Video Stereolization: Combining Motion Analysis with User Interaction

Miao Liao, Jizhou Gao, Ruigang Yang, Minglun Gong

Abstract

We present a semi-automatic system that converts conventional videos into stereoscopic videos by *combining motion analysis with user interaction*, aiming to transfer as much as possible labeling work from the user to the computer. In addition to the widely-used structure from motion (SFM) techniques, we develop two new methods that analyze the optical flow to provide additional qualitative depth constraints. They remove the camera movement restriction imposed by SFM so that general motions can be used in scene depth estimation – the central problem in mono-to-stereo conversion. With these algorithms, the user's labeling task is significantly simplified. We further developed a quadratic programming approach to incorporate both quantitative depth and qualitative depth (such as these from user scribbling) to recover dense depth maps for all frames, from which stereoscopic view can be synthesized. In addition to visual results, we present user study results showing that our approach is more intuitive and less labor intensive, while producing 3D effect comparable to that from current state-of-the-art interactive algorithms.

Index Terms

Semi-automatic 2D-3D Conversion, Stereo/3D Video/Movie, Motion Analysis, user labeling.

I. Introduction

Given the recent success of 3D movies from Hollywood, there is a renewed public interest in stereoscopic videos. While new contents can be captured by stereoscopic cameras or CG contents can be re-rendered in the stereo mode, how to convert the large collection of legacy monocular movies and videos to stereo remains an open problem.

Miao Liao, Jizhou Gao and Ruigang Yang are with the Department of Computer Science, University of Kentucky.

Minglun Gong is with the Department of Computer Science, the Memorial University of Newfoundland.

While there exist commercial softwares (e.g. Tri-Def $DDD^{\copyright}$) that automatically turn images into stereo ones, there still exists large space to improve the 3D effect. Understandably, converting a single image or a monocular video into stereo requires knowing the depth information for every pixel in the scene. Recovering 3D range from a single 2D image is an ill-posed problem, a universal approach that requires no user interaction will be extremely difficult, if not impossible, to achieve. The goal of our research is to reduce the amount of user interaction as much as possible by *taking advantage of the movement in the scene*.

While structure-from-motion techniques (SFM)[1] is an obvious choice and has been explored in several prior papers [2], [3], [4], we introduce two more methods to estimate scene depth. The first is based on the observation that in many *follow* shots, the foreground remains in the center of the screen and changes little while the background changes frequently. The second is based on the observation that as an object moves towards or far away from the camera, its image size varies accordingly. By analyzing the optical flow [5], [6], [7] between frames, we can perform automatic foreground segmentation and determine the relative depth changes of moving objects.

All the analysis based on movement, caused by object motion, camera motion or both, leads to constraints on the scene depth. For certain videos, these constraints are enough to automatically obtain a scene depth map from which stereo view synthesis is possible. For more complex videos or videos with little or no movement, user interaction is only needed in places where depth cues cannot be extracted automatically. The user interaction only requires sparse scribbles indicating relative depth (inequality and equality). *Combining the prior analysis through motion and the depth ordering by user interaction*, our system can generate dense depth maps that produce visually plausible 3D stereo images for a variety of scenes and shots as shown in Figure 1.

Our paper has the following contributions:

- We propose two novel techniques that automatically estimate the 3D information from video sequences. Unlike SFM that requires non-axis camera movement (e.g., dolly, crane), our techniques try to infer depth information under arbitrary camera/object movement, such as camera pan or zoom, which are frequently used in both everyday video and professional shots.
- We provide a user-friendly interface that requires users to label depth relationship other than depth value on the images. Our UI design benefits from the already defined 3D cues by the pre-processing of movement, providing the users with a more intuitive and less labor

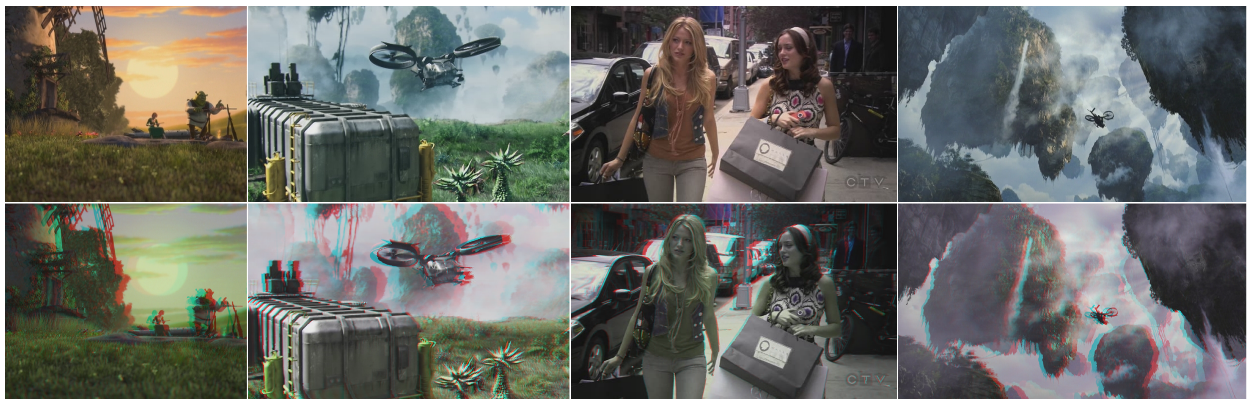


Fig. 1. The snapshots of 4 videos that have been converted to stereo views. The 1^{st} row shows the original frames, and the 2^{nd} shows the corresponding stereoscopic images.

intensive UI environment. In the worst case that none of the 3D cues can be inferred in the pre-processing step, our labeling can still simulate the direct depth labeling under the same amount of manual work.

- We formulate the sparse to dense depth propagation as a quadratic programming problem, that could elegantly integrate both relative (such as layer orders) and absolute (such as that from SFM) depth constraints.

The user interaction perspective of our system is mostly related to interactive stereo conversion methods (e.g., [8], [9]). Because we analyze the movement to provide as many 3D cues as possible, useability study showed that our system is more intuitive and takes less time to produce stereo videos of comparable quality.

II. Related Work

Mono video to stereo conversion can boil down the problem of inferring the relative or absolute depth for the underlying scene. Some recent work [10], [11], [12] proposed to use machine learning approaches to recover the 3D structure from a single image. While they have achieved some excellent results given this very difficult problem, the outcome depends largely on the training data set. Given the complexity and varieties of everyday videos, it is not clear if this type of data-driven approach can be made tractable for videos.

Automatic stereoscopic extraction can be achieved by generating dense depth map for every frame in a monocular video of static scenes [2]. Structure from motion algorithm is employed to compute the 3D positions of the tracked feature points as well as the camera poses. With the calculated camera poses, multiple view stereo is applied on each frame to produce the dense depth maps [13]. [3] and [4] can deal with dynamic scenes by segmenting rigid objects into layers and SFM [1] is applied on each layer respectively to reconstruct 3D structure. [14], [15] and [16] avoid the generation of dense depth map by synthesizing the other view from the other frames in the input video. Their methods involve less computational resources. However, they are designed to handle static scenes, and certain assumption has to be made on the camera paths so there would be an image in the sequence that can be borrowed for stereo-pair view synthesis. Finally it should be noted that SFM [1] can only deal with certain camera motions, such as dolly and crane shots [17]. For camera movement that does not change the center of projection, such as these commonly used pan/tilt/zoom shots [17], SFM [1] will fail. In our approach, we further analyze optical flow [5], [6], [7], which is defined for any type of moment, to infer depth. It compliments SFM to broaden the types of movement our system can handle.

In general, full automatic methods do not produce satisfactory results in complex scenes exhibiting both complex camera movement and non-rigid object motion. Some semi-automatic approaches resort to the assistance of user interaction. [18] developed a software (DeepSee Studio) that enables the lasso operation on an image to segment objects from the background. [19] requires the users to specify depth to some sparse points in certain key frames, followed by the spatial and temporal depth propagation. The propagation is implemented by classification, where the classifier is trained by the user input information. The most recent work [9] is similar to [19]. They also ask the users to assign some absolute depth value, which is color encoded, to a few key frames by scribbling. The user scribbling is used to train a classifier that classifies the rest pixels in the image sequence. Only those pixels with high confidence are assigned a depth value by the classifier. The depth is propagated to the entire video by a similar approach of colorization [20] that encourages same depth values for neighboring pixels, except for those edges separated ones. Instead of using a linear least square formulation as in [20], [9], our depth propagation formulation is based on quadratic programming since it allows both equality and non-equality assignment [21] (expressing the fact that one layer is in front of the other).

All these user-scribble based approaches rely solely on the user to provide vital depth information. The depth cues that can be automatically estimated from motion are ignored. In our system, we combine motion analysis with user interactions so that we can get the best of both

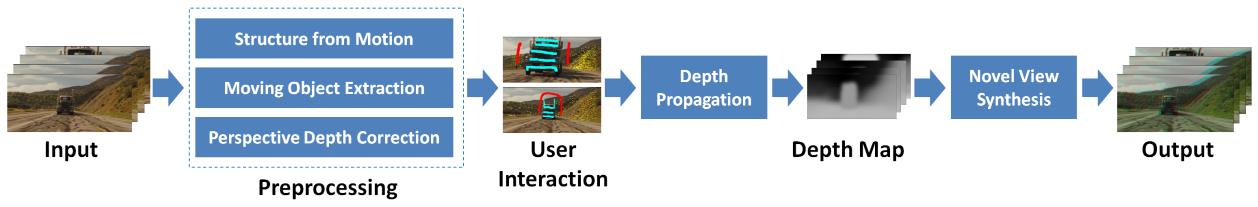


Fig. 2. The pipeline of our system.

worlds: motion analysis can provide depth constraints automatically, reducing the amount of user input required while user interaction can guarantee the final quality of the synthesized view.

III. System

Figure 2 shows the pipelines of our system. In the pre-processing step, the input image sequence is first passed through three individual automatic modules: *structure-from-motion* (SFM [1]), *moving object segmentation* (MOS), and *perspective depth correction* (PDC). The SFM algorithm is applied to the input image sequence with dominant rigidly moving objects to recover a sparse set of 3D points. The MOS module is used to automatically segment the foreground, it is particularly effective in a *follow* shot in which the foreground is relatively static and the background is rapidly changing. Finally, the PDC module inspects the size change of an object's image to estimate relative depth changes between frames. After automatic processing, the users are presented with images showing area with known depth (from SFM and MOS). If there are still undefined regions, the users need to label them in some key frames by simple scribbling. The user's input as well as all the automatically calculated depth cues will be integrated in a quadratic programming framework to generate dense depth maps for all frames.

Finally the novel view is generated via shifting every pixel horizontally by a certain amount base on the depth maps, simulating the perspective from the other eye. Since in our cases, the baseline between the synthesized view and the input view is small, we use a simple technique to deal with the gaps in the synthesized view due to disocclusion. We simply fill the uncolored region with neighboring pixels of larger depth values. We found this method worked well in practice. View synthesis is not the emphasis of our paper therefore we will not discuss it further.

A. Automatic Pre-Processing

1) Structure from Motion and Optical Flow Estimation: The structure from motion algorithm [1] is conventionally employed to recover the 3D positions of the feature points and the positions of the camera in a sequence of static scene. After the positions of the camera track are computed, a dense depth map for each frame can be obtained by multiple view stereo [2].

Our system does not make any assumption on the input image sequences. The scene may be captured by a camera with fixed viewpoint or may contain non-rigid motion and hence, the SFM algorithm does not always work. However, in most of the moving camera shots, SFM can automatically track sparse features on the static background, leaving only the moving foreground objects to be labeled by the users. Both spacial and temporal depth variations for these feature points can be accurately computed in our approach. By contrast, Guttmann et al.'s approach [9] requires users to careful label the depths at different potions of the background, as well as how the depth changes across different frames, which can be both cumbersome and imprecise for scenes with complex background.

The depth information extract by SFM are incorporated into the final solution through equality constraint. Basically, for each feature point $\mathbf{p}$ in frame t with assigned depth $D^t_{SFM}(\mathbf{p})$, we add the following constraint to the final linear equation:

$$d^t(\mathbf{p}) = D^t_{SFM}(\mathbf{p}) \tag{1}$$

where $d^t(\mathbf{p})$ is the depth value of pixel $\mathbf{p}$ at frame t.

We also estimate the optical flow between adjacent frames. Similar to [7], the optical flow is estimated by solving an optimization problem defined on region-tree by dynamic programming. The region-trees of over-segmented regions are constructed from the input images. The optical flow will be used in both perspective depth correction and moving object extraction.

2) Moving Object Extraction: In follow shots, where the camera tracks the moving foreground objects, it is often than not that the tracked (foreground) objects are visually consistent in the video sequence, whereas the appearance of the backdrops varies. Based on this observation, we can achieve the automatic segmentation by counting the statistical coherence of appearance variation at each pixel, followed by making a per-pixel decision whether it is foreground or not.

Given a sequence of video frames $\{I_1, I_2, \cdots I_m\}$, our goal is to segment the foreground object

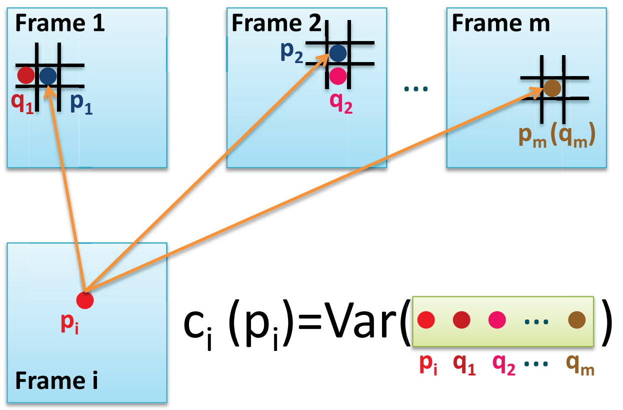


Fig. 3. An illustration to compute the color variance $c_i(\mathbf{p}_i)$ at pixel $\mathbf{p}_i$ in frame I_i. Suppose pixel $\mathbf{p}_i$ finds its correspondences $\mathbf{p}_1$ in frame I_1, $\mathbf{p}_2$ in frame I_2, and up to $\mathbf{p}_m$ in frame I_m as shown by the orange arrows, and $\mathbf{p}_1$'s neighboring pixel $\mathbf{q}_1$ has the smallest color difference with $\mathbf{p}_i$ and so on and so forth; then the cost $c_i(\mathbf{p}_i)$ is the variance of the color values from $\mathbf{p}_i, \mathbf{q}_1, \mathbf{q}_2 \cdots \mathbf{q}_m$.

$\mathcal{O} = \bigcap_{i=1}^{m} I_i$. However, segmenting all frames simultaneously needs to solve a large number of unknowns. Instead, we focus on segmenting one frame at each time, but with additional information propagated from other frames. We first establish dense correspondence maps between any pair of frames I_i and I_j by traversing through a connected path from I_i to I_j related by optical flows.

In the next step, we propose to compute a color variance map $\mathcal{C}_i$ for frame I_i, where each element $c_i(\mathbf{p}_i)$ corresponds to the color variance of the series consisting of pixel $\mathbf{p}_i$ in frame I_i and its correspondent pixels in other frames. Suppose we find $\mathbf{p}_i$'s correspondences $\mathbf{p}_1$ in frame I_1, $\mathbf{p}_2$ in frame I_2, and up to $\mathbf{p}_m$ in frame I_m; we can simply compute $c_i(\mathbf{p}_i)$ as the color variance of the series:

$$c_i(\mathbf{p}_i) = \text{Var}\left(\theta_i(\mathbf{p}_i), \theta_1(\mathbf{p}_1), \theta_2(\mathbf{p}_2), \cdots \theta_m(\mathbf{p}_m)\right), \tag{2}$$

where $\theta_j(\mathbf{p}_j)_{1 \le j \le m}$ is the color of the pixel $\mathbf{p}_j$ in frame I_j. However, to be more robust to image noise and alignment offsets, we use the approximate matching strategy to compute $\mathcal{C}_i$, that is instead of selecting $\mathbf{p}_i$'s exact matched pixel $\mathbf{p}_j$ in frame I_j, we search the $\mathbf{p}_j$'s neighboring pixel $\mathbf{q}_j$, where color $\theta_j(\mathbf{q}_j)$ is closest to the reference color $\theta_i(\mathbf{p}_i)$, and efficiently compute the color variance in a greedy manner as:

$$\begin{aligned} \mathbf{q}_j &= \arg\min_{\mathbf{q}_j \in N(\mathbf{p}_j)} \left|\theta_j(\mathbf{q}_j) - \theta_i(\mathbf{p}_i)\right|, \quad j \neq i, & (3) \\ c_i(\mathbf{p}_i) &= \mathrm{Var}\left(\theta_i(\mathbf{p}_i), \theta_1(\mathbf{q}_1), \theta_2(\mathbf{q}_2), \cdots \theta_m(\mathbf{q}_m)\right), & (4) \end{aligned}$$

where $N(\mathbf{p}_j)$ represents a search window centered at pixel $\mathbf{p}_j$ in frame I_j with the size of w, and in our experiments, we typically set $w = 7 \times 7$. This procedure is illustrated in Figure 3.

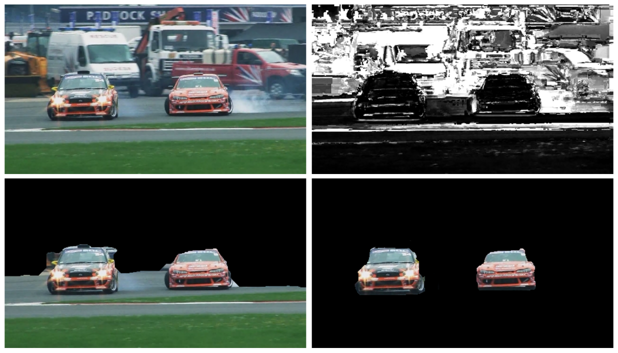


Fig. 4. An intermediate result of the moving object extraction. Given one frame I_i (upper left image), we can first compute its variance map $\mathcal{C}_i$ (upper right image). Based on $\mathcal{C}_i$, we can mask out the high-variance (bright in the variance map) parts using graph-cut algorithm to get the initial segmentation (lower left image). Furthermore, a user can provide only a few color seeds, e.g., the green color in the grass and gray in the ground, to remove the unexpected extracted parts and get the final segmentation result (lower right image).

Finally, we formulate the variance map $\mathcal{C}_i$ as an additional data term V in the binary labeling Markov Random Field (MRF) energy minimization framework for image segmentation. V is defined in equation 5:

$$V(\mathbf{p}) = \begin{cases} \exp(-c_i(\mathbf{p})/\alpha_f) & \text{if } \mathbf{p} \in \text{Background}, \\ 1 - \exp(-c_i(\mathbf{p})/\alpha_b) & \text{if } \mathbf{p} \in \text{Foreground}, \end{cases} \tag{5}$$

where $\alpha_f = 7 \times v_f$, $\alpha_b = 0.5 \times v_b$; v_f is set to the 0.1 percentile in $\mathcal{C}_i$ and v_b is set to the 99th percentile in $\mathcal{C}_i$. The explanation of equation 5 is that if the color variance $c_i(\mathbf{p})$ is small, which implies that the pixel $\mathbf{p}$ in frame I_i finds visually consistent matches in the rest of frames, the pixel should be considered as foreground, i.e., the cost of assigning it to background is high. On the contrary, a large color variance $c_i(\mathbf{p})$ indicates pixel $\mathbf{p}$ should be labeled as background. Combining the other widely used visual cues, such as color and edge information [22], [23], we adopt the graph-cut approach to solve the minimization problem using the min-cut/max-flow algorithm [22] and eventually extract the moving foreground objects from the video sequence. Figure 4 shows the segmentation result for one frame in a follow shot.

By applying the moving object extraction, we are not seeking accurate foreground extraction with sharp boundary. A rough segmentation is enough in our application, since we will use erosion and dilation to define the foreground and background regions and their depth relations (figure 8, middle image in the 1^{st} row).

When the foreground and background are defined, they will be treated as two different depth layers. However, not the entire background region is marked as further than the foreground, but only the areas that is above and to the both sides of the foreground object, which we call defined background. This is under the observation that most follow shots are following objects on the ground, so that the ground below the object is actually closer and should not be regarded as behind the object. Let $\mathbf{p}$ and $\mathbf{q}$ be arbitrary pixels that lie on the foreground and defined background respectively. The following constraints are plugged into the final formulation:

$$d(\mathbf{p}) - d(\mathbf{q}) \geq \Delta d \tag{6}$$

where Δd is a predefined threshold that denotes the minimum depth difference between two layers.

3) Perspective Depth Correction: Another useful vision cue is that, under perspective projection with fixed focal length, a rigid object gets bigger when it moves closer to the camera and appears smaller when it moves away. As shown in figure 5, the size change of the object shows up on the estimated optical flow as local flow expansion or shrinking. Here we use this cue to automatically infer depth changes of moving objects. It is noteworthy that this observation is based on the assumption of local rigidity, i.e., the local structure of the objects in the scene undergoes rigid motion in short time intervals even though deformable objects are allowed in the video.

The optical flow of an object does not show up as pure expansion or shrinking unless it is at the center of the image and is moving along the optical axis of the camera. In general cases, the optical flow of the object is a combination of the motion along the depth direction and the one parallel to the camera plane. To extract the portion of the flow caused by depth change,

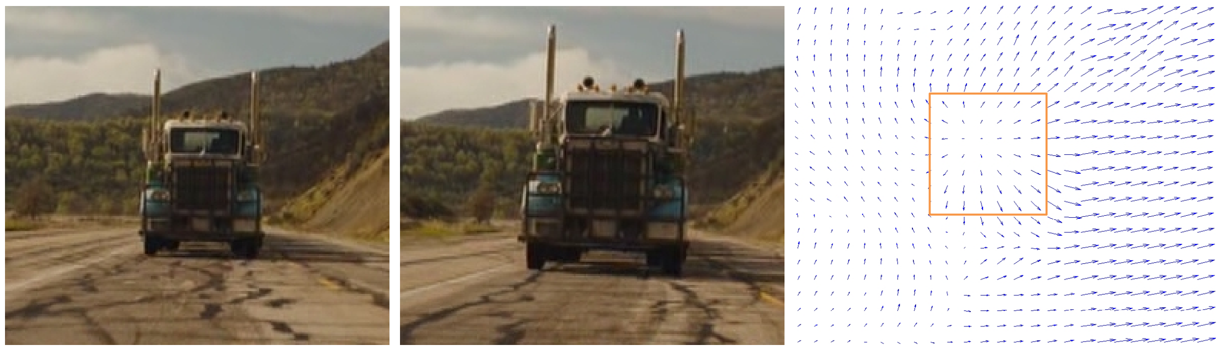


Fig. 5. The illustration of the expanding optical flow.

here we first apply Helmholtz-Hodge decomposition to decompose the 2D optical flow into a divergence-free vector field and a divergence field. The divergence-free field is ignored and the divergence field undergos the following test to see if it is caused by the depth change of a rigid object.

The test is performed based on a local structure consisting of a pixel and its 4 spatial neighbors, which we refer as a unit structure. The 5 pixels in the unit structure are traced into the next frame by the pre-computed optical flow. Figure 6 shows four of the possible shapes the unit structure could take in the next frame under the rigidity assumption. 1) If the unit structure is scaled evenly in all the directions, it suggests that the corresponding 3D points of the unit structure pixels are at the same depth and move toward (or away from) the camera. 2) If the unit structure is evenly scaled and rotated, the motion of the corresponding 3D structure is exactly the same as in the first case except that there is also in-image-plane rotation. 3) If the unit structure is unevenly scaled, it could result from the depth variation among the structure's points, causing different points moving at different observed velocities. 4) If the unit structure is skewed, it may caused by the off-image-plane rotation. Other shapes of the unit structure are possible, which may caused by either the combination of the above rigid motions or non-rigid motion. Here, we only look for the occurrences of the first two cases, and use them to perform depth correction.

In both cases that we consider, the unit structure is isotropically scaling along all direction and the scaling factor can be calculated using the average length of the 4 edges. Figure 7 explains the relation between scaling factor and depth change using a single line segment. As line segment L moves closer to the camera, its projection size on the image plane changes from l_1 to l_2.

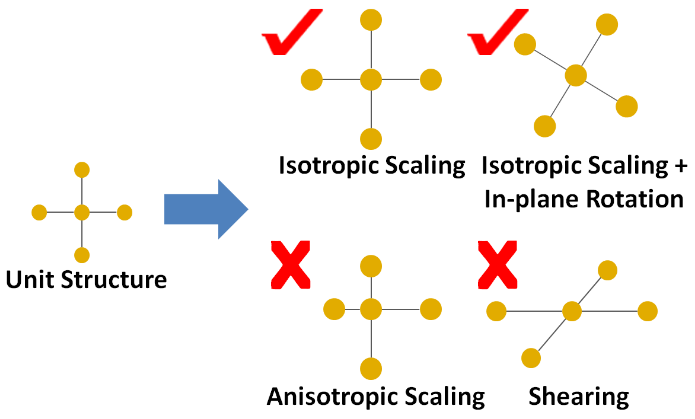


Fig. 6. The unit structure in frame t could be transformed into the 4 shapes in frame t+1 illustrated on the right hand side. We only make use of the first two cases to infer the depth change.

Assuming that the unknown line segment's depth changes from d_1 to d_2, we have:

$$\left.\begin{aligned} l_1 &= \frac{f}{d_1}L \\ \\ l_2 &= \frac{f}{d_2}L \end{aligned}\right\} \Longrightarrow s = \frac{l_2}{l_1} = \frac{d_1}{d_2} \tag{7}$$

where f is the focal length, s is the scaling factor of the line segment's image.

Equation 7 tells us that the length of the line segment's projection is inversely proportional to the depth of the line segment's position. Therefore, by calculating the scaling factor of the line segment, we find out its relative depth change without knowing the absolute depth value.

The same property holds for the unit structure. Let $d^t(\mathbf{p})$ be the depth of the center pixel $\mathbf{p}$ of the unit structure in frame t, $d^{t+1}(\mathbf{p}')$ be the depth of the corresponding pixel $\mathbf{p}'$ in the next frame, and s be the scaling factor of the unit structure, the following perspective depth correction constraint is included in the final quadratic programming formulation:

$$d^t(\mathbf{p}) - s \cdot d^{t+1}(\mathbf{p}') = 0 \tag{8}$$

So far, we only look for isotropic scaling of a unit structure between neighboring frames. However, an isotropic scaling could be the result of shape deformation rather that depth change. This is possible if we check only 2 neighboring frames, but a constant isotropic scaling in a sequence of consecutive frames is highly unlikely due to deformation. Therefore, in order to

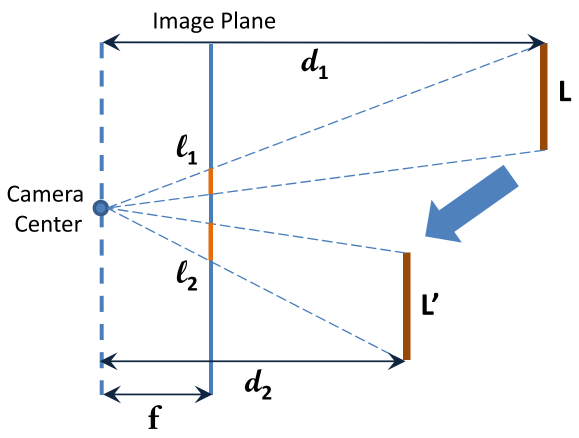


Fig. 7. The length of the image of a line segment changes when it moves closer to the camera. The ratio of the image length $\frac{l_1}{l_2}$ is inverse proportional the ratio of the depth $\frac{d_1}{d_2}$.

improve the robustness, we will only define depth change after observing a constant isotropic scaling in a sequence of neighboring frames.

B. User Interaction

All those automatically computed output will be presented in the user interface and be marked as defined regions. The reconstructed 3D feature points will be highlighted in yellow on the input image frames (figure 8, row 2, left image). The segmented foreground/background objects will be marked as at distinct depth (figure 8, row 1, middle image). The effect of the perspective depth correction will show up as red points in the user interface. With the assistance of defined regions, the users need to do some labeling work that points out the depth in the undefined regions.

At this point, we have to clarify that motion analysis is not applicable to any input video/shot, and it is not guaranteed to produce robust results in some scenarios. Since our system does not automatically determine what type of shot the input video is, it is all up to the user to decide if to apply a certain motion analysis to the input video. Furthermore, once the results from the motion analysis are presented in the user interface, the users will also have to decide whether to use the results in the final formulation. That means our system does not provide an automatic procedure to verify the correctness of the motion analysis results. The work is left to the users. Therefore, the users can choose to ignore those incorrect results from motion analysis. In the

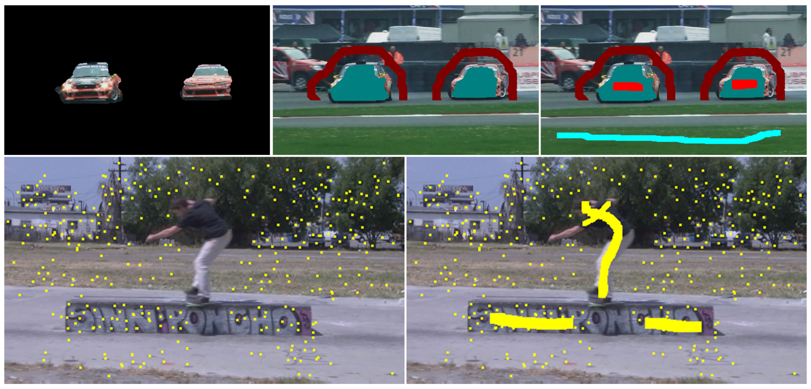


Fig. 8. The segmented results (row 1, left image) is input in the user interface and treated as predefined depth difference constraints (row 1, middle image). The users need to use depth difference brush to indicate the depth variation in the rest part of the image (row 1, right image). Generally, cyan regions are closer than red regions and dark color scribblings are from segmentation while light color scribblings are from user interaction. The reconstructed 3D points are marked as yellow (row 2, left image) in the user interface, and the users need to assign the computed depth value to undefined regions (row 2, right image).

worst case, the motion analysis procedures can not extract any useful 3D information. Then, the only thing the users can do is to mark the depth ordering manually using the brushes.

We introduce two types of brushes: the depth difference brush and depth equivalence brush. The former one is mainly intended to distinguish the surfaces that lie on different depth (figure 8, row 1, right image). And the latter one is employed to assign the recovered depth value to undefined regions (figure 8, row 2, right image). Both types of user inputs are are incorporated into the final solution through equality constraints. Let $\mathbf{p}_n$ and $\mathbf{p}_f$ be two pixels covered by the depth different brush, where $\mathbf{p}_n$ is inside the front area and $\mathbf{p}_f$ is inside the back area. Also assume that pixels $\mathbf{p}_i$ and $\mathbf{p}_j$ are two arbitrary pixels covered by the depth equivalence brush. The following equations will be added to the linear equations to constraint the depth of these pixels based on the user's input:

$$\begin{aligned} d(\mathbf{p}_n) - d(\mathbf{p}_f) &\geq \Delta d \\ d(\mathbf{p}_i) - d(\mathbf{p}_j) &= 0 \end{aligned} \tag{9}$$

C. Depth Propagation

To avoid solving an extremely large problem, we down-sample the input image sequences by a factor of 4 in both dimension, and up-sample the solution back the original resolution using the joint bilateral technique [24].

The propagation is based on the smoothness assumption that spatial and temporal neighboring pixels should have the same depth value if they share the same color. For each pixel $\mathbf{p}$ in the image sequence, we have

$$d(\mathbf{p}) - \sum_{\mathbf{q} \in N_{\mathbf{p}}} w_{\mathbf{pq}} d(\mathbf{q}) = 0 \tag{10}$$

where $d(\mathbf{p})$ is the depth value of pixel $\mathbf{p}$ and $w_{\mathbf{pq}}$ is the normalized weight between pixel $\mathbf{p}$ and $\mathbf{q}$, which is inversely proportional to the color difference of these two pixels. $N_{\mathbf{p}}$ is a set consisting of $\mathbf{p}$'s 8 spatial neighbors in the current frame and 1 temporal neighbor in the next frame, which is located using optical flow. The equation 10 for each pixel is stacked as a large sparse linear equation $Ax = 0$. where A is the weight matrix and x is the depth vector defined on every pixel.

The equality constraints from the SFM (equation 7), perspective depth correction (equation 8) and user's depth equivalence brush (equation 9) are stacked as a linear equation $Cx = d$.

And the inequality constraints from the moving object extraction (equation 6) and user's depth different brush (equation 9) are stacked as a linear inequality equation $Ex \geq f$.

Therefore, the propagation is formulated as the following optimization problem:

$$\min_x \|Ax\|_2^2 \qquad s.t. \quad Cx = d, Ex \geq f \tag{11}$$

It is equivalent to solve

$$\min_x (x^T A^T A x) \quad s.t. \quad Cx = d, Ex \geq f \tag{12}$$

This is a standard quadratic programming formulation. The solution can be found using the MOSEK optimization toolbox, which can deal with sparse large-scale problems.

IV. Results

We tested our system on a variety of video shots under different camera motions. We first demonstrate that the output from the three automatic preprocessing algorithms (SFM, moving object extraction and perspective depth correction) do help the users in the labeling step.

In the case of figure 9, the background scene is reconstructed by SFM, and the only undefined object/area is the moving person. By observing that the person is standing upon the rail, the user can simply assign the depth of the rail to the person by applying equal brush on the first and last frames. The depth value is propagated to in-between frames by temporal smoothness constraints.

Figure 12 shows the results of moving object extraction. This is a typical follow shot in which the subjects are followed by a panning camera. In this case, SFM fails due to the lack of parallax. However, the moving object extraction algorithm can easily segment out the cars from the constantly changing background audiences and foreground lawns. The results are input to the user interface as the depth difference constraints. In particular, we regard the segmented object to be in front of the area above and to the both sides of it, under observation that most follow shots are following objects on the ground. The remaining work to the users is differentiating the depth between the foreground lawn and the cars (figure 12, row 3).

Figure 10 shows the effectiveness of perspective depth correction. If the algorithm is not applied on the input video, the depth value will be propagated from the first frame to the last frame under the smoothness constraints defined on spatial and temporal neighbors. Thus, the same depth value on the truck in the first frame is passed onto the last frame, which is obviously invalid. Whereas, the perspective depth correction algorithm captures the depth change and produces correct depth value in the last frame.

Our system mainly relies on motion to produce additional information that facilitates the user interaction. However, if our three automatic subsystems can not do anything in the first place, our two brushes: depth difference brush and depth equality brush can still produce similar 3D effect as does the Guttmann's method. For example in figure 11, the scene, consisting of nearly static subjects, is captured by a stationary camera. Guttmann's method directly assigns different depth to different layers of the scene. Our method can accomplish the same effect by applying the depth difference brush a couple of times, in order to point out pairwise layers that lie on different depth.

All the shown stereo footage in this section as well as in prevous sections can be found in the accompanied video. Please wear a pair of red/cyan 3D glasses to watch them.

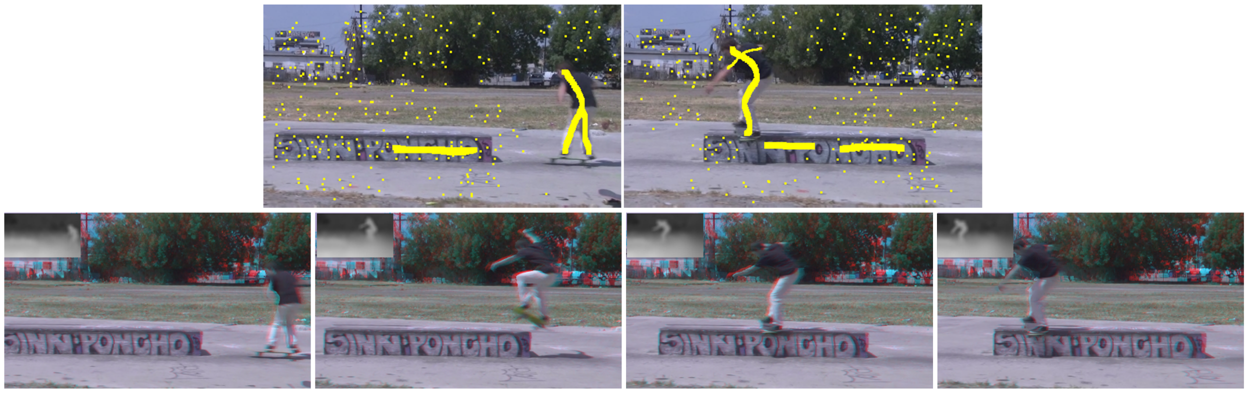


Fig. 9. The results obtained by equal brush labeling along with the assistance from SFM. 1st row shows the user labeling on the first and last frames. 2nd row shows some frames from the generated stereo video. The small gray-scale images are depth maps.

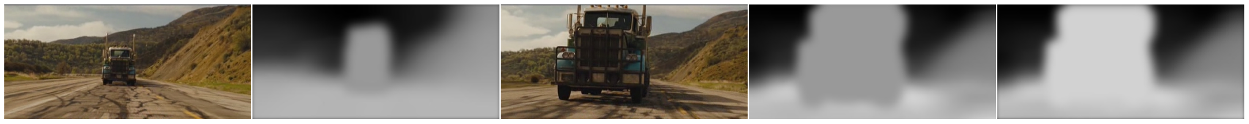


Fig. 10. The effectiveness of perspective depth correction algorithm. The 1st and 3rd images are the first and last frame from a video. The 2nd image is the depth map of the first frame and rightmost two images are depth maps of the last frame without and with the perspective depth correction respectively.

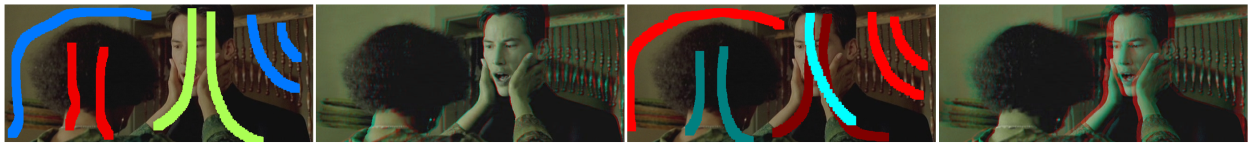


Fig. 11. The scene of stationary camera and static objects will fail our three algorithms in the preprocessing. However, our two labeling brushes: depth difference brush and depth equivalence brush can still produce similar visual 3D effect as does Guttmann's method. Guttmann's method ($1^{st} image$) assigns different depth (color coded) to different layers of the scene. Our method ($3^{rd} image$) achieves the same effect by applying the depth difference brush twice. The dark red&cyan scribbling pair indicate the depth difference of the old lady and the young man. And the light red&cyan pair point out the difference between the young man and the background. The results from both methods are visually comparable.

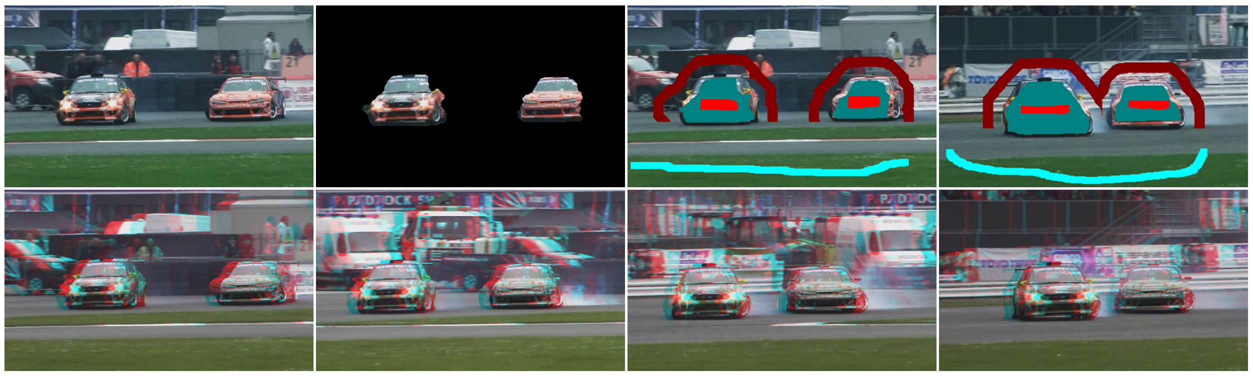


Fig. 12. The moving object extraction results and its integration into the uer interface. The left two images in the 1^{st} row are original image and segmented result. The right two images in the 1^{st} row show the look of the user interaction. The dark red&cyan pairs are automatically generated from the segmentation results and the light red&cyan pairs are marked by user. 2^{nd} row shows the stereo images from the video.

V. User Study

To further compare the our system with the one proposed by Guttmann et al. [9], two user studies are conducted. In the first study, the users are asked to convert videos to 3D using both systems. The qualities of the 3D videos generated by these users are evaluated in the second user study. Based on the design goal of our system, we hypothesize the following:

H1: Participants using our system will be able to complete the tasks quicker than using the Guttmann's method.

H2: Participants will find it easier to label the relative depths of different objects than labeling the disparity values.

H3: When given a choice of which system to use for future label tasks, participants will prefer our system.

H4: The improvement of our system over Guttmann's method is more noticeable when the input videos contain more motion.

A. System Usability Study

This user study is conducted in a controlled setting using a 2×4 (system $\times$ labeling task) between-subjects design. Each participant performs each of the four labeling tasks only once, two of which using our system and the other two using Guttmann's method. To further alleviate potential learning effects, a Graeco-Latin square was used to vary the order of exposure to the interface and the order of the task assignment. Prior to performing any of the tasks, participants were given a chance to play with both systems.

For each task, measurements of time to task completion and subjective user opinion were made. Pre-study questionnaires were administered to determine prior experience with computer and image/video editing. Post-study questionnaires measures perceived usefulness and ease-of-use, along with an indication of preference.

a) Participant Demographics: Twenty individuals were recruited from the CS graduate student population at two universities to participate in this study. Their response to the pre-study questionnaires shows that all of them are familiar with image/video editing, but do not have prior knowledge about 3D video conversion.

b) Time to Task Completion: The average time required to complete the four labeling tasks with the two system are illustrated in Figure 13. It shows that, for all four datesets, the users took less time when using our system to label the scene for 3D conversion. ANOVA tests preformed on these measurements confirm that the timing differences are statistical significant ($\mathbf{F(1,72) = 8.83, p < 0.01}$). This finding confirms our first hypothesis (H1). The time difference is mainly due to the advantage of relative depth labeling over absolute depth labeling, because with the later system, the users always tend to figure out an absolute depth value for an object and make sure this value be consistent with previous labeling. The huge time difference of ”gossip girl” comes from the merits of our motion analysis, where the background is already recovered by structure from motion algorithm.

c) Subjective Reactions: In the post-study questionnaires, participants were asked to indicate their degree of agreement to statements related to the usability of the two systems (using a five-point Likert scale). As shown in Table I, users generally find it difficult to label the depth directly, which is required by Guttmann et al.'s method. Majority users find labeling the relative depth easy and almost all users find our interface makes the 3D conversion task easier (H2). In addition, when asked “which UI would you prefer for future labeling tasks?”, all participants selected our UI (H3).

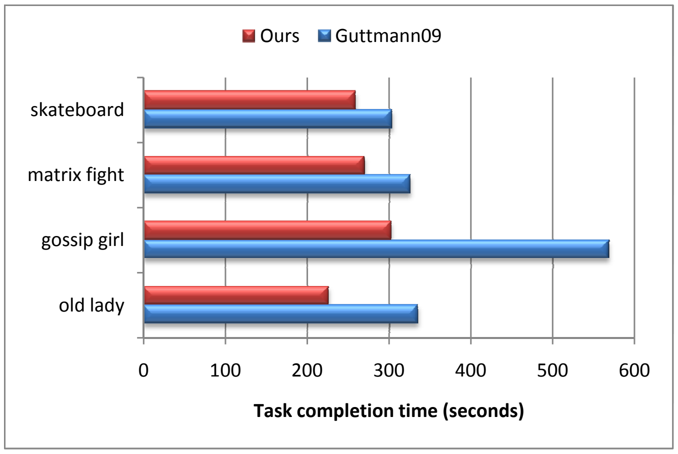


Fig. 13. The average time that the users took under different systems.

TABLE I

USERS' RESPONSE TO THE POST-STUDY QUESTIONNAIRES, WHERE UI_A REFERS TO GUTTMANN ET AL.'S METHOD AND UI_B REFERS TO OUR APPROACH

Statements	Strongly disagree	disagree	neutral	agree	strongly agree
It's easy to label the depth values directly using UI_A	3	10	3	4	0
It's easy to label the depth ordering using UI_B	1	0	0	8	11
The labeling task using UI_B is easier than using UI_A	1	0	1	6	12

B. Result Quality Study

To remove the bias that a user may have toward a given system, the result quality evaluation is performed in a separate user study. In this step, all 3D videos generated in the previous study are shuffled and shown to the users. The users are asked to assign a score to each 3D video, without knowing who converted the video and which system was used for the conversion. The score ranges from 0 to 4, with '0' being the poorest and '4' being the best.

The average scores for the 3D videos generated for each dataset using each system are illustrated in Figure 14. It shows that, for "skateboard", "matrix fight", and "gossip girl" datesets, the users find the results generated by our system to be much better than the ones generated by Guttmann's method. ANOVA tests confirm that the differences are statistical significant for all three datasets ($\mathbf{F(1, 254) = 114, p < 0.01}$ for "skateboard"; $\mathbf{F(1, 254) = 211, p < 0.01}$ for "fight"; and $\mathbf{F(1, 254) = 8.74, p < 0.01}$ for "gossip girl"). the "old lady" dataset, the average

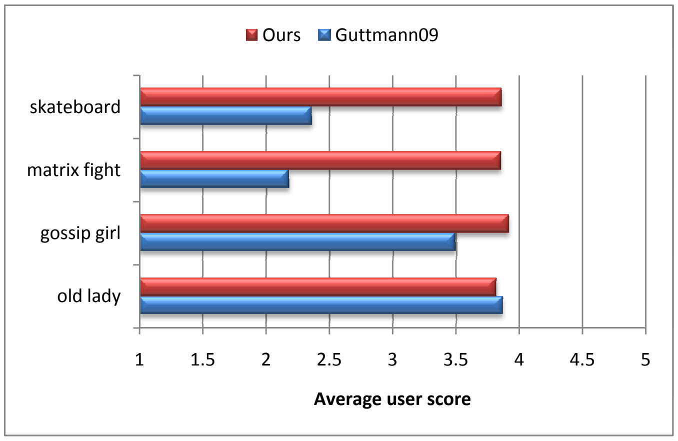


Fig. 14. The average score that the users gave to videos generated by different systems.

score for the video generated using the Guttmann's method is slightly higher, but the difference is statistically insignificant in ($\mathbf{F(1, 254) = 0.155, p > 0.5}$). This suggests that, when the scene contains little motion, the result qualities of both systems rely solely on the users' input. However, when the scene contains complex motion, the depth labels provided by the users become imprecise and the advantages of automatic pre-processing processes start to emerge (H4).

VI. Conclusion

This paper presented a hybrid framework to semi-automatically convert conventional videos to stereoscopic videos. Our system tries to make full use of the available motion information so that less user interaction is required. Compared to the previous methods [9], the major novelty of our framework lies in the utilization of motion prior analysis, such as automatic moving object extraction and perspective depth correction, to deal with arbitrary camera/object movement. In addition, we also proposed a new user interface that requires users to specify relative depth orders with the help of pre-computed 3D visual cues, instead of labeling the depth value directly. Finally, the depth information automatically extracted by computer algorithms and the interactively labeled by users are propagated to the whole video sequence through solving a quadratic programming problem. As demonstrated in the results and confirmed by the user study, our system is more user-friendly, as well as produces better quality of stereo videos for scenes with complex motions.

The success of our automatic processing modules depends on the camera/object motion. For example, with a dolly shot, SFM will work great; on the other hand, in a pan or follow-shot in sport video, the automatic segmentation method is effective. How to automatically detect these shot/motion types and apply appropriate motion analysis tool is an interesting topic of its own right. For the scope of this paper, we let the user decide which tool to use.

As in many computer vision algorithms, the scene depth estimation problem boils down to the issue of correspondences. If there is no feature to track, the resulting depth map will not be accurate. However, since the goal of our paper is view synthesis and the baseline (offset) of the synthesized view is small, even low-quality depth map can generate satisfactory result. One can look at the gossip-girl sequence, the depth map for their hair is not correct, yet without any special handling of occlusion, alpha, or precise boundary, one can barely notice the artifacts in the stereo movie.

Looking into the future, we are planning to automatically cut a video into multiple shots and try to utilize coherence among non-adjacent shots for the same scene. Moreover, we are interested in automatically categorizing the video sequences so that the computer can decide which visual cues can be utilized. How to explore other visual cues for automating the video stereolization process is also worth investigating.

References

[1] C. Tomasi, “Shape and motion from image streams under orthography: a factorization method,” *International Journal of Computer Vision*, vol. 9, pp. 137–154, 1992.

[2] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, “Consistent depth maps recovery from a video sequence,” *Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 6, 2009.

[3] S. Diplaris, N. Grammalidis, D. Tzovaras, and M. G. Strintzis, “Generation of stereoscopic image sequences using structure and rigid motion estimation by extended kalman filters,” *ICME*, vol. 2, 2002.

[4] K. Moustakas, D. Tzovaras, and M. G. Strintzis, “Stereoscopic video generation based on efficient layered strcture and motion estimation from a monoscopic image sequence,” *Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, 2005.

[5] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” *ARTIFICAL INTELLIGENCE*, vol. 17, pp. 185–203, 1981.

[6] M. J. Black and P. Anandan, “The robust estimation of multiple motions: parametric and piecewise-smooth flow fields,” *Comput. Vis. Image Underst.*, vol. 63, pp. 75–104, January 1996. [Online]. Available: http://portal.acm.org/citation.cfm?id=229144.229157

[7] C. Lei and Y. H. Yang, “Optical flow estimation on coarse-to-fine region-trees using discrete optimization,” in *ICCV*, 2009.

[8] C. Varekamp and B. Barenbrug, “Improved depth propagation for 2d to 3d video conversion using key-frames,” in *IETCVMP*, 2007.

[9] M. Guttmann, L. Wolf, and D. Cohen-or, “Semi-automatic stereo extraction from video footage,” in *ICCV*, 2009.

[10] D. Hoiem, A. A. Efros, and M. Hebert, “Geometric context from a single image,” in *International Conference on Computer Vision*. IEEE, 2005.

[11] A. Saxena, M. Sun, and A. Y. Ng, “Learning 3-d scene structure from a single still image,” in *International Conference on Computer Vision*, 2007.

[12] A. Saxena, S. H. Chung, and A. Y. Ng, “Learning depth from single monocular images,” in *NIPS*, 2006.

[13] G. Zhang, Z. Dong, J. Jia, L. Wan, T.-T. Wong, and H. Bao, “Refilming with depth-inferred videos,” *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 15(5), pp. 828–840, 2009.

[14] S. Knorr and T. Sikora, “An image-based rendering (ibr) approach for realistic stereo view synthesis of tv broadcast based on structure from motion,” in *ICIP*, 2007.

[15] E. Rotem, K. Wolowelsky, and D. Pelz, “Automatic video to stereoscopic video conversion,” in *SPIE*, 2005.

[16] G. Zhang, W. Hua, X. Qin, T.-T. Wong, and H. Bao, “Stereoscopic video synthesis from a monocular video,” *Transactions on Visualization and Computer Graphics*, vol. 13, no. 4, 2007.

[17] S. Katz, *Film Directing Shot by Shot: Visualizing from Concept to Screen*. Michael Wiese, 1991.

[18] P. Harman, “Home based 3d entertainment: An overview,” in *ICIP*, 2000, pp. Vol I: 1–4.

[19] P. Harman, J. Flack, S. Fox, and M. Dowley, “Rapid 2d to 3d conversion,” in *in Stereoscopic Displays and Virtual Reality Systems IX, Andrew*, 2002, pp. 78–86.

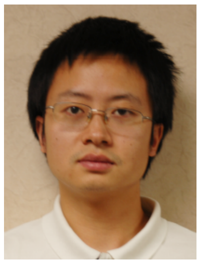
[20] A. L. Dani, D. Lischinski, and Y. Weiss, “Colorization using optimization,” *ACM Transactions on Graphics*, vol. 23, pp. 689–694, 2004.

[21] D. Skora, D. Sedlacek, S. Jinchao, J. Dingliana, and S. Collins, “Adding depth to cartoons using sparse depth (in)equalities,” *Eurographics*, 2010.

[22] Y. Boykov and M. Jolly, “Interactive graph cuts for optimal boundary and region segmentation of objects in ND images,” *IJCV*, 2001.

[23] Y. Li, J. Sun, C. Tang, and H. Shum, “Lazy snapping,” *SIGGRAPH*, 2004.

[24] J. Kopf, M. Cohen, D. Lischinski, and M. Uyttendaele, “Joint bilateral upsampling,” *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, vol. 26, no. 3, p. to appear, 2007.



Miao Liao received his B.S. degree from Tsinghua University in 2005. He is currently a PhD student in the Department of Computer Science at the University of Kentucky. He is working as a research assistant in the Center for Visualization and Virtual Environment. His research interests include computer vision, computer graphics, image processing and multimedia.

Jizhou Gao received the B.S. degree from the Computer Science Department, Zhejiang University, China in 2006. He is currently a Ph.D. candidate in the Computer Science Department at the University of Kentucky. His research interests include computer vision and data mining.

Ruigang Yang received the MS degree in Computer Science from Columbia University in 1998 and the PhD degree in Computer Science from the University of North Carolina, Chapel Hill, in 2003. He is an associate professor in the Computer Science Department, University of Kentucky. His research interests include computer vision, computer graphics, and multimedia. He is a member of the IEEE Computer Society and the ACM. He is a recipient of the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Program Award in 2004.

Minglun Gong is an associate professor at the Department of Computer Science, Memorial University of Newfoundland. He obtained his Ph.D. from University of Alberta in 2003, his M.Sc. from Tsinghua University in 1997, and his B.Engr. from Harbin Engineering University in 1994. After graduation, he was a faculty member at Laurentian University for four years before joined the Memorial University.

Minglun's research interests include a variety of topics in computer graphics, computer vision, image processing, pattern recognition, and optimization techniques. He has published over 50 technical papers in refereed journals and conference proceedings and served as program committee member and reviewer for international journals and conferences.