3D Structure Estimation from Monocular Video Clips

Arturo Donate and Xiuwen Liu
Florida State University
Department of Computer Science
Tallahassee, FL 32306
{donate,liux}@cs.fsu.edu

Abstract

This paper explores the idea of extracting three dimensional features from a previously recorded video, in an attempt to provide three dimensional information about a video clip in order to improve the performance of various video analysis tasks. Although video analysis is a very prevalent area of research, the use of 3D features is scarce in the literature due to the inherent difficulties associated with extracting accurate 3D representations of videos in cases where no previous knowledge of the scene or camera is known.

In this paper, we present a framework that attempts to compute a dense three dimensional representation of a scene using only the available video sequence. Our proposed system exploits the motion of the camera in order to estimate the relative 3D positions of salient features located in the video frames. Additionally, we incorporate the use of appearance-based models to estimate their relative poses and fit a 3D human model into the reconstructed scenes. We test our method using various video clips obtained from online databases in order to show the feasibility of this approach.

1. Introduction

With all the current and ever-increasing uses for video databases, the need for fast and accurate mining systems grows every day. On account of this, video analysis is quickly becoming a prominent field of active research. This field can be broken down into several smaller areas [21]. Some of these include multimodal analysis, video representation, video summarization, browsing, as well as video retrieval.

Many of these video analysis areas share similar methods for extracting relevant details from videos. Some of these methods include using low level measurements such as color histograms, while others use more common image descriptors such as SIFT [12] and SURF [1] features. While many of these tools have been proven to be successful in solving various video analysis tasks, none of them capture the inherent three dimensional structure of real world objects. As such, their descriptive powers will always be limited to some degree.

The aim of this work is to extract a dense 3D reconstruction of a video sequence. We assume no prior knowledge of the system as we attempt to not only provide a dense reconstruction of the static elements in the observed scene, but also provide 3D models of the humans present in the scene, in their relative poses.

Several recent publications have attempted to solve similar problems related to mapping scenes in 3D from a single camera. In [3], the authors present a system to perform simultaneous localization and mapping using a single monocular camera. The system can track several points in real-time and perform accurate localization of the camera. The world is modeled as a probabilistic 3D map that is continuously updated via an Extended Kalman Filter [18]. Since the primary focus of the system is to achieve high localization accuracy, the features of the probabilistic map are modeled using a densely populated covariance matrix. In order to achieve real-time performance, the number of features cannot be very large. Because of this, the framework is able to generate accurate locations for a set of 3D features, but the number of features is relatively sparse.

In [14], the authors attempt to perform camera localization and 3D reconstruction of the scene in real time using only a single video camera as input. Although this system was not designed to work with video sequences, using a single monocular video camera as input presents many of the same problems as we are dealing with. The authors first use salient corners and normalized cross correlation to find matching features across images. Then, the normalized 5-point algorithm [15] is used to determine the relative 3D camera pose across the initial set of frames. The 3D points are then computed via triangulation, and the camera and feature positions are updated via local bundle adjustment.

In [10], the authors present a framework for augmented reality using a single monocular camera. The authors run two separate threads, one for tracking and one for mapping. The tracking process receives new frames from the camera, projects existing features onto the image according to a prior pose estimate, searches the image for the features using a coarse-to-fine approach, then computes an updated camera pose estimate. The mapping process is responsible for generating the 3D map and maintaining it. Initially, the 3D points are calculated using the 5-point algorithm [15]. As the camera moves, this process searches for new features in the images and calculates their 3D positions via triangulation. The relative 3D positions of the camera and features are optimized using local bundle adjustment. Additionally, the system uses RANSAC on the 3D points stored in the map to estimate the location of a flat surface. Once a suitable surface is found, the system is able to perform several augmented reality tasks using the 3D map. Some of these tasks may include projection of various 3D objects onto the estimated flat surface.

The literature on 3D human tracking is extensive, and thus we will only review selected works. Several of these methods use either multiple cameras in order to obtain accurate depth information, or a single stationary monocular camera.

In [11], the authors attempt to perform real-time human body tracking in 3D. This proposed system uses color and depth information to track several parts of the human body in real time. In order to compute the depth of each tracked feature, the authors use several cameras and employ the use of multi-baseline stereo algorithms. Additionally, the cameras are assumed to be stationary, so subjects can be segmented using background subtraction.

In [13], the authors propose a method for robust tracking of a person's upper body and 3D pose estimate in real time using a single monocular camera. People are first detected using a probabilistic framework for head, torso and hand detection using adaboost. The relative pose of the person is then estimated using previously learned mixture models along with RANSAC. Afterwards, the detected person is matched to a 3D model using edge and silhouette matching criteria incorporating background subtraction (like the previous work, the camera is assumed to be static in this case).

In [20], the authors present a framework that uses temporal motion models along with principal component analysis in order to track a human body. Data is obtained from a stationary stereo camera. The authors argue that by posing the tracking problem as one of minimizing differentiable objective functions, they are able to use standard deterministic optimization methods, as opposed to probabilistic methods. In doing so, they show that tracking of a human body can be done at a much lower computational cost than traditional probabilistic methods.

The rest of the paper is organized as follows. Section 2 describes our approach for estimating the 3D structure of the background scene in a video. Section 3 explains how our framework estimates the relative pose and positions of people in the scene, and from this how we can insert a 3D model of a human into the previously-generated reconstruction. We show several examples on various video sequences in Section 4, and conclude our paper in Section 5.

2. 3D Scene Reconstruction

The task of reconstructing a scene in 3D from a single video input is very difficult due to the many inherent limitations of such a system. By being only a single view (as opposed to a stereo view), the system must rely on temporal information to provide enough information to accurately calculate the three dimensional structure of a scene. The problem becomes even more complex when dealing with dynamic scenes involving multiple independently moving objects. Since there is only one view of the scene, we rely on image frames at different points in time in order to obtain different views of the scene, but objects may not be located in the same place at different points in time, causing problems for the reconstruction step.

Our method takes advantage of available camera motion in video clips to obtain views of the same scene from different angles, in order to estimate the relative 3D geometry. In doing so, we make use of many fundamental concepts proposed in [14, 10].

The basic idea behind our approach is to first generate an initial 3D reconstruction of the scene using video frames obtained at different points in time. As new frames are observed by the system, the estimated 3D points are projected back onto the image and compared with the corresponding image feature locations. Bundle adjustment [9, 5] is used to optimize the estimated camera and feature locations in the 3D world.

2.1. Initialization

The system begins by generating an initial reconstruction from a pair of images. This step requires user interaction to obtain a pair of images with some camera translation in order to generate the initial reconstruction. Ideally, the user will select two frames where the camera is observing the same scene, but with only a translation of the camera position. Once the two frames are selected, the system finds corresponding points in the two images via optical flow and the eight point algorithm is used in order to estimate the scene geometry and relative camera position between the two frames. Afterwards, we use triangulation to estimate the relative 3D location of each point. These point locations are stored in our 3D map along with the camera location. To find these salient image points in each image, we use

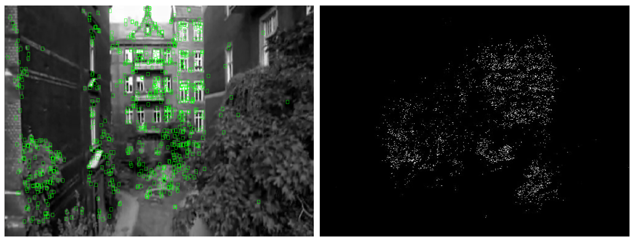
Figure 1. Sample reconstruction: left image shows the features, right image shows the reconstruction.

the Harris feature detector [8] already implemented into the OpenCV library [2]. Harris features were chosen for their ability to efficiently extract a substantial amount of corner features in image frames.

At this point, we have an initial 3D reconstruction of the scene as observed by the camera. Note that if the translation between the chosen frames is insufficient, this will result in a reconstruction where all the 3D points are coplanar (due to the lack of disparity between corresponding features). The system will continue without error, but the reconstructions will prove to be inaccurate.

2.2. Camera pose estimation

In order for our method to work correctly, we must update our camera location estimate for each frame in the video. To do this, we adopt the process initially proposed in [14]. Each time a new video frame is obtained, we first project the features currently stored in the 3D map onto the new frame.

Next, we find the salient features in the new frame using the Harris feature detector. The goal is to find the corresponding salient feature in the new frame that matches with the projected features (from our 3D map). To do this, we perform a search within a fixed distance from each projected feature, looking for the new salient feature with the lowest zero-mean SSD score. If this score is below a certain threshold value, we consider the salient feature and the 3D feature to be a match.

In order to correctly project the features onto the image, it is necessary to estimate the intrinsic camera parameters. For the results presented in this paper, we estimated these parameters manually by observing the calibration parameters of several cameras, and finding the set of values that yielded reasonable results. This process may be automated by using some of the automatic camera calibration methods available in the literature [4, 19, 6].

Once the relative positions of all the features are found, we use them to calculate an initial estimate of the camera location using Grunert's method [7, 14]. This provides an initial estimate for the camera pose, which will be refined later.

2.3. Updating the map

As new video frames are analyzed by the system, the 3D map is updated accordingly. Similar to previous works [14, 10, 3], we observe that not every frame of the video needs to be used for reconstruction. For example, if the camera is stationary and is observing a static scene, there is no point in re-calculating the 3D reconstruction every time a new frame is observed. This is computationally expensive and does not provide any useful information. Instead, we only insert new features into our 3D map if the current number of visible features falls below a threshold. Setting this value to a large number will generate very dense reconstructions, but will have an adverse effect on computational costs.

In our experiments, we found that a value of approximately 500 features provides a good balance between computational requirements and reconstruction quality. When the number of observed features in a given frame falls below this threshold, the system selects new features in the image and uses triangulation to estimate the 3D location of these points. Afterwards, these points are added into the 3D map. This threshold value will also depend on the scene being observed however, since the Harris detector is unable to extract salient features from textureless regions.

Since we are estimating 3D feature locations by exploiting camera motion, we can only accurately reconstruct static scenes to ensure that features are in the same location when observed at different points in time. Most video clips are dynamic in nature however, sometimes containing multiple moving objects, each with unique motions. At times, our method will attempt to match features across frames that belong to objects in motion. Since our camera location estimates rely on the 3D map features, we must remove any feature from the map that can potentially corrupt our camera location estimates. If a feature in the 3D map fails to be found in subsequent frames, we assume that this feature belongs to a moving object, and is thus deleted from the map after a certain number of failed matching attempts.

2.4. Optimizing camera and feature locations

As new features are added to the map, we update the camera position and feature positions using bundle adjustment. This process uses a Levenberg-Marquardt optimization to obtain the optimal least-squares solution for the system (with respect to camera and feature locations). As discussed in [14, 10], this bundle adjustment step can be very computationally expensive, so both of these previous works employ the use of local bundle adjustment. Instead of optimizing over all the previous frames, local bundle adjustment only takes into consideration the last N frames. Let α_i be our cost function at time i to be minimized by the local bundle adjustment. Let C_i be the set of camera locations for the most recent N positions, and let F_i be the set of feature locations (in 3D world coordinates) measured at each of the N previous camera locations. We can now define our cost function as

$$\alpha_i\left(C_i, F_i\right) = \sum_{C_i} \sum_{f_j \in F_i} d\left(f_j, P_i f_j\right) \tag{1}$$

where $d\left(f_j, P_i f_j\right)$ is the distance between the previously recorded feature location f_j and the re-projection of the feature with the computed projection matrix P (incorporating the new camera location parameters). In other words, we are using a least-squares optimization to minimize the re-projection error of camera and feature locations for each point in time.

Taking all previous frames into account when performing the optimization yields the best reconstruction results, but can become computationally expensive, especially since our goal is to generate dense reconstructions. In our experiments, we noticed that using local bundle adjustment and taking into consideration the most recent $N = 10$ frames in the system yields reasonable 3D reconstructions while still keeping computational costs relatively low.

Figure 1 shows a sample run of this process. The left image shows a video frame with the Harris features detected (illustrated with green boxes). The right image shows a sample reconstruction of the points. Although difficult to visualize in the image, the reconstruction successfully estimates the relative geometry of the scene. We can see correct localization for the features on the adjacent and far building walls, the foreground tree, as well as a few features on the ground plane.

3. Human Model Fitting

At this point we are able to take a video and generate a 3D reconstruction of the static elements in the observed scene. Moving objects present a much greater challenge to reconstruct since their location may differ by an unknown factor between consecutive video frames. This problem becomes even more challenging when the objects are nonrigid or articulate, such as people. In order to successfully solve this problem, we employ the use of 2D human pose estimation techniques, as well as our own 3D model of the human body. The basic idea is to estimate the relative 2D pose of the person, then fit a 3D model to the resulting 2D pose.

3.1. 2D pose estimation

The first step is to determine if a person is present in the scene. We can do this efficiently by running a face detector on selected frames. If a face is found (either frontal or profile), we assume there is a person in the scene. Next, we employ the use of the 2D human pose estimator proposed by Ramanan et al. [16, 17] to estimate the pose of the detected person.

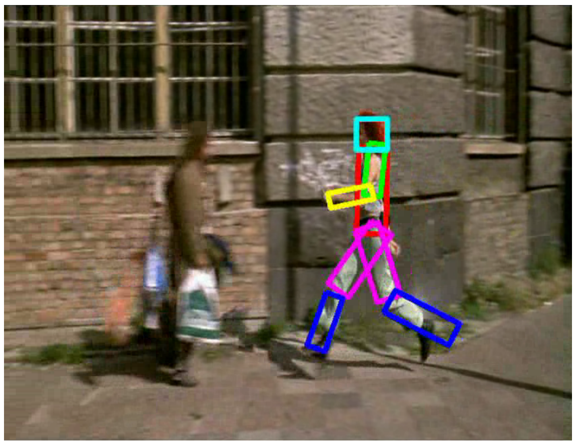
Figure 2. Sample pose estimate from the movie “Run Lola Run.”

In [16, 17], the authors present a framework for estimating the relative 2D pose of a person across a video sequence. Here, the authors use a simple 2D model of a human composed of a head, torso, upper and lower arms, as well as upper and lower legs. The system first builds an appearance-based model of each person, then tracks each person by detecting the model in subsequent frames. In order to build the model, candidate parts (for each part of the model) are first found using an edge-based part detector. These candidate parts are then clustered via the mean-shift algorithm in order to find patches with similar appearances across time.

Afterwards, the authors enforce a predefined motion model on the resulting candidate parts. Clusters that are smaller than a certain size, or that never move, are removed (effectively rendering the method useless for tracking people who stand still across the entire video sequence). Each torso cluster is considered a unique person. Once the torso is estimated, the rest of the limbs are inferred according to their model. For further details on this method, we refer the reader to [16, 17]. Figure 2 illustrates a pose estimate calculated using this method.

3.2. 3D pose estimation

At this point, we have accurately estimated the 2D pose of each person in the video. The next step is to infer a 3D model of a person from the already calculated 2D model. To do this, we use a simplified 3D model of a human with the same basic sections as the 2D model proposed by Ramanan et al. [16, 17].

Our 3D model is composed of block structures, each corresponding to one of the following body sections: head, torso, upper and lower arms, upper and lower legs. This model will be used to represent the person in the reconstructed 3D scene. Given such a mapping between a 2D model and a 3D model, there may exist several different 3D poses that correspond to a given 2D pose (i.e., more than

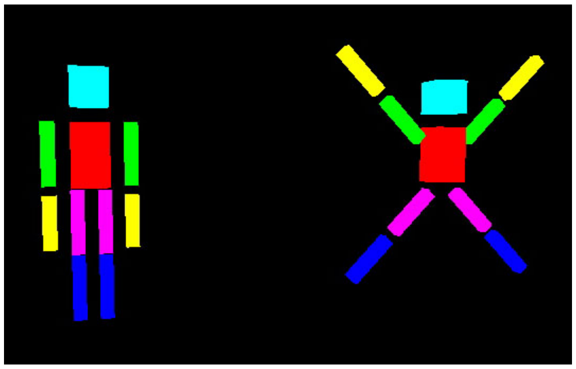

Figure 3. Two sample 3D model poses showing joint articulation.

one 3D pose may share the same projection in 2D space). For this reason, we chose to use a simplified 3D model with several constraints.

First of all, the overall model can only be rotated freely about the Y axis. Each upper arm section has an almost free range of motion, but cannot bend backwards towards the rear of the body, or be rotated so that it collides with the torso or the head. The lower arm is more restricted, only allowing 90 degrees of freedom to bend at the elbow joint. The upper leg segments can be rotated up to 45 degrees forwards, backwards, or away from each other. The legs are not allowed to bend inwards (making it impossible for our model to cross its legs). The lower leg segments have 135 degrees of freedom at the knee. Both the knee and elbow joints are also constrained so that the bending occurs in one direction only (i.e., if the arm is hanging straight down, the elbow can only be bent in such a way so that the lower arm is rotated towards the direction that the model is facing, and not towards the rear; similar constraints are applied to the knee to simulate typical human limitations). Figure 3 shows our 3D model in two possible poses.

The first step is to determine the relative orientation of the 3D model, which will be refined later. Since we are limited to a rotation about the Y axis, we need to figure out in which direction the model is facing. To do this, we look at the position of the 2D model over time. We assume that the person is always facing in the direction that they are walking. Therefore, if the 2D pose estimation retrieves a 2D model moving in the negative X direction, out initial 3D model orientation is to rotate the model so that it is facing this same direction. Because of this, we are not able to accurately model people walking backwards.

Next, we need to align the sections of our 3D model with the 2D pose estimation. The first thing we do is to calculate the relative angle of the torso (in relation to the image's Y axis). This angle will give us the amount of rotation to be applied to our 3D model's torso. From this, we calculate the relative angle between the torso and each upper arm segment in the 2D pose estimate, and apply them to the 3D model. Next, we calculate the angle between the upper and lower arm segments, and rotate the 3D model's lower arm accordingly. A similar process is repeated for the upper and lower leg segments.

At this point we have aligned our 3D model with the 2D pose estimation calculated from the video frames. The final step is to find the correct placement of the 3D model in the reconstructed 3D scene. Recall from Section 2 that our 3D reconstruction is only able to triangulate static points in the scene. Because of this, the 3D reconstruction cannot accurately recover any points on the moving person. Instead, we attempt to place our 3D model's feet in the 3D scene. We begin with the 2D pose estimation, and attempt to find the image feature with the shortest euclidean distance to the bottom of one of the lower leg segments. When this point has been found, we place our 3D model in the reconstructed scene so that the lower leg segment has the same 3D coordinate as the image feature selected. If more than one feature is found to have the overall shortest distance, the feature is chosen at random.

The final step is to refine the orientation of our 3D model. Recall that we initially set the orientation based on the direction of the 2D pose estimate. Now, since we have an estimated 3D location for our model across several video frames, we can refine this orientation by setting it equal to the direction that the 3D model is traveling in (equivalent to the direction between the features closest to our model's feet over time).

4. Experiments

We present several of our current results in this section. We use various video clips obtained from Youtube as input to our algorithm. As such, the videos are of reasonably low resolution and contain several compression artifacts. Since the method presented here cannot handle shot boundaries within video clips, we manually broke the videos into series of shots and present some of our experimental results here.

It is important to note that since our results are simply dense point clouds, they are difficult to illustrate in images. Therefore, a bit of imagination may be required when interpreting the sample reconstructions illustrated in this paper. To better illustrate our results, we have placed videos of sample reconstructions online at `http://www.youtube.com/view_play_list?p=5410456BCC285FE1`.

The first experiment is from the music video "Here I Am" by Bryan Adams, illustrated in Figure 4. In this clip, the camera is translating from left to right. The scene contains a man sitting on the ground leaning against a rock, while the background contains several rock formations. Here, the camera motion is not sufficient to accurately measure the curvature of the man's body, but it is sufficient to

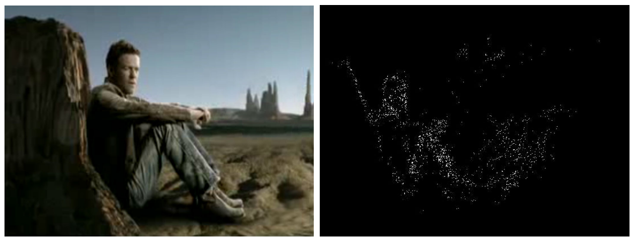
Figure 4. Bryan Adams music video and corresponding 3D reconstruction.

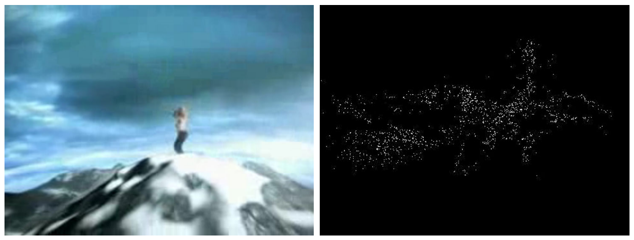
Figure 5. Shakira music video and corresponding 3D reconstruction.

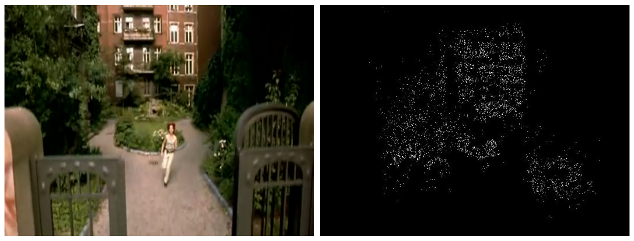
Figure 6. Sample frame from the motion picture "Run Lola Run."

successfully place the man and the rock on the foreground of the reconstruction, while clearly separating them from the background rocks.

The second experiment is from the music video "Whenever, Wherever" by Shakira. In this clip, we see several rock formations in the foreground, with a woman dancing on top of one of them. Here, our method successfully separates the two rocks and is even able to recover the smooth inclined surface shape of the visible rocks. Again, the method localizes the woman on the rocks correctly in the reconstruction, but the video quality is too poor to be able to generate an accurate reconstruction of her body curvature. Still, the relative geometry of the scene, as well as the locations of the objects, was recovered successfully.

The next set of experiments were computed from scenes from the film "Run Lola Run." In the first clip, illustrated in Figure 6, the camera motion corresponds to a translation in the negative Y direction. Here we see a building in the background, with trees on the sides of the foreground and a fountain near the center of the ground plane. Although the system successfully detected the woman's face at the end of the video clip, the 3D pose estimation was unable to estimate her pose.

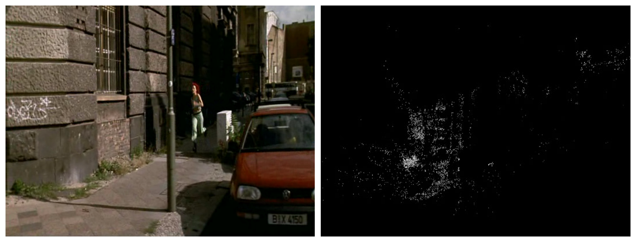
Figure 7. Sample frame from the motion picture "Run Lola Run."

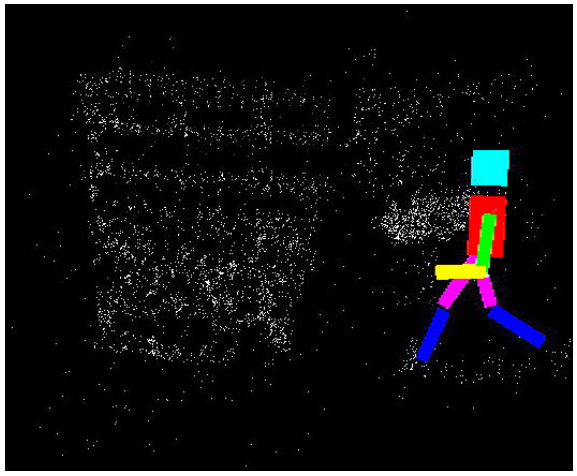
Figure 8. Sample frame from the motion picture "Run Lola Run."

Figure 7 shows yet another experiment using a clip from "Run Lola Run." The scene contains a woman running along a sidewalk, while the camera moves from left to right in order to follow her. Our reconstruction is able to recover the structure of the wall clearly, including a dense representation of high-texture areas. Unfortunately the Harris feature detector used finds very few points around the vehicles on the right side of the screen, so we were unable to recover any 3D representation of this part of the scene. This is likely due to the lack of salient textures and gradients in the region. As in the previous experiment, the face detection returns a positive detection at this point in the clip, but the 2D pose estimator fails to accurately locate the woman's body due to changes in lighting and scale.

Figure 8 shows results from the same scene as in the previous example, but obtained a few seconds later in the clip. Here, not only are we able to recover the 3D structure of the scene, but we are also able to accurately locate the woman using the method described in Section 3. The corresponding image frame is shown in Figure 2.

5. Conclusion

As video databases become more prevalent, the need for powerful video analysis and mining techniques increases. Many tasks such as shot-boundary detection, video summarization, and content-based video retrieval rely on measurements and techniques based on two-dimensional image features. Although some of these tools can be very powerful, they are not capable of capturing the inherent three dimensional structure of a scene. The goal of the research presented in this paper is to provide additional tools capable of capturing the three dimensional structure of a scene, in the hopes of facilitating and improving various video analysis tasks.

Several works have been published in recent years dealing with 3D estimation of scenes observed from monocular video cameras. In this paper, we borrow many of the proposed ideas to develop a framework capable of generating a dense 3D reconstruction of an observed scene with no previous knowledge of any scene parameters. By making use of the available camera motion within a video clip, we are able to generate a 3D reconstruction of the static objects in the scene by tracking salient features in the image frames, and measuring their relative changes in image coordinates as the camera moves about the world.

Our proposed system is also capable of locating people, estimating their relative pose, and inserting a 3D human model within the reconstructed scene. Ramanan et al.'s method is used for human pose estimation [16, 17] in the individual image frames of the clip. In order to extend these results to three dimensions, we make use of our simplistic 3D human model and align it with the results obtained from the 2D pose estimation. Once the 3D model is accurately fit to the 2D model, we place the 3D model into the previously constructed 3D scene by finding the point with the shortest distance to the person's feet.

Although the results presented here show great promise, much work remains to be done before this technique can be truly useful to other researchers. We are currently extending the proposed method to reconstruct low-textured regions of the videos, since the Harris detector fails to obtain features at these locations, in order to increase the density of the reconstructed point clouds. Additionally, we are currently working on improving the performance of the human pose estimation in order to increase robustness and better estimate human poses, while attempting to perform all calculations efficiently.

Acknowledgements

This work is partially funded by NSF grants CCF-0514743 and DMS-0713012, and the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 RR021813. Information on the National Centers for Biomedical Computing can be obtained from http://nihroadmap.nih.gov/bioinformatics.

References

[1] H. Bay, T. Tuytelaars, and L. J. V. Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[2] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, pages 120–126, November 2000.

[3] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1052–1067, 2007.

[4] J. Deutscher, M. Isard, and J. Maccormick. Automatic camera calibration from a single manhattan image. In *European Conference on Computer Vision*, pages 175–205, 2002.

[5] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, August 2002.

[6] L. Grammatikopoulos, G. Karras, E. Petsa, and I. Kalisperakis. A unified approach for automatic camera calibration from vanishing points. In *Image Engineering and Vision Metrology*, 2006.

[7] R. Haralick, C. nan Lee, K. Ottenberg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, pages 592–598, 1991.

[8] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge Press, 2003.

[10] G. Klein and D. W. Murray. Parallel tracking and mapping for small ar workspaces. In *International Symposium on Mixed Augmented Reality*, 2007.

[11] J.-F. Li, Y. Xu, Y. Chen, and Y. Jia. A real-time 3d human body tracking and modeling system. In *International Conference on Image Processing*, pages 2809–2812, 2006.

[12] D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

[13] A. S. Micilotta, E. jon Ong, and R. Bowden. Real-time upper body detection and 3d pose estimation in monoscopic images. In *European Conference on Computer Vision*, pages 139–150, 2006.

[14] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 2008.

[15] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, 2004.

[16] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 271–278, 2005.

[17] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, 29(1):65–81, January 2007.

[18] M. I. Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties, February 2004.

[19] T. Tamaki, T. Yamamura, and N. Ohnishi. An automatic camera calibration method with image registration technique. In *Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics*, pages 317–322, 2000.

[20] R. Urtasun and P. Fua. 3d human body tracking using deterministic temporal motion models. In *European Conference on Computer Vision*, pages 92–106, 2004.

[21] Z. Xiong, R. Radharkishnan, A. Divakaran, Y. Rui, and T. S. Huang. *A Unified Framework for Video Summarization, Browsing and Retrieval*. Elsevier, 2006.