represents K-Means update process of $\mathbb{P}, \mathbb{H}$. Finally, the update rule for heat map $\mathbb{H}_{(i,j)}$ is:

$$
\mathbb{H}_{(i,j)} = l(\frac{|\hat{\boldsymbol{f}}_{(i,j)}|}{|\hat{\boldsymbol{f}}|}, d^{'}_{(i,j)}, s_{(i,j)}) \tag{4.8}
$$

meaning heat value at position (i, j) is determined by linear combination of those parameters. The algorithm then find a projected frame to maximally increase overall confidence of current $\mathbb{H}$ with the following steps.

1. We maintain all selected frames with a utility set $\boldsymbol{Z}(\mathbb{I}(f_j, h_j))$, and each is measure by a utility value as index value, with its corresponded video index and image position as key value. The utility indicates the inverse reconstruction confidence, and is evaluated via sum of all values within the area covered by its projection on $\mathbb{H}$.

2. To maximize the overall construction confidence at each iteration, we seek the frame with the maximal utility, indicating the most unstable sub-region, then do 2-way key-frame selection, starting from the frame, to explore other un-visited frames with algorithm 1.

3. With searched new candidates from neighboring frames $f_l \in \Phi_N$, $\mathbb{P}$, $(\mathbb{C}_{\mathbf{0}})$ and $\mathbb{C}_{\mathbf{1}}$ are updated simultaneously with *Online Clustering Algorithm*. Meanwhile, all selected neighbors, if haven't been contained in $\hat{T}$ and visited set (closed list) Γ, are added into current utility set $\boldsymbol{Z}(\hat{T})$ for the next iteration(line 5).

With *summed area table* [7]($\mathbb{I}_{\mathbb{P}}$, line 1 of Alg.2), the heat values can be evaluated within near constant time, while the frame with maximized utility value can be calculated within constant time as well with appropriate data structure. The neighbors of current selected frame are given by Φ_N, a deterministic function , labeling all other frames as

Algorithm 2 Incremental update

Require: $\Theta, \mathbb{P}, \mathbb{H}$, input data $f_l, f_l \in (\cup_{k=1}^{N} V_k \setminus \Theta)$, Maximum budget T;

$t = 0,\ \Lambda = \emptyset,\ \mathbb{I}_{\mathbb{P}} = SAT(\mathbb{P}),\ \hat{T} = \boldsymbol{Z}(\mathbb{I}(f_j, h_j)),\ j \in \{1, ..., |\Theta|\}$;
while true **do**
  break if $t > T$;
  $\hat{T}_R$=$max_t \boldsymbol{Z}(\hat{T})_t, \Lambda = \Lambda \cup \hat{T}_R, \Gamma = \emptyset, U_\Gamma = \emptyset$; ▷ select the element with maximum utility
  for $f_l \in \Phi_N(\hat{T}_R)$ **do**
    if $f_l \nsubseteq \Lambda$ and $f_l \nsubseteq \hat{T}$ **then**
      $\Gamma = \Gamma \cup f_l$;
    end if
  end for
  if $\Gamma \neq \emptyset$ **then**
    $\hat{T} = \hat{T} \cup \Gamma, \hat{\mathbb{P}} = \mathscr{F}_{\mathbb{P}}(\mathbb{P}, \Gamma), \hat{\mathbb{H}} = \mathscr{F}_{\mathbb{H}}(\hat{\mathbb{P}}), \hat{\mathbb{I}}_{\mathbb{P}} = SAT(\hat{\mathbb{H}})$;
    $U_\Gamma = \cup_{l=1}^{|\Gamma|} \mathbb{I}(f_l, h_l)$;
  end if
  $\hat{T} = \hat{\boldsymbol{Z}}(\hat{T}, U_\Gamma)$;
  $\mathbb{H} = \hat{\mathbb{H}}, \mathbb{P} = \hat{\mathbb{P}}, \mathbb{I}_{\mathbb{P}} = \hat{\mathbb{I}}_{\mathbb{P}}$;
  $t = t + 1$;
end while
return $\hat{\mathbb{H}}, \hat{\mathbb{P}}$;

neighbors within fixde temporal sliding window. In our experiments the window size is set as $[-2^\tau, 2^\tau]$ so as to cover the gap skipped during key-frame selection procedure, where τ is defined in the first line of Alg.1.

Generally, the update procedure continues until all data are processed. Alternatively, one can also adjust T to a relatively small number for early stopping, giving more time for iterating over the whole process to refine $\mathbb{P}$, if the dataset is very large. For example, more time can be used for refining homographies during image matching step, by setting the stop criterion to a very high value, corresponding to a strict rule that guarantees the homography is probably accurate with very high probabilities. The reason is that sometimes at early steps, it is possible that the estimated homography h_j could not be estimated, or with relatively low confidence, due to the impurity of background panorama caused by existence of foreground objects.

Chapter 5: Experiments

This section provides both qualitative and quantitative experiment results of our framework, as well as result comparisons with other methods. In addition, discussions about experiment results and more details about parameter settings are also given at the end of this part.

5.1 Dataset

We focus on the panorama reconstruction problem of American football video collections. Our dataset is composed of large number of video clips which are captured by hand-held cameras from American football match play field. Our goal is to reconstruct panorama for each game that contains all play field scenes showed from videos. Figure 5.1 gives some examples of frames sampled from our dataset, and table 5.1 provides detailed information about the size of dataset.
With more detail, the following lists provide more information about our dataset to better understand the challenges we met during reconstruction process.

- dataset size. The dataset contains 10 games, with each having about 140-150 videos of 400 frame length on average. Table 5.1 shows detailed information about the size of dataset.

- play type. Based on type of plays, the dataset can be divided into following groups: KickOff, Offense, Defense, Punt, Punish and Noise.

Game index	1	2	3	4	5	6	7	8	9	10
# of videos	132	140	173	149	135	156	152	159	128	146

Table 5.1: Size of dataset. Our dataset includes 10 games, each game contains about 147 videos on average, and each video have about 400-500 frame length.

Figure 5.1: Examples of our dataset. Sampled from different games. Frames within the same game are arranged row by row.

- environment. Various background environment are also included, including dynamic illuminations (day/night time); Shadow;
- motion. Including foreground motion like player movement, and camera motion, like Zoom in/out, large camera motion (when tracking subsequent player events after play begins).

As shown above, there are lots of challenges for panorama reconstruction. One problem is how to select a subset which can summarize all background scene emerged from all videos; we have to deal with various issues like blurred image caused by motion, video qualities and light illumination; another problem is computation efficiency, which should be finished in reasonable time for real-world applications.

5.2 Results

5.2.1 Computation Time

Two evaluation measurements are used to demonstrate the efficiency of our framework, as shown in the following tables. Table 5.2 and 5.3 shows the computation time during initialization step for panorama $\mathbb{P}$ and heat map $\mathbb{H}$, respectively. Note that the total runtime, is computed by both tables. For example, in Game 2 there are about 130 videos in total, corresponding 130 MoS frames for initialization step. During update, Key-frame Selection algorithm will dynamically update the subset via 2-way search algorithm from neighboring frames of the candidate with the largest utility, i.e., highest heat values. In our experiments 5 or 6 update steps would be sufficient to reconstruct panoramas with competitive qualities (we applied 10 iteration steps for all games). Thus the total time used includes $2.90min$ (panorama initialization) + $3.85min$ (heat map initialization) + $3.5min \times 10$ (update) = 41.75 $minutes$.

As comparison, table 5.4 shows overall runtime based on standard panorama reconstruction pipeline. Additionally, table 5.4 also applied "fixed-step sampling" method to construct key-frame subset instead of applying all available videos like $Direct\ Method$ introduced in previous sections. In detail, we sample the frames based on fixed step for each video, in our experiment the step size is defined as a constant number 30. We applied $Random\ Forest$ classifier to refine features for both methods.

As can be seen from the two tables, our framework is much more efficient compared to standard panorama reconstruction approaches. One reason is that the selected key-frame subset has smaller number of frames with the power of key-frame selection algorithm, at the same time being capable of summarize almost all play field scenes with those frames which are more "representative". Meanwhile, our incremental update strategy can also avoid over-determined problem, that is, the reconstruction process can be stopped at any time as long as there's not much improvement, which can be evaluated via intermediate updated heat maps.

5.2.2 Feature Compression Ratio

Another measurement used is feature compression ratio based on $Random\ Forest$ (RF) classification described in section 3.2. Table 5.5 shows the experimental results with

Game	1	2	3	4	5	6	7	8	9	10	Average
Initialization time (minutes)	2.90	2.61	3.68	3.06	1.59	2.25	4.73	3.11	1.65	2.82	2.94

Table 5.2: Time used at the first step for panorama $\mathbb{P}$ initialization, measured in *minutes*.

Step	Initialization	Update
Process time (minutes)	3.85	3.5

Table 5.3: Time used for processing heat map $\mathbb{P}$ during initialization step and subsequent update steps, measured in *minutes*. Note that different from table 5.2, the consumed time only depends on image size other than number of extracted features, thus the runtime showed in this table is based on every 100 frames. *Online K-Means Clustering* method is used based on method [28]. Note that during current step, update will not be performed and previous result will be kept if there is only 1 overlapped pixel.

Game	1	2	3	4	5	6	7	8	9	10	Average
Process time (minutes)	84.67	178.53	76.95	158.18	251.86	39.20	110.40	109.14	163.55	120.58	129.31

Table 5.4: Overall runtime of standard panorama reconstruction method. An "fixed-step sampling" method is applied for subset selection, where a fixed step is used to sample a frame for each video. In our experiments the step size is set as 30, equaling video frame-rate.

Method	Runtime	Feature Compression Ratio (r / 1 - r)	Runtime Compression Ratio (r / 1 - r)
Key-frame selection + RF	1.57	0.2825 / 0.7175	0.403 / 0.5970
Key-frame selection	2.63	0 / 1	
Sampling + RF	3.58	0.2764 / 0.7236	0.4003 / 0.5997
Sampling	5.97	0 / 1	

Table 5.5: Average feature compression ratio (r / 1 - r, where r is compression ratio) and runtime (measured in *minutes*, without heat map construction) ratio during update steps. Row by row: the first 2 rows applied key-frame selection via Alg.1 w/w.o *RF* classifier, respectively; the last two rows applied 2-way forward-backward "fixed-step sampling" method based on selected frame via heat map, w/w.o *RF* classifier, respectively, where the step size is set as 30. Column by column: column *Runtime* indicates overall runtime; column *Feature Compression Ratio* indicates the percentage of removed features after classification; column *Runtime Compression Ratio* indicates the percentage of decreased runtime after applying classification approach. Difference between tab. 5.4 and this table: standard panorama reconstruction approach includes repeated process of integrating smaller images called "connected components", which are generated based on input frame sequences; the process will repeat until no such component are constructed that bringing more field scenes. See algorithm 3 with detailed explanation of this approach.

comparison of both feature compression ratio and runtime ratio. We designed two independent test cases to measure the performance of *RF* classifier: the first one took detected key-frames as input, i.e., *MoS* frames, while the second one used the same sampling strategy explained in previous section. Test results are shown in the last two columns, denoted as "Feature Compression Ratio" and "Runtime Ratio" respectively. The results show that by applying *RF* classifier, about 30% features will be throw away during feature matching process, resulting in about 40% gain of runtime efficiency on average.

5.2.3 Qualitative Results

In this section we will show generated panoramas based on the our framework as well as qualitative results generated with other methods for comparison. In detail, we imple-

Algorithm 3 Standard Panorama Reconstruction Approach

Require: Selected subset from video collections

```
while true do
    Do SIFT feature extraction
    Examine features via RF classification and throw away non-matchable candidates
    Do featurea matching and stimate homographies between images
    Do image blending based on images and estimated homographies
    if No improvements then
        break
    end if
    Update subset with generated connected components
end while
return Reconstructed panorama (connected components with the largest area of
background scene)
```

mented methods proposed in [4] with both linear blending and multi-band blending as post-process; additionally, we also implemented Graph-cut based blending method discussed in [3] at pixel level. Intuitively, below are the differences among those methods:

- Linear blending. Linear blending method integrates all available data (images) for each position (pixel), and the final pixel values are obtained by combining all pixel values with the same position in liner form. Alternatively, weighted linear blending is a more popular method, in which the weight of previous value equals number of pixels integrated in history. Thus the updated value, given a new pixel, will be calculated in the form of $\frac{N_p}{N_p+1}P_{old} + \frac{1}{N_p+1}P_{new}$, where N_p indicates number of integrated pixels up to now, and P_{old}, P_{new} indicate old value and the new coming pixel value, respectively.

- Multi-band blending. Multi-band blending tries to keep significant information in there is large gradient, such as edges or textures. The process is done by keeping those information via image pyramid implemented by Gaussian pyramid. By doing this high frequency bands will be blended over small ranges, while low frequency bands are blended over larger ranges.

- Graph-cut based blending method. This method tries to find local optimal solution with minimized cost function by updating current value repeatedly, given its neighboring pixel information. Cost function includes two parts: unary cost

and pairwise cost. In our experiments the unary cost is defined as average texture information within fixed windows; normally pixels with fewer texture information have larger probability being within background areas. Whereas pairwise cost is determined by consistency or smoothness among neighboring pixels that encourages results to be more smooth. The pairwise cost function is defined by Euclidean distance of representative features stacked by color and edge information as well.

- Our method. In our framework the final value of each pixel is directly determined by background cluster centers, which is similar to linear blending.

Figure 5.2- 5.12 show the generated panorama with different methods. Figure 5.2- 5.4 are generated by linear blending; figure 5.5- 5.7 apply multi-band blending proposed by [4]; figure 5.8, 5.9 use graph-cut approach discussed in [26]; and figure 5.10, 5.11 are generated via our method. In addition, figure 5.12 show the visualized results of heat map after update process is done.

5.3 Discussion

5.3.1 Performance

During our experiments, we found that the actual runtime on each game differs from each other. The reason is that during update process, the size of selected key-frame subset varies depending on both reconstructed panorama at current step, measured by number of extracted matchable features, and video qualities to which the selected frame belongs (with the maximum heat value), measured by video length and frame qualities. More matchable features meaning faster convergence rate for homography estimation, and larger video length indicates more time consumed for searching. As to the notation "budget" mentioned in Alg. 1, we define it as allowed computation time and available data that can be used. The later means that the process continues until all frames are measured and integrated incrementally. In our experiments we applied the first definition and set it as fixed number of update processes, which is 10. That being said, based on the discussion in section 5.2.1, the average runtime of our method would be 41.75 *minutes*. From table 5.4, we found that our approach is much more efficient than other methods. Additionally, we found that among other methods, graph-cut based method has the best

qualitative results, whilst took more time than any others. Furthermore, we also found that multi-band method isn't applicable to solve this problem. Both foreground and background areas in our dataset contain almost equivalent gradient of edge and texture information, multi-band method don't have the ability to separate both since it will keep any high frequency information from all frames.
From figure 5.12, we can see that almost all the scenes are covered by selected frames, at the same time with very low "heat" values, meaning that almost all pixels within the panorama are well restored with very high probabilities of being guaranteed to be correct, within the context of our separation rules defined in prior section.

5.3.2 Limitation

However, there still exist some problems that our approach could not well deal with. In specific, there are mainly 3 problems or limitations of our method.

- The overall performance largely depends on initialization step.
- Homography estimation step cannot guarantee the correctness or preciseness as well as bundle adjustment.
- Our method needs much larger memories than other methods.
- Geometry distortions.

Our method fails at initialization step when available matchable features are not sufficient. For example, for the game showed in figure 5.3 (row 1), we couldn't be able to initialize the model, since background areas are almost filled with homogeneous textures, while features representing foreground objects are almost removed during feature matchability estimation step. Bundle adjustment, on the other hand, keeps all features instead and thus performs better in this circumstance. As demonstrated by [16], the property of local consistency could be propagated continuously and thus be used to find more global consistent features. Our method loses this advantage since key-frames are selected discretely with dynamic search steps. This also explains the second item listed above.
Another issue is that our method takes much more memories, since we also have to keep track of information generated by online clustering process, in addition of representative

features of each pixel for our separation rule. Our program crashes when processing panoramas with large resolutions. However, this could also partially due to inaccurate estimate homographies, leading to highly distorted polygons when projected onto panorama plane.

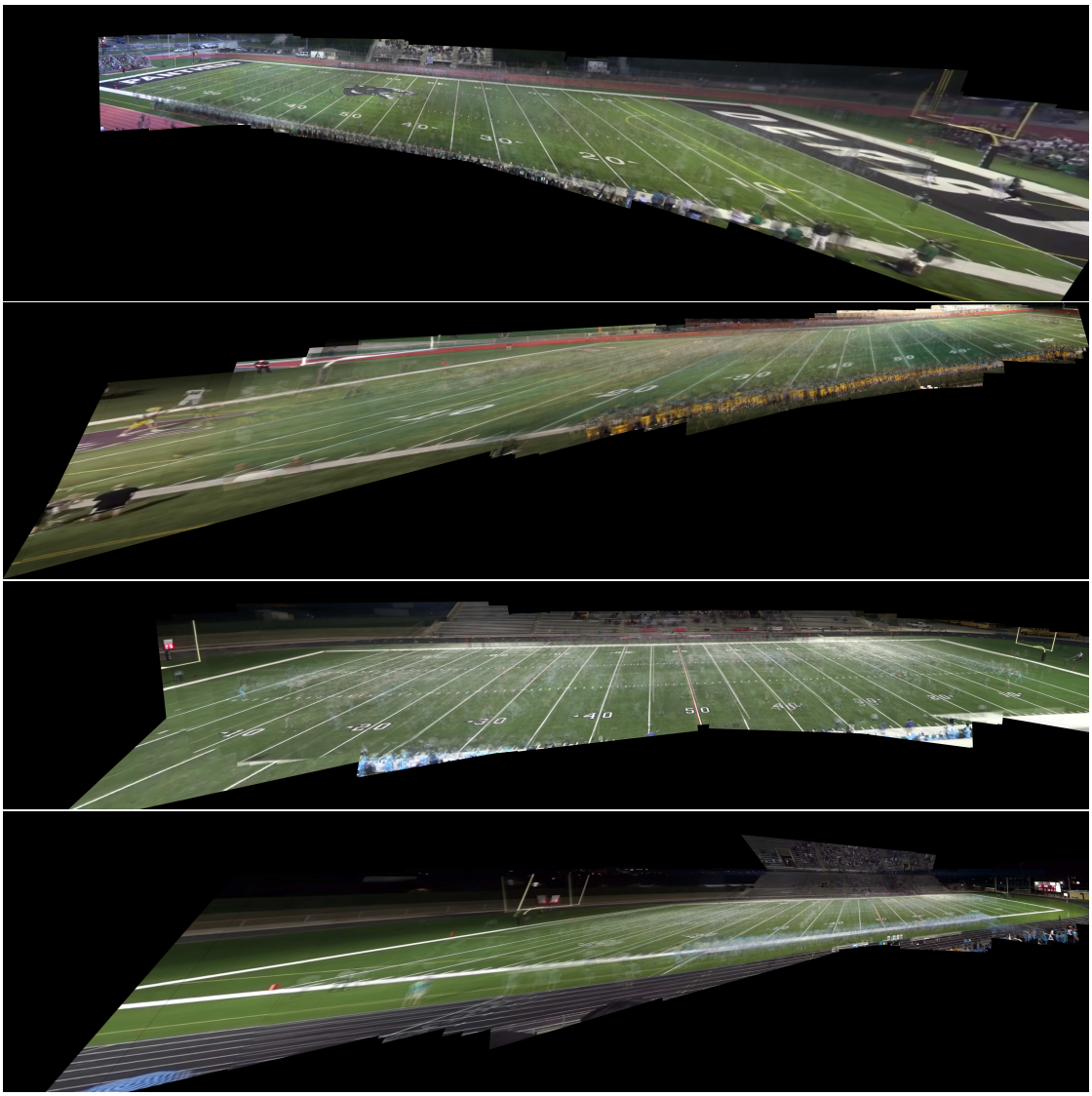

Figure 5.2: Linear blending results, part 1.

Figure 5.3: Linear blending results, part 2.

Figure 5.4: Linear blending results, part 3. Linear blending method integrates all available data (images) in pixel-wise via linear combination. This result (Linear blending result part 1 3) applied weighted linear blending, in which the updated value, given a new pixel, will be calculated in the form of $\frac{N_p}{N_p+1}P_{old} + \frac{1}{N_p+1}P_{new}$, where N_p indicates number of integrated pixels up to now, P_{old}, P_{new} indicate old and the new coming pixel values, respectively.

Figure 5.5: Multi-band blending results, part 1.

Figure 5.6: Multi-band blending results, part 2.

Figure 5.7: Multi-band blending results, part 3. Multi-band blending tries to keep significant information with high frequency, such as edges or textures. The process is done via constructed image pyramids.

Figure 5.8: Graph-cut blending results, part 1.

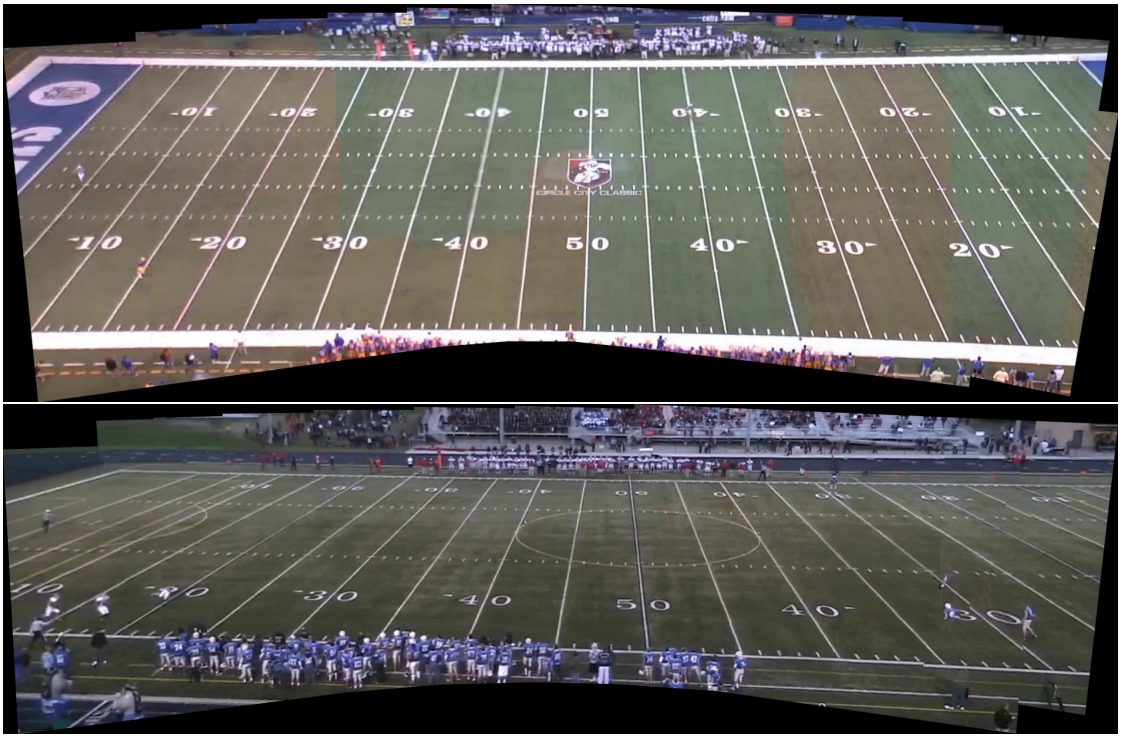

Figure 5.9: Graph-cut blending results, part 2. This method tries to find local optimum solution with minimized cost function, given its neighboring pixel information. Cost function includes two parts: unary cost and pairwise cost: unary cost is defined as average texture information within fixed windows, whereas pairwise cost is determined by consistency or smoothness among neighboring pixels that encourages results to be more smooth.

Figure 5.10: Incremental reconstruction results, part 1, generated by out method. Where each pixel represents the final cluster background center values. See part 2 for more results and discussions.

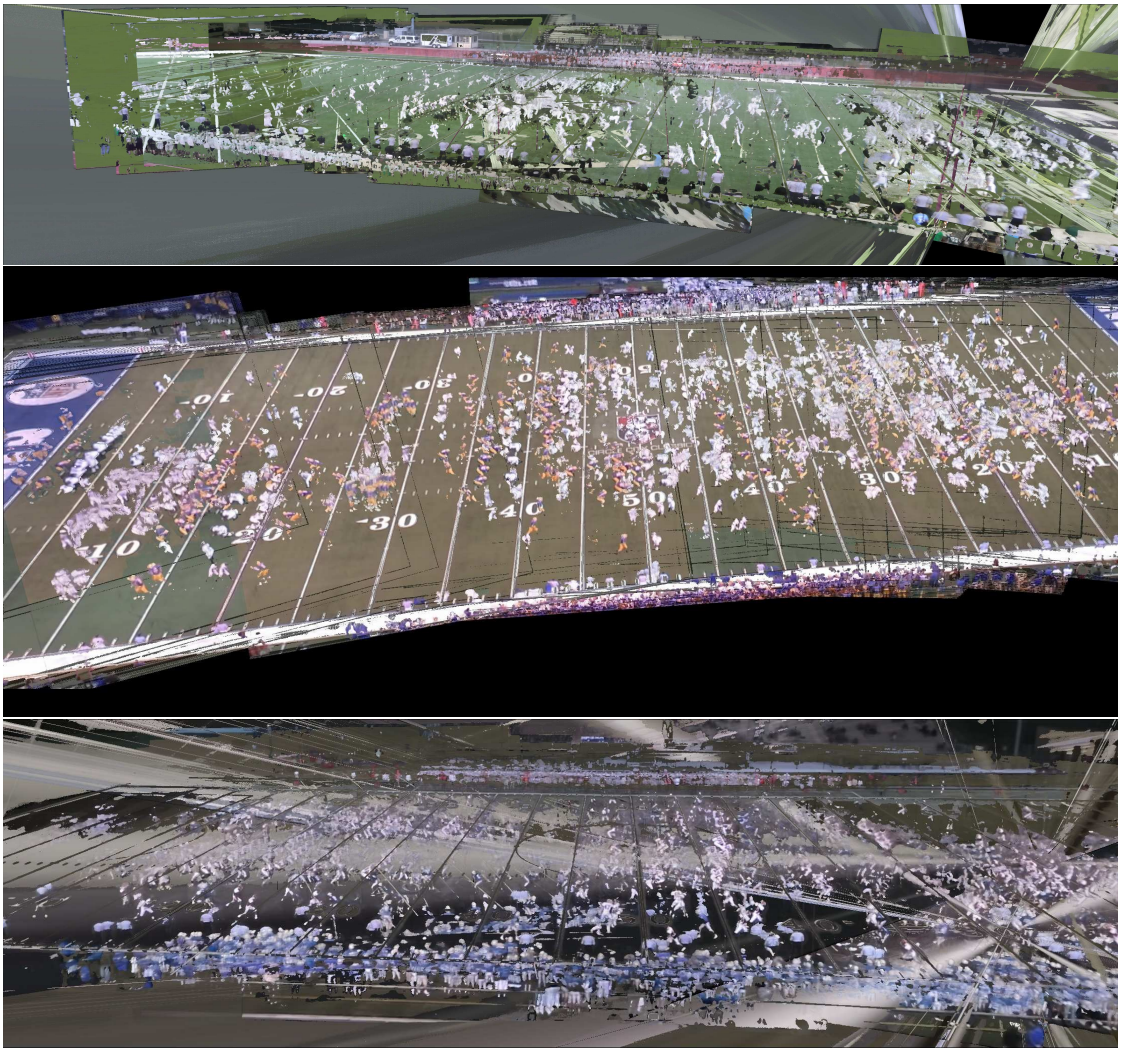

Figure 5.11: Incremental reconstruction results, part 2. Each image shows corresponded foreground (cluster centers) after termination of update process. Note that compared to those generated by other approaches, there are more noises showed in boundary areas, which is caused by inaccurate estimated homographies during update process. Other than standard panorama reconstruction approach which integrate all available data to refine homographies via bundler, the incremental process only contains limited neighboring images, as well as updated panorama up to current step, that means for some frames, homography estimation step would be messed up due to insufficient features.

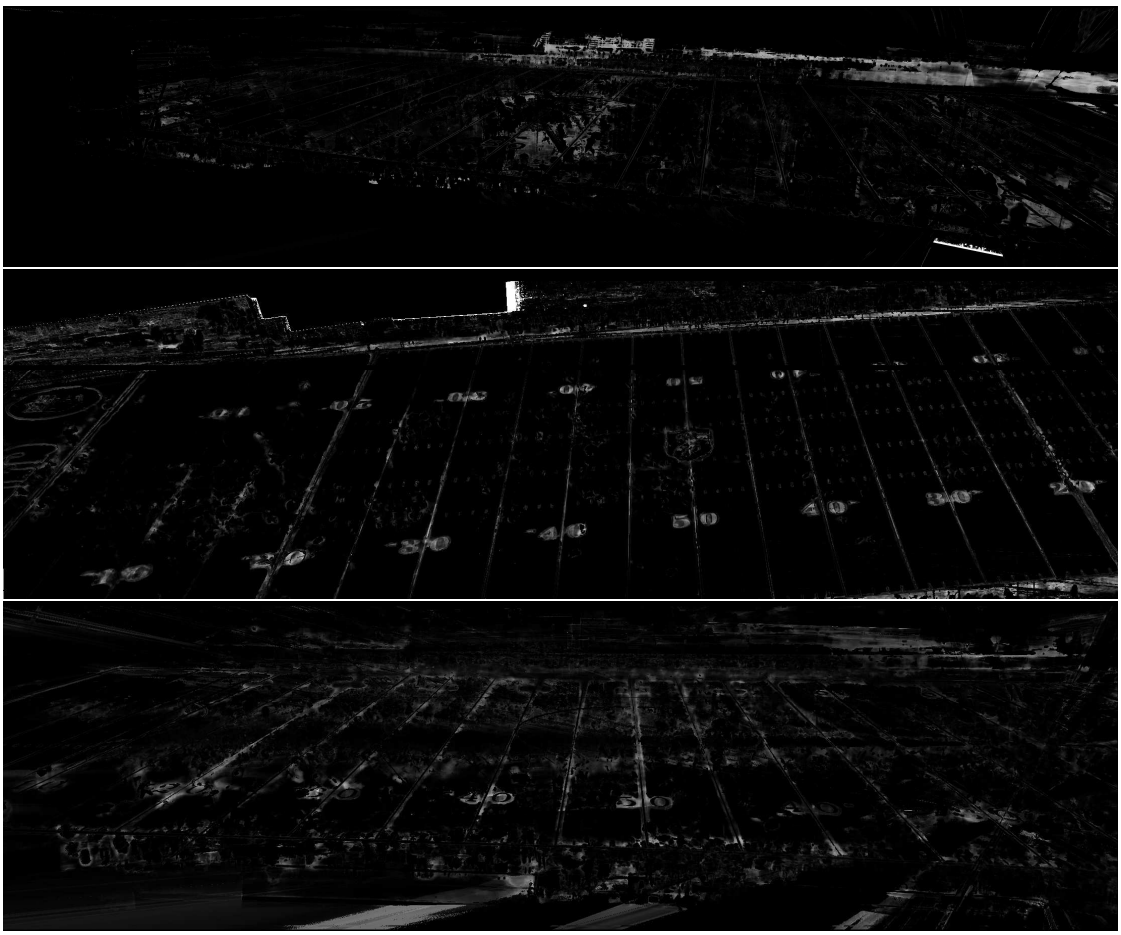

Figure 5.12: Visualized heat map after normalization. Each pixel represents the heat or inverse-confidence of reconstructed panorama at current step. Intuitively, higher value indicates lower confidence of reconstructed panorama. See figure 4.1 to better understand how key-frame selection works during update process.

Chapter 6: Conclusion and Future Work

In this report we presented a new panorama reconstruction framework for American football video collections. Specifically, our contribution includes: first, we apply a new strategy for electing a subset of images from the large input set, referred to as key-frame selection algorithm. Second, we present a new method for removing low-quality features as a pre-process step for feature matching. Third, we design a new strategy for iteratively selecting which new images to process for updating the panorama. Experiment results demonstrate that our approach is more efficient than state-of-the-art methods, while yielding competitive results. To the best of our knowledge, prior work has not considered efficiency, but only accuracy of panorama extraction. Our approach has an advantage of being flexible to various time budgets set by real-world applications.
Our plans for future works include:

- Higher order feature representation. Currently our use of SIFT features only allow us to capture local potential information. However, the discriminative power of local descriptors has limitations to both geometric verification and large images. Alternatively, [34] bundled various types of features into local groups and improved discrimative representation power with competitive experiment results for large-scale image search problem.

- Geometric distortion. As shown in our experiments, our method also suffers from distortions due to image qualities. The distortion error propagate spatially and accumulated together, resulting in highly curved field boundaries. Typical distortion elimination methods include first order radical distortion [31], or treating distortion factor as intrinsic camera parameters [8]. We tested the first method in our experiments, however there's no observed significant improvements.

- Computational performance. There are still lots of spaces to improve our framework. Specifically, the fact that processing works at pixel level is high independent, with the power of General-purpose GPU (GPGPU) framework, the clustering process can be highly parallelized, resulting in polynomial or even exponential order

of runtime improvement. On the other hand, our framework has limited ability of memory management, which has been discussed in previous chapter. A tutorial on computational efficiency in computer vision is provided by [25].

Bibliography

[1] Indriyati Atmosukarto, Bernard Ghanem, Shaunak Ahuja, Karthik Muthuswamy, and Narendra Ahuja. Automatic recognition of offensive team formation in american football plays. In *CVPR workshop*, pages 991–998. IEEE, 2013.

[2] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *CVPR*, pages 1000–1006. IEEE, 1997.

[3] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

[4] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 74(1):59–73, 2007.

[5] S. Brutzer, B. Hoferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *CVPR*, pages 1937–1944. IEEE, 2011.

[6] Sen-Ching S. Cheung and Chandrika Kamath. Robust background subtraction with foreground validation for urban traffic video. *EURASIP J. Appl. Signal Process.*, pages 2330–2340, 2005.

[7] Franklin C Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, 18(3):207–212, 1984.

[8] C BROWN Duane. Close-range camera calibration. 1971.

[9] Fida El Baf, Thierry Bouwmans, and Bertrand Vachon. Comparison of background subtraction methods for a multimedia learning space. In *SIGMAP*, pages 153–158, 2007.

[10] Bernard Ghanem, Tianzhu Zhang, and Narendra Ahuja. Robust video registration applied to field-sports video analysis. In *ICASSP*, 2012.

[11] J-Y Guillemaut, Joe Kilner, and Adrian Hilton. Robust graph-cut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes. In *ICCV*, pages 809–816. IEEE, 2009.

[12] Xiaojie Guo, Xinggang Wang, Liang Yang, Xiaochun Cao, and Yi Ma. Robust foreground detection using smoothness and arbitrariness constraints. In *ECCV*, pages 535–550. Springer, 2014.

[13] Daniel Gutchess, M Trajkovics, Eric Cohen-Solal, Damian Lyons, and Anil K Jain. A background model initialization algorithm for video surveillance. In *ICCV*, volume 1, pages 733–740. IEEE, 2001.

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[15] Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Predicting matchability. In *CVPR*, pages 9–16. IEEE, 2014.

[16] Robin Hess and Alan Fern. Improved video registration using non-distinctive local image features. In *CVPR*, pages 1–8. IEEE, 2007.

[17] Yuri Ivanov, Aaron Bobick, and John Liu. Fast lighting independent background subtraction. *IJCV*, 37(2):199–207, 2000.

[18] Jiaya Jia and Chi-Keung Tang. Image registration with global and local luminance alignment. In *ICCV*, pages 156–163. IEEE, 2003.

[19] Juho Kannala, Mikko Salo, and Janne Heikkilä. Algorithms for computing a planar homography from conics in correspondence. In *BMVC*, pages 77–86, 2006.

[20] Kyungnam Kim, Thanarat H Chalidabhongse, David Harwood, and Larry Davis. Background modeling and subtraction by codebook construction. In *ICIP*, volume 5, pages 3061–3064. IEEE, 2004.

[21] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[22] Frank Liu, Rob Hess, and Alan Fern. Efficiently constructing mosaics from video collections. In *WACV*, pages 223–230. IEEE, 2015.

[23] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[24] Behrooz Mahasseni, Sheng Chen, Alan Fern, and Sinisa Todorovic. Detecting the moment of snap in real-world football videos. In *IAAI*, 2013.

[25] C. Marwa, S. Fatma, and T. Rached. Computer vision application in graphic processors. 2012.

[26] Kenji Okuma, James J Little, and David G Lowe. Automatic rectification of long image sequences. In *ACCV*, page 9, 2004.

[27] Matthew J Roach, John SD Mason, and Mark Pawlewski. Video genre classification using dynamics. In *ICASSP*, volume 3, pages 1557–1560. IEEE, 2001.

[28] Michael Shindler, Alex Wong, and Adam W. Meyerson. Fast and accurate k-means for large datasets. In *NIPS*, pages 2375–2383. 2011.

[29] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[30] Christian Thurau and Václav Hlavác. Pose primitive based human action recognition in videos or still images. In *CVPR*, pages 1–8. IEEE, 2008.

[31] Ben Tordoff and David W Murray. The impact of radial distortion on the self-calibration of rotating cameras. *Computer Vision and Image Understanding*, 96(1):17–34, 2004.

[32] Ba Tu Truong and Svetha Venkatesh. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 3(1):3, 2007.

[33] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 19(7):780–785, 1997.

[34] Zhong Wu, Qifa Ke, Michael Isard, and Jian Sun. Bundling features for large scale partial-duplicate web image search. In *CVPR*, pages 25–32. IEEE, 2009.

[35] Hui Zeng, Xiaoming Deng, and Zhanyi Hu. A new normalized method on line-based homography estimation. *PR Letters*, 29(9):1236–1244, 2008.

APPENDICES