EE392J Final Project Report

3-D Motion Estimation and Applications

Chuo-Ling Chang

chuoling@stanford.edu

I. Introduction

This project is composed of two main parts: 3-D motion estimation, and applications using the estimated 3-D motion. For 3-D motion estimation, 3-D shape reconstruction is first applied and followed by model-based 3-D motion estimation. Three applications are then demonstrated including object tracking and segmentation, 3-D stabilization, and motion-compensated interpolation.

II. 3-D Motion Estimation

The approach for 3-D motion estimation in this project has two stages: 3-D shape reconstruction and model-based 3-D motion estimation. The object shape in a video sequence is usually unknown. A structure from motion algorithm is first applied to reconstruct an approximate 3D shape from several views of the video sequence. With the approximate object shape, a model-based motion estimation algorithm can then be applied to estimate the 3-D motion of the object.

1. 3-D Shape Reconstruction

The object shape and motion in the video sequence is unknown in the beginning. To estimate the object shape, we need to solve a structure-from-motion problem, i.e. simultaneously estimate shape and motion.

Take several views with certain variety of viewing angles from the video sequence as reference views. By identifying feature correspondences in these reference views, the epipolar geometry and 3-D motion between the reference views can be estimated. With the estimated 3-D motion, the 3-D position of these feature points can then be computed. The procedure is shown in figure 1.

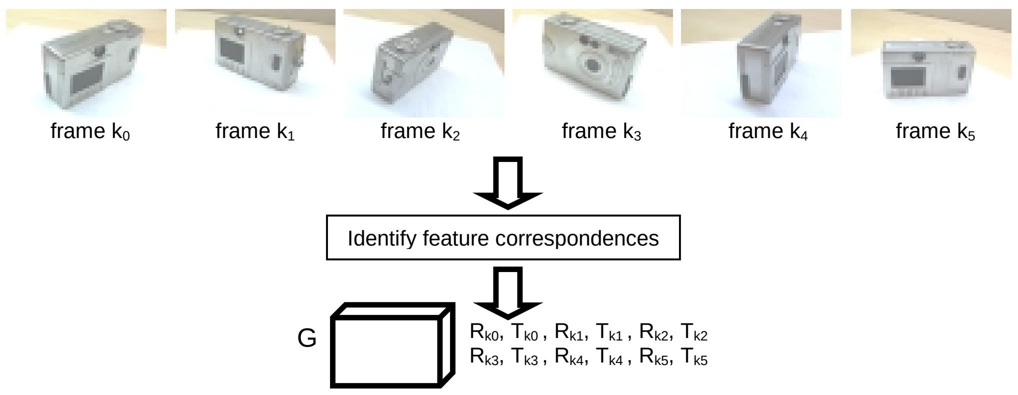

frame k_0
frame k_1
frame k_2
frame k_3
frame k_4
frame k_5
Identify feature correspondences
G
R_{k0}, T_{k0} , R_{k1}, T_{k1} , R_{k2}, T_{k2}
R_{k3}, T_{k3} , R_{k4}, T_{k4} , R_{k5}, T_{k5}

Figure 1

Take one of every 100 frames of *digital camera* sequence as the reference frames. By identifying the correspondences of the vertices on the camera, the 3-D position of these vertices (hence the approximate shape G) and object motions (R_k,T_k) (or the relative camera motions) can be estimated. The 3-D shape reconstruction part of project is done with the aid of off-the-shelf software [1].

2. Model-Based 3-D Motion Estimation

Since the approximate object shape is acquired in the 3-D shape reconstruction stage, motion estimation of the remaining frames can be solved using a model-based approach [2], i.e. estimate 3-D motion of an object with known shape.

The model-based approach is based on optical flow equations. Since the displacement in image domain due to 3-D motion of the object can be linearized as a linear function of object shape and 3-D motion [2], the optical flow equation can be rewritten for every pixel in the object:

$$\frac{\partial I_k}{\partial x} d_{x,k}(G,R,T) + \frac{\partial I_k}{\partial y} d_{y,k}(G,R,T) = I_{1,k} - I_{2,k}, \forall k$$

Spatial gradients and frame difference can be computed from the images and object shape G is known and fixed, hence the object motion R, T can be estimated using linear regression method.

The optical flow approach has several assumptions and limitations:

a. Constant intensity assumption:

The intensity of one surface point of the object should be the same in different views.

b. Linear signal model:

The intensity should be a linear function in image domain, i.e. the gradient should be constant.

c. Small motion:

Due to the linearization of the displacement and signal model, only small motion is valid.

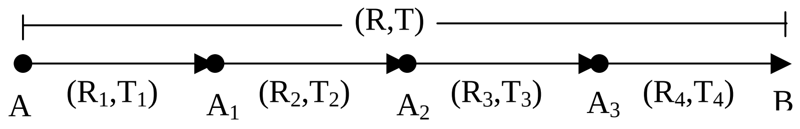
d. Small deviations of shape model from real object shape

To apply to a real video sequence, these assumptions usually don't hold. Therefore, the motion estimation is not stable and is sensitive to noise. Several auxiliary techniques can be applied to improve the robustness and accuracy of motion estimation [2].

(1). Iterative algorithm

For large motion, an iterative algorithm is needed. The motion from frame A to frame B, say (R,T), is to be estimated in the first iteration. However if (R,T) is too large for the small motion assumption, only a portion of it, say (R_1,T_1) can be estimated. A new frame A_1 can then be generated by warping A with the motion (R_1,T_1). Now frame A_1 and B can be used for the next iteration. The total motion (R,T) will be the concatenation of the results in each iteration (R_1,T_1), (R_2,T_2)…etc. This concept can be expressed as a simple 1-D analogy shown in figure 2.

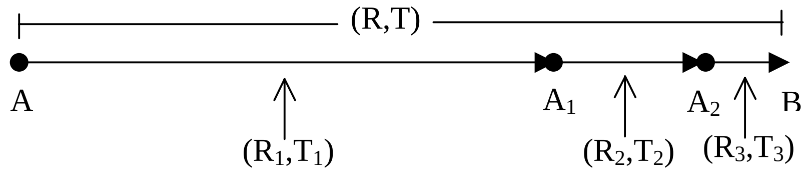
Figure 2



(2). Multi-scale algorithm (coarse-to-fine)

Small motion in a low-resolution image is equivalent to larger motion in full-resolution image. Therefore, with multi-scale algorithm a large motion can be estimated at the lower-resolution stage and then refined at the higher-resolution stage. The concept is shown in figure 3. Combining multi-scale with iterative approach can reduce the number of iterations and also prevent from being caught in local minimum.

Figure 3



(3). Quadratic signal model

The linear signal model assumption of optical flow equation can be improved to a quadratic signal model by changing the expression of gradient to average of gradients in two adjacent frames [2].

(4). Low-pass-filter

The signal is assumed to be linear (or quadratic), at least for a small range in the image. If there are noises or some high frequency components, this linear signal model doesn't hold and the gradient could drive the estimated motion to a wrong direction. Therefore, it is necessary to low-pass-filter the image before compute the spatial and temporal gradients.

(5). Gradient threshold

For areas in the object with constant intensity (zero gradient), there could be many motions which all satisfy the optical flow equations (aperture problem). These areas could bias the estimation for the whole object. It is usually good to use the pixels that have gradients higher than a certain threshold.

(6). Boundary elimination

Since the shape model is just an approximation of the object shape, it doesn't perfectly fit the actual object. Therefore, background could be identified as part of the object at object boundary. For motion estimation, these boundary pixels should be eliminated and only the interior pixels of the object should be considered.

(7). Outlier removal

The constant intensity assumption doesn't hold for the object with certain extent of reflection. This part of object, which largely violates the constant intensity assumption, should be rejected as outliers when establishing the optical flow equations. In addition, inaccurate model also introduce some outliers due to the inconsistence with shape model and 3-D motion.

To reject these outliers, a soft-threshold robust estimator is used as follows:

(i). Compute the least square solution for system of optical flow equations

$$\frac{\partial I_k}{\partial x} d_{x,k}(G,R,T) + \frac{\partial I_k}{\partial y} d_{y,k}(G,R,T) = I_{1,k} - I_{2,k}, \forall k$$

(ii). Compute the optical flow equation error for each pixel

$$e_k = \frac{\partial I_k}{\partial x} d_{x,k}(G,R,T) + \frac{\partial I_k}{\partial y} d_{y,k}(G,R,T) - (I_{1,k} - I_{2,k}), \forall k$$

(iii). Compute the weight for each pixel as a function of optical equation error and satisfies

$$w_k = f(e_k), \forall k$$
$$e_k < \hat{e}_k \Leftrightarrow w_k > \hat{w}_k$$

(iv). Compute the weighted least square solution

$$w_k \left(\frac{\partial I_k}{\partial x} d_{x,k}(G,R,T) + \frac{\partial I_k}{\partial y} d_{y,k}(G,R,T)\right) = w_k (I_{1,k} - I_{2,k}), \forall k$$

The pixels with higher optical flow equation errors will be treated as outliers and put less weight. Therefore, their impact for the estimation of entire object will be reduced. In this way, the threshold doesn't need to be explicitly determined and the outliers can still be naturally rejected. The algorithm can also be applied in an iterative manner.

With all these auxiliary techniques, 3-D motion estimation is more robust and accurate. An example of motion estimation result is shown in figure 4. The image at top is warped from previous frame using estimated motion between previous and current frame, i.e. a synthesized current frame with texture from previous frame. The image at bottom is the difference between synthesized current frame and the original current frame. Zero difference is shifted to have color gray in the image. The complete video sequence of synthesized and difference images are attached as *dc_motionEst_syn.avi* and *dc_motionEst_err,avi*.

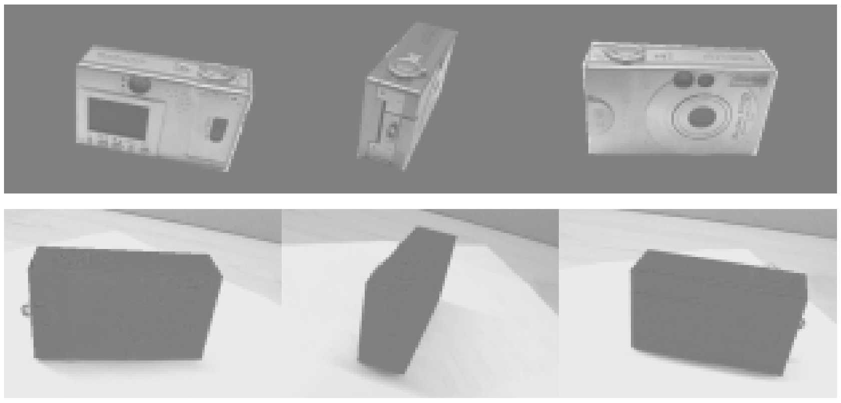


Figure 4. Results for motion estimation (top) synthesized frame (bottom) difference between synthesized and original frame

III. Applications

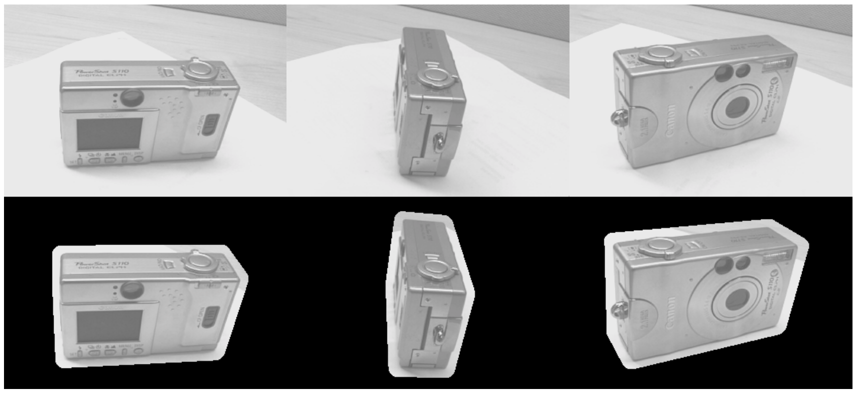
With the estimated object shape and motion, different applications can be applied for the original sequence. In this project, three examples are investigated: object tracking and segmentation, 3-D stabilization, and motion compensated interpolation.

1. Object Tracking and Segmentation

From the estimated 3-D shape and motion, 2D projection of the object on the image is known. Object through the video sequence can be tracked and background can be segmented.

If only the foreground object is of interest, for video communication it is useful to spend more bandwidth on the foreground object. Furthermore, for MPEG-4 video object coding, a complete description the object, i.e. 3-D shape, motion, and texture, could be obtained in this manner and further video manipulation would be possible.

An example of tracking the object and segmentation is shown on figure 5. The complete video sequence is attached as *dc_tracked.avi*.



Frame 50 Frame 150 Frame 250

Figure 5. Object tracking and segmentation (top) original frame (bottom) foreground object segmented

2. 3-D Stabilization

From object tracking mentioned above, a conventional 2-D video stabilization is already achieved by shifting the object image to the center of each frame. In addition, a 3-D stabilization, i.e. stabilize the 3-D object motion (or relative 3-D camera motion) is also possible.

Since motion at every frame in the video sequence and the object shape is already known, new views can be synthesized by referencing texture from existing views. 3-D stabilization is a simple demonstration of synthesizing new views.

Stabilization of translation is achieved by setting a new object motion path with zero translation, i.e. the 3-D position of the object center is fixed. This new sequence is synthesized by referencing texture from the corresponding frame in the original sequence.

Stabilization of rotation can also be applied to remove the jitter in rotational motion. The concept is shown in figure 6. An up-sampled new object motion path is first designed by inserting new frames to achieve constant rotation between frames. The rotation axis and rotation angles are interpolated from existing frames. Then the new object motion path is down-sampled to the original rate but now has smoother rotations.

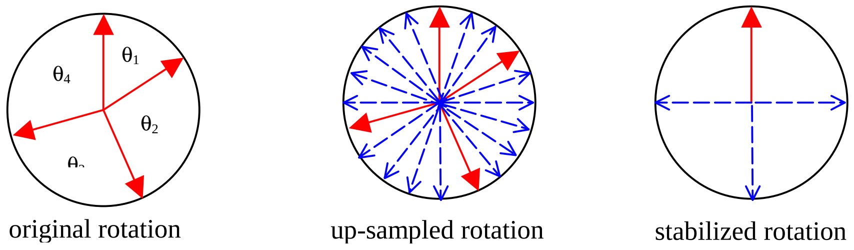


Figure 6. Stabilization for rotation

Several frames of a stabilized sequence with replaced background are shown in figure 7. The complete video sequence is attached as *dc_stabilized.avi*.

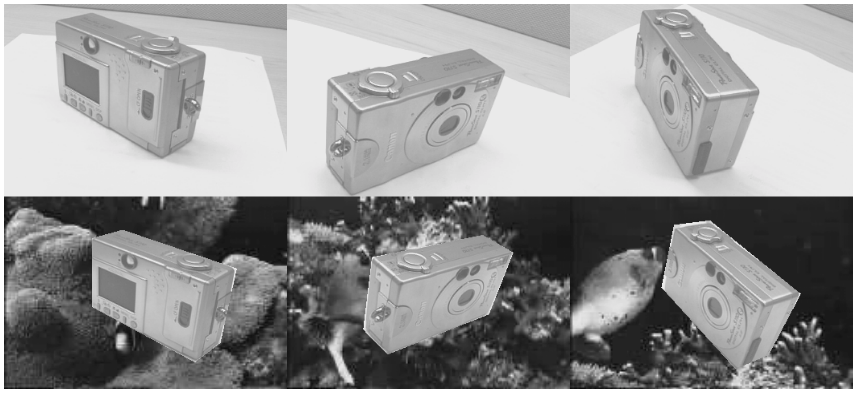

Frame 110 Frame 220 Frame 330

Figure 7. 3-D stabilization (top) original frame (bottom) stabilized frame with background replaced

3. Motion-Compensated Interpolation

The main goal of this part is to down-sample the frame-rate of original sequence for storage or transmission and restore the original sequence by motion-compensated interpolation for display. In other words, represent the original sequence with lower temporal resolution and minimize the loss in quality.

The first experiment is using 6 frames to represent the 500-frame *digital camera* sequence. Therefore, every 100 frames are interpolated from two reference frames, i.e. the leftmost and rightmost frame in each 100-frame segment.

A simple way to interpolate frame *I* from frame *L* and frame *R* is as follow:

1. Compute 3-D positions of the object points shown in frame *I* by known object shape and motion
2. Project these 3-D object points to frame *L* and frame *R*
3. Interpolate within frame *L* and *R* for the intensity of the projected points
4. Compute the weighted average intensity of the projected point in frame *L* and frame *R*. The weight is inversely proportional to the distance between frame *I* and frame *L*, *R*

Basically, one interpolated frame I_L is generated by motion compensation from frame *L* and the other interpolated frame I_R is generated from frame *R*. The final interpolated frame I_W is obtained by taking weighted average of I_L and I_R. For example, to interpolate frame 30 from frame 1 and frame 100, I_1 is weighted by 0.7 and I_{100} is weighted by 0.3. The result of this algorithm is shown on figure 7(a).

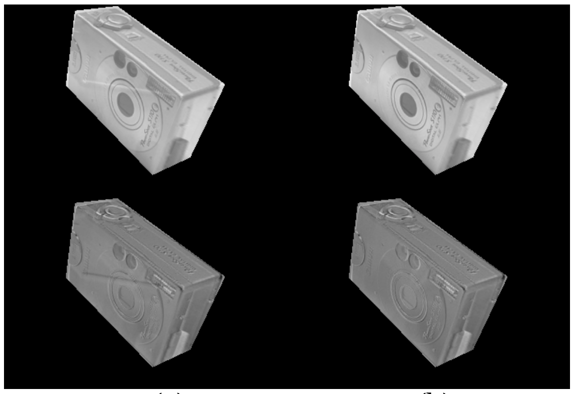

(a) (b)

Figure 7. Frame 320 interpolated from frame 300 and frame 400. (top) interpolated frame (bottom) difference between interpolated and original frame. (a) scheme 1 (b) scheme 2

An obvious artifact is observed due to the invisibility in one of the reference frame. For example, face *M* of the object could be visible in frame 1 and frame 30, but not in frame 100. Therefore, to interpolate a point on face *M* in frame 30, only frame 1 needs to be considered. The intensity from frame 100 is actually from another face covering face *M* and hence is not a correct value. The result of modified algorithm by considering visibility is shown on figure 7(b) where the artifacts on figure 7(a) are removed.

The weight in the previous cases is simply inversely proportional to the distance between frames and is not optimized. An optimized weight for combining I_L and I_R can be computed in the sense of minimizing the mean squared error between I_W and I. In other words, by solving the least square problem below.

$$w \cdot I_{L,k} + (1-w) \cdot I_{R,k} = I$$

This equation is applied for the pixels of the object both visible in I_L and I_R. For those invisible in one of the reference frames, weight is set either to 1 or 0.

So far the weight is the same for the entire object (except the invisible part as mentioned above). However, different triangle meshes of the object can have different weights by estimating the weight separately for each triangle mesh. This is reasonable since the weight should depend on the surface orientation of each triangle meshes and viewing angles of different views.

In addition, an extra term accounts for intensity offset can also be added to the weight estimation problem.

$$w \cdot I_{L,k} + (1-w) \cdot I_{R,k} + offset = I$$

The weighted average without offset can only result in a value between two reference intensities. The offset added in the equation provides flexibility to interpolate a bright image from two dark images or vice versa.

To summarize, 5 schemes are proposed here for 3-D motion compensated interpolation:

1. weight inversely proportional to distance, visibility not considered
2. weight inversely proportional to distance, visibility considered
3. weight estimated for the entire object, visibility considered
4. weight estimated for each triangle, visibility considered
5. weight and offset estimated for each triangle, visibility considered

Results for scheme 1,2 are already shown in figure 7. Results for scheme 3,4,5 are shown in figure 8.

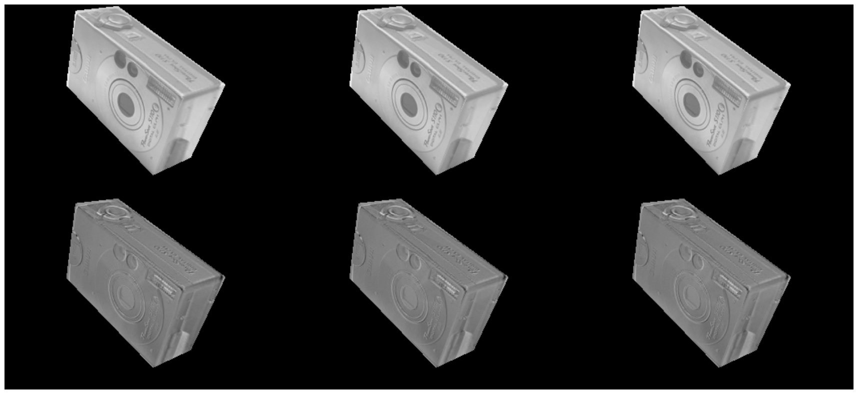


Figure 8. Frame 320 interpolated from frame 300 and frame 400. (top) interpolated frame (bottom) difference between interpolated and original frame. From left to right: scheme 3, 4,5

Table 1 lists PSNR between the interpolated and original frames when interpolating the 500-frame *digital camera* sequence with 6 reference frames using different schemes.

Table 1. Interpolating the 500-frame *digital camera* sequence using 6 reference frames

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5
PSNR	25.20 dB	25.10 dB	25.36 dB	25.51 dB	26.16 dB

The artifacts in scheme 1 are removed in other schemes. However, the PSNR for scheme 2 is not better than scheme 1. A possible reason is that even scheme 1 is averaging wrong intensities for the invisible part, the averaging operation still provides some noise reduction effects. However, scheme 2 is still much more visually pleasing than scheme 1.

The PSNR for scheme 3,4,5 are higher than scheme 1,2 since the weights are optimized. Scheme 4 and scheme 5 might have discontinuities along the triangle mesh boundaries as shown in figure 9 since the weights (and offsets) are estimated separately for each triangle mesh. However, the PSNR in scheme 4,5 are still higher than scheme 3.

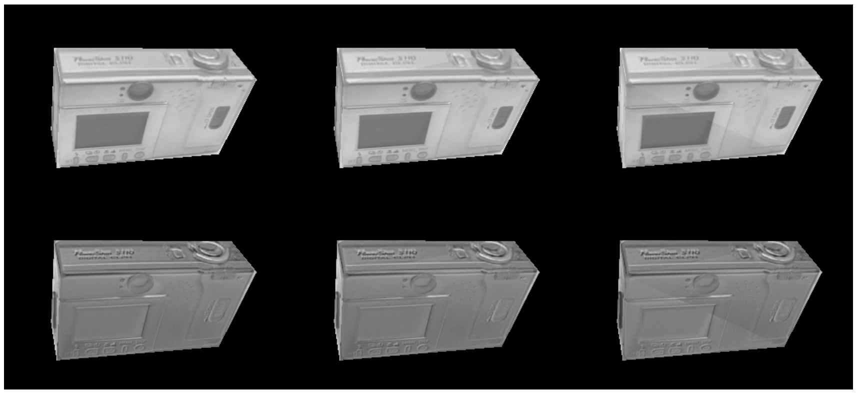
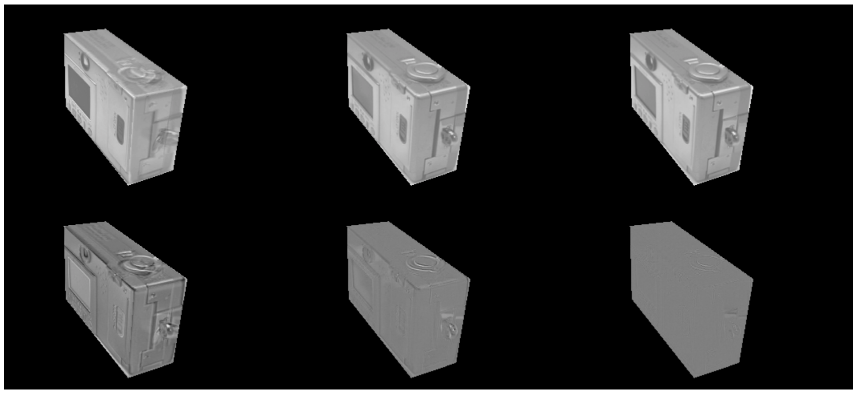


Figure 9. Frame 470 interpolated from frame 400 and frame 500. (top) interpolated frame (bottom) difference between interpolated and original frame. From left to right: scheme 3, 4,5

The number of reference frames can be increased to improve quality of motion compensated interpolation. Results by using different number of reference frames with scheme 3 are shown in figure 10(a) and 10(b). The complete video sequence is attached as *dc500_mci_6_21_81.avi*.



Frame 130

Figure 10(a). Motion compensated interpolation for the 500-frame sequence using different number of reference frames. For left to right: 6, 21, 81 reference frames.

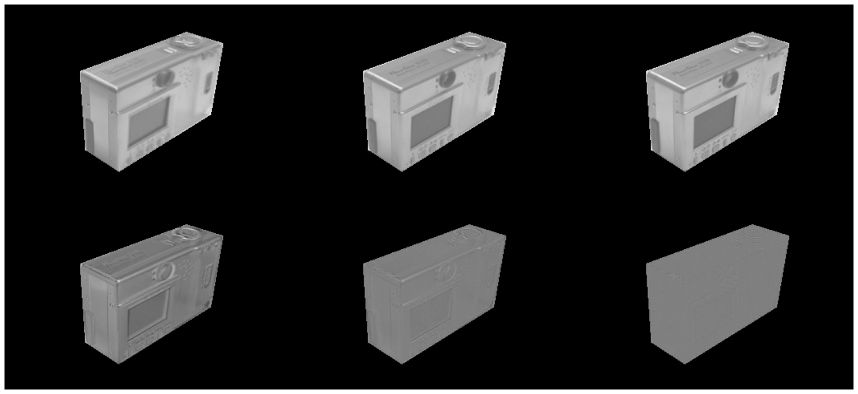


Figure 10(b). Motion compensated interpolation for the 500-frame sequence using different number of reference frames. For left to right: 6, 21, 81 reference frames.

With different schemes, different amount of overhead for the weights and offsets are also introduced. For a fair comparison, rate-distortion curves are plotted including the rate due to overhead as shown in figure 11. The rate for the uncompressed 352x240 frame in YUV 4:2:0 format is 1013760 bits/frame. The weights range from 0~1 and are quantized to 4 bits per weight. The offsets are also quantized to 5x10^{-5} of the intensity range (0~255) and can be represented by 12 bits. There are 10 triangle meshes for the *digital camera* sequence. One set of motion parameters are quantized to 114 bits.

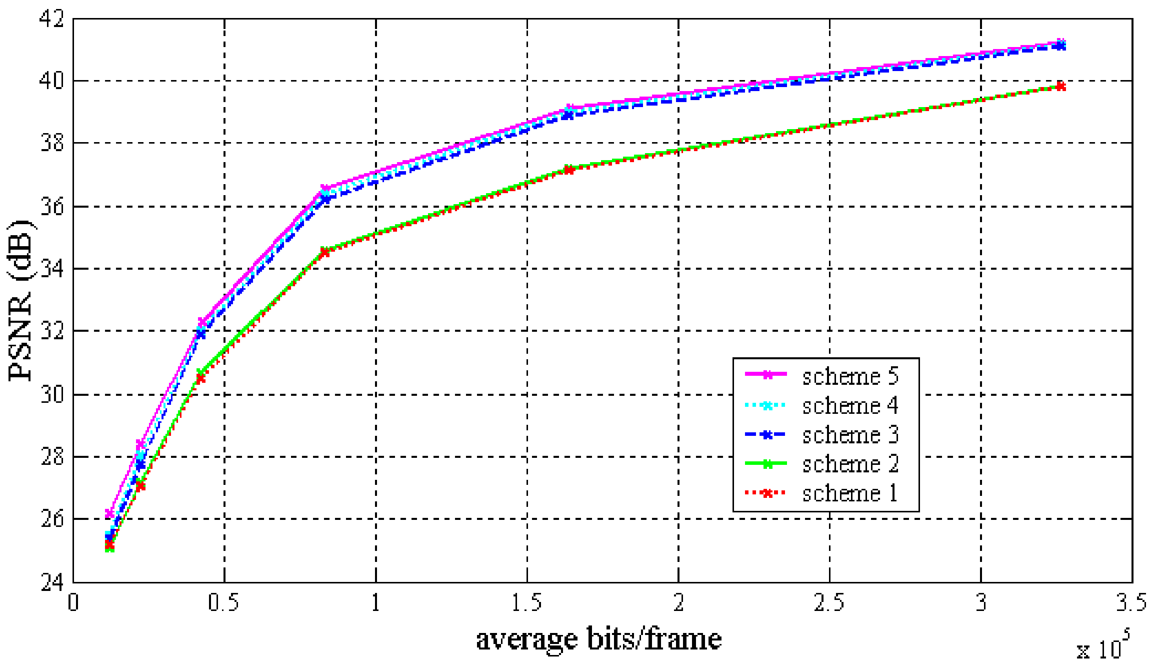


Figure 11. Motion compensated interpolation for the 500-frame *digital camera* sequence with different schemes. 6, 11, 21, 41, 81, 161 reference frames are used. Source rate is 1013760 bits/frame

From the R-D curve, scheme 5 could improve 2 dB in PSNR compared to scheme 1 for certain rage of rate. Scheme 3,4,5 have similar performance although scheme 5 is always the best. With 81 reference frames (compression ratio about 6), 39 dB can already be achieved. Even for 21 reference frames (compression ratio about 25), 32 dB can still be achieved.

To conclude, different schemes should be adopted for different applications. For applications without residual coding, scheme 3 is a good choice since it doesn't have the discontinuity effects. For applications with residual coding, scheme 5 is a better choice since it minimizes the energy of residual and residual coding can compensate the discontinuities. However, the impact of rate from overhead should be considered, especially for low-rate case where overhead could take a significant amount of the entire rate.

Another example of motion compensated interpolation for *car* sequence is shown in figure 12. The complete video sequence is attached as *car150_mci_2_5_17.avi*.

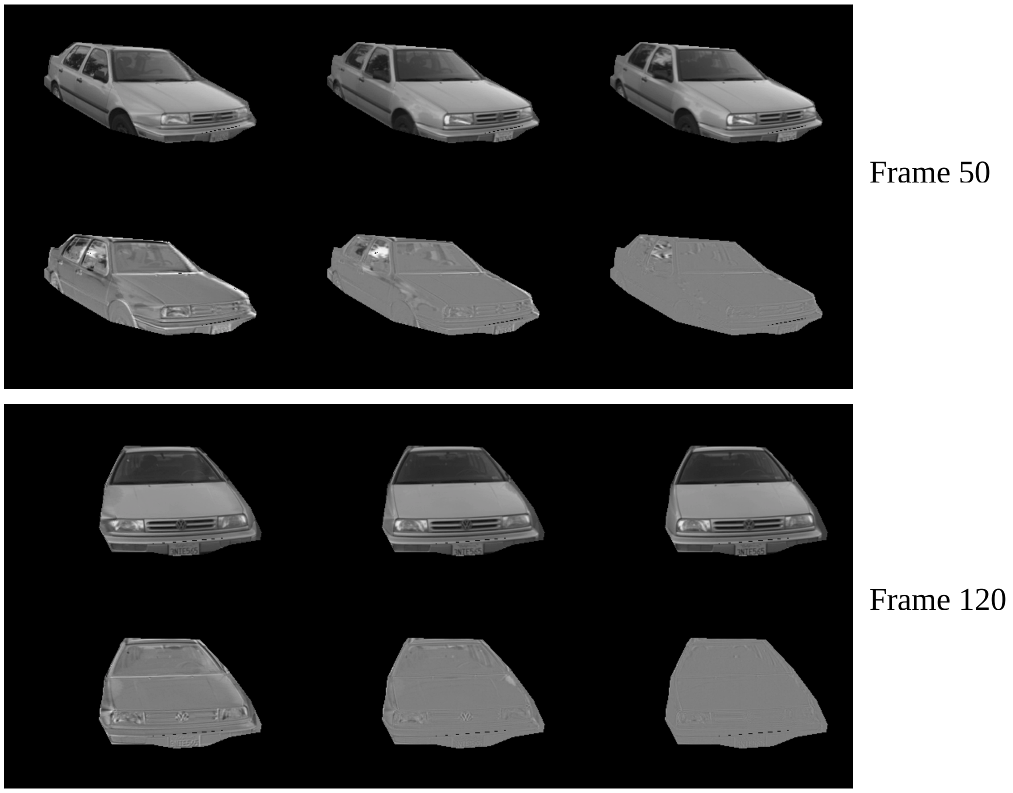


Figure 12. Motion compensated interpolation of the 150-frame sequence using different number of reference frames. For left to right: 2, 5, 17 reference frames.

IV Conclusion

In this project, estimation of 3-D motion as well as 3-D shape for an object in video sequence is achieved. These shape and motion information can be used for various video-processing applications in order to represent the video sequence more efficiently and variously.

Reference:

[1] PhotoModeler demo-version: www.photomodeler.com

[2] Chapter 7, "Video Processing and Communications" by Wang, Ostermann, and Zhang

Attached video file:

1. *dc_motionEst_syn.avi* 2. *dc_motionEst_err.avi* 3. *dc_tracked.avi*

4. *dc_stabilized.avi* 5. *dc500_mci_6_21_81.avi* 6. *car150_mci_2_5_17.avi*

Attached source code:

1. motionEst.c++ 2 .objectStabilized.c++ 3. MCInterpolate.c++

(These are main programs only, c++ classes are not provided since too many are involved)