Misalignment Correction for Depth Estimation using Stereoscopic 3-D Cameras

Michael Santoro, Ghassan AlRegib, Yucel Altunbasak

School of Electrical and Computer Engineering, Georgia Institute of Technology
Atlanta, GA 30332 USA
{msantoro,alregib,yucel}@gatech.edu

***Abstract*—This paper presents a misalignment correction method for reducing depth errors that result from camera shift. The proposed method uses a real-time motion estimation approach to correct for alignment errors between stereo cameras. Unlike existing methods in the literature, the natural disparity between stereo views is incorporated into a constrained motion estimation framework. The proposed method is shown to accurately estimate synthetic misalignments due to translation, rotation, scaling, and perspective transformation. In addition, real images taken from a stereo camera rig confirm that the proposed method is capable of significantly reducing misalignments due to camera yaw, pitch, and roll.**

I. Introduction

Stereo vision continues to grow in importance as more uses of 3-D technology emerge. Applications such as 3-D displays, cameras, robot navigation, and driver assistance systems take advantage of the natural disparity between two cameras to provide an estimate of 3-D parameters. However, the misalignment or mismatch between stereo cameras has been shown to affect 3-D depth estimates [1]. To correct for the misalignment or mismatch between left and right views, offline camera calibration is performed during the manufacturing process. However, in handheld, automotive, or robotic applications, cameras are often subject to environmental factors such as mechanical stresses, vibrations, or large temperature variations. These factors cause calibration parameters to drift, which can significantly affect the accuracy of the 3-D measurements [1][2]. Therefore, to maintain correspondence between left and right views, it is necessary to provide an online approach to misalignment correction.

It is important to distinguish misalignment correction from continuous self-calibration methods, which were discussed in [2]. While the goal of self-calibration is to determine the extrinsic and intrinsic camera parameters, misalignment correction aims to reduce the mismatch between camera views without regard to the camera parameters. To determine the mismatch between views, it is necessary to determine the transformation model that maps pixels from the left to right image or vice versa. Such a transformation model should accommodate for multiple sources of error such a baseline shift, rotation, pitch, yaw, non-parallel sensors, and lens distortion. Unfortunately, a model which includes all error sources is prohibitively expensive and not suitable for a real-time implementation.

To enable real-time misalignment correction, we make two assumptions regarding the calibration/alignment: 1) the cameras are initially calibrated to within a small error tolerance, and 2) the camera drift/misalignment results in small displacements between left and right images. The first assumption is necessary in order to determine an initial estimate of the horizontal disparity between left and right images. Otherwise, it is not possible to distinguish the alignment error from the natural horizontal disparity that exists between views. The second assumption allows us to use a block-based model to estimate pixel mappings between images, i.e., we approximate the different sources of alignment error as vertical and horizontal translations.

Other misalignment approaches have used an affine transformation model [3] and scale-invariant feature transform (SIFT) [4] to establish correspondence between left and right images. In [3], the authors remove the horizontal shift component from the affine model, which is equivalent to assuming that there is no camera misalignment due to horizontal shift, yaw, or non-parallel sensors. However, as shown in [1], depth estimates are most sensitive to camera motion which results in a relative yaw between views. In [4], the authors do not provide the details of their SIFT approach or experimental data, but such approaches generally require a large number of feature points and a method for outlier detection.

In the next sections, we examine how camera alignment errors affect depth estimates and propose a misalignment correction scheme based on constrained motion estimation using block matching. In section II, we summarize the works of [1][2], which present both theoretical and experimental results for the effects of alignment errors on depth estimates. In section III, we use synthetically-generated images to show that the our block-based motion estimation algorithm is capable of detecting various alignment errors. The proposed misalignment correction scheme is presented in section IV. Experimental results from a stereo camera rig are presented in section V, and the conclusions of the paper are presented in section VI.

II. Effect of Alignment Errors

In this section, we summarize the results of several works which provide both equations and experimental data relating alignment errors to 3-D depth errors [1][2].

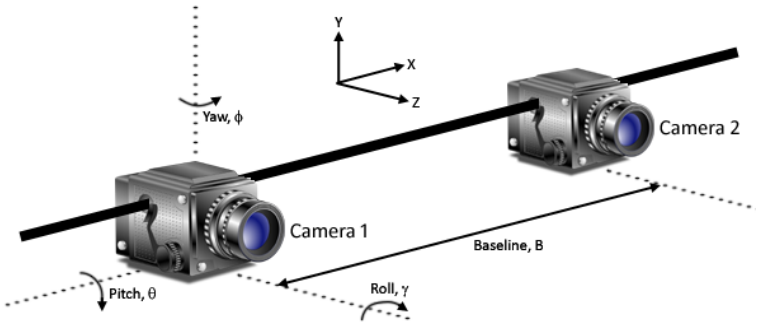

Fig. 1. Stereo camera rig showing error sources θ, γ, ϕ, and B.

The different types of alignment error for stereo cameras (with the exception of image sensor misalignment and scaling) are shown in Fig. 1.

As shown in Fig. 1, depth errors may result from baseline shift, rotation (roll), pitch, and yaw. In [1], the authors found that the most critical alignment parameters in order of importance were yaw, image sensor tilt, pitch, roll, baseline error, and focal length error. The equations relating the sensitivity of these parameters to depth are shown in Table I.

TABLE I
RELATIVE CHANGES IN DEPTH FOR DIFFERENT MISALIGNMENTS

Error Source	**Relative Change in Depth**
Yaw Error $\Delta\phi$	$\frac{\Delta Z}{\Delta\phi} \approx -\frac{Z^2}{B}(1+X_2)$
Sensor Tilt $\Delta\phi$	$\frac{\Delta Z}{\Delta\phi} \approx -\frac{X_2^2}{B}$
Pitch Error $\Delta\theta$	$\frac{\Delta Z}{\Delta\theta} \approx \frac{Z^2}{B}X_2Y_2$
Roll Error $\Delta\gamma$	$\frac{\Delta Z}{\Delta\gamma} \approx \frac{Z^2}{B}Y_2$
Baseline Error ΔB	$\frac{\Delta Z}{\Delta B} \approx -\frac{Z}{B}$
Focal Length Error Δf_2	$\frac{\Delta Z}{\Delta f_2} \approx -\frac{Z^2}{Bf_2^2}$

To generate the equations in Table I, it is assumed that Camera 1 is perfectly calibrated, and Camera 2 is perfectly calibrated except for the error source listed. The error sources in the left column of the table are the deviation from perfect alignment with respect to camera 2, where X_2, Y_2 represents the true 3-D coordinate of the object with respect to camera 2, B is the baseline distance, Z is the true absolute depth, and f_2 is the focal length. The error sources which contribute the most to alignment errors can be explained by the order of the depth Z in the right column; for example, the depth error increases quadratically for yaw misalignment but decreases linearly for baseline misalignment.

The results in Table I show that in order to minimize depth errors, the yaw, pitch, and sensor tilt cannot be ignored. These errors will introduce a prospective transformation since the 2-D displacement of objects will depend on their depth, and this type of transformation cannot handled by the affine model of [3].

III. Block-based Detection of Alignment Errors

In this section, we demonstrate that our block-based motion estimation algorithm is capable of detecting alignment errors with a high degree of accuracy.

A. Overview of Block-Based Algorithm

Our algorithm performs true motion estimation using quarter-pixel motion vectors (MVs), and it has achieved high rankings compared to other state-of-the-art algorithms in the Middlebury benchmark [5] (see "BlockOverlap" in online benchmark). The Middlebury database contains image sequences with various types of motion: translation, rotation, scaling, perspective, etc.

As part of the motion estimation process, our algorithm assigns a confidence value to each MV. The confidence value is used to characterize the validity of the MV, i.e., a higher confidence value indicates that the chosen MV is more likely to represent the true motion. More details can be found in [6].

Assigning a confidence value to each MV is important for real stereo images, where there may be occlusions, brightness variations, etc. In such cases, the MVs will have low confidence values and thus cannot be used to estimate the amount of misalignment. These MVs must be handled separately when estimating the depth; however, this topic is outside the scope of the paper. In order to use motion to estimate the horizontal disparity in section IV, we assume that misalignment correction is performed on scenes that do not contain independent object motion. Although this may somewhat limit the proposed misalignment correction approach, it is much less limiting than the need to perform manual calibration.

B. Detecting Alignment Errors

We now show that our block-based motion estimation algorithm is capable of estimating different types of misalignments on synthetic sequences. We use a global image translation to model baseline shift between cameras, rotation to model

roll, scaling to model focal length differences, and perspective transformation to model yaw, pitch, and sensor tilt.

The 300-frame Foreman sequence was used to test each type of misalignment error. For each frame of the sequence, we created a synthetic frame with the specified misalignment and used our motion estimation algorithm to estimate the amount of misalignment. For the translational, scaling, and rotational misalignments, we compare our estimated misalignment to that of [3]. Perspective misalignments were not considered in [3].

1) Translation: The translational misalignment was created by shifting all pixels in each frame by the specified amount. The motion was then estimated between the shifted and original frames. A comparison between our algorithm and [3] is shown in Table II for three different vertical shift amounts.

TABLE II
COMPARISON OF ERRORS FOR TRANSLATIONAL MISALIGNMENT.

Translation Params.	**Method of [3]**			**Proposed Method**		
True Shift	-10	24	48	-10	24	48
Estimated Shift	-10.17	24.55	46.97	-10.00	24.00	48.00
Mean Error	0.17	0.55	1.03	0.00	0.00	0.00
Standard Deviation	0.28	0.45	0.79	0.00	0.00	0.00

2) Rotation: The rotational misalignment was created by multiplying each pixel position in the original frame by a rotation matrix, i.e.,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\gamma & \sin\gamma & \frac{W}{2}(1-\cos\gamma) - \frac{H}{2}\sin\gamma \\ -\sin\gamma & \cos\gamma & \frac{W}{2}\sin\gamma + \frac{H}{2}(1-\cos\gamma) \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{1}$$

where $\{x', y'\}$ are the mapped pixel coordinates, $\{x, y\}$ are the original pixel coordinates, γ is the rotation angle, and W, H are the frame width and height, respectively.

For non-integer positions of x' and y' in (1), bilinear interpolation was used to estimate the pixel intensities. Next, the motion was estimated between the rotated and original frames. The angle of rotation was found from the MVs using the arc length approximation

$$\gamma_d = \frac{s}{r}, \tag{2}$$

where γ_d is the change in rotation angle, s is the displacement of a given pixel, and r is the distance from the pixel to the center of the frame. The rotation angle was estimated for each pixel $\{x', y'\}$ in the rotated frame. A comparison between our algorithm and [3] is shown in Table III.

3) Scaling: The scaling misalignment was created by multiplying each pixel position in the original frame by a scaling matrix, i.e.,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 & \frac{W}{2}(1-s_x) \\ 0 & s_y & \frac{H}{2}(1-s_y) \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{3}$$

TABLE III
COMPARISON OF ERRORS FOR ROTATIONAL MISALIGNMENT.

Rotation Params.	**Method of [3]**			**Proposed Method**		
True Angle	$-3°$	$4°$	$7°$	$-3°$	$4°$	$7°$
Estimated Angle	-2.95	3.95	5.66	-2.99	3.99	6.98
Mean Error	0.05	0.05	1.34	0.004	0.007	0.02
Standard Deviation	0.22	0.23	0.48	0.002	0.002	0.005

where $\{x', y'\}$ are the mapped pixel coordinates, $\{x, y\}$ are the original pixel coordinates, the scaling factor $s_x = s_y$, and W, H are the frame width and height, respectively.

For non-integer positions of x' and y' in (3), bilinear interpolation was used to estimate the pixel intensities. Next, the motion was estimated between the scaled and original frames. The scaling factor was computed by comparing the ratio of displacements in the original and scaled frames as follows:

$$s_x = s_y = \frac{\sqrt{(x - \frac{W}{2})^2 + (y - \frac{H}{2})^2}}{\sqrt{(j_x - \frac{W}{2})^2 + (i_y - \frac{H}{2})^2}}, \tag{4}$$

where $\{x, y\}$ are the pixel coordinates in the original frame, $\{i_y, j_x\}$ are the motion-compensated pixel coordinates in the scaled frame, and W, H are the frame width and height, respectively. The scaling factor was estimated for each pixel in the scaled frame. A comparison between our algorithm and [3] is shown in Table IV.

TABLE IV
COMPARISON OF ERRORS FOR SCALING MISALIGNMENT.

Scaling Params.	**Method of [3]**			**Proposed Method**		
True Scale	-3	4	5	-3	4	5
Estimated Scale	-2.83	3.66	3.75	-2.99	3.99	4.99
Mean Error	0.17	0.34	1.25	0.007	0.007	0.005
Standard Deviation	0.35	0.32	0.30	< 0.00	< 0.00	< 0.00

4) Perspective Transformation: A perspective transformation can be used to model yaw, pitch, and roll (rotation) misalignments. Since rotation was covered in section III-B2, we focus on yaw and pitch misalignments in this section.

To generate frames with yaw and pitch misalignments, it is necessary to represent 3-D coordinates in a 2-D plane such that an object has a smaller projection when it is far away from the center of projection and a larger projection when it is closer. The 3-D perspective mapping is given in (5),

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\gamma & -\cos\phi\sin\gamma + \sin\phi\sin\theta\cos\gamma & (\sin\phi\sin\gamma + \cos\phi\sin\theta\cos\gamma)f \\ \cos\theta\sin\gamma & \cos\phi\cos\gamma + \sin\phi\sin\theta\sin\gamma & (-\sin\phi\cos\gamma + \cos\phi\sin\theta\sin\gamma)f \\ -\dfrac{\tan\theta}{f\cos\phi} & \dfrac{\tan\phi}{f} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (5)$$

which is a modification of the 3-D rotation matrix in x-y-z convention for Euler angles of θ, γ, and ϕ (pitch, roll, and yaw, respectively). The general 3-D rotation matrix was modified in order to provide realistic values for the focal length, f, and the angle of view, α. The focal length and angle of view are related as follows:

$$\alpha = 2\arctan\frac{d}{2f}, \quad (6)$$

where d is the size of the image sensor in the horizontal or vertical direction. We chose $\alpha = 60^\circ$ and $d = 16mm$ for a more realistic depiction of the different 3-D rotations.

In order to display the 3-D coordinates in a 2-D plane, the coordinates of $\{x', y'\}$ in (5) must be normalized by the depth component, z'. The 2-D mapping equations are given as

$$\begin{aligned} x'' &= \frac{x'}{z'} - l_x \\ y'' &= \frac{y'}{z'} - l_y, \end{aligned} \quad (7)$$

where l_x and l_y are used to shift the projected frame such that the center of projection is located at $(W/2, H/2)$, and W and H are the frame width and height, respectively.

For non-integer positions of x'' and y'' in (7), bilinear interpolation was used to estimate the pixel intensities. Next, the motion was estimated between the projected and original frames. To determine the yaw, pitch, and roll parameters, it is first necessary to estimate the original mapping matrix used to map the original image coordinates onto the projected frame. We re-write the linear system of (5) as

$$\mathbf{x}' = \mathbf{H}\mathbf{x}, \quad (8)$$

where $\mathbf{x}'$ represents the projected frame coordinates, $\mathbf{H}$ is the modified rotation matrix, and $\mathbf{x}$ represents the original frame coordinates.

The modified rotation matrix can be found by determining the plane to plane homography $\hat{\mathbf{H}}$ between the original and projected frames, where $\hat{\mathbf{H}}$ is an estimate of $\mathbf{H}$. Given four sets of pixel coordinates in the projected frame and the corresponding four sets of motion-compensated pixel coordinates in the original frame[1], an estimate of $\hat{\mathbf{H}}$ can be determined. More details can be found in [7]. Following the estimation of $\hat{\mathbf{H}}$, the angles of the yaw, pitch, and roll can be determined by equating the entries of $\hat{\mathbf{H}}$ with the original $\mathbf{H}$ given in (5).

For small angles ($< 10^\circ$) of yaw, pitch, and roll, pixels near the center of projection will undergo very small displacements in the projected frame. As a result, the estimate of $\hat{\mathbf{H}}$ becomes ill-conditioned near the center of projection. To overcome this limitation, we manually selected four pixel locations near the image corners in the projected frame and determined their corresponding motion-compensated pixels in the original frame. An illustration of the pixel selection is shown in Fig. 2.

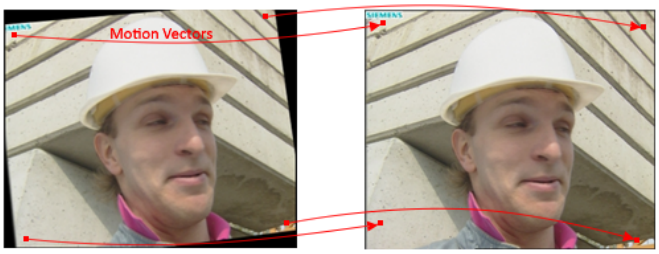

Fig. 2. Pixel selection in original and projected frames.

Since perspective transformations were not handled by the method of [3], only the results of our algorithm are shown in Table V.

TABLE V
ERRORS FOR YAW ϕ AND PITCH θ MISALIGNMENTS.

Params.	ϕ	ϕ	ϕ	θ	θ	θ
	$-\mathbf{3}^\circ$	$\mathbf{4}^\circ$	$\mathbf{7}^\circ$	$-\mathbf{3}^\circ$	$\mathbf{4}^\circ$	$\mathbf{7}^\circ$
Estimated Angle	-2.30	4.68	7.31	-2.93	4.15	7.06
Mean Error	0.72	0.68	0.31	0.07	0.15	0.06
Standard Deviation	0.52	0.22	0.19	0.10	0.08	0.30

The larger errors for the small angles in Table V are mainly due to the difficulty in estimating $\hat{\mathbf{H}}$ for small angles. In order to estimate small angles accurately, very high precision motion vectors are needed; however, our algorithm only computes quarter-pixel accurate MVs.

Since real stereo cameras generally suffer from multiple misalignments, we also generated a projected frame with all three types of misalignments – yaw, pitch, and roll. As described in the previous experiments, the matrix $\hat{\mathbf{H}}$ was estimated using four sets of pixel coordinates in the original and projected frame. The results for the three simultaneous misalignments are shown in Table VI. The results for the synthetically-generated sequences of Tables II-VI show that our algorithm is capable of detecting different types of misalignments with high accuracy.

IV. Proposed Misalignment Correction

In this section, we first discuss the constrained motion estimation model that will be used to preserve the stereo disparity

[1]The motion-compensated pixel coordinates must be shifted by l_x and l_y prior to estimating $\hat{\mathbf{H}}$

TABLE VI
ERRORS FOR YAW ϕ, PITCH θ, AND ROLL γ MISALIGNMENTS.

Params.	ϕ	θ	γ
	7°	**7°**	**7°**
Estimated Angle	7.14	7.17	6.98
Mean Error	0.14	0.17	0.18
Standard Deviation	0.46	0.27	0.48

and estimate alignment errors. The constrained motion model will then be demonstrated on real images from a stereo camera rig.

The constrained motion model was introduced by Farsiu et al. [8] in the context of super-resolution. In [8], the authors introduced the Jacobi identity for three consecutive frames i, j, and k as follows:

$$V_{i,k} = V_{i,j} + V_{j,k}, \tag{9}$$

where $V_{i,k}$ is the motion vector between frames i and k. Equation (9) states that motion between frames i and k must be the composition of the motion between frames i, j and j, k. We note that (9) assumes a translational motion model, which was demonstrated to be an effective means of estimating alignment errors in section III-B.

We can extend (9) to misalignment correction by decomposing the misalignment problem into two steps: 1) determine the initial horizontal shifts $\underline{X}_D$ between right and left views[2], and 2) determine the horizontal and vertical shifts between right and left views resulting from misalignment, $\underline{X}_M$. The vectors $\underline{X}_D$ and $\underline{X}_M$ represent a lexicographic ordering for the estimated motion with respect to the right view. The misalignment error can then be written as

$$\underline{E}_M = \underline{X}_M - \underline{X}_D, \tag{10}$$

where $\underline{E}_M$ is a vector of misalignment errors for each pixel in the right view. We now examine each of the above steps in detail.

Step 1): To determine the initial horizontal shifts between right and left views, we assume that the two cameras have been pre-calibrated. It is important that the initial calibration is accurate since it will serve as a baseline for the misalignment correction algorithm. Since calibration is already performed during the stereo camera manufacturing process, no further overhead is incurred by requiring an initial calibration. The initial calibration produces the intrinsic and extrinsic cameras parameters which are used to remove lens distortion and provide image rectification. Following distortion removal and rectification, the left and right image planes will be vertically aligned. Next, the baseline horizontal shifts $\underline{X}_D$ can be determined by estimating the horizontal motion between right and left views. It is only necessary to determine $\underline{X}_D$ once since it can be stored in memory for future access.

Step 2): If the cameras remain calibrated, it is not necessary to perform the second step; i.e., $\underline{E}_M = \underline{0}$. However, real cameras will shift due to environmental factors that introduce horizontal and/or vertical displacements between the right and left views. Given two new images taken from misaligned cameras, distortion removal and rectification is first performed using the initial calibration parameters. However, since the initial calibration data no longer applies to the new images, there will exist both horizontal and vertical displacements between rectified views. The motion between rectified views is represented by $\underline{X}_M$. The alignment error can then be computed using (10).

V. RESULTS FROM STEREO CAMERA RIG

To demonstrate the proposed misalignment correction method from section IV, we used seven stereo image pairs taken from a camera rig similar to that of Fig. 1. Only alignment errors due to yaw, pitch, and roll were considered, i.e., it is assumed that the baseline B does not change between cameras, and the differences in focal length are sufficiently small.

For the first stereo image pair, which will be used as the baseline for determining $\underline{X}_D$, we roughly aligned the left and right cameras such that the optical axes were parallel to one another. To perform the calibration of the two cameras, we used the OpenCV implementation of [9][10]. The method of [10] is a manual calibration method which requires multiple images of a chessboard pattern in different orientations. We performed three separate calibrations, acquiring 30 chessboard images in each calibration. The intrinsic and extrinsic camera parameters for the three calibrations were averaged and taken to be the initial parameters. The relative yaw, pitch, and roll (ϕ, θ, γ) from the rotation matrix were found to be

$$\begin{bmatrix} \phi \\ \theta \\ \gamma \end{bmatrix} = \begin{bmatrix} 3.93^\circ \\ 0.63^\circ \\ -0.57^\circ \end{bmatrix}. \tag{11}$$

Following the determination of the calibration parameters, we again used the OpenCV implementation of [9][10] to perform image rectification and distortion removal. The rectified images for the left and right views are shown in Fig. 3(a) and Fig. 3(b), respectively.

To generate the remaining six stereo image pairs, the yaw, pitch, or roll of the right camera was modified while the position of the left camera remained unchanged. After each adjustment to the right camera, three calibrations were performed to determine the relative rotation matrix. The relative yaw, pitch, and roll for the six stereo image pairs are given in Table VII. For Experiments #2 and #3 in Table VII, the yaw ϕ of the right camera was modified in both directions while the left camera remained unchanged. Similarly, the pitch θ was modified in both directions in Experiments #4 and #5, and the

[2]Either the left or right view may be chosen as the reference frame for motion estimation.

(a) Left rectified image. (b) Right rectified image.

Fig. 3. Left and right rectified images for initial calibration parameters.

TABLE VII
RELATIVE YAW ϕ, PITCH θ, AND ROLL γ FROM CALIBRATION DATA.

Experiment #	ϕ	θ	γ
2	$-2.03°$	$1.76°$	$-0.54°$
3	$6.46°$	$1.73°$	$-0.40°$
4	$3.10°$	$-2.70°$	$-0.03°$
5	$2.24°$	$2.21°$	$-0.17°$
6	$3.61°$	$2.10°$	$1.28°$
7	$1.42°$	$2.35°$	$-1.89°$

roll γ was modified in both directions in Experiments #6 and #7. Since our camera rig was imperfect, it did not allow us to completely isolate each angle individually (e.g., changing the pitch angle resulted in small deviations in yaw and roll angles).

For each of the experiments shown in Table VII, the captured left and right images were rectified using the initial calibration parameters. Next, the motion was estimated between the right and left images to determine $\underline{X}_M$. Since $\underline{X}_D$ was previously determined from the initial calibration parameters, the translational alignment error $\underline{E}_M$ was found from (10).

Using the alignment error $\underline{E}_M$, the pixel positions in the right image were motion-compensated to determine if alignment errors had been reduced. To estimate the alignment errors, we compare motion-compensated pixel positions in the right image to the initial right image (the baseline). As in section III-B4, the point mapping method was used determine the matrix that maps four sets of points in the initial right image to the motion-compensated points. These points were manually selected in the initial images, and the corresponding motion-compensated points were determined. We only considered pixels whose MVs had high confidence values (see section III-A). The angular errors for the yaw, pitch, and roll between the initial pixels and motion-compensated pixels are given in Table VIII. As shown in Table VIII, the proposed misalignment correction method significantly reduces the misalignment due to yaw and roll and reduces the pitch misalignment to within $1.5°$.

TABLE VIII
ANGULAR DIFFERENCES IN YAW ϕ, PITCH θ, AND ROLL γ.

Experiment #	ϕ	θ	γ
2	$< 0.01°$	$-0.55°$	$< 0.01°$
3	$< 0.01°$	$1.48°$	$< 0.01°$
4	$< 0.01°$	$1.25°$	$< 0.01°$
5	$< 0.01°$	$0.20°$	$< 0.01°$
6	$< 0.01°$	$-0.55°$	$< 0.01°$
7	$< 0.01°$	$-1.37°$	$< 0.01°$

VI. CONCLUSION

The misalignment correction approach proposed in this paper uses a block-matching-based constrained motion framework to preserve the natural disparity between stereo views while correcting for vertical and horizontal misalignments that result from camera shift. The proposed approach requires only an initial calibration to determine the natural disparity, and the vertical and horizontal misalignments are estimated using real-time block-based motion estimation. The ability of the block-based motion estimation algorithm to correct for misalignments was first demonstrated using synthetic sequences with translation, rotation, scaling, and perspective transformations in section III. In section V, it was shown that proposed algorithm is capable of reducing all types of misalignments to within $1.5°$. For the yaw angle misalignment, which has the greatest effect on depth estimates, the proposed algorithm reduces the misalignment error to less than $0.01°$.

REFERENCES

[1] W. Zhao and N. Nandhakumar, "Effects of camera alignment errors on stereoscopic depth estimates," *Pattern Recognition*, vol. 29, no. 12, pp. 2115 – 2126, 1996.

[2] T. Dang, C. Hoffmann, and C. Stiller, "Continuous stereo self-calibration by camera parameter tracking," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1536 –1550, July 2009.

[3] I. E. Pekkucuksen, A. U. Batur, and B. Zhang, "A real-time misalignment correction algorithm for stereoscopic 3d cameras." SPIE, 2012.

[4] F. Zilly, P. Eisert, and P. Kauff, "Real-time analysis and correction of stereoscopic hdtv sequences." CVMP, 2009.

[5] M. Santoro, G. AlRegib, and Y. Altunbasak, "Motion estimation using block overlap minimization," in *Submitted to International Workshop on Multimedia Signal Processing (MMSP)*, 2012.

[6] ——, "Block-overlap-based validity metric for hybrid de-interlacing," in *Submitted to International Conference on Image Processing (ICIP)*, 2012.

[7] A. Criminisi, I. Reid, and A. Zisserman, "A plane measuring device," *Image and Vision Computing*, vol. 17, no. 8, pp. 625 – 634, 1999.

[8] S. Farsiu, M. Elad, and P. Milanfar, "Constrained, globally optimal, multi-frame motion estimation," in *IEEE Workshop on Statistical Signal Processing*, 2005, pp. 1396–1401.

[9] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330 – 1334, Nov. 2000.

[10] J. Y. Bouguet, "Camera calibration toolbox for Matlab," 2008. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/.