Joint Subspace Stabilization for Stereoscopic Video

Feng Liu[†]
[†] Portland State University
fliu@cs.pdx.edu

Yuzhen Niu[♮,†]
[♮]Fuzhou University
yuzhen@cs.pdx.edu

Hailin Jin[‡]
[‡]Adobe Research
hljin@adobe.com

Abstract

Shaky stereoscopic video is not only unpleasant to watch but may also cause 3D fatigue. Stabilizing the left and right view of a stereoscopic video separately using a monocular stabilization method tends to both introduce undesirable vertical disparities and damage horizontal disparities, which may destroy the stereoscopic viewing experience. In this paper, we present a joint subspace stabilization method for stereoscopic video. We prove that the low-rank subspace constraint for monocular video [10] also holds for stereoscopic video. Particularly, the feature trajectories from the left and right video share the same subspace. Based on this proof, we develop a stereo subspace stabilization method that jointly computes a common subspace from the left and right video and uses it to stabilize the two videos simultaneously. Our method meets the stereoscopic constraints without 3D reconstruction or explicit left-right correspondence. We test our method on a variety of stereoscopic videos with different scene content and camera motion. The experiments show that our method achieves high-quality stabilization for stereoscopic video in a robust and efficient way.

1. Introduction

Stereoscopic video provides an immersive viewing experience by invoking the binocular depth cue. Thanks to the recent success of 3D movies, there is a resurgence of interests in stereoscopic video. Nowadays we have capable hardware for displaying and capturing stereoscopic video. However, the development in stereoscopic video processing is still in its infancy. This paper addresses an important stereoscopic video processing problem, video stabilization.

Video stabilization is the problem of removing undesired camera shake from a video. It has been shown that a good video stabilization algorithm can significantly improve the visual quality of an amateur video, making it close to the level of a professional one [14]. Stabilization algorithms play an even more important role in stereoscopic video because the effect of camera shake is more pronounced in stereo [19]. In particular, the temporal jitter in a shaky stereoscopic video can cause an excessive demand on the accommodation-vergence linkage [11]. This may bring physical discomforts to viewers if they were asked to watch the shaky video over an extended period.

Applying a homography-based monocular stabilization method [20] to each view of a stereoscopic video is problematic as it often damages the original horizontal disparities and induces the vertical disparities. Both will compromise the stereoscopic viewing experience. Existing 3D reconstruction-based stabilization methods can be well extended to handle stereoscopic video [3, 5, 14]. Once the 3D camera motion and scene structure are estimated, we can smooth the camera motion and synthesize a new pair of left and right video to follow the smooth camera motion. The left and right video are consistently stabilized in this way and the disparity problems are handled implicitly. 3D reconstruction-based methods, however, require structure from motion, which is typically brittle and time-consuming.

In this work, we present a joint subspace stabilization method for stereoscopic video. This method handles disparity problems in video stabilization without 3D reconstruction or even explicit left-right correspondence estimation. According to Irani [10], the feature trajectories from a short monocular video lie in a low-rank subspace. We extend this to stereoscopic video and prove that the feature trajectories from the left and right video of a stereoscopic video share a common subspace spanned by a small number of eigen-trajectories. This means that we can represent any trajectory as a linear combination of eigen-trajectories. We can then smooth the left and right feature trajectories consistently by smoothing the common eigen-trajectories. In this way, no vertical disparities will be introduced and horizonal disparities will be smoothed. These two are the key properties of a successful stereoscopic video stabilization method.

This paper has two main contributions. First, we prove that the low-rank subspace constraint for monocular video [10] also holds for stereo video with no increase in the rank. Our second contribution is a high-quality practical algorithm for stereo video stabilization. It does not require explicit left-right correspondence. By combining the trajectories from the left and right video, our method is robust to the insufficiency of long trajectories and to the degeneration in scene content, camera motion, and tracking error.

2. Related Work

Existing work on monocular video stabilization can be categorized into 2D and 3D reconstruction-based methods. 2D methods restrict themselves to 2D motion models between frames [6, 9, 12, 18, 20]. They are robust and efficient but have limited smoothing capabilities because 2D motion models cannot account for the parallax. If they are independently applied to the left and right view of a stereo video, they can often introduce vertical disparities which may damage the stereoscopic viewing experience.

3D reconstruction-based video stabilization methods compute 3D models of the scene and use image-based rendering techniques to render the stabilized video [2, 3, 5, 14]. Compared to 2D methods, these methods have better smoothing capabilities. In principle, these 3D reconstruction-based methods can be applied to stereo video. However, they require 3D reconstruction which is a fragile process and is sensitive to scene content, camera motion, etc., and have limited practical applicability. The latest work along this line uses a RGBD camera to capture the scene depth and facilitates 3D reconstruction [16].

Two recent methods compute middle-level representations between a full 3D reconstruction of the scene and a 2D motion between frames [7,15]. They can achieve similar stabilization effects to the 3D reconstruction-based methods but retain most of the robustness and efficiency of the 2D methods. Unlike 3D reconstruction-based methods, applying them independently to each view of a stereoscopic video often does not produce satisfying results as shown later on. We extend [15] to stereoscopic video.

There is limited work on video stabilization beyond monocular cameras. Liu *et al*. generalize their 2D method to stereoscopic video [13]. Their method uses similarity transformations and includes an additional term to account for the vertical disparities between left and right video. The use of similarity transformations limited its smoothing capabilities like most of the 2D methods. Smith *et al*. studied video stabilization for light-field cameras [22]. Their method performs 3D scene reconstruction and uses depth-guided warping to create novel views. It shares the same limitations of the 3D reconstruction-based stabilization methods in terms of robustness and efficiency. Moreover, it has been only demonstrated on camera arrays with 5 or more cameras.

Finally, it has been shown in previous research that left-right correspondence is not necessary for some computer vision problems, such as aligning non-overlapping videos [4], and non-rigid reconstruction from two videos [23]. This paper shows that left-right correspondence is not needed for stereo video stabilization.

3. Joint Subspace Video Stabilization

Like monocular video stabilization methods, our method first tracks and smooths feature trajectories from input video, and then synthesizes stabilized video guided by the smooth feature trajectories. Specifically, we first track feature trajectories by applying the KLT algorithm [21] to the left and right video *separately*. We rule out trajectories on moving objects using the epipolar constraint and the trajectory length threshold [15]. We denote with $(x_L^i(t), y_L^i(t))$ the location of the i-th feature trajectory in the t-th frame of the left video and with $(x_R^i(t), y_R^i(t))$ a similar quantity in the right video. Note that in general there is no relationship between $(x_L^i(t), y_L^i(t))$ and $(x_R^i(t), y_R^i(t))$ even for the same i and t because we track separately. The stabilization task is to obtain two sets of new feature trajectories $\{\hat{x}_L^i(t), \hat{y}_L^i(t)\}$ and $\{\hat{x}_R^i(t), \hat{y}_R^i(t)\}$ that guide the rendering of a pair of stabilized videos. In order to stabilize a stereoscopic video, the new feature trajectories need to meet two requirements.

1. Both $\{\hat{x}_L^i(t), \hat{y}_L^i(t)\}$ and $\{\hat{x}_R^i(t), \hat{y}_R^i(t)\}$ should be smooth and consistent with the original scene viewed by two moving cameras.
2. $\{\hat{x}_L^i(t), \hat{y}_L^i(t)\}$ and $\{\hat{x}_R^i(t), \hat{y}_R^i(t)\}$ should be consistent with a moving stereo camera instead of two arbitrarily positioned cameras.

As discussed previously, 3D reconstruction can be used to meet these two requirements, but it is time-consuming and often brittle. Below we first show that if an affine camera model is assumed, we can easily perform 3D reconstruction using matrix factorization for stereo video stabilization. We then describe our joint-subspace method that extends the matrix factorization-based method to handle the challenging case of a perspective camera.

3.1. Affine Stereo Video Stabilization

We define the following $2(n_L + n_R) \times k$ displacement matrix

$$\begin{bmatrix} x_L^1(1) - x_L^1(r) & x_L^1(2) - x_L^1(r) & \cdots & x_L^1(k) - x_L^1(r) \\ y_L^1(1) - y_L^1(r) & y_L^1(2) - y_L^1(r) & \cdots & y_L^1(k) - y_L^1(r) \\ & & \vdots & \\ x_L^{n_L}(1) - x_L^{n_L}(r) & x_L^{n_L}(2) - x_L^{n_L}(r) & \cdots & x_L^{n_L}(k) - x_L^{n_L}(r) \\ y_L^{n_L}(1) - y_L^{n_L}(r) & y_L^{n_L}(2) - y_L^{n_L}(r) & \cdots & y_L^{n_L}(k) - y_L^{n_L}(r) \\ x_R^1(1) - x_R^1(r) & x_R^1(2) - x_R^1(r) & \cdots & x_R^1(k) - x_R^1(k) \\ y_R^1(1) - y_R^1(r) & y_R^1(2) - y_R^1(r) & \cdots & y_R^1(k) - y_R^1(k) \\ & & \vdots & \\ x_R^{n_R}(1) - x_R^{n_R}(r) & x_R^{n_R}(2) - x_R^{n_R}(r) & \cdots & x_R^{n_R}(k) - x_R^{n_R}(r) \\ y_R^{n_R}(1) - y_R^{n_R}(r) & y_R^{n_R}(2) - y_R^{n_R}(r) & \cdots & y_R^{n_R}(k) - y_R^{n_R}(r) \end{bmatrix} \quad (1)$$

where n_L and n_R are the numbers of tracking features in the left and right views respectively, k is the number of video frames, and r is the index of the reference frame.

Proposition 1. *The displacement matrix as defined in equation* (1) *has a rank up to* 8 *for affine cameras.*

Proof. Without loss of generality, we define the global reference frame to be that of the left camera at time $t = 0$. Let $(R(t), T(t))$ be the rigid motion of the left camera at time t. By definition, we have $R(0) = I$ and $T(0) = 0$. We assume the relative rigid motion of the left and right cameras does not change over time which we denote with $(\Delta R, \Delta T)$. We further assume that the left and right cameras are in the standard stereo configuration, i.e., there is no relative rotation and the relative translation is along the x-axis. We have $\Delta R = I$ and $\Delta T = [\Delta T_1, 0, 0]^T$. We assume both cameras follow the weak perspective projection model and share the same intrinsic parameters which do not change over time. Let $[X_L, Y_L, Z_L]$ be the coordinates of a point in the global reference frame that is visible in the left camera. According to the weak perspective projection model, the projection in the left camera is given by

$$\mathbf{x}_L(t) = \pi\left(\begin{bmatrix} f & 0 & c_1 \\ 0 & f & c_2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ 0 & 0 & 0 & d_0 \end{bmatrix}\begin{bmatrix} X_L \\ Y_L \\ Z_L \\ 1 \end{bmatrix}\right)$$

where f is the focal length, $[c_1, c_2]^T$ is the image center, $d_0 \in \mathbb{R}$ is a constant, R_{ij} is the (i,j)-th component of $R(t)$ and t_i is the i-th component of $T(t)$. Note that for simplicity we have dropped the time index. $\pi(\cdot)$ is the perspective projection, i.e. $\pi([X, Y, Z]) = [X/Z, Y/Z]$. The 2D image displacement in the left image is given by

$$\begin{aligned}\begin{bmatrix} u_L \\ v_L \end{bmatrix} &= \mathbf{x}_L(t) - \mathbf{x}_L(0) \\ &= \frac{f}{d_0}\begin{bmatrix} (R_{11}-1)X_L + R_{12}Y_L + R_{13}Z_L + t_1 \\ R_{21}X_L + (R_{22}-1)Y_L + R_{23}Z_L + t_2 \end{bmatrix}\end{aligned}$$

Let $[X_R, Y_R, Z_R]$ be the coordinates of another point in the global reference frame that is visible in the right camera. Note that we do not assume any correspondence on the points between the left and right images. The rigid motion of the right camera can be expressed in terms of that of the left camera as $(R(t), T(t) + \Delta T)$. The projection in the right camera is given by

$$\mathbf{x}_R(t) = \pi\left(\begin{bmatrix} f & 0 & c_1 \\ 0 & f & c_2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 + \Delta T_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ 0 & 0 & 0 & d_0 \end{bmatrix}\begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix}\right)$$

and the 2D image displacement in the right image is given by

$$\begin{aligned}\begin{bmatrix} u_R \\ v_R \end{bmatrix} &= \mathbf{x}_R(t) - \mathbf{x}_R(0) \\ &= \frac{f}{d_0}\begin{bmatrix} (R_{11}-1)X_R + R_{12}Y_R + R_{13}Z_R + t_1 \\ R_{21}X_R + (R_{22}-1)Y_R + R_{23}Z_R + t_2 \end{bmatrix}\end{aligned}$$

We define the following three matrices

$$E = \begin{bmatrix} R_{11}-1 & R_{12} & R_{13} & t_1 & R_{21} & R_{22}-1 & R_{23} & t_2 \end{bmatrix}^T$$

$$C_L = \frac{f}{d_0}\begin{bmatrix} X_L & Y_L & Z_L & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X_L & Y_L & Z_L & 1 \end{bmatrix}$$

$$C_R = \frac{f}{d_0}\begin{bmatrix} X_R & Y_R & Z_R & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X_R & Y_R & Z_R & 1 \end{bmatrix}$$

We obtain

$$\begin{bmatrix} u_L & v_L & u_R & v_R \end{bmatrix}^T = \begin{bmatrix} C_L \\ C_R \end{bmatrix} E, \tag{2}$$

which concludes the proof. □

In Equation 2, C_L and C_R depend only on the scene points and E depends only on the motion. This shows that for an affine camera, the displacement matrix from a stereoscopic video can be factored into a camera matrix E and scene matrices C_L and C_R. This is effectively a 3D reconstruction similar to 3D reconstruction for monocular video in the case of an affine camera. We can then perform stereoscopic video stabilization in the following steps.

1. Estimate feature trajectories from the left and right video separately and assemble a joint trajectory displacement matrix according to Equation 1.

2. Apply matrix factorization to the joint displacement matrix and obtain the scene matrices C_L and C_R and the camera matrix E.

3. Smooth the camera matrix and compose with C_L and C_R to obtain the smoothed trajectories.

4. Warp the left and right video guided by the smooth trajectories using content-preserving warping [14].

This approach to stereoscopic video stabilization clearly will not bring in disparity problems as it only smoothes the motion of the stereo camera rig. A major problem with this approach is that affine camera model is often not a good approximation of a real-world camera.

3.2. Perspective Stereo Video Stabilization

We are inspired by the recent subspace video stabilization method for monocular video that handles a more general camera, i.e. perspective camera [15]. This monocular stabilization method was based on the subspace observation that the feature trajectories of a short video imaged by a perspective camera lie in a low-rank subspace [10]. As shown in [10], a trajectory displacement matrix for a monocular video can be factorized into two low-rank matrices C and E in a similar way to Equation 2. For a perspective camera, matrix C and E are more complicated than those for an affine camera. But it has been shown in [15], matrix E, called “eigen-trajectory matrix”, can be smoothed in the same way as the camera matrix for the affine camera, and the smooth trajectories can be obtained by composing matrix C and the smoothed eigen-trajectory matrix.

In the following, we first prove that the subspace constraint is also valid for stereoscopic video imaged by perspective cameras. In fact, the trajectories from the left and right video share a common subspace. This means that the matrix factorization in Equation 2 is also valid for perspective cameras. Accordingly, we can apply a similar approach for stereoscopic video captured by affine cameras to those captured by perspective cameras. We will further prove that this method will not cause disparity problems for perspective cameras.

Proposition 2. *The displacement matrix as defined in Equation 1 has a rank up to* 6 *for perspective cameras with instantaneous motion that have constant focal lengths.*

Proof. We follow the same notation as in the previous proof. Instead of the weak perspective projection model, we assume the cameras follow the standard perspective projection model. Let $\mathbf{X}_L = [X_L, Y_L, Z_L]^T$ be the coordinates of a point visible in the left camera. The feature point displacement in the left video is given by

$$\begin{bmatrix}u_L, v_L\end{bmatrix}^T = f\pi(R\mathbf{X}_L + T) - f\pi(\mathbf{X}_L) \tag{3}$$

We assume the camera motion is instantaneous that both the rotation and translation are small. We can rewrite Equation 3 in terms of its Taylor expansion as follows:

$$\begin{aligned} u_L =& \frac{f}{Z_L^2}\Big(- X_L Y_L \omega_1 + (X_L^2 + Z_L^2)\omega_2 - Y_L Z_L \omega_3 \\ & + Z_L t_1 - X_L t_3\Big) \end{aligned} \tag{4}$$

$$\begin{aligned} v_L =& \frac{f}{Z_L^2}\Big(- (Y_L^2 + Z_L^2)\omega_1 + X_L Y_L \omega_2 + X_L Z_L \omega_3 \\ & + Z_L t_2 - Y_L t_3\Big) \end{aligned} \tag{5}$$

where $[\omega_1, \omega_2, \omega_3]^T$ are the exponential coordinates of R. Let $\mathbf{X}_R = [X_R, Y_R, Z_R]^T$ be a point visible in the right image. The displacement in the right video is given by

$$[u_R, v_R]^T = f\pi(R\mathbf{X}_R + T + \Delta T) - f\pi(\mathbf{X}_R + \Delta T) \tag{6}$$

The Taylor expansion is given by

$$\begin{aligned} u_R =& \frac{f}{Z_R^2}\Big(- (X_R + \Delta T_1)Y_R\omega_1 + (X_R(X_R + \Delta T_1) + Z_R^2)\omega_2 \\ & - Y_R Z_R \omega_3 + Z_R t_1 - (X_R + \Delta T_1)t_3\Big) \end{aligned} \tag{7}$$

$$\begin{aligned} v_R =& \frac{f}{Z_R^2}\Big(- (Y_R^2 + Z_R^2)\omega_1 + X_R Y_R \omega_2 \\ & + X_R Z_R \omega_3 + Z_R t_2 - Y_R t_3\Big) \end{aligned} \tag{8}$$

Combining Equations 4, 5, 7, and 8, we obtain

$$\begin{bmatrix}u_L & v_L & u_R & v_R\end{bmatrix}^T = f\begin{bmatrix}C_L \\ C_R\end{bmatrix}E, \tag{9}$$

where

$$E = \begin{bmatrix}\omega_1 & \omega_2 & \omega_3 & t_1 & t_2 & t_3\end{bmatrix}^T,$$

$$C_L = \frac{1}{Z_L^2}\begin{bmatrix} -X_L Y_L & X_L^2 + Z_L^2 & -Y_L Z_L & Z_L & 0 & -X_L \\ -(Y_L^2 + Z_L^2) & X_L Y_L & X_L Z_L & 0 & Z_L & -Y_L \end{bmatrix},$$

$$C_R(1:3) =$$
$$\frac{1}{Z_R^2}\begin{bmatrix} -(X_R + \Delta T_1)Y_R & X_R(X_R + \Delta T_1) + Z_R^2 & -Y_R Z_R \\ -(Y_R^2 + Z_R^2) & X_R Y_R & X_R Z_R \end{bmatrix},$$

is the left 2×3 part of C_R and

$$C_R(4:6) = \frac{1}{Z_R^2}\begin{bmatrix} Z_R & 0 & -(X_R + \Delta T_1) \\ 0 & Z_R & -Y_R \end{bmatrix},$$

is the right 2×3 part of C_R. This concludes the proof. □

We would like to make several important remarks.

- Proposition 2 is an extension of the result from Irani [10]. However, it is interesting to see that there is no increase in the rank when we go from a monocular video to a stereoscopic one. Furthermore, we note that the left and right video share the same low-dimensional subspace E. Like [10], we can prove a similar rank constraint for varying focal lengths.
- The trajectory matrix can be obtained as a summation of the displacement matrix and a matrix with the reference point repeated over time. Since the latter matrix is a rank-1 matrix, the trajectory matrix has a rank up to 7 for instantaneous motion and a constant focal length. We use the same notations (C, E) to denote the factorization results of the displacement matrix and the trajectory matrix for consistency.
- Equation 9 holds only for the ideal case where the camera motion is instantaneous. In practice, the assumption on instantaneous motion does not hold for long videos. However, we can generalize Proposition 2 to such situations as discussed in [15]. First, the trajectory matrix is highly incomplete. The low-rank constraint does not have to apply to the missing elements in the trajectory matrix. For instance, it is fine for stabilization if the missing elements are not reconstructed accurately. Second, we can still use the rank constraint as long as the error between the measured trajectory matrix and the reconstructed trajectory matrix is low. Third, we can increase the rank to allow more flexibility. Experimentally, we find that for all the videos we have tested rank 9 is sufficient in terms of obtaining a low reconstruction error. Rank 9 is also used in the monocular subspace stabilization method [15].

3.2.1 Summary: joint subspace stabilization

We now summarize our stereoscopic video stabilization method for perspective cameras as follows. As we estimate the eigen-trajectory matrix E in Equation 9 jointly from the trajectories of the left and right video, we call our method *joint subspace video stabilization.*

1. Estimate feature trajectories from the left and right video separately and assemble a joint trajectory matrix as follows.

$$M_{LR} = \begin{bmatrix} M_L \\ M_R \end{bmatrix}, \tag{10}$$

   where M_L is the left trajectory matrix defined as

$$\begin{bmatrix} x_L^1(1) & x_L^1(2) & \cdots & x_L^1(k) \\ y_L^1(1) & y_L^1(2) & \cdots & y_L^1(k) \\ & & \vdots & \\ x_L^n(1) & x_L^n(2) & \cdots & x_L^n(k) \\ y_L^n(1) & y_L^n(2) & \cdots & y_L^n(k) \end{bmatrix}. \tag{11}$$

   $(x_L^i(t), y_L^i(t))$ is the location of the i^{th} feature trajectory at frame t. M_R is defined in a similar way for the right video.

2. Apply the moving factorization algorithm [15] to the joint trajectory matrix and obtain the coefficient matrices C_L and C_R and an eigen-trajectory matrix E.

3. Smooth the eigen-trajectory matrix E by low-pass filtering or canonical path fitting and obtain a smooth eigen-trajectory $\hat{E}$. The smoothed trajectory matrices $\hat{M}_L$ and $\hat{M}_R$ can be computed by composing $\hat{E}$ with C_L and C_R.

4. Warp the left and right video guided by $\hat{M}_L$ and $\hat{M}_R$ using a content-preserving warping method [14].

3.2.2 Disparity

We now examine how this joint subspace stabilization method works on disparities in a stereoscopic video. Let $\{x_L(t), y_L(t)\}$ and $\{x_R(t), y_R(t)\}$ be two corresponding feature trajectories of the left and right video, respectively. The disparity sequence between these two trajectories is

$$\begin{aligned} \{d(t)\} &= \{(x_R(t) - x_L(t), y_R(t) - y_L(t))\} \\ &= ((C_R^x - C_L^x)E, (C_R^y - C_L^y)E) \end{aligned} \tag{12}$$

and the disparity sequence after separate smoothing is

$$\begin{aligned} \{\hat{d}(t)\} &= \{(\hat{x}_R(t) - \hat{x}_L(t), \hat{y}_R(t) - \hat{y}_L(t))\} \\ &= ((C_R^x - C_L^x)\hat{E}, (C_R^y - C_L^y)\hat{E}) \end{aligned} \tag{13}$$

This shows that both horizontal disparities and vertical disparities are smoothed. Particularly, if the input video has zero vertical disparities (assuming a rectified stereo rig and no noise in feature tracking), $C_R^y = C_L^y$. Then the output trajectories will have no vertical disparities either. Furthermore, horizontal disparities are smoothed. Horizontal disparities play an important role in depth perception. Jittery horizontal disparities may lead to inconsistent depth perception or distortions in 3D space. The proposed joint subspace algorithm is able to remove the jitter in horizontal disparities. Meanwhile, as horizontal disparities in a stereoscopic video typically change insignificantly temporally, smoothing will not change disparity magnitudes significantly.

3.2.3 Joint subspace vs. separate subspace

Equation 9 shows that the feature trajectories from the left and right video share the same subspace. This actually implies that ideally, applying the monocular subspace stabilization method [15] to each video separately, called separate subspace stabilization, will lead to the same stabilization result as our joint subspace method. However, separate subspace stabilization often cannot work well in practice. The subspaces separately estimated for the left and right video are often different when there are feature tracking errors or the set of feature trajectories for the two videos differ from each other significantly. The discrepancy between the left and right stabilization results will then be introduced.

We performed a simulation to compare the joint subspace method with the separate subspace method. We created a random rank-9 matrix M of size $(2m + 1) \times 40$ as a feature trajectory matrix. We partitioned it into M_1, M_2, and M_3, where the first two matrices each have m rows and the last one only has one row. We added random noises with the same distribution to both M_1 and M_2 and dropped the last 5 elements of M_3. We used the subspaces estimated from the noisy versions of M_1 and M_2 to reconstruct the 5 missing elements in M_3. We measured the two reconstruction errors and compared them against the reconstruction error using a subspace computed jointly from M_1 and M_2. We also computed the difference between the reconstruction using M_1 and M_2. This difference can be regarded as the vertical disparity if we consider M_3 to be the y-component of a feature trajectory. We varied the value of m from 10 to 100 and repeated the whole procedure 1000 times for each m value.

The results are shown in Figure 1. We made two observations. First, the joint subspace method produces consistently a smaller reconstruction error than the method using each individual subspace. More importantly, the discrepancy between the reconstructions using M_1 and M_2 is very significant, as indicated by the blue curve. Second, the errors using individual subspaces and the difference between the reconstruction from the two individual subspaces go down as the number of feature trajectories increases. This

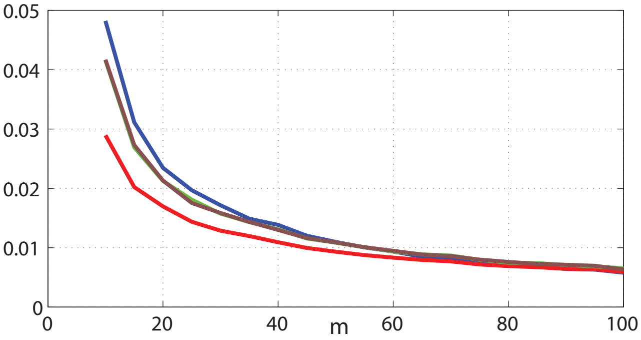


Figure 1. Joint vs. separate subspace. The curves show the reconstruction errors as the number of feature trajectories increases. **Red**: the error using the joint subspace. **Green and Brown**: the errors using the two individual subspaces. **Blue**: the reconstruction difference between the two individual subspaces. One can observe that the error using the joint subspace is consistently smaller than the ones using the individual subspaces. Moreover, the errors using the individual subspaces and their difference go down as the number of feature trajectories increases.

is because that the subspaces computed using the noisy data get better as the number of feature trajectories increases and as a result the reconstruction errors and the reconstruction difference go down.

One concern with the joint subspace method is the possibly increasing subspace fitting error as only one subspace is estimated instead of two. We examined the fitting error for 20 stereoscopic videos, each of which has a frame size 640×360. We considered only the first 40 frames of each video and only used trajectories that spanned that entire duration, thus yielding a complete matrix that we can factorize using SVD. We found that the mean factorization errors from the joint factorization are slightly bigger than those from the separate factorization, as expected. On average, the joint factorization errors are about 6.6%, 7.6%, and 9.2% more than the separate factorization ones when we use rank 7, 8, and 9 respectively. But we also found that the absolute errors are small enough to be negligible. The mean factorization errors from the joint factorizations are 0.083 , 0.074, 0.067 pixels for rank 7, 8, and 9 respectively.

4. Experiments

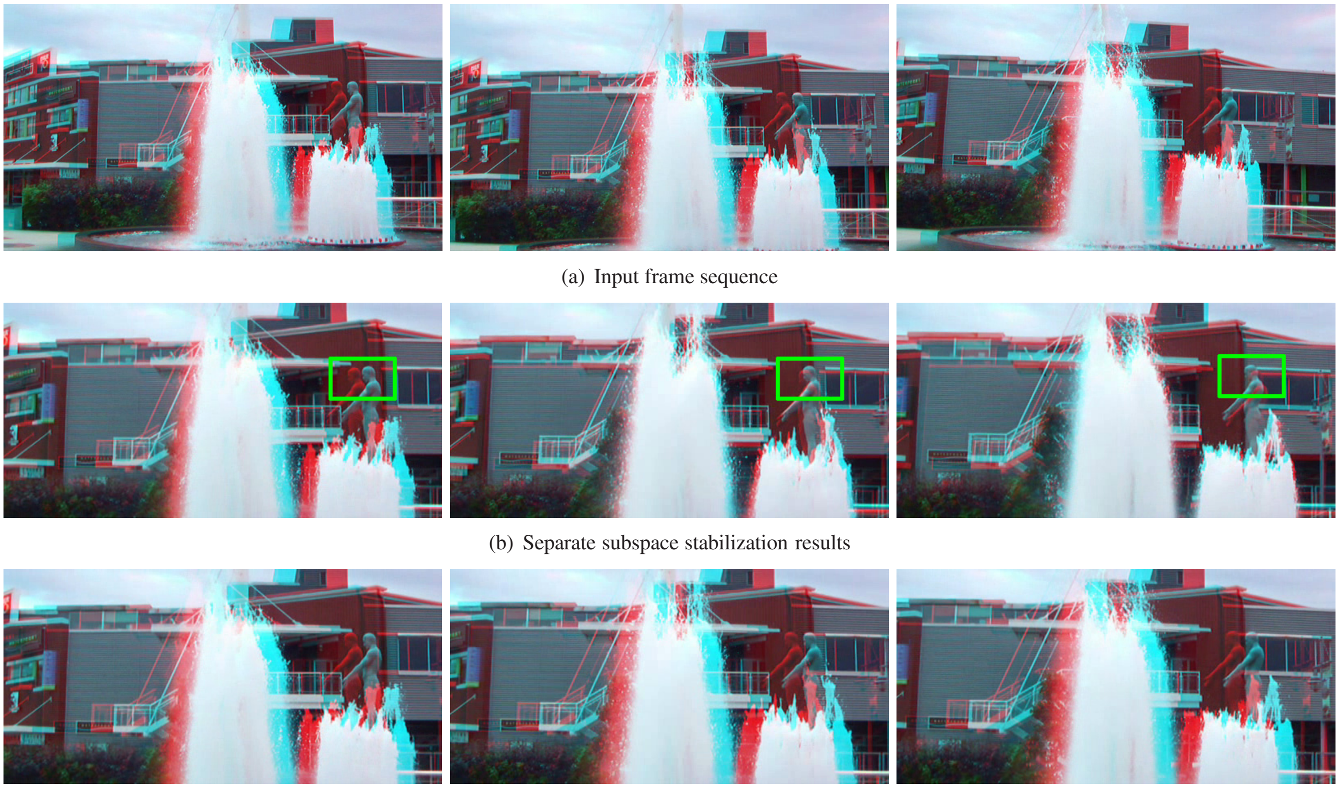
We tested our approach on a collection of 57 stereoscopic videos, ranging from 10 to 80 seconds, captured by a variety of stereoscopic cameras in many different scenes. We compared our joint subspace stabilization method against the separate subspace stabilization method of applying [15] independently to the left and right view. In both methods, we applied a low-pass filter with the same window size (60 frames for most of the examples). We did not compare our method against any 2D or 3D stabilization methods because that would almost amount to comparing [15] against 2D or 3D methods which is already covered in [15]. We refer readers to the video demo (in anaglyph) for more results[1].

We examined all the results from both methods. We consider a result unsuccessful if either the algorithm fails to process the video or the resulting stereoscopic video is uncomfortable to watch on a stereoscopic display. Out of the 57 videos, 55 of our results are successful. The two videos with which our method failed (shown in Figure 3) have strong motion blur and as a result there are very few long feature trajectories although we combine them from the two views. The separate subspace method only worked on 48 videos. In addition to the same two videos that failed our method, the separate subspace method failed on another 7 videos where the camera moves very quickly and there are not a sufficient number of long feature trajectories in at least one of the two views. Our method worked well with these videos as it can use feature trajectories from both views.

As discussed in Section 3.2.3, when there are abundant feature trajectories, the separate subspace method can smooth the two views consistently. This is evidenced by the fact that this method can successfully process 48 out of 57 videos that we tested on. When either view of an input video did not have enough long trajectories, this method failed and could only process a portion of the video that had enough feature trajectories. Figure 2 (b) shows another problem of the separate subspace method. Since the subspace constraint is applied to each view independently, the feature trajectories between two views can often be smoothed inconsistently, which sometimes leads to vertical disparities, as described in Section 3.2.3. Moreover, the horizontal disparities across frames are drastically changed. As shown in Figure 2 (b), the horizontal disparities quickly change over a short period of time, and even reverse the sign. This is incorrect as the disparities in the input sequence change little, as shown in Figure 2 (a). This is particularly problematic as it can cause a bad viewing experience. They should have been stabilized, as shown in our result (Figure 2 (c)).

We examined the vertical and horizontal disparities as well as their second derivatives in the output videos. In order to compute the second derivatives of the disparities, we need to track corresponding feature pairs. We first assemble the left and right video into a single frame sequence $\{F_L(1), F_R(1), F_R(2), F_L(2), ...\}$ and then apply the standard KLT algorithm to track feature trajectories from this sequence. We found that when the KLT algorithm is used to track features across the left and right view, it loses significantly more features than across two temporally consecutive frames. To address this problem, we estimate SIFT features [17] and use them to align the corresponding left and right video frame as SIFT features can be reliably matched between the left and right view. We then applied the KLT algorithm to the aligned frames. We found that this strategy can increase the number of feature trajectories across

[1] www.cs.pdx.edu/~fliu/project/joint-subspace



(a) Input frame sequence

(b) Separate subspace stabilization results

(c) Joint subspace stabilization results

Figure 2. Separate subspace vs. joint subspace. Stabilizing each view independently causes vertical disparities and drastically disturbs the horizontal disparities, as shown in (b).

the left and right view and thus be able to provide more feature trajectories for the robust statistics of the stabilization results. We further fit a fundamental matrix to the point correspondence and remove the matching points that are inconsistent with the fundamental matrix. We finally manually remove the remaining incorrect matches by examining the entire videos. The final outputs are a number of matched feature trajectories. We then compute the horizontal and vertical disparities and their second derivatives. *Note, these matched feature trajectories were not used in stabilization. They were only used to evaluate the stabilization results.*

We found that our method is able to produce stereoscopic video with smaller vertical disparities. In particular, the average vertical disparity by the separate subspace method is 0.69 pixels while the average vertical disparity by our method is 0.68 pixels. As a reference, all the input videos are resized to 640 × 360 and the average vertical disparity in the input videos is 0.68 pixels. We also computed the average of the top 1% of vertical disparities. The average vertical disparities of the separate subspace method and our method are 1.61 and 1.33 pixels. The average second derivatives of horizontal disparities of the input videos and the output videos from both methods are 0.20 pixels. Since the horizontal disparities are inversely proportional to the depth and the jitters of the camera motion in depth is often small, so the jitters in most feature trajectories are small except for scene points close to the camera. We therefore examine the top 1% of the second derivatives of horizontal disparities. We find that both the separate subspace and joint subspace method reduce it from 1.47 to 1.36 pixels. These statistics were run on the 48 videos that both our method and the subspace method work well with. That is, we did not include the 9 videos that failed the separate subspace method and had no stabilization results. These results show that our joint subspace method can successfully avoid introducing extra vertical disparities and smooth horizontal disparities. The reason that the separate subspace method does not produce significantly large vertical disparities or the second disparity derivatives is that these 48 videos have abundant long feature trajectories and thus can be handled by the separate subspace method, as discussed in Section 3.2.3.

Performance. Similar to the monocular subspace method [15], our method consists of 3 major parts in terms of computation: feature tracking, matrix factorization, and content-preserving warping. We performed all the experiments on a computer with a 3.00 GHz Intel Dual Core CPU and 4GB memory. The KLT tracker achieves 7 fps on each view when it is tuned to track roughly 500 feature points per frame on each view of an input video of size 640 × 360. The time for the moving matrix factorization part is negligi-

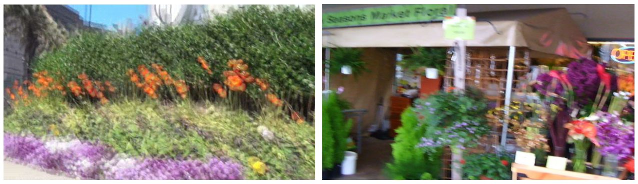


Figure 3. Failure examples. These two videos suffer from serious motion blur and failed our method.

ble. Our implementation of content-preserving warps [14] achieves 10 fps on each view. Overall, our implementation achieves 4 fps (namely 2 stereo frame pairs per second).

4.1. Discussion

Our method has a few important advantages. First, our method maintains the disparity constraints without explicit stereo correspondence. Second, our method in general inherits the advantages from [15]. In particular, it is able to produce high-quality stabilization results without computing a 3D reconstruction and it is robust, efficient, and allows a streaming implementation. Third, our method is more robust than [15] because it can make use of the feature trajectories in both views. However, our method still requires enough long feature trajectories for matrix factorization and has difficulties to handle videos with dominating scene motion, excessive shake, or strong motion blur, such as the examples shown in Figure 3. Like [15], our method does not explicitly handle rolling shutter problem. We can borrow methods from rolling shutter removal research [1, 8] to handle the rolling shutter artifacts.

5. Conclusion

This paper presented a joint subspace stabilization method for stereoscopic video. We proved that the subspace constraint is valid for stereoscopic video. Particularly, the left and right video share the same subspace. We showed that we can stabilize stereoscopic video without any explicit correspondence computation. Moreover, our method is more robust to the presence of short trajectories than the monocular subspace stabilization method. We validate our method on a variety of stereoscopic videos with different scene content and camera motion. Our experiments show that our method is able to achieve high-quality stereoscopic video stabilization in a robust and efficient way.

Acknowledgements. This work was supported in part by NSF grants IIS-1321119, CNS-1205746, and CNS-1218589.

References

[1] S. Baker, E. Bennett, S. B. Kang, and R. Szeliski. Removing rolling shutter wobble. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392 – 2399, 2010.

[2] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, M. Cohen, B. Curless, and S. B. Kang. Using photographs to enhance videos of a static scene. In *Eurographics Workshop on Rendering*, 2007.

[3] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *IEEE CVPR*, pages 609–614, Dec. 2001.

[4] Y. Caspi and M. Irani. Aligning non-overlapping sequences. *Int. J. Comput. Vision*, 48(1):39–51, 2002.

[5] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. *International Journal of Computer Vision*, 63(2):141–151, July 2005.

[6] M. L. Gleicher and F. Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5(1):1–28, 2008.

[7] A. Goldstein and R. Fattal. Video stabilization using epipolar geometry. *ACM Transactions on Graphics*, 2012.

[8] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Effective calibration free rolling shutter removal. *IEEE ICCP*, 2012.

[9] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *IEEE CVPR*, 2011.

[10] M. Irani. Multi-frame correspondence estimation using subspace constraints. *Int. J. Comput. Vision*, 48(1):39–51, 2002.

[11] M. Lambooij, W. IJsselsteijn, and I. Heynderickx. Visual discomfort in stereoscopic displays: a review. In *Stereoscopic Displays and Virtual Reality Systems XIV*, 2007.

[12] K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, and M. Ouhyoung. Video stabilization using robust feature trajectories. In *IEEE ICCV*, 2009.

[13] C.-W. Liu, T.-H. Huang, M.-H. Chang, K.-Y. Lee, C.-K. Liang, and Y.-Y. Chuang. 3d cinematography principles and their applications to stereoscopic media processing. In *ACM International Conference on Multimedia*, 2011.

[14] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. *ACM Transactions on Graphics*, 28(3):44:1–44:9, 2009.

[15] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. *ACM Transactions on Graphics*, 30(1):4:1–4:10, 2011.

[16] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun. Video stabilization with a depth camera. In *IEEE CVPR*, 2012.

[17] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. of Computer Vision*, 60(2), 2004.

[18] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 28(7), 2006.

[19] B. Mendiburu. *3D Movie Making: Stereoscopic Digital Cinema from Script to Screen*. Focal Press, 2009.

[20] C. Morimoto and R. Chellappa. Evaluation of image stabilization algorithms. In *DARPA Image Understanding Workshop*, pages 295–302, May 1997.

[21] J. Shi and C. Tomasi. Good features to track. In *IEEE CVPR*, pages 593–600, 1994.

[22] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala. Light field video stabilization. In *IEEE ICCV*, 2009.

[23] L. Wolf and A. Zomet. Wide baseline matching between unsynchronized video sequences. *Int. J. Comput. Vision*, 68(1):43–52, 2006.