Ray Divergence-Based Bundle Adjustment Conditioning for Multi-View Stereo

Mauricio Hess-Flores[1], Daniel Knoblauch[2], Mark A. Duchaineau[3], Kenneth I. Joy[1], and Falko Kuester[2]

[1] Institute for Data Analysis and Visualization, University of California, Davis, USA
{mhessf,kijoy}@ucdavis.edu
[2] University of California, San Diego, USA
{dknoblau,fkuester}@ucsd.edu
[3] Lawrence Livermore National Laboratory, Livermore, CA, USA
{duchaine}@llnl.gov

Abstract. An algorithm that shows how ray divergence in multi-view stereo scene reconstruction can be used towards improving bundle adjustment weighting and conditioning is presented. Starting with a set of feature tracks, ray divergence when attempting to compute scene structure for each track is first obtained. Assuming accurate feature matching, ray divergence reveals mainly camera parameter estimation inaccuracies. Due to its smooth variation across neighboring feature tracks, from its histogram a set of weights can be computed that can be used in bundle adjustment to improve its convergence properties. It is proven that this novel weighting scheme results in lower reprojection errors and faster processing times than others such as image feature covariances, making it very suitable in general for applications involving multi-view pose and structure estimation.

Key words: Multi-view reconstruction, ray divergence, weighted bundle adjustment, confidence ellipsoids, image feature covariances

1 Introduction

During the past years there has been a surge in the amount of work dealing with multi-view reconstruction of scenes, for industry and in many other modern applications. State-of-the-art algorithms [1] provide very accurate matching, camera poses and scene structure, based on sparse features such as those obtained with the SIFT [2] or related algorithms. These recent algorithms are capable of reconstructing large scenes from even unstructured image sets, obtained for example from the Internet. In such scenarios, camera parameters such as location, orientation and intrinsics may be available or accurately estimated for some of the cameras but not all. This could also be the case even in structured sets of images acquired with the same camera. Because of this reason, despite very accurate feature matching, the accuracy of a multi-view reconstruction still relies on accurate camera parameter calibrations. This creates a great need to identify

where and why errors are present in these parameters, specifically without the need to know ground-truth, since this is not always available. In the absence of ground-truth data, multi-view algorithms usually resort to *bundle adjustment* [3] to reduce reprojection error, which is the most meaningful geometric measure of accuracy in the lack of any ground-truth. However, this can be an expensive element in a scene reconstruction pipeline for high numbers of scene points and cameras, despite recent and efficient sparse implementations such as *SBA* [3], and must be used wisely. Furthermore, it requires a good enough starting point close to the global minimum for convergence.

Our main goal in this paper is to show how simple *ray divergence* when attempting scene reconstruction is an inexpensive yet powerful tool that can aid in bundle adjustment convergence for multi-view stereo. Ray divergence is defined as the shortest distance between rays emanating from each respective camera center and through each pixel position of a given feature track, as will be further described in Subsection 2.1. Our work is partially inspired by the recent algorithm of Knoblauch et al. [4], which measures per-correspondence ray divergence when attempting scene reconstruction from a set of initial unconstrained dense correspondences and then decomposes the total error map into errors related to camera parameters and correspondence errors. To the knowledge of the authors there had been no other previous work on such an error factorization without using ground truth knowledge. The ray divergence metric relies on the input feature matches being unconstrained, which is what allows for measuring geometric errors. Using matches generated for example through epipolar geometry-based guided matching would yield no reconstruction error, since these are generated such that they lie on the same epipolar plane with the point they represent in 3D space.

As far as other previous work on camera parameter error analysis, it has been done for the most part with respect to ground-truth values, such as the methodology to test the accuracy of camera pose estimation presented in Rodehorst et al. [5]. The work in Zhao et al. [6] deals with how extrinsic and intrinsic calibration inaccuracies contribute towards depth estimate errors, but for the specific case of a stereo camera pair with a known baseline and other relative positioning information. Benchmarks also exist for reconstruction accuracy [7], though the analysis is done versus ground-truth values, and our algorithm is based on ray divergence rather than the accuracy of exact recovered positions.

In our algorithm, we compute ray divergence per feature track and use it as a joint measure of all camera parameter inaccuracies, without the need for ground-truth knowledge and prior to actually computing the 3D structure. We start out similarly to Knoblauch et al. [4], first computing ray divergences for all available feature matches but with the important difference that we use robust SIFT features instead of dense correspondences, keeping in mind that such feature matches are also unconstrained and therefore it is possible to extract a geometric error unlike in guided matching. We also assume that these feature matches are highly accurate, and this is generally true since sparse SIFT matches are less prone to mismatching due to occlusions, repetitive patterns and texture-

less regions than dense correspondences. To further ensure that we have very accurate matches, epipolar geometry-based RANSAC outlier removal [8] is applied prior to computing ray divergences. This in turn allows us to assume that the total ray divergence error corresponds only to camera-related inaccuracies, such that we can avoid the error decomposition in Knoblauch et al. to obtain camera parameter errors.

As will be discussed, the validity of ray divergence as a measure of camera parameter uncertainty can be proven, since it correlates well with Beder et al.'s confidence ellipsoid roundness measure for computed 3D scene points [9] in the case when image feature covariances are set to identity. Furthermore, since ray divergence encodes camera inaccuracy information, we show how it can be used in weighted bundle adjustment to improve its convergence properties. It is shown how this scheme outperforms weighting based on more-expensive image feature covariance metrics [10, 11] or Beder et al.'s confidence measure. The entire procedure is first derived for the two-view case, but later shown how this can easily be extended to multiple views. In summary, our algorithm presents a very practical and inexpensive way to measure camera parameter uncertainty in the absence of ground-truth information and use that uncertainty to improve bundle adjustment conditioning. The entire procedure will be described in detail in Section 2, followed by experimental results (Section 3) and conclusions (Section 4).

2 Proposed Algorithm

Our analysis will begin with the two-view case, where it is first shown in Subsection 2.1 how to compute ray divergence, and in Subsection 2.2 how to set up weighted bundle adjustment based on ray divergence values. The extension to multiple views will be outlined in Subsection 2.3.

2.1 Two-View Ray Divergence Calculation

The first step in our algorithm is to compute ray divergence per feature match, similarly to Knoblauch et al. [4], except we start with sparse SIFT features [2] instead of dense correspondences. In the case of *perfect* feature matches, camera intrinsics and extrinsics and no radial distortion, rays starting from each camera center and through the respective image plane feature location should intersect at an exact position in 3D space, but due to any inaccuracies this generally will not occur. We define ray divergence as the shortest distance between such rays, as depicted on the left image of Fig. 1. As mentioned earlier, due to accurate feature matching ray divergence is assumed to correspond entirely to camera parameter inaccuracies, which turns out to be a good approximation even if there are small matching errors. Matches will never be perfect in reality, but we filter bad matches through RANSAC on the epipolar geometry, using a $3.84\sigma^2$ inlier threshold on Sampson error [8].

Ray directions D_i for the two cameras are calculated per Eq. 1, with x_i and y_i being the pixel coordinates in each image. The absolute orientation R_i

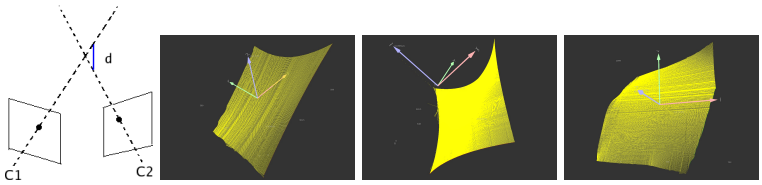

Fig. 1. Concept of ray divergence d (*left*), and sample dense camera parameter error maps for image pairs from different datasets, to depict their smooth variation.

and position C_i for each of the two cameras is computed by factorizing the *essential matrix*, which can be computed from feature matches using *N-point* algorithms [8]. The cameras' intrinsic parameters (such as focal length and principal point, with no pixel skew) are assumed to be at least roughly known in order to create each 3×3 matrix K_i.

$$D_i = R_i * K_i^{-1} * \begin{pmatrix} x_i & y_i & 1 \end{pmatrix}^T \ . \tag{1}$$

Given the camera center locations C_i, the shortest distance between the two rays corresponds to the Euclidean distance between the nearest distance points P_i on each ray as shown in Eq. 2, with t_i defining the distance to move along each ray. Finally, the ray divergence d can be obtained from $d_i = |P_1 - P_2|^2$. This error comprises any inaccuracies with the camera poses, intrinsics or radial distortion, and influences scene reconstruction in a global, smooth manner [4].

$$P_i = C_i + t_i * D_i \ . \tag{2}$$

The ray divergence d is then computed for all available feature matches. In Knoblauch et at. [4], the resulting set of divergences corresponds to the total reconstruction error which is a function of both feature matching errors and camera-related errors, but as mentioned earlier we assume here that the entire error corresponds to the cameras. Therefore, we can say that ray divergence d_i for a given feature match is a function of relative rotation between the two cameras R_{rel}, relative translation T_{rel}, intrinsic parameters for the two cameras K_1 and K_2, and radial distortion, which we'll represent as distorted pixel coordinates (x_{ri}, y_{ri}), such that $d_i = f(R_{rel}, T_{rel}, K_1, K_2, x_{ri}, y_{ri})$.

To show how errors in these parameters affect ray divergence in a global, smooth manner, and for visualization purposes since it becomes more difficult to show using sparse matches, we computed dense correspondences through a standard optical flow method to obtain a total ray divergence map for a few test sequences. Each was factorized into camera-parameter error maps, modelled as smooth B-spline surfaces, and correspondence error maps (remaining high-frequency components). The resulting camera-parameter error maps are shown in Fig. 1. Starting with sparse features, a smooth but sparse set of surface points is obtained as shown in Fig. 2 for the *Palmdale* dataset, which shows grayscale-coded ray divergence values for all available matches. In general, it has been

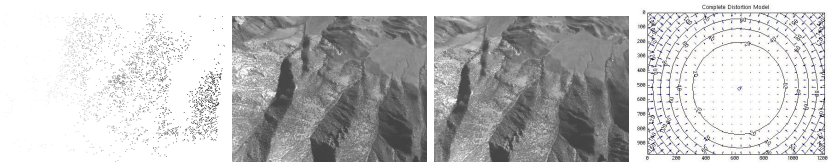

Fig. 2. Ray divergences (*left*) for the set of matches from a pair of *Palmdale* dataset images (*middle*), displayed such that lighter colors indicate higher divergences. The true radial distortion map for the used camera, in pixels, is also displayed (*right*).

observed that the highest divergences tend to occur towards the edges of images (as seen in Fig. 2, where most matches are on the left-hand side of the images) in part because of radial distortion, and it becomes clear that we want such matches to have less of an influence in bundle adjustment because of their higher ray divergence, as discussed further in Subsection 2.2.

2.2 Bundle Adjustment Weighting with Ray Divergences

Now that ray divergences have been computed, and assuming that these are a function mainly of camera parameter inaccuracies, it will be shown how these values can be used as input weights to bundle adjustment in order to improve its convergence properties. However, one further step before applying bundle adjustment is to obtain initial estimates for the scene's structure. We use Lindstrom's triangulation algorithm [12] due to its superior accuracy and speed with respect to standard linear triangulation [8].

Weighted Bundle Adjustment. The objective of bundle adjustment is to adjust pose and structure estimates in such a way that the total reprojection error of the 3D points with respect to their corresponding 2D feature track positions in each camera is minimized [8]. The cost function which is traditionally minimized can be expressed as the sum of squares of reprojection errors between each 3D point and the feature matches which yielded it, as shown in Eq. 3 for the general case of N 3D points seen in M cameras.

$$min(a_j, b_i) \sum_{i=1}^{N} \sum_{j=1}^{M} v_{ij} (d(Q(a_j, b_i), x_{ij}))^2 \ . \tag{3}$$

Here, x_{ij} is the position of the i_{th} feature on image j. The binary variable v_{ij} equals '1' if point i is visible in image j ('0' otherwise). The vectors a_j and b_i parametrize each camera j and 3D point i, respectively, with $Q(a_j, b_i)$ as the reprojection of point i on image j. Finally, d^2 is the Euclidean distance in each image between each original correspondence and its associated reprojection. This minimization involves a total of $3N + 11M$ parameters, and can be achieved using the Levenberg-Marquardt algorithm. The *SBA* implementation [3] was used,

since it exploits the sparse block structure of the normal equations solved at each iteration to greatly speed up the process.

The Levenberg-Marquardt algorithm is based on solving the *augmented normal equations* at each iteration. In *weighted* bundle adjustment, each input feature is weighted differently with the objective of improving convergence by giving less weight to those features that are more likely to be inaccurate. In practice, these weights are implemented as covariances. The normal equations have the form shown in Eq. 4, but when using weighted bundle adjustment, the equations change to the form shown in Eq. 5, where Σ corresponds to a block-diagonal matrix consisting of 2×2 covariance matrices for each input feature, J is the parameter Jacobian matrix, δ_p the parameter update step, μ the damping term and ϵ the error vector.

$$(J^T J + \mu I)\delta_p = J^T \epsilon \; . \tag{4}$$

$$(J^T \Sigma_x^{-1} J + \mu I)\delta_p = J^T \Sigma_x^{-1} \epsilon \; . \tag{5}$$

Comparison with Reconstructed Point Confidence Ellipsoid Roundness. Before proceeding, we wish to analyze the validity of ray divergence as a measure of camera errors, such that it can aid in bundle adjustment. Beder et al. [9] present an algorithm to determine the best initial pair for a multi-view reconstruction. Their analysis is based on computing a confidence ellipsoid for each computed 3D scene point X, such that its roundness measures the quality of each obtained point. For two views, the covariance matrices of image feature matches x' and x" are given by C' and C" respectively [10, 11]. Then, the covariance matrix C_{XX} of the distribution of the scene point coordinates X is proportional to the upper left 4×4 submatrix $N^{-1}_{1:4,1:4}$ for the inverse of the 5×5 matrix N given by Eq. 6. The A and B matrices encode information related to the projection matrices for the two cameras, the image coordinates of the feature match yielding the scene point, and the 3D point coordinates.

$$N = \begin{pmatrix} A^T \left(B \begin{pmatrix} \text{C'} & 0 \\ 0 & \text{C"} \end{pmatrix} B^T \right)^{-1} A & X \\ X^T & 0 \end{pmatrix} \; . \tag{6}$$

Now, if the homogeneous vector $X = [X_0^T, X_h]^T$ is normalized to Euclidean coordinates, the covariance matrix of the distribution of the Euclidean coordinates is given by Eq. 7, where J_e corresponds to the Jacobian of a division of X_0 by X_h.

$$C^{(e)} = J_e C_{XX} J_e^T \; . \tag{7}$$

Finally, if we perform the singular value decomposition of the matrix $C^{(e)}$, the roundness R of the confidence ellipsoid is obtained as the square root of the quotient of the smallest singular value λ_3 and the largest singular value λ_1, per $R = \sqrt{\frac{\lambda_3}{\lambda_1}}$. The value of R lies between 0 and 1, and only depends on the relative geometry of the two poses, the feature positions and the 3D point; radial

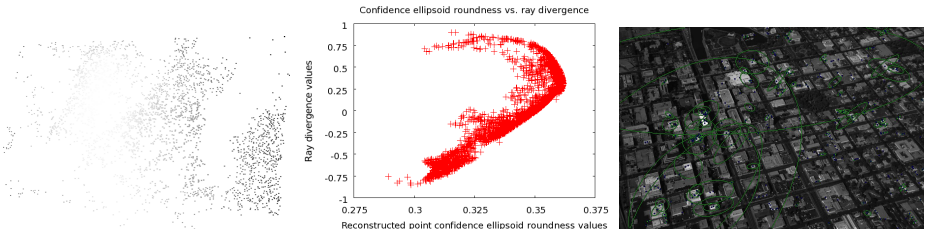

Fig. 3. Reconstructed point confidence ellipsoid roundness values using identity image feature covariances (*left*) for the set of matches from a pair of *Palmdale* dataset images, where lighter colors indicate lower roundness values. The middle image shows its correlation with ray divergence. The right image displays Zeisl's covariance metric values [11] for SIFT features in a *Stockton* image as green ellipsoids.

distortion is not modelled.

Something very important to note here is that image feature covariances [10, 11] are defined completely by the intensity variations in local neighborhoods and thus may look rather random to visual inspection, with no clear pattern as the image is traversed, as seen on the right in Fig. 3. On the other hand, the surface of ray divergences has a much smoother shape, which is a function of all camera parameter inaccuracies. So if we filter out all features that have high image covariances, matches obtained between remaining 'good' features are still bound to the information ray divergence provides, in order to know if they're overall good or bad matches for reconstruction purposes. This is the power of using ray divergence to weight bundle adjustment, since it provides information beyond just the feature matching uncertainty. For example, two *perfect* matches could still yield a non-zero ray divergence due to camera inaccuracies. Therefore, using ray divergence or even the values provided by Beder et al.'s metric [9], though more expensive to compute and not inclusive of radial distortion, provide a stronger constraint towards weighting bundle adjustment than image-based covariances [10, 11]. The right side of Fig. 3 shows the result of applying Zeisl's image covariance metric [11] on a select group of SIFT features, displayed as ellipses with size proportional to covariance values. The left side shows the smooth transitions in values for Beder et al.'s confidence ellipsoid roundness [9] using identity image feature covariances, and the middle shows its correlation with ray divergence. Though it is not an exact correlation because of differences near the edges of images, where the behavior is slightly different, the bulk of points show a very good correlation (a coefficient of 0.93 for the main linear part of this particular plot), such that higher divergences, in absolute value, exhibit lower roundness.

Gaussian Weighting. A close look at a ray divergence histogram reveals a smooth curve, typically reaching a maximum near zero. If we assume that the probability $p(d)$ that a given feature match exhibits a ray divergence d is given

by Eq. 8, where μ_d corresponds to the mean ray divergence and σ_d to its standard deviation for a given two-view set of feature matches, we can essentially assume that ray divergence histogram values follow a Gaussian probability density function (pdf) and use these values as weights for bundle adjustment. The average and standard deviation are computed directly from the ray divergences for the available set of feature matches. Since these weights must be input as 2×2 covariance matrices, we assume an isotropic probability distribution and set the diagonal elements with equal pdf-based values, while setting the remaining two elements to zero. It is very important to note that we want to penalize low pdf values since these correspond typically to higher divergences. Therefore, we 'invert' the pdf values and place this number along the diagonal; their original values are obtained again later from matrix inversion while solving the augmented weighted normal equations. This results in higher covariances providing lower weights.

$$p(d) = \frac{1}{2\pi\sigma_d^2} e^{\frac{|d-\mu_d|^2}{2\sigma_d^2}} \quad . \tag{8}$$

The advantages of using Gaussian values as weights is that positive weights are always obtained, no matter what the divergence values are or if they show zero-crossings. The area under the computed Gaussian curve is always unity, by definition, and this is helpful towards mathematical stability since very large variations between the smallest and largest assigned weights is not typical. Also, exponentials are much cheaper to compute than for example a singular value decomposition, as needed in Beder et al.'s algorithm [9]. Finally, ray divergence transitions are smooth such that high ray divergences should be assigned higher covariances than lower ones.

2.3 Extension to Multiple Views

The extension to multiple views is rather simple, and is based directly on the two-view case. In a sequential multi-view pipeline, since covariances have to be specified as 2×2 covariance matrices for each feature of a given feature track, for each feature in a new image we simply assign the Gaussian-based weight corresponding to the ray divergence for the feature's match to the prior image. Average and standard deviation are obtained from the set of pairwise matches between the two most recent images, in order to compute the pdf prior to computing each individual weight. Covariances for the features in the very first image can be initialized to identity, or by computing them from images [10, 11] for better initial accuracy. This way of chaining pairwise consecutive estimates works well no matter what the number of frames as long as pairwise ray divergence estimates are well-conditioned, which can usually be achieved through a prior *frame decimation* [13]. An analysis of this baseline effect on divergences is shown in Section 3. For non-sequential cases, the average of all ray divergence values for all matches to a given feature could potentially be used, though we have yet to test this case.

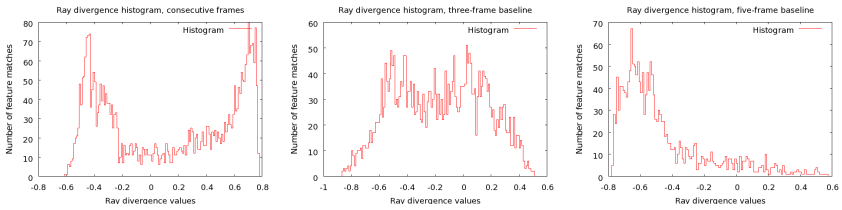

Fig. 4. Ray divergence histograms at increasing baselines (*left to right*), for pairwise frames from the *Stockton* dataset.

3 Results

The algorithm was tested on real scenes such as *Stockton*, *Palmdale*, *castle-P19* [7] and *Medusa* [14], as well as synthetic scenes such as *Megascene1* and *Coneland*. All tests were conducted on a single-core *Intel Xeon* machine at $2.80GHz$ with 1 GB of RAM, on one thread. For all tests, we assume that the same camera is used per dataset and have initial values available for the focal length and principal point, though these in some cases were inaccurate. Images were not undistorted prior to testing, and were acquired sequentially.

One important initial experiment consisted in analyzing the behavior of ray divergence given different baselines. For this, we started out with one frame of the *Stockton* sequence and then obtained ray divergences at different baselines from that particular frame. In Fig. 4, results show that Gaussian fitting works well for 'good' baselines, which are typically achieved by applying frame decimation [13] or other choosing algorithms [1] such that the baseline is not too small for linear triangulation but not too small or large for pose estimation degeneracies to occur. This was also verified in several other datasets. The middle image shows the most smooth histogram, and that is where frame decimation picked the best keyframe. In general, with good baselines ray divergence histograms are smooth and can generally be approximated well by Gaussian fitting. With other baselines, ray divergences would not be suitable for Gaussian fitting and for bundle adjustment, since the values are more heavily affected by noise. A good frame decimation is key to our algorithm's success. Table 1 shows the reprojection error and processing time results for these different baselines, where it is shown that the frame decimation keyframe yielded the lowest reprojection error and processing time per point.

In the next experiment, we compared processing times and reprojection errors obtained using weighted bundle adjustment under four different conditions: bundle adjustment weighted by image feature covariances [10], by confidence ellipsoid roundness with and without including image feature covariances, and based on ray divergences. This was only performed on *good* two-view baselines, obtained with prior frame decimation. Table 2 shows the results for some test datasets. Average values for all test parameters were obtained across pairwise frame analysis for all consecutive pairs of each dataset. Unweighted bundle adjustment was not compared, since the comparison would not be direct. Time

Table 1. Number of points, final total reprojection error R (pixels), bundle adjustment iterations I, processing time t in seconds and min/max ray divergence for Gaussian-pdf ray divergence-weighted bundle adjustment at different baselines, for the *Stockton* dataset. Best results were obtained for the three-frame frame decimation keyframe.

Baseline	Points	R	I	t	min_d	max_d
Consecutive	3605	0.049	150	4.24	−0.606	0.774
3 frames	3369	0.013	33	0.83	−0.863	0.508
5 frames	1831	0.200	73	0.87	−0.774	0.561
8 frames	476	0.111	30	0.09	−0.297	0.537

Table 2. Iterations I, final total reprojection error R (pixels) and processing time t (seconds) in (I, R, t) format obtained using bundle adjustment under four different weighting schemes: image feature covariances (CBA), reconstructed point confidence ellipsoid roundness with ($UWBA$) and without including image feature covariances ($UIBA$), and Gaussian-pdf with ray divergences ($RDBA$).

Dataset	CBA	$UWBA$	$UIBA$	$RDBA$
Stockton	43, 0.621, 0.90	40, 0.171, 0.84	37, 0.072, 0.79	38, 0.015, 0.78
Palmdale	23, 4.687, 0.45	22, 1.692, 0.38	20, 0.831, 0.41	22, 0.113, 0.37
castle-P19	150, 281.13, 0.99	150, 4150, 0.95	150, 1046.1, 0.88	97, 90.036, 0.62
Dinosaur	26, 2.631, 0.06	22, 0.286, 0.05	24, 0.09, 0.05	24, 0.162, 0.05
Megascene1	49, 12.14, 0.04	42, 0.179, 0.03	45, 0.074, 0.03	46, 0.124, 0.04
Coneland	150, 28052, 1.10	150, 1880.38, 0.99	115, 599.88, 0.79	126, 81.86, 0.90

is consumed by the *SBA* software [3] to read-in covariance data, and there is matrix inversion for covariance matrices and multiplication of these with Jacobian matrix elements at each iteration, so processing times are typically higher when using covariances. Even so, our bundle adjustment weighting outperforms unweighted bundle adjustment as far as final reprojection error in almost every case, as seen on the right in Fig. 6 where NBA represents the unweighted case. It can be seen that ray divergence-based weighting outperforms every other type of weighting in just about every category, though it's slightly slower and with a higher reprojection error than the more-expensive $UIBA$ in a few cases. Overall, our weighting scheme provides the best combination of processing time, final reprojection error and computational complexity in computing weights. As far as complexity, Beder's algorithm ($UWBA$ and $UIBA$) for example includes the inversion of a 5×5 matrix and two singular value decompositions of a 4×4 and a 3×3 matrix, whereas ray divergence computation does not involve SVD or inversions at all. The feature covariance method CBA is also more expensive, requiring multiple exponential evaluations for each covariance matrix, whereas our method computes a single exponential value.

Having proven that the algorithm performs very well on pairwise reconstructions, it was applied as explained in Subsection 2.3 to perform multi-view reconstructions using our sparse multi-view reconstruction pipeline. Fig. 5 shows on

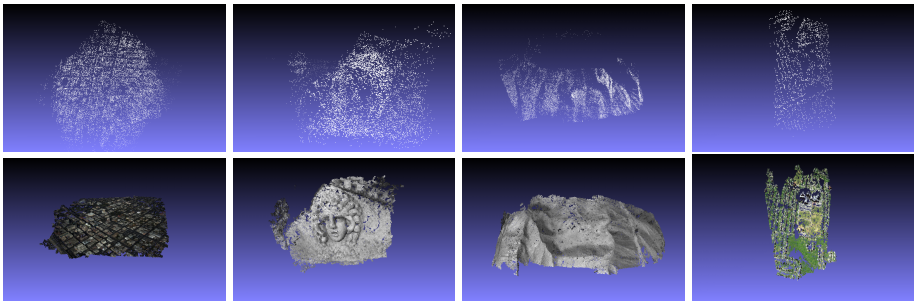

Fig. 5. Top row: sparse multi-view reconstructions for the *Stockton* (*left*), *Medusa* (*middle left*), *Palmdale* (*middle right*) and *Megascene1* (*right*) datasets. Their respective dense reconstructions using the PMVS algorithm [15] are shown on the bottom.

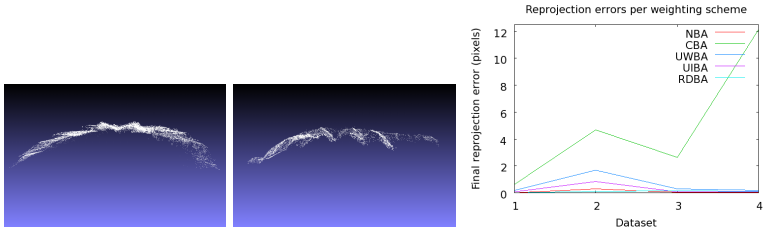

Fig. 6. Side view of a multi-view reconstruction showing the effect of using distorted images (*left*) versus images undistorted with parameters recovered per our algorithm (*middle*), for the *Palmdale* dataset. Total reprojection errors are lower than with other weighting schemes (*right*), as shown for a few datasets.

the top row sparse reconstructions that were obtained while applying sequential multi-view reconstruction, bundle-adjusting with each added image using ray divergence-based weighting. These high-quality sparse reconstructions allow for other algorithms to be applied, such as dense reconstructions with the PMVS algorithm [15] as shown on the bottom row of Fig. 5. Fig. 6 shows the effect on scene reconstruction of using original distorted images versus versions that were undistorted using parameters recovered with our weighted bundle adjustment.

4 Conclusions

An algorithm that makes use of scene reconstruction ray divergence for weighting bundle adjustment and improving its convergence properties was introduced. It was shown that ray divergence, which is a function of all camera parameter inaccuracies, is more efficient to compute and outperforms other weighting schemes such as those based on image feature covariances. There is no dependence on ground-truth information, and results show an improved convergence on different real and synthetic scene types.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

1. Snavely, N., Seitz, S.M., Szeliski, R.: Photo Tourism: Exploring Photo Collections in 3D. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, New York, NY, USA, ACM (2006) 835–846
2. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal On Computer Vision **60** (2004) 91–110
3. Lourakis, M., Argyros, A.: The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece (2000)
4. Knoblauch, D., Hess-Flores, M., Duchaineau, M., Kuester, F.: Factorization of Correspondence and Camera Error for Unconstrained Dense Correspondence Applications. 5th International Symposium on Visual Computing, Las Vegas, Nevada (2009) 720–729
5. Rodehorst, V., Heinrichs, M., Hellwich, O.: Evaluation of Relative Pose Estimation Methods for Multi-Camera Setups. In: International Archives of Photogrammetry and Remote Sensing (ISPRS '08), Beijing, China (2008) 135–140
6. Zhao, W., Nandhakumar, N.: Effects of Camera Alignment Errors on Stereoscopic Depth Estimates. Pattern Recognition **29** (1996) 2115–2126
7. Strecha, C., von Hansen, W., Gool, L.J.V., Fua, P., Thoennessen, U.: On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery. In: CVPR'08. (2008) –1–1
8. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. 2nd edn. Cambridge University Press (2004)
9. Beder, C., Steffen, R.: Determining an Initial Image Pair for Fixing the Scale of a 3D Reconstruction from an Image Sequence. In: DAGM-Symposium'06. (2006) 657–666
10. Brooks, M.J., Chojnacki, W., Gawley, D., van den Hengel, A.: What Value Covariance Information in Estimating Vision Parameters? In: ICCV'01. (2001) 302–308
11. Zeisl, B., Georgel, P., Schweiger, F., Steinbach, E., Navab, N.: Estimation of Location Uncertainty for Scale Invariant Features Points. In: BMVC09. (2009) xx–yy
12. Lindstrom, P.: Triangulation Made Easy. In: CVPR. (2010) 1554–1561
13. Knoblauch, D., Hess-Flores, M., Duchaineau, M., Joy, K.I., Kuester, F.: Non-Parametric Sequential Frame Decimation in Low-Memory Streaming Environments. 7th International Symposium on Visual Computing, Las Vegas, Nevada (2011) 363–374
14. Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual Modeling with a Hand-Held Camera. International Journal of Computer Vision **59** (2004) 207–232
15. Furukawa, Y., Ponce, J.: Accurate, Dense, and Robust Multi-View Stereopsis. In: IEEE Conference on Computer Vision and Pattern Recognition. (2007) 1–8