Differential Camera Tracking through Linearizing the Local Appearance Manifold

Hua Yang Marc Pollefeys Greg Welch Jan-Michael Frahm Adrian Ilie

Computer Science Department
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175, USA

Abstract

The appearance of a scene is a function of the scene contents, the lighting, and the camera pose. A set of n-pixel images of a non-degenerate scene captured from different perspectives lie on a 6D nonlinear manifold in $\Re^n$. In general, this nonlinear manifold is complicated and numerous samples are required to learn it globally.

In this paper, we present a novel method and some preliminary results for incrementally tracking camera motion through sampling and linearizing the local appearance manifold. At each frame time, we use a cluster of calibrated and synchronized small baseline cameras to capture scene appearance samples at different camera poses. We compute a first-order approximation of the appearance manifold around the current camera pose. Then, as new cluster samples are captured at the next frame time, we estimate the incremental camera motion using a linear solver. By using intensity measurements and directly sampling the appearance manifold, our method avoids the commonly-used feature extraction and matching processes, and does not require 3D correspondences across frames. Thus it can be used for scenes with complicated surface materials, geometries, and view-dependent appearance properties, situations where many other camera tracking methods would fail.

1. Introduction

In this paper we address the challenging problem of tracking in scenes with highly view-dependent appearances. For example, scenes with curved reflective surfaces, semi-transparent surfaces, and specular reflections all change in appearance as the viewpoint changes. This confounding behavior typically makes motion estimation very difficult.

Traditionally, tracking is formulated as a search problem in the parameter space of the transformation. There are two main classes of tracking approaches. The first approach estimates motion using salient features of the scene. As a classic example, Lounget-Higgins [15] extracted features from a stereo image pair, and used the epipolar geometry to estimate the relative camera motion. This method was later extended to multiple views for simultaneously estimating the camera internal parameters, the structure and the motion [1, 18]. The second approach extracts the tracking information from the whole appearance observations. For instance, feature matching is replaced by optical flow in differential settings [5, 13, 11].

The second approach typically relies on an invariant scene appearance or a parametric model that accommodates the varying scene appearance. For example, Murase and Nayar [17] proposed an eigenspace-based recognition method that projects object appearance samples onto a hyperspace parameterized by object pose and illumination. In [3, 10, 6], an object under different lighting conditions was represented as an illumination linear subspace. Since projecting the global appearance of an object into a PCA space requires precise segmentation and normalization, these methods can encounter difficulties in a general case. To alleviate these difficulties, [7] proposed to represent the target object as surface patches and perform recognition in the local appearance space. A similar model is used for object tracking [16], in which the object is described as a collection of local templates whose displacements can be computed efficiently using a linear solver. Bascle and Deriche modeled the appearance of an object using texture appearance templates [2]. A parametric representation of the scene can also be formulated as a global statistics of the objects' appearance [4, 8]. Recently, Elgammal proposed to learn a generative appearance model of the target object offline, and employ the model to compute the geometric transformation given the change of the object appearance [9].

In this paper, we present a method that tracks the camera motion through linearization of the local appearance mani-

fold. A camera cluster with small baselines is used to acquire local appearance samples. Then these samples are used to compute a linear approximation of the local appearance manifold and to estimate the camera motion parameters in this linear space. Our proposed method does not require any 3D or 2D correspondences, thus it accommodates scenes with view-dependent appearance variances. In contrast to the methods based on learning an invariant appearance representation, our method avoids the learning process that requires training data.

2. Problem statement

Our novel tracking approach targets scenes with highly view-dependent appearances. As far as we know, this class of scenes is not handled by any existing technique. It is very difficult to model appearance behavior in general, and for this specific class of scenes it is particularly hard to find a *global* model due to the highly nonlinear appearance function. Instead we use multiple *local* samples that represent the manifold about the current viewpoint. These local samples are concurrently extracted from the scene while performing the tracking.

Consider a camera that undergoes complete 6D motion (3D translation and 3D rotation) in a static scene. At each frame time, it captures from its current pose P an n-pixel intensity image I of the scene. Each appearance sample I represents a point in a high-dimensional space $\Re^n$. As the camera changes its pose P, I also changes, moving along a manifold in $\Re^n$. One can see that there exists a mapping from P to I, denoted as $I = f(P)$. Since the transformation space of the camera pose has six degrees of freedom, the dimensionality of the appearance manifold is at most six. The dimension will be smaller for degenerate cases — for example, when a camera looks at a very distant scene. For these cases, f is not invertible and the motion can not be fully recovered. In this paper, we consider only cases where the appearance manifold is not degenerate, and accordingly f is invertible.

In general the appearance manifold of a scene is highly nonlinear. Therefore, numerous samples need to be acquired to learn a representation of it. Although learning f globally would be an ideal solution, this task is mostly infeasible in practice due to changes of lighting conditions and movements of scene objects during capture, which violate the static scene assumption. However, as discussed above, we know that the dimensionality of the appearance manifold is at most six. Hence, with six samples one can generate a linear approximation of the appearance manifold as $dI = FdP$, where dI is the difference image, dP is the camera motion, and F is the Jacobian matrix. This holds as long as the camera motion is within the range of an acceptable linear approximation. To achieve our goal we need to solve three fundamental problems:

1. How to capture local appearance samples?
2. How to derive a local linear approximation given appearance samples?
3. Is a linear approximation sufficient?

We will address these problems in the following sections.

3. Tracking

This section introduces our novel tracking framework. We will introduce a technique to capture the appearance samples during tracking and discuss an algorithm to linearize the local appearance manifold and track the camera motion within the linear space.

3.1. Linearize the appearance manifold

Consider a camera that undergoes complete 6D motion. Assume at time t, we simultaneously acquire a reference image I_0 from the current camera pose P_0, and m perturbed images I_k at nearby perspectives P_k $(k = 1, \ldots, m)$. Thus we can compute m difference images dI_k and camera motion dP_k by subtracting I_0 and P_0 from I_k and P_k. We want to use these samples to linearize the appearance function $I(P)$ around P_0:

$$I = I_0 + F\,(P - P_0) \tag{1}$$

$$dI = F\,dP. \tag{2}$$

Here I and dI are n-pixel images represented as $n \times 1$ vectors, P and dP are 6×1 pose vectors, and F is the Jacobian (partial derivative) matrix $\partial I / \partial P$ of size $n \times 6$. Given m samples of dI_k and known dP_k, we can combine these sample vectors into matrices and write the linear equation as:

$$[dI_1, dI_2, \ldots, dI_m] = F\ [dP_1, dP_2, \ldots, dP_m] \tag{3}$$

If m is greater than 6 and the images and poses are not degenerate, the equation is of rank 6 and can compute the least square solution of F using the Moore-Penrose pseudo inverse as

$$F = [dI_1, dI_2, \ldots, dI_m]\ [dP_1, dP_2, \ldots, dP_m]^{+} \tag{4}$$

The above discussion addresses the appearance manifold linearization problem in the general case where $m \geq 6$. For an efficient system, one would like to use minimum number of perturbed cameras whose poses expand a 6D motion space. In this case, $m = 6$ and the Jacobian becomes

$$F = [dI_1, dI_2, \ldots, dI_6]\ [dP_1, dP_2, \ldots, dP_6]^{-1} \tag{5}$$

Once a linear approximation is computed for the local appearance manifold, we can estimate the camera motion

using a linear solver. Assume at frame $t+1$, an updated reference image $\tilde{I}_0$ is captured at the new camera pose $\tilde{P}_0$. We can then compute a temporal difference image $d\tilde{I}_0 = \tilde{I}_0 - I_0$ and estimate the camera motion $d\tilde{P} = \tilde{P}_0 - P_0$ as the least square solution of Equation (6).

$$F\, d\tilde{P} = d\tilde{I}_0 \tag{6}$$

3.2. Sample with a camera cluster

As shown in Fig. 1 we have constructed a prototype *differential camera cluster* consisting of four synchronized and calibrated small baseline cameras: one *center* camera C_0 and three cameras C_1, C_2 and C_3 that are offset from the center. The coordinate frame of the cluster is defined to align with C_0. We call the other three cameras C_1, C_2 and C_3 *translational* cameras as they capture images from translated viewpoints.[1] At any point in time, the *center* camera and the *translational* cameras can be used to obtain four simultaneous appearance samples of the local appearance manifold. See the example images indicated by the green (solid) arrows in Fig. 1. In addition we generate three warped images by rotating the image plane of C_0 around its three coordinate axes. See the example images indicated by the red (dashed) arrows in Fig. 1. One can consider these warped images as having been captured by three virtual *rotational* cameras C_4, C_5 and C_6, each with the same camera center as C_0 but with rotated axes. Thus at any frame the cluster effectively "captures" seven local appearance samples $I_0, \ldots, I_6$. In a non-degenerate case these images can be used to derive a first order approximation of the local appearance manifold as described in section 3.1.

The camera cluster provides seven real-time appearance samples. Because the *rotational* images are warped versions of the *center* image, these four samples are from the same manifold. However, the *translational* samples are captured using different cameras. To use these samples for linearizing the local manifold captured by the center camera, we need geometric and photometric consistency across cameras. Assuming the radial distortion is removed from all cameras, we can achieve geometric consistency by using the *intrinsic* parameters of the *center* camera for the *translational* cameras. Specifically we decompose the projection matrix [12] to obtain camera intrinsic and extrinsic parameters. After that we generate three virtual *translational* cameras using the *intrinsic* parameters of the *center* camera and the *extrinsic* parameters of the real *translational* cameras. We then generate the translational images using homography mappings. To ensure photometric consistency across the cameras we used the approach presented in [14]. The approach consists of a closed-loop process that tunes the camera hardware settings such that the colors values of a 24-sample color target are consistent in all camera images. This is followed by a software post-processing step that uses the same 24 color samples to compute a mapping that further improves photometric consistency.

[1] In a general case, their axes do not need to be parallel with those of the *center* camera.

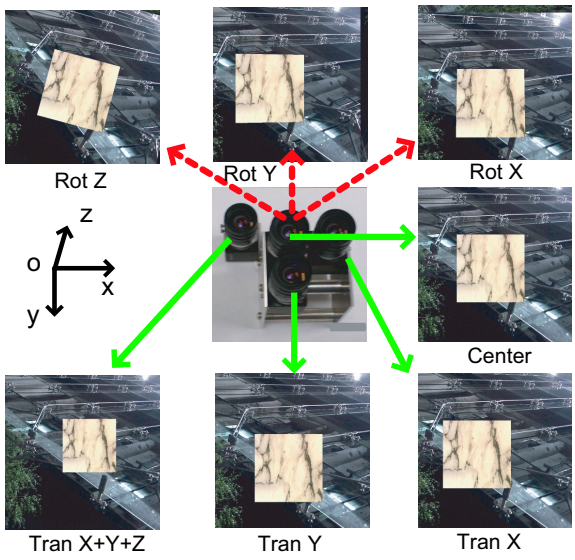

Figure 1. A prototype *differential camera cluster* (center) and illustrative images. We obtain seven images total: one center, three translated, and three rotated. Note that the images shown above were rendered with exaggerated baselines to make the differing perspectives more apparent.

4. Linearity of the local appearance manifold

4.1. An SVD analysis

In theory, given six non-degenerate samples one can always generate a linear approximation of the appearance manifold, but such a linearization is accurate only within a limited region. The size of each locally-linear region is determined by the local smoothness of the manifold. Typically, the scene appearance is a highly nonlinear function and its locally-linear regions are quite small. This means that any differential camera cluster should have very narrow baselines and a very high frame rate to acquire samples and restrict motions within a small locally-linear region. This is a great challenge. Even if these requirements can be satisfied, due to the inherent electronic noise of the camera, the signal-to-noise ratio of the spatial and temporal difference images might not be big enough to recover motion accurately. We try to alleviate this problem by blurring the images to smooth the appearance manifold.

Let us examine the smoothness of the appearance manifold for a synthetic 3D scene consisting of two textured planes (see the images in Fig. 2). The two planes are placed at different depths to help distinguishing the parallax ef-

fects from out-of-plane rotation and in-plane translation. Several rectangular white textures are pasted (around the periphery) to provide some low frequency texture to the dimmed background. The resolution of the synthetic camera is $n = 640 \times 640$. As shown in Fig. 2(a), we use a uniform distribution to randomly perturb the camera pose and generate $m = 50$ images from nearby perspectives. The maximum magnitude of the perturbation is carefully defined so that the imaging of a point on the frontal plane shifts at most 4 pixels away from its center position. The images are blurred using a Gaussian filter and sub-sampled at a 4-to-1 rate (see Fig. 2(b)). This sampling rate is higher than the Nyquist rate. We then acquire difference images by computing a mean image and subtracting it form the sub-sampled images. The pixel intensities of these difference images are reordered into column vectors and grouped into a $n \times 50$ matrix. We then apply SVD decomposition to the matrix.

To demonstrate the effects of smoothing the appearance manifold, we use three different Gaussian kernels of size $[20 \times 20, 40 \times 40, 80 \times 80]$ and $\sigma = [3, 6, 12]$ to filter the original perturbed images and generate three sets of blurred images. The SVD results on matrices constructed using these blurred images are shown in Fig. 2(d,e,f). One can see from Fig. 2(e,f) that there are six significant singular values, which implies that the smoothed local appearance manifold can be appropriately linearized and the motion can be recovered. Yet in Fig. 2(d), the fourth, fifth and sixth singular values are not easily distinguished from the rest. While in general this could be an indication of a degenerate scene (under-constrained), in this case it is an indication that there is considerable non-linearity embedded in the appearance manifold for the region where the samples are captured. To address this we can use smaller baselines to sample closer to the center pose, or smooth the images more heavily.

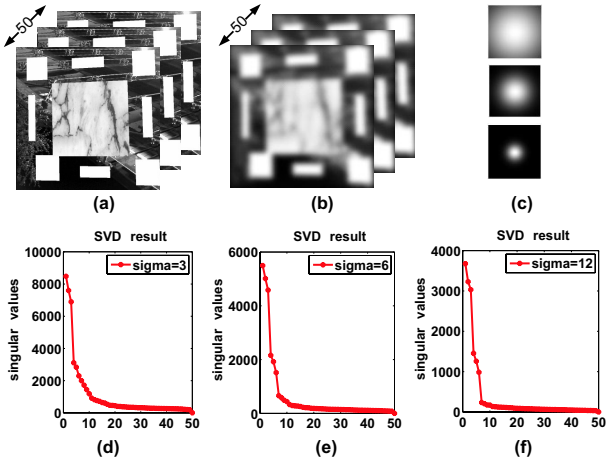

Figure 2. SVD analysis using synthetic images of a 3D scene with textured foreground and background planes. The foreground plane is the marbled square in the middle of the images. (a) The original images captured from 50 close perspectives. (b) The blurred images. Three Gaussian filters with $\sigma = [3, 6, 12]$ are used. (c) The power spectrum of the three Gaussian filter within $[-40, 40]$ of the frequency domain. (d) (e) (f) SVD results using the three different filters.

4.2. A quantitative analysis using sine waves

The SVD analysis shows that a non-linear appearance manifold can be smoothed using a low-pass filter. One can thus sample inside a locally-linear region and compute a linear approximation of the filtered manifold. In this section, we present a quantitative analysis of sampling and estimating motion on the smoothed manifold. In particular, for a given spacial sampling density (camera baseline), we try to determine the threshold frequency for the low-pass filter and derive an estimate of the estimation error. In the analysis, we use sine signals as the scene contents, for any image can be decomposed into a series of sine waves of different frequencies.

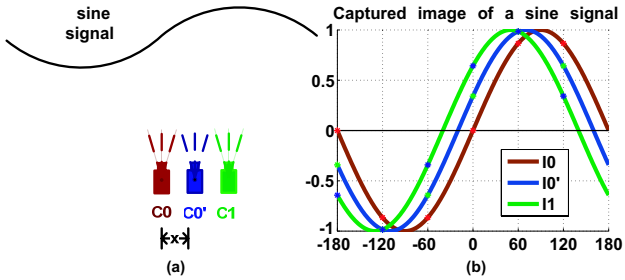

Figure 3. 1D orthographic cameras capture images of a sine signal.

Without loss of generality, let us consider the problem in one dimension. Suppose at current frame, two parallel 1D orthographic cameras C_0 and C_1 capture images I_0 and I_1 of a sine signal as shown in Fig. 3(a). At the next frame, C_0 moves x degrees and captures a new image $\tilde{I_0}$. All three images contain r samples from one period of the sine signal, starting from initial phases a_0, a_1 and $\tilde{a_0} = a_0 + x$ as shown in Fig. 3(b). Using these $3r$ samples, we can compute the motion estimate $\tilde{x}$. We can write these three 1D images as:

$$I_j = sin(a_j + r\ k) \tag{7}$$

where $j \in [0, 1, \tilde{0}]$ and $k \in [0, 1, \ldots, \lfloor 359/r \rfloor]$. Substituting Equation (7) into Equation (1) and (2), we get the following equation for estimating the motion $\tilde{x}$:

$$\tilde{x} = (a_1 - a_0)\ (\tilde{I_0} - I_0)/(I_1 - I_0) \tag{8}$$

The estimation error $(\tilde{x} - x)/b$ for different motion x using different baselines $b = a1 - a0$ are shown in Fig. 4. One can see from Fig. 4(a) that the estimation error remains relatively small for motion x within $[0, b]$ and increases quickly

beyond this region. Region $[0, b]$ is the linear region for motion estimation. In other words, we can only expect to recover camera motion that is smaller or close to the baseline. As a zoom in of Fig. 4(a), Fig. 4 (b) shows that while using a bigger baseline enlarges the linear region, it also causes greater estimation errors for motions within the linear range. For instance, using a baseline $b = 120$ the estimation error for $x = 30$ is 5 degrees. In practice, we empirically choose $b = 90$ to achieve a balance between the size of the tracking volume and the tracking accuracy. This means for a particular scene, we can consider adjusting the camera baseline to be a quarter of the highest frequency in the image. Or for a fixed baseline, we should use a low-pass filter whose threshold wavelength is four times the length of the baseline. Then we can expect to achieve good estimation for a motion within the baseline.

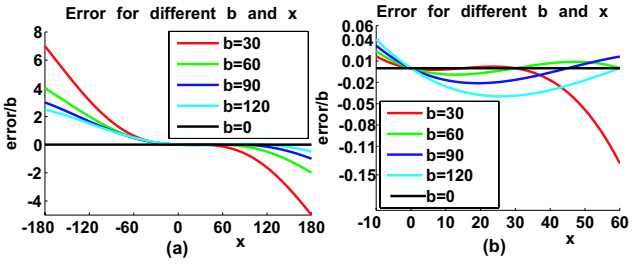

Figure 4. Analysis of motion estimation error. (a) Estimation error $(\tilde{x} - x)/b$ for specific baselines $b = a1 - a0 = [30, 60, 90, 120]$ degrees, and $x \in [-180, 180]$ degrees. (b) Zoom in of (a) for $x \in [-10, 60]$ degrees.

While the analysis of 1D orthogonal cameras observing sine signals derives some interesting conclusions, it can not be directly extended to 3D scenes and perspective cameras. Here we provide some empirical verification. Consider the synthetic data set used in section 4.1. The shift of a point in the image plane is in $[0, 4]$ pixels. According to the previous analysis, to obtain a linear region of $[0, 4]$, the pixel displacement with respect to the scene and the camera baseline should be 4 pixels and the threshold wavelength should be 16 pixels. That is a 40 cycle signal for a 640×640 image. From Fig. 2(c) and Table 1, we can see that 95% energy of a Gaussian filter of $\sigma = 6$ is covered by its low-frequency spectrum inside $[-40, 40]$. Thus the threshold of this low-pass filter is 40 cycles. Whereas for the filter with $\sigma = 3$, only 59% energy is within the 40 cycle low-frequency spectrum. As shown in Fig. 2(d,e), a good linear approximation can be computed for images blurred with the first filter but not for the second one. The results support the analysis.

The above analysis shows that the appropriate tracking volume is $[0, b]$, which means the estimation error is asymmetric. To solve this problem, one can consider choosing the center camera according to the motion direction. For instance, in the 1D case, if the cameras move to the left (see Fig. 3(a)), we can choose C_1 to be the center camera. Another solution is to keep C_0 as the center camera but use I_0 and I_1 captured at the new frame to compute intensity derivatives. These solutions can be extended to 3D cases. To use this technique, we need to detect the direction of the motion. This can be achieved by doing motion estimation twice, first for the direction and then for the real estimation. While this solution appears awkward, the additional computation is affordable, as it only involves solving a linear system that can be done efficiently.

Gaussian kernel σ	3	6	12
Percentage energy covered by the low frequency spectrums [-40,40]	59%	96%	99%

Table 1. Energy distribution of Gaussian filters in the frequency domain.

5. Experimental Results

We first present some results based on synthetic data. We simulated a scene consisting of a textured planar patch and a curved mirror, both contained inside a cube. The inner six surfaces of the (surrounding) cube were textured using the same image. The simulated camera cluster was placed in front of the curved mirror. Thus the camera cluster viewed some of the scene beyond the edges of the mirror, and some of the reflection. Traditional tracking techniques would not perform well on this data, since the epipolar constraint does not hold for the distorted scene reflection from the curved mirror as the camera moves.

Fig. 5 presents some tracking results on the synthetic scene over 40 frames. An original image (640×640) is shown in Fig. 5(a), where the border of the curved mirror is marked in green. One can see that the reflection of the planar patch is distorted by the curved mirror. When generating the image sequences, we restricted the maximum extent of the camera motion so that the pixel motion for frontal scene points would be less than 4 pixels. We generated blurred images (shown in Fig. 5(e)) using a Gaussian kernel of 160×160 with $\sigma = 24$ and sub-sampled them at 32 cycles (20 pixels). We choose a σ that was larger than the analysis result shown in section 4.2 to accommodate the reflection of the rear scene, which could move faster in the image plane than the front scene. We show the translation and rotation estimates in Fig. 5 (b-d) and (f-h). The horizontal axes represent frame numbers and the vertical axes represent the accumulated motion across the previous frames. The red (solid) curves represent the true value, and blue (dashed) curves represent the estimated value. The translation and rotation of the camera is defined in a global coordinate system that is aligned with the coordinate system of the center camera at the initial frame.

In a second experiment, we tracked a real camera cluster. The cluster was built using four PointGrey Flea digital color cameras (see Fig. 1). The baselines are 34 mm in the X and Y direction, and 66 mm in the Z direction. As described in section 3.2, geometric and photometric consistencies were enforced across the four cameras. The color images were converted to grayscale for scene appearance samples.

To obtain some form of a ground-truth reference motion path we moved the camera cluster along some predetermined grid points marked on a table (the X-Z plane) while imaging the scene. Prior to data collection, the cameras were manually adjusted to obtain parallel principal axes across frames. The results are shown in Fig. 6. An original image and its blurred version is shown in Fig. 6(a) and (e). The resolution of the original image is 1024×768. The image was blurred using a Gaussian kernel of 160×160 with $\sigma = 24$, and then sub-sampled at a ratio of 20 to 1. The accumulated translations and rotations are shown in Fig. 6(b-d) and (f-h). Again, the translations and rotations are defined in the coordinate of the center camera at the initial frame. One can see that the algorithm achieves good estimates of X and Y translations, and the estimated rotations and Z translations are small. We believe the exhibited error is due primarily to inherent drift in any (incremental) approach, and registration error introduced by our manual alignment process.

Our third experiment demonstrates the tracking of a hand-held camera cluster over 200 frames. As the ground truth motion was unknown, we illustrate the tracking accuracy using projection error. Seven reference scene points were used. Their 2D coordinates in two camera images of the initial frame were extracted and matched using a standard OpenCV KLT tracker at sub-pixel accuracy. We then back-projected and computed their 3D positions in the world coordinates defined by the pose of the center camera at the first frame. As the cluster moved, its incremental motion at every frame was estimated and the accumulated motion between the current frame and the first frame was computed. These accumulated motion parameters were then used to compute a projection matrix of the center camera at its current pose. We used the estimated projection matrix to project the reference 3D points onto the current center image, and indicated the projections with white patches surrounded by red circles (see Fig. 7). Fig. 7(a) shows the reference points in the center image of the initial frame. Their projections using the estimated motion parameters are shown in Fig. 7(b-f). Since the reference points are initially back-projected using small baseline stereo, the computed 3D positions may be considerably off from their true values. Therefore the final projection result contains error from both the incremental tracking and the initial triangulation process.

In the fourth experiment, a hand-held camera cluster was tracked in a scene with semi-transparency. We pointed the cameras to a large window and captured image sequences at dawn. The indoor lighting environment was adjusted such that the cameras see both the outside and the reflection of the inside (see Fig. 8). Typical feature based tracking algorithm would encounter difficulties extracting and matching features, since the image is a superimposition of two layers undergoing different motion. Like we did in the third experiment, we computed the projection matrices from the camera motion estimate and project the reference scene points onto the center image (Fig. 8(b-f)). However, as the standard KLT method does not handle semi-transparency well, the reference points are manually selected in the initial frame (Fig. 8(a)) at an integer pixel accuracy.

6. Conclusions

We presented a novel method for incrementally tracking camera motion through sampling and linearizing the local appearance manifold. We have demonstrated the method using both simulation and a real prototype camera cluster.

One area of future work we envision is related to the imaging component of the cluster. We have some ideas for using custom optics to effectively achieve the same center, translational, and rotational images using a *single* image sensor. This could make the unit more compact, while also helping address color, geometry, and speed issues.

In addition, the simple and regular nature of the computation could lead to a fast system for on-line estimation, perhaps using specialized embedded hardware such as FPGAs. As the processing speed increases, the inter-frame time could be decreased, improving the linear approximations and some other aspects of the approach.

Finally, because the method is inherently incremental (estimating and integrating dP), it is likely that in practice one would want to periodically use some of the images for more conventional feature-based drift correction. Like MPEG and other video encoding schemes, future hardware could send a continuous dP stream with periodic key frames to the host computer, enabling fast incremental tracking with drift estimation and correction.

Acknowledgements

We acknowledge the support of our colleagues at UNC-CS. Gary Bishop inspired the insight of sine wave analysis. John Thomas built the rig for the camera cluster. Herman Towles and Philippos Mordohai helped with the system and experiment setup. This effort is primarily supported by National Library of Medicine contract N01-LM-3-3514: "3D Telepresence for Medical Consultation," and in part by NSF grant EIA-0303590, NSF Career award IIS-0237533 and a David and Lucille Packard Foundation Fellowship.

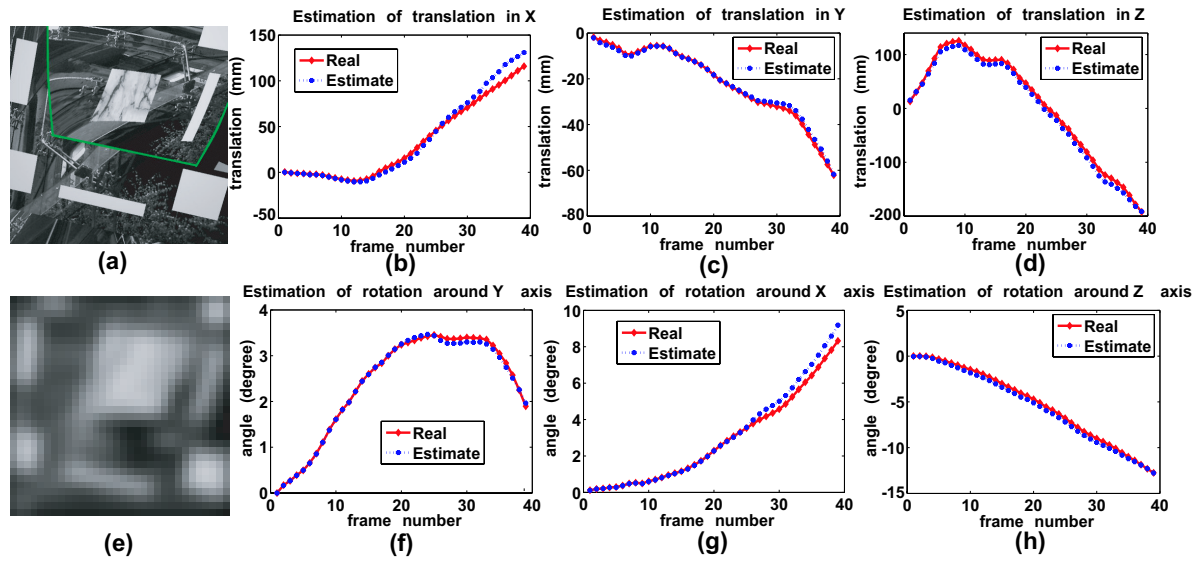

Figure 5. Tracking in a synthetic scene with a curved mirror over 40 frames. (a) An image of the synthetic scene. The border of the curved mirror is marked in green. (e) A blurred image. (b)-(d) Estimation of camera translations (in mm) in X, Y and Z directions. (f)-(h) Estimation of camera rotation angles (in degree) around Y, X and Z axes. Red (solid) curves represent the true values of the accumulated motion, blue (dashed) curves represent the estimated values of the accumulated motion.

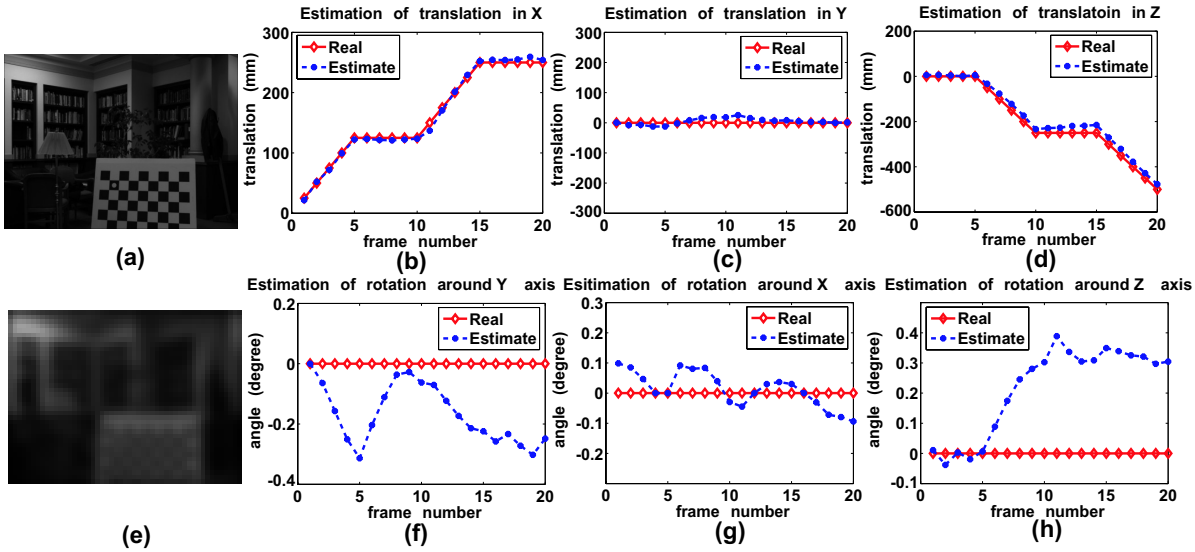

Figure 6. Tracking a controlled camera motion over 20 frames. (a) A image of the real scene. (e) A blurred image. (b)-(d) Estimation of camera translations (in mm). (f)-(h) Estimation of camera rotation angles (in degree). Red (solid) curves represent the true values of the accumulated motion, blue (dashed) curves represent the estimated values of the accumulated motion.

References

[1] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17(6), 1995.

[2] B. Bascle and R. Deriche. Region tracking through image sequences. *In Proc. of International Conference on Computer Vison*, 2005.

[3] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible illumination conditions. *International Journal of Computer Vision*, 28(3), 1998.

[4] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. *In Proc. of Computer Vison and Pattern Recognition*, 1998.

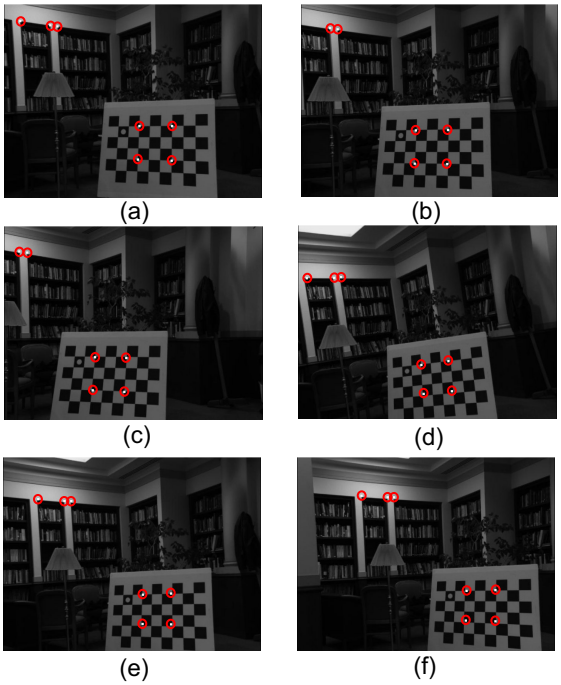

Figure 7. Tracking a hand-held camera motion over 200 frames. The tracking results are illustrated through projecting seven 3D scene points onto the *center* image. The projection matrix is computed using the motion estimate. The reference scene points are marked as white patches surrounded by red circles. (a) The extracted reference points in the *center* image at the initial frame. (b)-(f) The projected points in the *center* image at frame 20, 60, 160, 180 and 200.

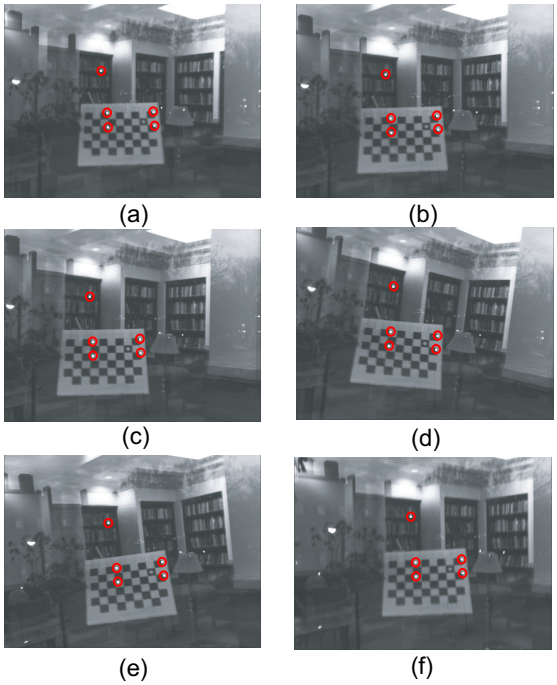

Figure 8. Tracking a hand-held camera cluster moving in a scene with semi-transparency for 60 frames. The tracking results are illustrated through projecting five 3D scene points onto the *center* image. The projection matrix is computed using the motion estimate. The reference points are marked as white patches surrounded by red circles. (a) The manually selected points in the *center* image at the initial frame. (b)-(f) The projected points in the *center* image at frame 10, 20, 30, 50 and 60.

[5] A. Bruss and B. Horn. Passive navigation. *Computer Vision, Graphics and Image Process*, 21, 1983.

[6] M. L. Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An apporach based on registration of texture-mapped 3d models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(4), 2000.

[7] V. Colin de Verdire and J. L. Crowley. Visual recognition using local appearance. In *In Proc. of European Conference on Computer Vison*, 1998.

[8] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *In Proc. of Computer Vison and Pattern Recognition*, 2000.

[9] A. M. Elgammal. Learning to track: Conceptual manifold map for closed-form tracking. *In Proc. of Computer Vison and Pattern Recognition*, 2005.

[10] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(10), 1998.

[11] K. Hanna. Direct multi-resolustion estimation of ego-motion and structure from motion. *In Proc. of IEEE workshop on Visual Motion*, 1991.

[12] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[13] D. Heeger and A. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *Int. Journal of Computer Vision*, 7(2), 1992.

[14] A. Ilie and G. Welch. Ensuring color consistency across multiple cameras. *In Proc. of International Conference on Computer Vison*, 2005.

[15] H. Louguet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293, 1981.

[16] J. Matas, K. Zimmermann, T. Svoboda, and A. Hilton. Learning efficient linear predictors for motion estimation. *Proceedings of 5th Indian Conference on Computer Vision, Graphics and Image Processing*, 2006.

[17] H. Murase and S. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal on Computer Vision*, 14(1), 1995.

[18] M. Pollefeys, R. Koch, and L. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknow internal camera parameters. *In Proc. of International Conference on Computer Vison*, 1998.