Title
Accurate Real-Time Reconstruction of Distant Scenes Using Computer Vision: The Recursive Multi-Frame Planar Parallax Algorithm

Author
Templeton, Todd

Publication Date
2009

Peer reviewed|Thesis/dissertation

Accurate Real-Time Reconstruction of Distant Scenes Using Computer Vision: The Recursive Multi-Frame Planar Parallax Algorithm

by

Todd Russell Templeton

B.S.E. (Princeton University) 2004
M.S. (University of California, Berkeley) 2006

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor S. Shankar Sastry, Chair
Professor Ruzena Bajcsy
Professor Martin Banks

Fall 2009

Accurate Real-Time Reconstruction of Distant
Scenes Using Computer Vision:
The Recursive Multi-Frame Planar Parallax Algorithm

by Todd Russell Templeton

Abstract

Accurate Real-Time Reconstruction of Distant
Scenes Using Computer Vision:
The Recursive Multi-Frame Planar Parallax Algorithm

by

Todd Russell Templeton

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor S. Shankar Sastry, Chair

In this dissertation, we detail the Recursive Multi-Frame Planar Parallax (RMFPP) algorithm, a recursive extension of Irani et al.'s Multi-Frame Planar Parallax (MFPP) batch algorithm that allows real-time reconstruction of distant static scenes using computer vision, with expected error that increases only linearly with depth. We present an overview and comprehensive derivation of the theoretical foundation on which the RMFPP algorithm is built, including the seminal planar-parallax work by Sawhney. We derive a recursive cost function that preserves more of the problem's nonlinearity than does the cost function in the MFPP algorithm, which allows a more accurate recursive procedure. In order to obtain a recursive algorithm, we remove the geometry-refining optimization that is present in the MFPP algorithm; however, we empirically show that our algorithm degrades gracefully in the presence of geometric error. We present results using both synthetic and real imagery that show that the RMFPP algorithm is at least as accurate as the original MFPP batch algorithm in many circumstances, is preferred to both fixed- and dynamic-baseline two-frame methods, and is suitable for real-time use.

For Cheryl.

Contents

List of Figures

Acknowledgments

This dissertation would not have been possible without the support of my advisor, Prof. S. Shankar Sastry. From our initial meeting—when he offered me the opportunity to work on what would become my dissertation before the beginning of my first semester—he has consistently provided me with the freedom to pursue my interests, and the (sometimes very expensive) equipment required to transform them into reality.

I would like to extend special appreciation to Dr. Christopher Geyer, the postdoctoral scholar with whom I worked at the beginning of my graduate career. Despite the fact that he began our time together by asking me to write two routines in assembly as a programming test—and then discarded them in favor of the original C versions for reasons of maintainability—I owe him a great debt for his instigation of the Bearland helicopter mapping and landing project, and for the beginnings of many of the ideas in this dissertation.

I would also like to thank Prof. Ruzena Bajcsy and Prof. Martin Banks for serving on my committee; Dr. Jonathan Sprinkle for his support in all things technical, political, and administrative; Dr. David Shim, Dr. Hoam Chung, and Travis Pynn for their support of my experiments at Richmond Field Station; and the Boeing Phantom Works crew at Southern California Logistics Airport for their support of my experiments in Victorville.

Parts of this dissertation are reprinted, in original or modified form, from [5]. They are ©2006 IEEE and are reprinted with permission.

Chapter 1

Introduction

In this dissertation we address the problem of accurately reconstructing a distant scene with a single camera, and doing so at framerate. An important application is the creation of digital elevation maps (DEMs) for purposes ranging from visualization (e.g. Google Earth and NASA's World Wind [19]) and hydrological analysis [8], to robot path planning [18] and autonomous landing [23]. Though active technologies such as radar and LIDAR are available and tend to have very high accuracy, passive sensors are often cheaper and have a smaller form factor, consume less power, and (being emissionless) are more difficult to detect. For a small platform such as a micro air vehicle (MAV) [15], passive sensors may be the only viable option.

1.1 Comparison to Previous Work

A common method for passive range estimation is the use of a stereo camera pair [3, 13]. Stereo systems, unfortunately, cannot attain the desired accuracy given constraints on resolution and platform space. For a stereo system, the variance of depth estimates ($\hat{z}$)

grow *quartically* with depth—that is, the variance of expected differences between the true depth and the estimate $\mathbb{E}\left[(z-\hat{z})^2\right] = \mathsf{O}(z^4)$—which in our case is unacceptable.[1]

In a rigid scene, however, we can treat multiple images—obtained as the vehicle moves through space—as a multiple-camera system. We describe here a recursive method based on the multi-frame planar parallax framework [17, 7] that uses multiple image pairs with baselines larger than are physically attainable on the platform in order to reduce variance. We show that in theory we can recover depth with a variance that is asymptotically quadratic in the depth.

The novelty of this result is a method that is (*i*) *recursive* in the sense that the cost of incorporating measurements from a new image is proportional only to the number of pixels in the image and does *not* depend on the number of frames already seen; (*ii*) dense, in the sense that it provides estimates of depth for any sufficiently-textured region; (*iii*) more accurate than instantaneous stereo; and (*iv*) direct (see the discussion, pro: [6], con: [24]), by which we mean that the algorithm does not depend on the matching of features, but rather expresses a cost function directly in terms of the image, and the gradients of the cost function are computed by linearization of the brightness constancy constraint.

Other methods feature some, but not all, of these elements. For example, bundle adjustment [25] is the optimal estimator for determining structure and motion from multiple views when correspondences are known and correct. However, it provides neither dense structure, nor the ability to recover structure recursively. Stereo and multi-baseline methods are the most-favored methods for recovering dense structure, e.g. [16, 20]. Matthies and Shafer [13], and later Xiong and Matthies [26], investigate sources of error in stereo; the primary drawback of stereo is its inaccuracy, as discussed above. Planar parallax is a related framework based on registration using a plane in the scene [17]. Irani et al. [7]

[1] For example, at a depth of 100 meters, a baseline of 1 meter, and a focal length $f = 500$, the standard deviation of $\hat{z}$ is approximately 20 meters.

propose a method for estimating planar parallax, and from it depth, using more than two views, though it is not recursive. Their work is the closest in spirit to ours. We improve on this method in that we present a Recursive Multi-Frame Planar Parallax (RMFPP) algorithm. Other closely related works are Zucchelli et al. [27], in which they sparsely estimate structure and motion and update a dense structure map; and Matthies et al. [14], in which they propose a Kalman filter for updating disparities.

Here we employ a direct method that takes advantage of the observation that for a smoothly-moving camera, although the initial small-baseline disparity estimates may lead to inaccurate depth estimates, they are nevertheless accurate disparity measurements. Furthermore, later improvement in the depth estimates will not significantly change the refined small-baseline disparities. Therefore, in the cost function described in [7], the image need not be rewarped and relinearized. Instead the linearized terms are kept and are incorporated into the cost function for later frames.

The method described here is subject to several assumptions. We reiterate that this method updates estimates of structure only; we assume that the positions and orientations of the cameras have been previously determined. We have *not* found a method to recursively estimate structure *and* motion that is both dense and direct—the possibility of such an algorithm seems unlikely but remains open. We also rely on the usual assumptions: validity of the brightness constancy constraint within some regions, rigidity of the scene, and the presence of sufficient texture. Finally, the motions between the camera positions should be sufficiently small—though this constraint can be lessened by the use of image pyramids.

1.2 Achieving Greater Accuracy With Multiple Frames

In this section we model depth errors in a stereo pair and in an idealized multiple-baseline system. For a fixed-baseline stereo pair, the predicted standard deviation is quadratic in the depth, i.e. $\mathbb{E}\big[(\hat{z}-z)^2\big]^{1/2} = \mathsf{O}(z^2)$. Using the simple case of a camera moving in a straight line at constant velocity, we show that by appropriately weighting pairwise estimates we can theoretically attain errors between $\mathsf{O}(z)$ and $\mathsf{O}\big(z^{1/2}\big)$, depending on the correlation between disparity estimates.

Fixed-baseline stereo. Consider a rectified stereo pair separated by a baseline b, observing a point at depth z. The relationship between disparity and depth is given by $z = f\,b/\delta$, where δ is the disparity and f is the focal length. Ignoring quantization errors and mismatches, we can obtain an approximation of the variance of the depth estimate at any single pixel, namely:

$$\begin{aligned}
\sigma_{\hat{z}}^2 &= \mathbb{E}\big[\,(\hat{z}-z)^2\,\big] \\
&= \mathbb{E}\left[\left(\frac{f\,b}{\delta+\epsilon} - \frac{f\,b}{\delta}\right)^2\right] \\
&= \int \left(\frac{f\,b}{\delta+\epsilon} - \frac{f\,b}{\delta}\right)^2 p(\epsilon)\,d\epsilon \\
&= \frac{f^2\,b^2}{\delta^2}\int \left(\frac{\epsilon}{\delta+\epsilon}\right)^2 p(\epsilon)\,d\epsilon \\
&\approx \frac{f^2\,b^2}{\delta^2}\int \left(\frac{1}{\delta}\,\epsilon\right)^2 p(\epsilon)\,d\epsilon \\
&= \frac{f^2\,b^2}{\delta^4}\int \epsilon^2\,p(\epsilon)\,d\epsilon \\
&= \frac{z^4}{f^2\,b^2}\,\sigma_\epsilon^2
\end{aligned} \tag{1.1}$$

where ϵ is the zero-mean error in the disparity estimate, and where we have taken the first-order Taylor series approximation of $\epsilon/(\delta+\epsilon)$ in ϵ about 0. Generally σ_ϵ depends on the image derivatives along the scanline. Using a single stereo measurement would be

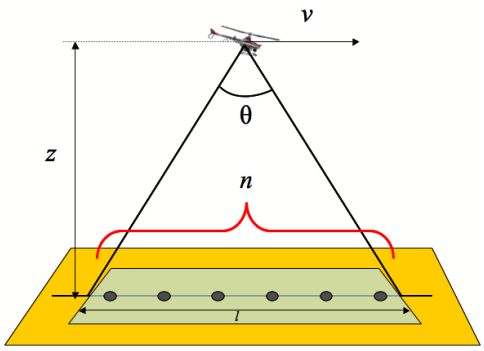


Figure 1.1: Idealized flight for purposes of analyzing depth accuracy. ©2006 IEEE. Reprinted, with permission, from [5].

ill-advised at depths greater than $f\,b/\sigma_\epsilon$—corresponding to a disparity of σ_ϵ—above which the predicted standard deviation $\sigma_{\hat{z}}$ would become larger than the depth.

Can we achieve greater accuracy in range using only passive means? We can reduce uncertainty by increasing b, or by utilizing the independence in the measurements, if there is any. Often it is not practical to increase the baseline between two vehicle-mounted cameras beyond some fixed limit, but on a moving vehicle we can get wider baselines for free. If the camera's motion can be recovered (either by structure-from-motion methods or by some combination of inertial and GPS systems) then we can use multiple measurements to reduce error. If the measurements are to some degree independent, then we can drive down uncertainty.

Monocular camera at constant velocity and framerate. We consider the following situation, depicted in Figure 1.1: an aerial vehicle flying at constant altitude above terrain with average relative height z meters, at constant horizontal velocity v. Assume that the position of the vehicle is known without error at all times, and that a downward-pointing onboard camera with field of view θ captures an image every t seconds. The baseline between the 0-th and k-th frame is $b_k = kvt$ and the number of times a point at depth z from the camera is seen is at most $n = l/vt = 2z\tan\frac{\theta}{2}/vt$.

Let us use a single pair of frames separated by a wider baseline and determine the resulting depth error. Choose the 0-th frame and the $c \cdot n$-th frame, where $c < 1$—that is, a frame a constant fraction in between the first and the last frame in which there is overlap in the two views of the ground. Then the predicted accuracy (variance) is $z^2\sigma_\epsilon^2/4c^2f^2\tan^2\frac{\theta}{2}$.[2] This is a significant improvement over the fixed-baseline case. Instead of being quadratic in the depth, here the predicted standard deviation is proportional to the depth.

Can we do better than error linear in depth by using multiple measurements? We can get the most out of multiple measurements when they are known to be independent. However, in simulated experiments with $1/f$ noise, we find that errors in disparity estimates are correlated with correlation coefficient up to 0.6; i.e. if $\epsilon_{i,j}$ is the error in the estimate of disparity between frames i and j, then we find there to be correlation between errors $\epsilon_{0,1}$ and $\epsilon_{0,2}$. Let us construct a linear estimator that is blind to the generally-unknown correlation coefficient, and derive the estimator's variance while assuming a nonzero correlation.

Let $\hat{z}_k$ be the depth calculated using the k-th estimated disparity, $\hat{\delta}_k = \delta_k + \epsilon_k$, between frames 0 and k. Using Equation 1.1, $\hat{z}_k$'s variance is $\sigma_k^2 = z^4\sigma_\epsilon^2/f^2k^2t^2v^2$; the minimum-variance linear estimate of z using $m = c \cdot n$ estimates $\hat{z}_k$ is $\hat{z} = \sum_{k=1}^{m} w_k\hat{z}_k$, where $w_k = \sigma_k^{-2}/\sum_{j=1}^{m}\sigma_j^{-2}$. Since the 0-th image does not change it is reasonable to assume that, for any fixed pixel in the first image, σ_{ϵ_k} does not change with k. However, as we have discussed, we cannot guarantee statistical independence of the ϵ_k.

Predicted error. It is difficult, if not impossible, to empirically determine the correlation among ϵ_k for real images. However, if we have evidence that the correlation is bounded, then we can gauge the effect of correlation on the accuracy of the linear estimator. Assume that, for some $0 \leq \rho \leq 1$, $\mathbb{E}[\epsilon_j\,\epsilon_k] < \rho\,\sigma_\epsilon^2$ for all j and k. Then the covariance between $\hat{z}_j$ and $\hat{z}_k$ is:

[2]Note that f and θ are usually coupled with the resolution of the camera, but are constant for a fixed camera.

$$
\begin{aligned}
\operatorname{cov}(\hat{z}_j, \hat{z}_k) &= \mathbb{E}\big[\, (\hat{z}_j - z)\,(\hat{z}_k - z)\,\big] \\
&= \mathbb{E}\left[\left(\frac{f\,b_j}{\delta_j + \epsilon_j} - \frac{f\,b_j}{\delta_j}\right)\left(\frac{f\,b_k}{\delta_k + \epsilon_k} - \frac{f\,b_k}{\delta_k}\right)\right] \\
&= \iint \left(\frac{f\,b_j}{\delta_j + \epsilon_j} - \frac{f\,b_j}{\delta_j}\right)\left(\frac{f\,b_k}{\delta_k + \epsilon_k} - \frac{f\,b_k}{\delta_k}\right) p(\epsilon_j, \epsilon_k)\, d\epsilon_j\, d\epsilon_k \\
&= \frac{f^2\, b_j\, b_k}{\delta_j\, \delta_k} \iint \left(\frac{\epsilon_j}{\delta_j + \epsilon_j}\right)\left(\frac{\epsilon_k}{\delta_k + \epsilon_k}\right) p(\epsilon_j, \epsilon_k)\, d\epsilon_j\, d\epsilon_k \\
&\approx \frac{f^2\, b_j\, b_k}{\delta_j\, \delta_k} \iint \left(\frac{1}{\delta_j}\,\epsilon_j\right)\left(\frac{1}{\delta_k}\,\epsilon_k\right) p(\epsilon_j, \epsilon_k)\, d\epsilon_j\, d\epsilon_k \\
&= \frac{f^2\, b_j\, b_k}{\delta_j^2\, \delta_k^2} \iint \epsilon_j\, \epsilon_k\, p(\epsilon_j, \epsilon_k)\, d\epsilon_j\, d\epsilon_k \\
&= \frac{z^4}{f^2\, b_j\, b_k}\, \mathbb{E}[\epsilon_j\, \epsilon_k] \\
&< \rho\, \frac{z^4}{f^2\, b_j\, b_k}\, \sigma_\epsilon^2
\end{aligned}
\tag{1.2}
$$

where we have taken the first-order Taylor series approximation of $\epsilon_j/(\delta_j + \epsilon_j)$ in ϵ_j, and $\epsilon_k/(\delta_k + \epsilon_k)$ in ϵ_k, about 0. The expected squared error, as a function of ρ, for the linear estimator defined above is:

$$
\begin{aligned}
\mathbb{E}\big[(\hat{z} - z)^2\big] &= \sum_{k=1}^{m} \operatorname{var}(w_k\, \hat{z}_k) \;+\; 2 \sum_{k=2}^{m} \sum_{j=1}^{k-1} \operatorname{cov}(w_j\, \hat{z}_j, w_k\, \hat{z}_k) \\
&= \sum_{k=1}^{m} w_k^2\, \operatorname{var}(\hat{z}_k) \;+\; 2 \sum_{k=2}^{m} \sum_{j=1}^{k-1} w_j\, w_k\, \operatorname{cov}(\hat{z}_j, \hat{z}_k) \\
&= \left(\sum_{k=1}^{m} \frac{1}{\operatorname{var}(\hat{z}_k)}\right)^{-2} \left[\sum_{k=1}^{m} \frac{1}{\operatorname{var}(\hat{z}_k)} \;+\; 2 \sum_{k=2}^{m} \sum_{j=1}^{k-1} \frac{\operatorname{cov}(\hat{z}_j, \hat{z}_k)}{\operatorname{var}(\hat{z}_j)\, \operatorname{var}(\hat{z}_k)}\right] \\
&< \left(\sum_{k=1}^{m} \frac{f^2\, k^2\, v^2\, t^2}{z^4\, \sigma_\epsilon^2}\right)^{-2} \left[\sum_{k=1}^{m} \frac{f^2\, k^2\, v^2\, t^2}{z^4\, \sigma_\epsilon^2} \;+\; 2\, \rho \sum_{k=2}^{m} \sum_{j=1}^{k-1} \frac{f^2\, j\, k\, v^2\, t^2}{z^4\, \sigma_\epsilon^2}\right] \\
&= \frac{z^4\, \sigma_\epsilon^2}{f^2\, v^2\, t^2} \left(\sum_{k=1}^{m} k^2\right)^{-2} \left[\sum_{k=1}^{m} k^2 \;+\; 2\, \rho \sum_{k=2}^{m} \left(k \sum_{j=1}^{k-1} j\right)\right] \\
&= \frac{z^4\, \sigma_\epsilon^2}{f^2\, v^2\, t^2} \left(\sum_{k=1}^{m} k^2\right)^{-2} \left[\sum_{k=1}^{m} k^2 \;+\; \rho \sum_{k=2}^{m} k^2\, (k-1)\right] \\
&\approx \frac{z^4\, \sigma_\epsilon^2}{f^2\, v^2\, t^2} \left(\frac{1}{3}\, m^3\right)^{-2} \left[\frac{1}{3}\, m^3 \;+\; \frac{1}{4}\, \rho\, m^4\right] \\
&= \frac{z^4\, \sigma_\epsilon^2}{f^2\, v^2\, t^2} \left[3\, \frac{1}{m^3} \;+\; \frac{9}{4}\, \rho\, \frac{1}{m^2}\right]
\end{aligned}
$$

$$= \frac{z^4\,\sigma_\epsilon^2}{f^2\,v^2\,t^2}\left[3\left(\frac{v\,t}{2\,c\,z\,\tan\frac{\theta}{2}}\right)^3 + \frac{9}{4}\,\rho\left(\frac{v\,t}{2\,c\,z\,\tan\frac{\theta}{2}}\right)^2\right]$$

$$= \frac{z^2\,\sigma_\epsilon^2}{f^2}\left[\frac{3}{8}\,\frac{v\,t}{c^3\,z\,\tan^3\frac{\theta}{2}} + \frac{9}{16}\,\rho\,\frac{1}{c^2\,\tan^2\frac{\theta}{2}}\right]$$

$$\triangleq \left(c_1\,z + c_2\,\rho\,z^2\right)\cdot\sigma_\epsilon^2 \tag{1.3}$$

for $z >> vt$, and where c_1 and c_2 depend on t, v, f, θ, and c. If there is no correlation, i.e. $\rho = 0$, then the resulting estimator has standard deviation proportional to the square root of the depth. When there is nonzero correlation, then the standard deviation is asymptotically linear. In Figure 1.2 we plot Equation 1.3 using the following values: $t = 3.75^{-1}$ sec; $v = 14$ meters/sec; $\theta = 50°$; $c = 1/4$; $\rho = 0$, $1/3$, $2/3$ and 1; and z ranging between 20 and 200 meters. Note that at $z = 17.11$ meters the number of measurements is $m = 1$.

To summarize, by combining multiple measurements and wider baselines we can obtain between linear and quadratic variance in depth estimates as a function of true depth, a vast improvement over the quartic variance obtained with a stereo pair. In the following chapters we describe a recursive algorithm for densely reconstructing terrain, in which we try to achieve this performance.

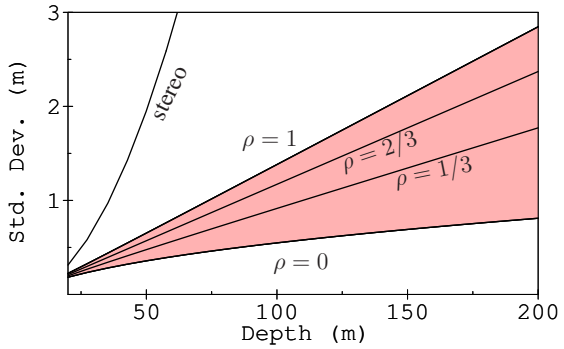


Figure 1.2: Predicted standard deviations for stereo and multi-baseline as a function of depth. The single curve shows depth error of stereo when using the smallest baseline (the first two frames in the idealized aerial image sequence). The (red) shaded area shows the range of errors of a multiple-baseline linear estimator for $0 \leq \rho \leq 1$. ©2006 IEEE. Reprinted, with permission, from [5].

Chapter 2

Multi-Frame Planar Parallax

The multi-frame planar parallax (MFPP) method is a generalization of stereo rectification to more than two frames that was first described by Sawhney [17], and was later extended by Irani et al. [7]. Whereas stereo rectification yields images where the disparity (or optical flow) is parallel to scanlines and is inversely proportional to depth, MFPP registration yields images such that the ratio of disparities along epipolar lines can be expressed in terms of a view-independent shape parameter that encodes depth.

This chapter provides a self-contained introduction to both two-frame and multi-frame planar parallax, using (and introducing) the notation that will be used throughout the remainder of this dissertation. The two-view derivations are more-detailed versions of Sawhney's in [17], and are given here in multi-frame notation for consistency. The multi-frame derivations are based on those of Irani et al. in [7].

2.1 Geometry

Suppose one or more stationary or moving cameras take images $\mathcal{I}_1, \ldots, \mathcal{I}_m$ of a rigid scene, and denote forward and inverse (calibrated) image projection as

$$\pi\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} \quad \text{and} \quad \pi^*\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

respectively. Denote by $\mathsf{K_i} \in SL!(3)$ the intrinsic calibration of the i-th view. Let the rotation and translation taking the i-th coordinate system to the first one be given by ($\mathsf{R_i} \in SO(3), \boldsymbol{T_i} \in \mathbb{R}^3$), i.e. all corresponding points $\boldsymbol{P_1}$ and $\boldsymbol{P_i}$ in the first and i-th coordinate systems satisfy $\boldsymbol{P_1} = \mathsf{R_i}\,\boldsymbol{P_i} + \boldsymbol{T_i}$. By combining these definitions, we have that a point $\boldsymbol{P_i}$ in the coordinate system of the i-th camera can be projected into the first camera by $\boldsymbol{p_1} = \pi[\mathsf{K_1}\,(\mathsf{R_i}\,\boldsymbol{P_i} + \boldsymbol{T_i})]$. Note that $\mathsf{R_1} = \mathsf{I}$ and $\boldsymbol{T_1} = \boldsymbol{0}$ so that the first frame acts as a reference frame.

Figure 2.1 shows the geometry of planar parallax for the reference frame and an arbitrary i-th frame. We represent the world point of interest, $\boldsymbol{P}$, by $\boldsymbol{P_1}$ in the coordinate system of the first view and by $\boldsymbol{P_i}$ in the coordinate system of the i-th view, and its projections into $\mathcal{I}_1$ and $\mathcal{I}_i$ by $\boldsymbol{p_1}$ and $\boldsymbol{p_i}$, respectively. We denote the image of the i-th viewpoint in $\mathcal{I}_1$ (one of the epipoles in the stereo pair defined by the first and i-th views) by $\boldsymbol{e_i} = \pi(\mathsf{K_i}\,\boldsymbol{T_i})$, although in practice we will often use the homogeneous epipole $\boldsymbol{E_i} = (e_x^{(i)}, e_y^{(i)}, e_z^{(i)})^T = \mathsf{K_1}\,\boldsymbol{T_i} \in \mathbb{R}^3$ to avoid division by zero when $e_z^{(i)}$ is zero or numerical errors when $e_z^{(i)}$ is small, i.e. when there is small z displacement between the views.

We choose a virtual reference plane in the scene, represented by $\boldsymbol{N_i}^T\,\boldsymbol{P_i} + d_i = 0$ in the i-th view. Here, $\boldsymbol{N_i} \in \mathbb{R}^3$ is unit normal of the reference plane in the coordinate system of the i-th view, and $d_i \in \mathbb{R}$ is the perpendicular distance of the i-th viewpoint from the virtual plane. This reference plane allows us to project a point $\boldsymbol{p_i}$ in $\mathcal{I}_i$ into $\mathcal{I}_1$ without knowing the

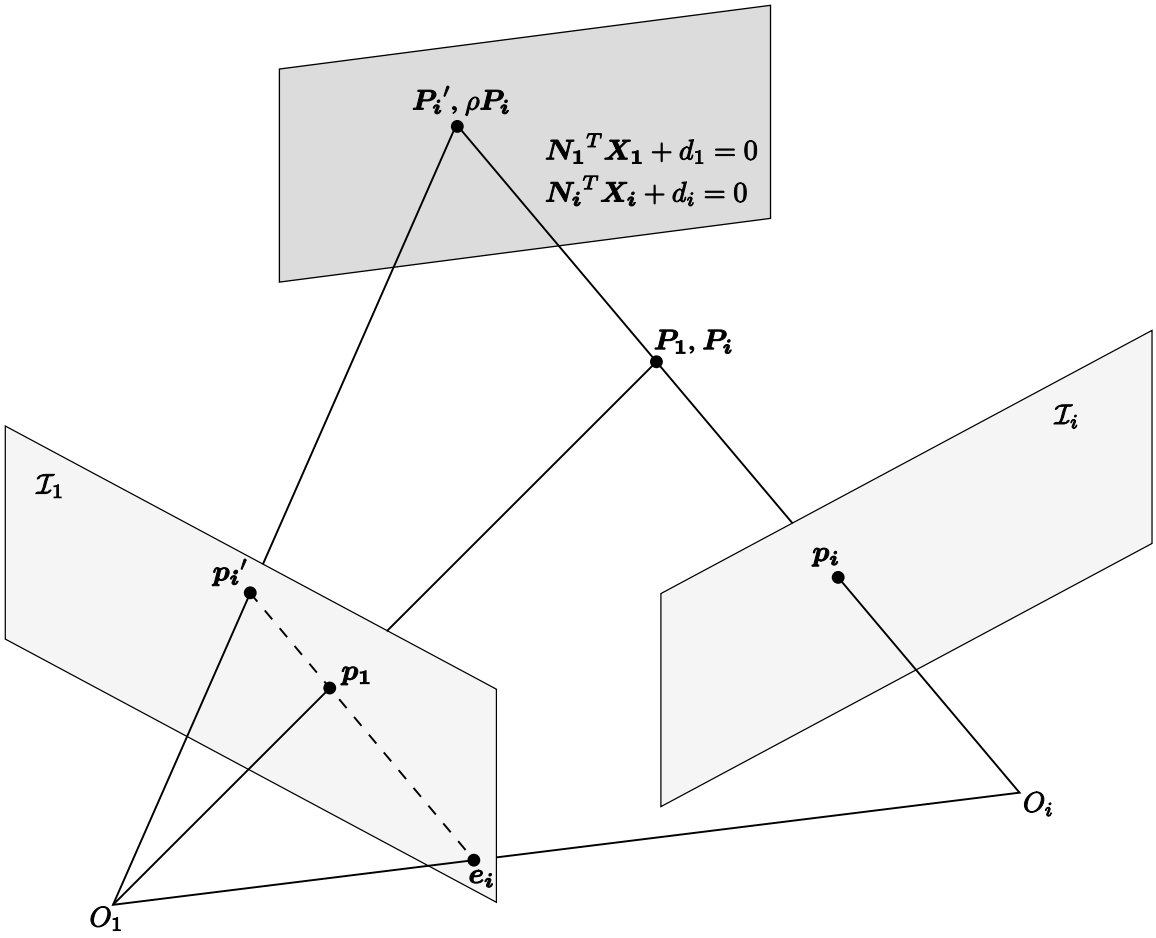


Figure 2.1: Geometry of two-view planar parallax.

distance of $\boldsymbol{P}_i$ from the i-th viewpoint. Although the projected point $\boldsymbol{p}'_i$ is not necessarily equal to $\boldsymbol{p}_1$ unless $\boldsymbol{P}$ is on the plane, the point $\boldsymbol{p}'_1$ has two convenient properties:

1. The point $\boldsymbol{p}_1$ is on the epipolar line through $\boldsymbol{p}'_i$. Alternatively, the point $\boldsymbol{p}'_i$ is on the epipolar line through $\boldsymbol{p}_1$. Figure 2.1 provides a geometric proof.

2. The position of $\boldsymbol{p}'_i$ along the epipolar line through $\boldsymbol{p}_1$ is a simple function of $\boldsymbol{N}_1$, d_1, $\mathsf{R_i}$, $\boldsymbol{T}_i$, K_1, $\mathsf{K_i}$, $\boldsymbol{p}_1$, and a view-independent shape parameter $\mathcal{G}(\boldsymbol{p}_1)$. This function will be derived in the following sections.

2.2 Homographies

Given the reference plane and the geometry between the first and i-th views, we construct the homographies $\mathsf{H_i} \in \mathbb{R}^{3\times3}$ such that each $\mathsf{H_i}$ transforms the first frame to the i-th frame via the reference plane, i.e. the projections $\boldsymbol{p'_i}$ and $\boldsymbol{p_i}$ of a point on the reference plane in the first and i-th frames, as in Figure 2.1, satisfy $\boldsymbol{p_i} = \pi[\mathsf{H_i}\,\pi^*(\boldsymbol{p'_i})]$. These homographies are given by:

$$\mathsf{H_i} = \mathsf{K_i}\mathsf{R_i}^{-1}\left(\mathsf{I} + \frac{1}{d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\mathsf{K_1}^{-1} \tag{2.2}$$

Proof:

$$\begin{aligned}
\rho\boldsymbol{P_i} &= \mathsf{R_i}^{-1}\left(\boldsymbol{P'_i} - \boldsymbol{T_i}\right) \\
&= \mathsf{R_i}^{-1}\left[\boldsymbol{P'_i} - \boldsymbol{T_i}\left(-\frac{1}{d_1}\boldsymbol{N_1}^T\boldsymbol{P'_i}\right)\right] \\
&= \mathsf{R_i}^{-1}\left(\mathsf{I} + \frac{1}{d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\boldsymbol{P'_i} \\
&\propto \mathsf{R_i}^{-1}\left(\mathsf{I} + \frac{1}{d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\mathsf{K_1}^{-1}\pi^*(\boldsymbol{p'_i}) \\
\Leftrightarrow \boldsymbol{p_i} &= \pi\left[\mathsf{K_i}\mathsf{R_i}^{-1}\left(\mathsf{I} + \frac{1}{d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\mathsf{K_1}^{-1}\pi^*(\boldsymbol{p'_i})\right] \\
&= \pi[\mathsf{H_i}\,\pi^*(\boldsymbol{p'_i})] \text{ where } \mathsf{H_i} = \mathsf{K_i}\mathsf{R_i}^{-1}\left(\mathsf{I} + \frac{1}{d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\mathsf{K_1}^{-1}
\end{aligned} \tag{2.3}$$

Note that we can write the inverse of $\mathsf{H_i}$ analytically:

$$\mathsf{H_i}^{-1} = \mathsf{K_1}\left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\mathsf{R_i}\mathsf{K_i}^{-1} \tag{2.4}$$

Proof:

$$\begin{aligned}
\boldsymbol{P'_i} &= \mathsf{R_i}\,\rho\boldsymbol{P_i} + \boldsymbol{T_i} \\
&= \mathsf{R_i}\,\rho\boldsymbol{P_i} + \boldsymbol{T_i}\left(-\frac{1}{d_i}\boldsymbol{N_i}^T\rho\boldsymbol{P_i}\right) \\
&= \mathsf{R_i}\,\rho\boldsymbol{P_i} + \boldsymbol{T_i}\left[-\frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\left(\mathsf{R_i}^{-1}\boldsymbol{N_1}\right)^T\rho\boldsymbol{P_i}\right] \\
&= \mathsf{R_i}\,\rho\boldsymbol{P_i} + \boldsymbol{T_i}\left(-\frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{N_1}^T\mathsf{R_i}\,\rho\boldsymbol{P_i}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right) \mathsf{R_i}\,\rho\boldsymbol{P_i} \\
&\propto \left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right) \mathsf{R_i}\mathsf{K_i}^{-1}\pi^*(\boldsymbol{p_i}) \\
\Leftrightarrow \boldsymbol{p'_i} &= \pi\left[\mathsf{K_1}\left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\mathsf{R_i}\mathsf{K_i}^{-1}\pi^*(\boldsymbol{p_i})\right] \\
&= \pi\left[\mathsf{H_i}^{-1}\,\pi^*(\boldsymbol{p_i})\right] \text{ where } \mathsf{H_i}^{-1} = \mathsf{K_1}\left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\mathsf{R_i}\mathsf{K_i}^{-1} \qquad (2.5)
\end{aligned}
$$

2.3 Geometric Constraint

In this section, we derive the fundamental planar parallax constraint, the position of $\boldsymbol{p'_i}$ along the epipolar line through $\boldsymbol{p_1}$:

$$
\boldsymbol{\delta}_i(\boldsymbol{p_1}, \mathcal{G}(\boldsymbol{p_1})) \triangleq \boldsymbol{p_1} - \boldsymbol{p'_i} = -\frac{\mathcal{G}(\boldsymbol{p_1})}{d_i - \mathcal{G}(\boldsymbol{p_1})\; e_z^{(i)}} \begin{bmatrix} e_z^{(i)}\, p_x - e_x^{(i)} \\ e_z^{(i)}\, p_y - e_y^{(i)} \end{bmatrix} \qquad (2.6)
$$

where $\boldsymbol{p_1} = (p_x, p_y)^T$, $\mathcal{G}(\boldsymbol{p_1}) = p_h/p_z$ is a view-independent scalar defined at each point in the first image, p_z is the depth of $\boldsymbol{P}$ in the first view, and $p_h = \boldsymbol{N_1}^T\boldsymbol{P_1} + d_1$ is the signed perpendicular distance of $\boldsymbol{P}$ from the reference plane.

Proof:

$$
\begin{aligned}
\boldsymbol{P'_i} &= \left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right) \mathsf{R_i}\,\rho\boldsymbol{P_i} \\
&\propto \left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right) \mathsf{R_i}\boldsymbol{P_i} \\
&= \left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right) \mathsf{R_i}\left[\mathsf{R_i}^{-1}\left(\boldsymbol{P_1} - \boldsymbol{T_i}\right)\right] \\
&= \left(\mathsf{I} - \frac{1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}\boldsymbol{N_1}^T\right)\left(\boldsymbol{P_1} - \boldsymbol{T_i}\right) \\
&= \boldsymbol{P_1} - \left[1 + \frac{\boldsymbol{N_1}^T\left(\boldsymbol{P_1} - \boldsymbol{T_i}\right)}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\right]\boldsymbol{T_i} \\
&= \boldsymbol{P_1} - \frac{\boldsymbol{N_1}^T\boldsymbol{P_1} + d_1}{\boldsymbol{N_1}^T\boldsymbol{T_i} + d_1}\boldsymbol{T_i}
\end{aligned}
$$

$$
\begin{aligned}
&= \boldsymbol{P_1} - \frac{p_h}{d_i}\boldsymbol{T_i} \\
\Rightarrow \pi^*(\boldsymbol{p'_i}) &= \mathsf{K}_1 \left[\frac{1}{p_z - \frac{p_h}{d_i} e_z^{(i)}} \left(\boldsymbol{P_1} - \frac{p_h}{d_i}\boldsymbol{T_i} \right) \right] \\
&= \mathsf{K}_1 \left[\frac{d_i}{d_i\, p_z - p_h\, e_z^{(i)}} \left(\boldsymbol{P_1} - \frac{p_h}{d_i}\boldsymbol{T_i} \right) \right] \\
&= \frac{d_i}{d_i\, p_z - p_h\, e_z^{(i)}} \left(p_z\, \pi^*(\boldsymbol{p_1}) - \frac{p_h}{d_i}\boldsymbol{E_i} \right) \\
\Leftrightarrow \pi^*(\boldsymbol{p_1}) - \pi^*(\boldsymbol{p'_i}) &= \pi^*(\boldsymbol{p_1}) - \frac{d_i}{d_i\, p_z - p_h\, e_z^{(i)}} \left(p_z\, \pi^*(\boldsymbol{p_1}) - \frac{p_h}{d_i}\boldsymbol{E_i} \right) \\
&= \left(1 - \frac{d_i\, p_z}{d_i\, p_z - p_h\, e_z^{(i)}} \right) \pi^*(\boldsymbol{p_1}) + \frac{p_h}{d_i\, p_z - p_h\, e_z^{(i)}} \boldsymbol{E_i} \\
&= -\frac{p_h\, e_z^{(i)}}{d_i\, p_z - p_h\, e_z^{(i)}} \pi^*(\boldsymbol{p_1}) + \frac{p_h}{d_i\, p_z - p_h\, e_z^{(i)}} \boldsymbol{E_i} \\
&= \frac{p_h}{d_i\, p_z - p_h\, e_z^{(i)}} \left(-e_z^{(i)}\, \pi^*(\boldsymbol{p_1}) + \boldsymbol{E_i} \right) \\
&= -\frac{\mathcal{G}(\boldsymbol{p_1})}{d_i - \mathcal{G}(\boldsymbol{p_1})\; e_z^{(i)}} \left(e_z^{(i)}\, \pi^*(\boldsymbol{p_1}) - \boldsymbol{E_i} \right) \\
\Leftrightarrow \boldsymbol{p_1} - \boldsymbol{p'_i} &= -\frac{\mathcal{G}(\boldsymbol{p_1})}{d_i - \mathcal{G}(\boldsymbol{p_1})\; e_z^{(i)}} \begin{bmatrix} e_z^{(i)}\, p_x - e_x^{(i)} \\ e_z^{(i)}\, p_y - e_y^{(i)} \end{bmatrix}
\end{aligned}
\tag{2.7}
$$

The difference $\boldsymbol{\delta}_i$ is called the *planar parallax*. Note that we have derived this relationship using the homogeneous epipole $\boldsymbol{E_i}$ instead of the (projected) epipole $\boldsymbol{e_i}$ in order to handle the case when the (projected) epipole is at infinity; in that case we find, as expected, that all epipolar lines are parallel, i.e. that all terms in Equation (2.6) that include p_x and p_y are zero.

If we know the correspondence between $\boldsymbol{p_1}$ and $\boldsymbol{p'_i}$ a priori, we can write a simpler form of $\boldsymbol{\delta}_i$ in terms of $\boldsymbol{p'_i}$:

$$
\boldsymbol{\delta}_i(\boldsymbol{p'_i}, \mathcal{G}(\boldsymbol{p_1})) \triangleq \boldsymbol{p_1} - \boldsymbol{p'_i} = -\frac{\mathcal{G}(\boldsymbol{p_1})}{d_i} \begin{bmatrix} e_z^{(i)}\, p'_x - e_x^{(i)} \\ e_z^{(i)}\, p'_y - e_y^{(i)} \end{bmatrix}
\tag{2.8}
$$

where $\boldsymbol{p'_i} = \left(p'_x, p'_y\right)^T$.

Proof:

$$\begin{aligned}
\pi^*(\boldsymbol{p'_i}) &= \frac{d_i}{d_i\, p_z - p_h\, e_z^{(i)}} \left(p_z\, \pi^*(\boldsymbol{p_1}) - \frac{p_h}{d_i} \boldsymbol{E_i} \right) \\
\Leftrightarrow \pi^*(\boldsymbol{p_1}) &= \frac{d_i\, p_z - p_h\, e_z^{(i)}}{d_i\, p_z}\, \pi^*(\boldsymbol{p'_i}) + \frac{p_h}{d_i\, p_z} \boldsymbol{E_i} \\
\Leftrightarrow \pi^*(\boldsymbol{p_1}) - \pi^*(\boldsymbol{p'_i}) &= \left(\frac{d_i\, p_z - p_h\, e_z^{(i)}}{d_i\, p_z} - 1 \right) \pi^*(\boldsymbol{p'_i}) + \frac{p_h}{d_i\, p_z} \boldsymbol{E_i} \\
&= -\frac{p_h\, e_z^{(i)}}{d_i\, p_z}\, \pi^*(\boldsymbol{p'_i}) + \frac{p_h}{d_i\, p_z} \boldsymbol{E_i} \\
&= \frac{p_h}{d_i\, p_z} \left(-e_z^{(i)}\, \pi^*(\boldsymbol{p'_i}) + \boldsymbol{E_i} \right) \\
&= -\frac{\mathcal{G}(\boldsymbol{p_1})}{d_i} \left(e_z^{(i)}\, \pi^*(\boldsymbol{p'_i}) - \boldsymbol{E_i} \right) \\
\Leftrightarrow \boldsymbol{p_1} - \boldsymbol{p'_i} &= -\frac{\mathcal{G}(\boldsymbol{p_1})}{d_i} \begin{bmatrix} e_z^{(i)}\, p'_x - e_x^{(i)} \\ e_z^{(i)}\, p'_y - e_y^{(i)} \end{bmatrix}
\end{aligned} \tag{2.9}$$

Although we have defined $\mathcal{G}(\boldsymbol{p_1}) = p_h/p_z$, p_h and p_z are not independent. Using the fact that $\boldsymbol{P_1} = p_z\, {\mathsf{K}_1}^{-1}\, \pi^*(\boldsymbol{p_1})$:

$$\begin{aligned}
p_h &= \boldsymbol{N_1}^T \boldsymbol{P_1} + d_1 \\
&= \boldsymbol{N_1}^T \left(p_z\, {\mathsf{K}_1}^{-1}\, \pi^*(\boldsymbol{p_1}) \right) + d_1 \\
&= \left(\boldsymbol{N_1}^T\, {\mathsf{K}_1}^{-1}\, \pi^*(\boldsymbol{p_1}) \right) p_z + d_1
\end{aligned} \tag{2.10}$$

Therefore, we can use $\mathcal{G}(\boldsymbol{p_1})$ and p_z interchangeably:

$$\begin{aligned}
\mathcal{G}(\boldsymbol{p_1}) &= \frac{p_h}{p_z} \\
&= \frac{\left(\boldsymbol{N_1}^T\, {\mathsf{K}_1}^{-1}\, \pi^*(\boldsymbol{p_1}) \right) p_z + d_1}{p_z} \\
&= \boldsymbol{N_1}^T\, {\mathsf{K}_1}^{-1}\, \pi^*(\boldsymbol{p_1}) + \frac{d_1}{p_z}
\end{aligned} \tag{2.11}$$

and

$$
\begin{aligned}
p_z = \frac{p_h}{\mathcal{G}(\boldsymbol{p_1})} &= \frac{\left(\boldsymbol{N_1}^T \, \mathsf{K_1}^{-1} \, \pi^*(\boldsymbol{p_1})\right) p_z + d_1}{\mathcal{G}(\boldsymbol{p_1})} \\
&\Leftrightarrow \left(\mathcal{G}(\boldsymbol{p_1}) - \boldsymbol{N_1}^T \, \mathsf{K_1}^{-1} \, \pi^*(\boldsymbol{p_1})\right) p_z = d_1 \\
&\Leftrightarrow p_z = \frac{d_1}{\mathcal{G}(\boldsymbol{p_1}) - \boldsymbol{N_1}^T \, \mathsf{K_1}^{-1} \, \pi^*(\boldsymbol{p_1})}
\end{aligned} \tag{2.12}
$$

2.4 Combining Multiple Frames

The above derivations describe the relationship between the first frame and an arbitrary i-th frame. In this section, we begin to use *all* of the non-reference frames to estimate $\mathcal{G}$, i.e. the $\mathcal{G}(\boldsymbol{p_1})$ value for each pixel in the reference frame, from which we can find the depth p_z of the pixel in the reference frame using Equation (2.12).

As we will, from this point forward, primarily be referring to pixels in the reference image, we define $\boldsymbol{p} = \boldsymbol{p_1}$.

Using the brightness constancy constraint, we have that:

$$
\mathcal{I}_i^r(\boldsymbol{p} - \boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p}))) - \mathcal{I}_1(\boldsymbol{p}) \approx 0 \tag{2.13}
$$

where $\mathcal{I}_i^r$ is the plane-registered image obtained by warping $\mathcal{I}_i$ by $\mathsf{H_i}$:

$$
\mathcal{I}_i^r(\boldsymbol{q}) = \mathcal{I}_i[\pi(\mathsf{H_i} \, \pi^*(\boldsymbol{q}))] \tag{2.14}
$$

In principle, we can solve for $\mathcal{G}$, and the extrinsic parameters $(\mathsf{R_i}, \boldsymbol{T_i})$ $i = 2, \ldots, m$ if they are unknown, by minimizing the sum-of-squares residual of the brightness constancy constraint over all images and all pixels:

$$
\sum_{i=2}^{m} \sum_{\boldsymbol{p}} \Big[\mathcal{I}_i^r(\boldsymbol{p} - \boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p}))) - \mathcal{I}_1(\boldsymbol{p}) \Big]^2 \tag{2.15}
$$

In practice, we make the solution more robust by assuming small differences between neighboring $\mathcal{G}(\boldsymbol{p})$ and evaluating the error of each $\mathcal{G}(\boldsymbol{p})$ over $\mathrm{win}(\boldsymbol{p})$, a $k \times k$ window around $\boldsymbol{p}$:

$$\varepsilon(\mathcal{G}, \mathsf{R}_2, \boldsymbol{T_2}, \ldots, \mathsf{R}_\mathsf{m}, \boldsymbol{T_m}) = \sum_{i=2}^{m} \sum_{\boldsymbol{p}} \sum_{\boldsymbol{q} \in \mathrm{win}(\boldsymbol{p})} \Big[\mathcal{I}_i^r(\boldsymbol{q} - \boldsymbol{\delta}_i(\boldsymbol{q}, \mathcal{G}(\boldsymbol{p}))) - \mathcal{I}_1(\boldsymbol{q}) \Big]^2 \quad (2.16)$$

2.5 The Multi-Frame Planar Parallax Algorithm

In [7], Irani et al. assume that the intrinsic parameters $\mathsf{K_i}$ $i=1,\ldots,m$ and extrinsic parameters $(\mathsf{R_i}, \boldsymbol{T_i})$ $i=2,\ldots,m$ are unknown, but that a dominant plane exists in the scene and can be identified independently (and indirectly) in each frame by estimating the homography $\mathsf{H_i}$ between that frame and the reference frame. As is shown in Equation (2.6), warping each frame $\mathcal{I}_i$ to $\mathcal{I}_i^r$ by $\mathsf{H_i}$ as in Equation (2.14) reduces any concern about the intrinsic parameters $\mathsf{K_i}$ $i = 1,\ldots,m$ and extrinsic parameters $(\mathsf{R_i}, \boldsymbol{T_i})$ $i = 2,\ldots,m$ to concern only about the homogeneous epipoles $\boldsymbol{E_i}$ $i = 2,\ldots,m$ and plane distances d_i $i = 2,\ldots,m$, which must be estimated along with $\mathcal{G}$. Note that $\boldsymbol{N_i}$ and d_i cannot be uniquely determined from the estimated $\mathsf{H_i}$ [12].

The MFPP algorithm minimizes an approximation of Equation (2.16) over all frames simultaneously. Irani et al. begin by approximating the brightness constancy constraint in Equation (2.13) as:

$$\mathcal{I}_i^\tau(\boldsymbol{p}) - \begin{bmatrix} \mathcal{I}_x(\boldsymbol{p}) \\ \mathcal{I}_y(\boldsymbol{p}) \end{bmatrix}^T \boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})) \approx 0 \quad \text{where}$$

$$\mathcal{I}_i^\tau(\boldsymbol{p}) = \mathcal{I}_i^r\Big(\boldsymbol{p} - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\Big) - \mathcal{I}_1(\boldsymbol{p}) + \begin{bmatrix} \mathcal{I}_x(\boldsymbol{p}) \\ \mathcal{I}_y(\boldsymbol{p}) \end{bmatrix}^T \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big) \quad (2.17)$$

Here, $\mathcal{I}_x$ and $\mathcal{I}_y$ are the x and y image derivatives of $\mathcal{I}_1$, and a tilde denotes the current estimate. The primary assumption is that $\boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})) - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)$ is small.

Proof:

$$\mathcal{I}_1(\boldsymbol{p}) \approx \mathcal{I}_i^r(\boldsymbol{p} - \boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p}))) = \mathcal{I}_i^r\Big[\boldsymbol{p} - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big) - \big(\boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})) - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\big)\Big]$$

$$\Rightarrow \mathcal{I}_i^r\Big(\boldsymbol{p} - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\Big) \approx \mathcal{I}_1\Big[\boldsymbol{p} + \big(\boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})) - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\big)\Big]$$

$$\approx \mathcal{I}_1(\boldsymbol{p}) + \begin{bmatrix} \mathcal{I}_x(\boldsymbol{p}) \\ \mathcal{I}_y(\boldsymbol{p}) \end{bmatrix}^T \big(\boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})) - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\big)$$

$$\Leftrightarrow \mathcal{I}_i^r\Big(\boldsymbol{p} - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\Big) - \mathcal{I}_1(\boldsymbol{p}) - \begin{bmatrix} \mathcal{I}_x(\boldsymbol{p}) \\ \mathcal{I}_y(\boldsymbol{p}) \end{bmatrix}^T \big(\boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})) - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\big) \approx 0$$

$$\Leftrightarrow \mathcal{I}_i^\tau(\boldsymbol{p}) - \begin{bmatrix} \mathcal{I}_x(\boldsymbol{p}) \\ \mathcal{I}_y(\boldsymbol{p}) \end{bmatrix}^T \boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})) \approx 0 \quad \text{where}$$

$$\mathcal{I}_i^\tau(\boldsymbol{p}) = \mathcal{I}_i^r\Big(\boldsymbol{p} - \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big)\Big) - \mathcal{I}_1(\boldsymbol{p}) + \begin{bmatrix} \mathcal{I}_x(\boldsymbol{p}) \\ \mathcal{I}_y(\boldsymbol{p}) \end{bmatrix}^T \boldsymbol{\delta}_i\Big(\boldsymbol{p}, \tilde{\mathcal{G}}(\boldsymbol{p})\Big) \tag{2.18}$$

Substituting Equation (2.6) into this approximation of the brightness constancy constraint, we find that:

$$\mathcal{I}_i^\tau(\boldsymbol{p}) + \frac{\mathcal{G}(\boldsymbol{p})}{d_i - \mathcal{G}(\boldsymbol{p})\ e_z^{(i)}} \left[\mathcal{I}_x(\boldsymbol{p})\left(e_z^{(i)}\, p_x - e_x^{(i)}\right) + \mathcal{I}_y(\boldsymbol{p})\left(e_z^{(i)}\, p_y - e_y^{(i)}\right)\right] \approx 0 \tag{2.19}$$

where $\boldsymbol{p} = (p_x, p_y)^T$.

After initializing $\mathcal{G}(\boldsymbol{p}) = 0\ \forall \boldsymbol{p}$, $\boldsymbol{E_i} = (0, 0, 1)^T\ \ i{=}2, \ldots, m$, and $d_i = 1\ i{=}2, \ldots, m$, the MFPP algorithm alternates between a local phase, which refines $\mathcal{G}$ while holding the $\boldsymbol{E_i}$'s and $\boldsymbol{d_i}$'s constant, and a global phase, which refines the $\boldsymbol{E_i}$'s and $\boldsymbol{d_i}$'s while holding $\mathcal{G}$ constant. The error function for the local phase is:

$$\varepsilon_{\text{MFPPL}}(\mathcal{G}(\boldsymbol{p})) = \sum_{i=2}^{m} \sum_{\boldsymbol{q} \in \text{win}(\boldsymbol{p})} \left[\begin{array}{l} \mathcal{I}_i^\tau(\boldsymbol{q})\left(\tilde{d}_i - \mathcal{G}(\boldsymbol{p})\ \tilde{e}_z^{(i)}\right) + \\ \qquad \mathcal{G}(\boldsymbol{p})\left[\mathcal{I}_x(\boldsymbol{q})\left(\tilde{e}_z^{(i)}\, q_x - \tilde{e}_x^{(i)}\right) + \mathcal{I}_y(\boldsymbol{q})\left(\tilde{e}_z^{(i)}\, q_y - \tilde{e}_y^{(i)}\right)\right] \end{array}\right]^2 \tag{2.20}$$

where $\boldsymbol{q} = (q_x, q_y)^T$; the error function for the global phase is:

$$\varepsilon_{\mathrm{MFPPG}}(\boldsymbol{E_i}, d_i) = \sum_{\boldsymbol{p}} w_i(\boldsymbol{p}) \left[\begin{array}{l} \mathcal{I}_i^{\tau}(\boldsymbol{p}) \left(d_i - \tilde{\mathcal{G}}(\boldsymbol{p})\ e_z^{(i)} \right) + \\ \qquad \tilde{\mathcal{G}}(\boldsymbol{p}) \left[\mathcal{I}_x(\boldsymbol{p}) \left(e_z^{(i)}\, p_x - e_x^{(i)} \right) + \mathcal{I}_y(\boldsymbol{p}) \left(e_z^{(i)}\, p_y - e_y^{(i)} \right) \right] \end{array} \right]^2 \tag{2.21}$$

where $w_i(\boldsymbol{p}) = \left(\tilde{d}_i - \tilde{\mathcal{G}}(\boldsymbol{p})\ \tilde{e}_z^{(i)} \right)^{-2}$ and $\boldsymbol{p} = (p_x, p_y)^T$. Note that the weights $w_i(\boldsymbol{p})$ ensure that each pixel contributes approximately equally to the optimal $\boldsymbol{E_i}$ and d_i during the global phase; Irani et al. do not find it necessary to ensure that each frame contributes approximately equally to the optimal $\mathcal{G}(\boldsymbol{p})$ in the local phase.

Because many of the terms in Equation (2.12) are unknown in this case, the traditional form of the MFPP algorithm is only able to calculate $\mathcal{G}(\boldsymbol{p})$, not p_z. However, in the remainder of this dissertation, when we use the MFPP algorithm we will assume that the intrinsic parameters $\mathsf{K_i}\ i = 1, \ldots, m$ and extrinsic parameters $(\mathsf{R_i}, \boldsymbol{T_i})\ i = 2, \ldots, m$ are known, which will eliminate the global phase of the optimization and also allow us to calculate p_z from $\mathcal{G}(\boldsymbol{p})$.

Chapter 3

The Recursive Multi-Frame Planar Parallax Algorithm

The Multi-Frame Planar Parallax (MFPP) algorithm summarized in Section 2.5 is a batch method; every gradient computation requires a summation over all images, and then each image is rewarped once $\mathcal{G}$ is updated. With increasing numbers of frames, it will not be feasible to perform such an update at framerate. We would prefer an algorithm that, when a new frame is added, has a constant-time update using sufficient information about the preceding images that has been stored in a set of statistics. This chapter derives such an algorithm: the Recursive Multi-Frame Planar Parallax (RMFPP) algorithm.

3.1 Enabling Observations

We begin with a simple example at a single pixel. Consider estimating p_z from disparities δ_k, as in Section 1.2. Suppose $\delta_k = (k-1) \cdot \frac{1}{p_z}$. The disparity as a function of k is linear, as shown in Figure 3.1 (a), and its slope is the reciprocal of p_z. Image $\mathcal{I}_1$ does not change

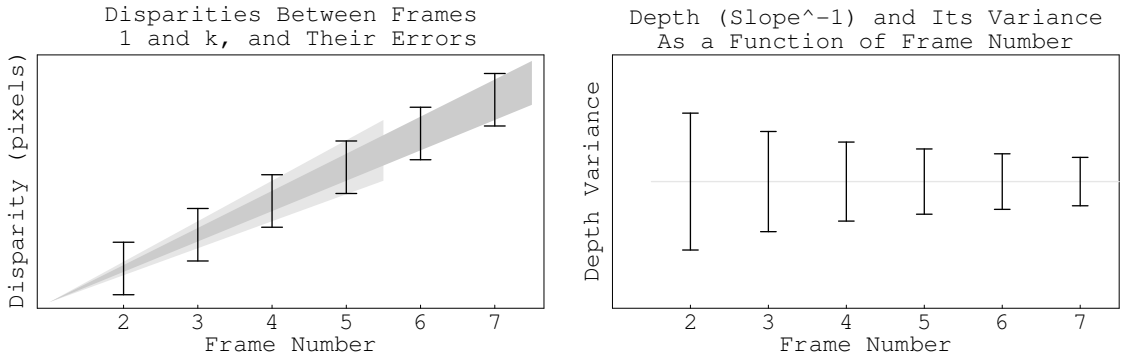


Figure 3.1: (a) Disparity as a function of frame number for translating camera; error bars indicate error independent of frame number. (b) Uncertainty in p_z as a function of frame number.

so the error bars in disparity have constant width. The wider triangle ending at frame 5 represents the set of lines with slope $(p_z^{-1}) \pm$ one standard deviation from the mean slope. The triangle ending at frame 7 shows the same but after incorporating frames 6 and 7.

Now consider Figure 3.1 (b). We show the standard deviation of an estimate of p_z as a function of frame number, assuming that at the k-th frame we incorporate all frames up to and including k. The uncertainty in p_z is transformed to uncertainty in disparities through scaling by $\partial\delta_k/\partial p_z = (k-1)/p_z^2$.

We observe: (1) small refinements to p_z later in the sequence do not significantly affect δ_k's near the beginning of the sequence; (2) as long as these triangles' widths do not exceed the radius for which the linearity of intensity is valid, there is no need to rewarp the images. If the disparities do not exceed the range of linearity, we can create a recursive algorithm.

3.2 Recursive Cost Function

We now turn the minimization of the MFPP algorithm's batch cost function $\varepsilon_{\mathrm{MFPPL}}$, as defined in Equation (2.20), into a recursive procedure. We begin by restoring the nonlinearity in $\mathcal{G}(\boldsymbol{p})$ from the constraint in Equation (2.19); instead of multiplying both sides of the constraint by the denominator as in the MFPP algorithm, we linearize the cost function around current estimates of $\mathcal{G}(\boldsymbol{p})$ below. We replace the sum over the window $\mathrm{win}(\boldsymbol{p})$ with the average over the window, to account for the possibility of the window being different in different frames due to the invalidity of one or more input terms—we discuss the issue of validity when we give the complete algorithm in Section 3.5. We also introduce weights α_i on the contribution of each frame in the sequence to the total non-recursive error $\varepsilon_{\mathrm{NR}}$:

$$\begin{aligned}
\varepsilon_{\mathrm{NR}}(\mathcal{G}(\boldsymbol{p})) &= \sum_{i=2}^{m} \frac{\alpha_i}{\left|\mathrm{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\mathrm{win}(\boldsymbol{p})} \left[\mathcal{I}_i^{\tau}(\boldsymbol{q}) + \frac{\mathcal{G}(\boldsymbol{p})}{d_i - \mathcal{G}(\boldsymbol{p})\ e_z^{(i)}} \left[\begin{array}{l} \mathcal{I}_x(\boldsymbol{q}) \left(e_z^{(i)}\, q_x - e_x^{(i)} \right) + \\ \qquad \mathcal{I}_y(\boldsymbol{q}) \left(e_z^{(i)}\, q_y - e_y^{(i)} \right) \end{array} \right] \right]^2 \\
&= \sum_{i=2}^{m} \frac{\alpha_i}{\left|\mathrm{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\mathrm{win}(\boldsymbol{p})} \left[\mathcal{I}_i^{\tau}(\boldsymbol{q}) + \frac{\mathcal{G}(\boldsymbol{p})}{d_i - \mathcal{G}(\boldsymbol{p})\ e_z^{(i)}}\ \mathcal{I}_i^{\kappa}(\boldsymbol{q}) \right]^2 \\
&\triangleq \sum_{i=2}^{m} \frac{\alpha_i}{\left|\mathrm{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\mathrm{win}(\boldsymbol{p})} r_i(\boldsymbol{q}, \mathcal{G}(\boldsymbol{p}))^2 \\
&\triangleq \sum_{i=2}^{m} \alpha_i\, c_i(\mathcal{G}(\boldsymbol{p}))
\end{aligned} \tag{3.1}$$

where $\boldsymbol{q} = (q_x, q_y)^T$. Note that we have simplified the notation by defining $\mathcal{I}_i^{\kappa}(\boldsymbol{q})$:

$$\mathcal{I}_i^{\kappa}(\boldsymbol{q}) = \mathcal{I}_x(\boldsymbol{q}) \left(e_z^{(i)}\, q_x - e_x^{(i)} \right) + \mathcal{I}_y(\boldsymbol{q}) \left(e_z^{(i)}\, q_y - e_y^{(i)} \right) \tag{3.2}$$

In the recursive formulation we propose a cost function that is linearized in past terms but iterated until convergence on the latest image. We denote by $\tilde{\mathcal{G}}^{(i)}$ the final estimate of $\mathcal{G}$ after the last iteration on the i-th frame or, in the case of the most recent image, the estimate of $\mathcal{G}$ after the most recent iteration. Then, for example after the i-th frame has arrived, we

define $\varepsilon^{(i)}_{\mathrm{RMFPP}}$ to be the per-pixel cost up to and including the i-th frame:

$$\begin{aligned}
\varepsilon^{(i)}_{\mathrm{RMFPP}}(\mathcal{G}(\boldsymbol{p})) &\triangleq \sum_{j=2}^{i} \alpha_j \, c_j(\mathcal{G}(\boldsymbol{p})) \\
&= \alpha_i \, c_i(\mathcal{G}(\boldsymbol{p})) + \sum_{j=2}^{i-1} \alpha_j \, c_j \Big[\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) + \Big(\mathcal{G}(\boldsymbol{p}) - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) \Big] \\
&\approx \alpha_i \, c_i(\mathcal{G}(\boldsymbol{p})) + \sum_{j=2}^{i-1} \alpha_j \underbrace{\left[\begin{array}{l} \dfrac{c_j'' \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big)}{2!} \Big(\mathcal{G}(\boldsymbol{p}) - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big)^2 + \\ \quad \dfrac{c_j' \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big)}{1!} \Big(\mathcal{G}(\boldsymbol{p}) - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) + \\ \quad c_j \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) \end{array} \right]}_{\text{2nd-order Taylor series in } \mathcal{G}(\boldsymbol{p}) \text{ at } \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})} \\
&\triangleq \alpha_i \, c_i(\mathcal{G}(\boldsymbol{p})) + \sum_{j=2}^{i-1} \alpha_j \left[A_j(\boldsymbol{p}) \; \mathcal{G}(\boldsymbol{p})^2 + B_j(\boldsymbol{p}) \; \mathcal{G}(\boldsymbol{p}) + C_j(\boldsymbol{p}) \right] \\
&\triangleq \alpha_i \, c_i(\mathcal{G}(\boldsymbol{p})) + \Sigma A^{(i-1)}(\boldsymbol{p}) \; \mathcal{G}(\boldsymbol{p})^2 + \Sigma B^{(i-1)}(\boldsymbol{p}) \; \mathcal{G}(\boldsymbol{p}) + \Sigma C^{(i-1)}(\boldsymbol{p})
\end{aligned} \tag{3.3}$$

where

$$\begin{aligned}
A_j(\boldsymbol{p}) &= \frac{1}{2} c_j'' \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) \\
B_j(\boldsymbol{p}) &= -\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \; c_j'' \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) + c_j' \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) \\
C_j(\boldsymbol{p}) &= \frac{1}{2} \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})^2 \; c_j'' \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \; c_j' \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big) + c_j \Big(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p}) \Big)
\end{aligned} \tag{3.4}$$

Here, A_j, B_j, and C_j are respectively the coefficients of $\mathcal{G}(\boldsymbol{p})^2$, $\mathcal{G}(\boldsymbol{p})$ and 1 in the Taylor series expansion of the per-pixel cost in the j-th frame; $\Sigma A^{(i-1)}$, $\Sigma B^{(i-1)}$, and $\Sigma C^{(i-1)}$ are the corresponding coefficients in the Taylor series expansion of the total past per-pixel cost, i.e. cumulative sums of the weighted A_j, B_j, and C_j through the $(i-1)$-st frame. Note that each A_j in ΣA, for example, is the result of a linearization (of c_j) about a different point from the terms before it, namely whichever was the latest estimate of $\mathcal{G}$.

3.3 Minimization of the Recursive Cost Function

Minimization of $\varepsilon^{(i)}_{\mathrm{RMFPP}}$ takes a particularly convenient form if we use Newton's method, which also features a fast rate of convergence. Then we can iteratively refine $\mathcal{G}$ by using:

$$\begin{aligned}
\mathcal{G}(\boldsymbol{p}) &\leftarrow \tilde{\mathcal{G}}^{(i)}(\boldsymbol{p}) - \frac{\varepsilon^{(i)'}_{\mathrm{RMFPP}}\Big(\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\Big)}{\varepsilon^{(i)''}_{\mathrm{RMFPP}}\Big(\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\Big)} \\
&= \tilde{\mathcal{G}}^{(i)}(\boldsymbol{p}) - \frac{\alpha_i\, c_i'\Big(\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\Big) + 2\,\Sigma A^{(i-1)}(\boldsymbol{p})\;\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p}) + \Sigma B^{(i-1)}(\boldsymbol{p})}{\alpha_i\, c_i''\Big(\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\Big) + 2\,\Sigma A^{(i-1)}(\boldsymbol{p})} \\
&= \frac{\alpha_i\left[\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\;c_i''\Big(\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\Big) - c_i'\Big(\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\Big)\right] - \Sigma B^{(i-1)}(\boldsymbol{p})}{\alpha_i\, c_i''\Big(\tilde{\mathcal{G}}^{(i)}(\boldsymbol{p})\Big) + 2\,\Sigma A^{(i-1)}(\boldsymbol{p})} \\
&= -\frac{\alpha_i\, B_i(\boldsymbol{p}) + \Sigma B^{(i-1)}(\boldsymbol{p})}{2\,\alpha_i\, A_i(\boldsymbol{p}) + 2\,\Sigma A^{(i-1)}(\boldsymbol{p})} \\
&= -\frac{\Sigma B^{(i)}(\boldsymbol{p})}{2\,\Sigma A^{(i)}(\boldsymbol{p})}
\end{aligned} \tag{3.5}$$

Equivalently, we can iteratively minimize the recursive cost function over frames $2 \ldots i$:

$$\begin{aligned}
\varepsilon^{(i)}_{\mathrm{RMFPP}}(\mathcal{G}(\boldsymbol{p})) &= \alpha_i\, c_i(\mathcal{G}(\boldsymbol{p})) + \Sigma A^{(i-1)}(\boldsymbol{p})\;\mathcal{G}(\boldsymbol{p})^2 + \Sigma B^{(i-1)}(\boldsymbol{p})\;\mathcal{G}(\boldsymbol{p}) + \Sigma C^{(i-1)}(\boldsymbol{p}) \\
&\approx \Sigma A^{(i)}(\boldsymbol{p})\;\mathcal{G}(\boldsymbol{p})^2 + \Sigma B^{(i)}(\boldsymbol{p})\;\mathcal{G}(\boldsymbol{p}) + \Sigma C^{(i)}(\boldsymbol{p})
\end{aligned} \tag{3.6}$$

In both cases, the only terms that change between iterations are $A_i(\boldsymbol{p})$, $B_i(\boldsymbol{p})$, and $C_i(\boldsymbol{p})$.

To fully specify the minimization procedure for $\varepsilon^{(i)}_{\mathrm{RMFPP}}$, we must derive A_j and B_j. We begin by deriving c_j' and c_j'':

$$\begin{aligned}
c_j'(\mathcal{G}(\boldsymbol{p})) &= \frac{1}{\big|\mathrm{win}(\boldsymbol{p})\big|}\sum_{\boldsymbol{q}\in\mathrm{win}(\boldsymbol{p})}\big[r_j(\boldsymbol{q},\mathcal{G}(\boldsymbol{p}))^2\big]' \\
&= \frac{2}{\big|\mathrm{win}(\boldsymbol{p})\big|}\sum_{\boldsymbol{q}\in\mathrm{win}(\boldsymbol{p})} r_j(\boldsymbol{q},\mathcal{G}(\boldsymbol{p}))\;r_j'(\boldsymbol{q},\mathcal{G}(\boldsymbol{p})) \\
c_j''(\mathcal{G}(\boldsymbol{p})) &= \frac{2}{\big|\mathrm{win}(\boldsymbol{p})\big|}\sum_{\boldsymbol{q}\in\mathrm{win}(\boldsymbol{p})}\big[r_j(\boldsymbol{q},\mathcal{G}(\boldsymbol{p}))\;r_j'(\boldsymbol{q},\mathcal{G}(\boldsymbol{p}))\big]' \\
&= \frac{2}{\big|\mathrm{win}(\boldsymbol{p})\big|}\sum_{\boldsymbol{q}\in\mathrm{win}(\boldsymbol{p})}\big[r_j'(\boldsymbol{q},\mathcal{G}(\boldsymbol{p}))^2 + r_j(\boldsymbol{q},\mathcal{G}(\boldsymbol{p}))\;r_j''(\boldsymbol{q},\mathcal{G}(\boldsymbol{p}))\big]
\end{aligned}$$

$$\approx \frac{2}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} r'_j(\boldsymbol{q}, \mathcal{G}(\boldsymbol{p}))^2 \tag{3.7}$$

where $r'_j = \partial r_j / \partial \mathcal{G}(\boldsymbol{p})$ and we ignore second-order terms of r_j. We also derive r'_j:

$$\begin{aligned} r'_j(\boldsymbol{q}, \mathcal{G}(\boldsymbol{p})) &= \left[\mathcal{I}^{\tau}_j(\boldsymbol{q}) + \frac{\mathcal{G}(\boldsymbol{p})}{d_j - \mathcal{G}(\boldsymbol{p})\ e_z^{(j)}}\ \mathcal{I}^{\kappa}_j(\boldsymbol{q}) \right]' \\ &= \frac{\left(d_j - \mathcal{G}(\boldsymbol{p})\ e_z^{(j)}\right) + \mathcal{G}(\boldsymbol{p})\ e_z^{(j)}}{\left(d_j - \mathcal{G}(\boldsymbol{p})\ e_z^{(j)}\right)^2}\ \mathcal{I}^{\kappa}_j(\boldsymbol{q}) \\ &= \frac{d_j}{\left(d_j - \mathcal{G}(\boldsymbol{p})\ e_z^{(j)}\right)^2}\ \mathcal{I}^{\kappa}_j(\boldsymbol{q}) \end{aligned} \tag{3.8}$$

Finally, we can fully specify A_j and B_j:

$$\begin{aligned} A_j(\boldsymbol{p}) &= \frac{1}{2} c''_j\left(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right) \\ &\approx \frac{1}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} r'_j\left(\boldsymbol{q}, \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right)^2 \\ &= \frac{1}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} \frac{d_j^2}{\left(d_j - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ e_z^{(j)}\right)^4}\ \mathcal{I}^{\kappa}_j(\boldsymbol{q})^2 \end{aligned}$$

$$\begin{aligned} B_j(\boldsymbol{p}) &= -\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ c''_j\left(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right) + c'_j\left(\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right) \\ &\approx -\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ \frac{2}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} r'_j\left(\boldsymbol{q}, \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right)^2 + \\ &\qquad \frac{2}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} r_j\left(\boldsymbol{q}, \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right)\ r'_j\left(\boldsymbol{q}, \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right) \\ &= \frac{2}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} r'_j\left(\boldsymbol{q}, \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right) \left[r_j\left(\boldsymbol{q}, \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right) - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ r'_j\left(\boldsymbol{q}, \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\right) \right] \\ &= \frac{2}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} \frac{d_j}{\left(d_j - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ e_z^{(j)}\right)^2} \mathcal{I}^{\kappa}_j(\boldsymbol{q}) \left[\begin{array}{c} \mathcal{I}^{\tau}_j(\boldsymbol{q}) + \frac{\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})}{d_j - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ e_z^{(j)}} \mathcal{I}^{\kappa}_j(\boldsymbol{q}) - \\ \frac{\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ d_j}{\left(d_j - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ e_z^{(j)}\right)^2} \mathcal{I}^{\kappa}_j(\boldsymbol{q}) \end{array} \right] \\ &= \frac{2}{\left|\text{win}(\boldsymbol{p})\right|} \sum_{\boldsymbol{q}\in\text{win}(\boldsymbol{p})} \frac{d_j}{\left(d_j - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ e_z^{(j)}\right)^2} \mathcal{I}^{\kappa}_j(\boldsymbol{q}) \left[\mathcal{I}^{\tau}_j(\boldsymbol{q}) - \frac{\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})^2\ e_z^{(j)}}{\left(d_j - \tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})\ e_z^{(j)}\right)^2} \mathcal{I}^{\kappa}_j(\boldsymbol{q}) \right] \end{aligned} \tag{3.9}$$

3.4 Cost Function Comparison

We compare the RMFPP cost function from Equation (3.6) to the MFPP cost function from Equation (2.20), where we have added weights $\alpha_j/|\text{win}(\boldsymbol{p})|$ to the MFPP cost function (and so have renamed it $\varepsilon_{\text{MFPPW}}$), and have simplified the notation by using $\mathcal{I}_j^{\kappa}(\boldsymbol{q})$. We define $\varepsilon_{\text{MFPPW}}(\mathcal{G}(\boldsymbol{p})) = \varepsilon_{\text{MFPPW}}^{(m)}(\mathcal{G}(\boldsymbol{p}))$, where $\varepsilon_{\text{MFPPW}}^{(i)}(\mathcal{G}(\boldsymbol{p}))$ is the cost for frames $2 \ldots i$:

$$
\begin{aligned}
\varepsilon_{\text{MFPPW}}^{(i)}(\mathcal{G}(\boldsymbol{p})) &= \sum_{j=2}^{i} \frac{\alpha_j}{|\text{win}(\boldsymbol{p})|} \sum_{\boldsymbol{q} \in \text{win}(\boldsymbol{p})} \left[\mathcal{I}_j^{\tau}(\boldsymbol{q}) \left(d_j - \mathcal{G}(\boldsymbol{p})\ e_z^{(j)} \right) + \mathcal{G}(\boldsymbol{p})\ \mathcal{I}_j^{\kappa}(\boldsymbol{q}) \right]^2 \\
&= \sum_{j=2}^{i} \frac{\alpha_j}{|\text{win}(\boldsymbol{p})|} \sum_{\boldsymbol{q} \in \text{win}(\boldsymbol{p})} \left[\left(\mathcal{I}_j^{\kappa}(\boldsymbol{q}) - \mathcal{I}_j^{\tau}(\boldsymbol{q})\ e_z^{(j)} \right) \mathcal{G}(\boldsymbol{p}) + \mathcal{I}_j^{\tau}(\boldsymbol{q})\ d_j \right]^2 \\
&= \sum_{j=2}^{i} \frac{\alpha_j}{|\text{win}(\boldsymbol{p})|} \sum_{\boldsymbol{q} \in \text{win}(\boldsymbol{p})} \left[\begin{array}{l} \left(\mathcal{I}_j^{\kappa}(\boldsymbol{q}) - \mathcal{I}_j^{\tau}(\boldsymbol{q})\ e_z^{(j)} \right)^2 \mathcal{G}(\boldsymbol{p})^2 + \\ \quad 2 \left(\mathcal{I}_j^{\kappa}(\boldsymbol{q}) - \mathcal{I}_j^{\tau}(\boldsymbol{q})\ e_z^{(j)} \right) \mathcal{I}_j^{\tau}(\boldsymbol{q})\, d_j\, \mathcal{G}(\boldsymbol{p}) + \\ \quad \mathcal{I}_j^{\tau}(\boldsymbol{q})^2\, d_j^2 \end{array} \right] \\
&\triangleq \sum_{j=2}^{i} \alpha_j \left[A_j^{\text{MFPP}}(\boldsymbol{p})\, \mathcal{G}(\boldsymbol{p})^2 + B_j^{\text{MFPP}}(\boldsymbol{p})\, \mathcal{G}(\boldsymbol{p}) + C_j^{\text{MFPP}}(\boldsymbol{p}) \right] \\
&\triangleq \Sigma A_{\text{MFPP}}^{(i)}(\boldsymbol{p})\, \mathcal{G}(\boldsymbol{p})^2 + \Sigma B_{\text{MFPP}}^{(i)}(\boldsymbol{p})\, \mathcal{G}(\boldsymbol{p}) + \Sigma C_{\text{MFPP}}^{(i)}(\boldsymbol{p}) \\
\Rightarrow \mathcal{G}(\boldsymbol{p}) &\leftarrow -\frac{\Sigma B_{\text{MFPP}}^{(i)}(\boldsymbol{p})}{2\, \Sigma A_{\text{MFPP}}^{(i)}(\boldsymbol{p})}
\end{aligned}
\tag{3.10}
$$

Therefore, analogous to $A_j(\boldsymbol{p})$ and $B_j(\boldsymbol{p})$, we can write:

$$
\begin{aligned}
A_j^{\text{MFPP}}(\boldsymbol{p}) &= \frac{1}{|\text{win}(\boldsymbol{p})|} \sum_{\boldsymbol{q} \in \text{win}(\boldsymbol{p})} \left(\mathcal{I}_j^{\kappa}(\boldsymbol{q}) - \mathcal{I}_j^{\tau}(\boldsymbol{q})\ e_z^{(j)} \right)^2 \\
B_j^{\text{MFPP}}(\boldsymbol{p}) &= \frac{2}{|\text{win}(\boldsymbol{p})|} \sum_{\boldsymbol{q} \in \text{win}(\boldsymbol{p})} \left(\mathcal{I}_j^{\kappa}(\boldsymbol{q}) - \mathcal{I}_j^{\tau}(\boldsymbol{q})\ e_z^{(j)} \right) \mathcal{I}_j^{\tau}(\boldsymbol{q})\, d_j
\end{aligned}
\tag{3.11}
$$

We note that $\varepsilon_{\text{MFPPW}}^{(i)}(\mathcal{G}(\boldsymbol{p}))$ is independent of $\tilde{\mathcal{G}}^{(j)}(\boldsymbol{p})$, $j = 2, \ldots, i$; this is not the case for $\varepsilon_{\text{RMFPP}}^{(i)}(\mathcal{G}(\boldsymbol{p}))$, where we have restored more of the problem's nonlinearity. We also note that the cost functions $\varepsilon_{\text{MFPPW}}^{(i)}(\mathcal{G}(\boldsymbol{p}))$ and $\varepsilon_{\text{RMFPP}}^{(i)}(\mathcal{G}(\boldsymbol{p}))$ are equivalent when $d_j = d_i$ and $e_z^{(j)} = 0$, $j = 2, \ldots, i$; this occurs when the camera's line of sight is perpendicular to, and the camera maintains a constant distance from, the reference plane.

3.5 Complete Algorithm

We are now ready to give the complete Recursive Multi-Frame Planar Parallax algorithm. The state variables $\mathcal{G}$, $\Sigma\mathsf{A}$, $\Sigma\mathsf{B}$, residual, and numValid, each the same dimensions as $\mathcal{I}_1$, are initialized to 0. Every image $\mathcal{I}_i$ utilizing the reference image $\mathcal{I}_1$ is processed as follows:

process_image(α_i, $\mathcal{I}_i$, $\mathsf{K_i}$, $\mathsf{R_i}$, $\boldsymbol{T_i}$, $\mathcal{I}_1$, $\mathcal{I}_x$, $\mathcal{I}_y$, $\mathsf{K_1}$, $\boldsymbol{N_1}$, d_1,
$\mathcal{G}$, $\Sigma\mathsf{A}$, $\Sigma\mathsf{B}$, residual, numValid):

compute $\mathcal{I}_i^r$ as in Equation (2.14)
compute $\mathcal{I}_i^\kappa$ as in Equation (3.2)
for up to n_{iter} iterations:
  $\{\Sigma\mathsf{AIter}, \Sigma\mathsf{BIter}\} = \{\Sigma\mathsf{A}, \Sigma\mathsf{B}\}$
  for each pixel $\boldsymbol{p}$:
    calculate $\boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p}))$ as in Equation (2.6)
    calculate $\mathcal{I}_i^w(\boldsymbol{p}) \triangleq \mathcal{I}_i^r(\boldsymbol{p} - \boldsymbol{\delta}_i(\boldsymbol{p}, \mathcal{G}(\boldsymbol{p})))$
    compute $\mathcal{I}_i^r(\boldsymbol{p})$ as in Equation (2.17)
  end for
  for each pixel $\boldsymbol{p}$:
    calculate $\{\mathsf{A_i}(\boldsymbol{p}), \mathsf{B_i}(\boldsymbol{p})\}$ as in Equation (3.9)
    $\{\Sigma\mathsf{AIter}(\boldsymbol{p}), \Sigma\mathsf{AIter}(\boldsymbol{p})\} += \alpha_i \{\mathsf{A_i}(\boldsymbol{p}), \mathsf{B_i}(\boldsymbol{p})\}$
    estimate $\mathcal{G}(\boldsymbol{p})$ as in Equation (3.5) using $\{\Sigma\mathsf{AIter}, \Sigma\mathsf{BIter}\}$
  end for
  if average change in valid region of $\mathcal{G}$ is small, then break
end for
$\{\Sigma\mathsf{A}, \Sigma\mathsf{B}\} = \{\Sigma\mathsf{AIter}, \Sigma\mathsf{BIter}\}$
for each pixel $\boldsymbol{p}$ where $\mathcal{I}_i^w(\boldsymbol{p})$ is valid:
  $\mathsf{residual}(\boldsymbol{p}) += \alpha_i |\mathcal{I}_1(\boldsymbol{p}) - \mathcal{I}_i^w(\boldsymbol{p})|$
  $\mathsf{numValid}(\boldsymbol{p})++$
end for

We mention several specific implementation issues below, which are: (1) how to prevent changes to points that go out of view; (2) how to handle regions of low texture; (3) how to decide whether a pixel is an outlier; (4) how to integrate measurements from multiple runs into a single coherent map; and (5) when to choose a new reference image.

The first two implementation issues are special cases of a more general strategy that is required for implementation of the algorithm: at each step where a new value is computed, the inputs must be evaluated to determine whether the output is valid, and the validity of values must be propagated through all calculations. For example, the image derivatives $\mathcal{I}_x$ and $\mathcal{I}_y$ are not valid at the edge pixels if a 3x3 matrix is convolved with the image to determine the derivative; therefore, the edge pixels of $\mathcal{I}_i^\kappa$ are also not valid. Pixels of $\mathcal{I}_i^r$ that the homography $\mathsf{H_i}$ maps to a pixel outside of $\mathcal{I}_i$ are marked as invalid; this occurs when a point goes out of view. In addition, $\mathsf{A_i}(\boldsymbol{p})$ and $\mathsf{B_i}(\boldsymbol{p})$ are invalid if $\mathsf{A_i}(\boldsymbol{p})/\mathsf{B_i}(\boldsymbol{p})$ is too large or NaN—in particular, the quotient is NaN where both image derivatives are zero (insufficient texture).

After we have processed all images $\mathcal{I}_i$ utilizing the reference image $\mathcal{I}_1$ as above, we generate a depth estimate, using Equation (2.12), for each pixel in the reference frame that satisfies a few basic criteria (discussed below). For each pixel in the reference frame that passes the previous test, we calculate a 3D world point (X, Y, Z) using $(\mathsf{R_1}, \boldsymbol{T_1})$; we also calculate an estimate of the uncertainty in the world Z coordinate, var(Z). Finally, we filter the 3D world points for outliers (discussed below).

We motivate the calculation of var(Z) by observing that the update equation for $\mathcal{G}(\boldsymbol{p})$, Equation (3.5), can be written as the quotient of two weighted averages:

$$\begin{aligned}\frac{-\Sigma B^{(i)}(\boldsymbol{p})}{2\,\Sigma A^{(i)}(\boldsymbol{p})} &\triangleq \frac{-\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,B_j(\boldsymbol{p})}{2\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,A_j(\boldsymbol{p})} \\ &= \frac{-\left(\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\right)^{-1}\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,B_j(\boldsymbol{p})}{2\left(\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\right)^{-1}\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,A_j(\boldsymbol{p})}\end{aligned}$$

$$\triangleq \frac{-\Sigma\alpha^{(i)}(\boldsymbol{p})^{-1}\,\Sigma B^{(i)}(\boldsymbol{p})}{2\,\Sigma\alpha^{(i)}(\boldsymbol{p})^{-1}\,\Sigma A^{(i)}(\boldsymbol{p})} \tag{3.12}$$

where we index α_j by $\boldsymbol{p}$ to allow $\alpha_j(\boldsymbol{p})$ to be zero if $\boldsymbol{p}$ is not valid in frame j. Because we have multiple estimates $A_j(\boldsymbol{p})$ of $A(\boldsymbol{p})$, we calculate the empirical weighted average and weighted variance of $A(\boldsymbol{p})$. Then we calculate the variance of the weighted sum $\left(\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\right)^{-1}\sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,A_j(\boldsymbol{p})$. We perform the same steps for the $B_j(\boldsymbol{p})$ values. We propagate the variances of the weighted sums through the Jacobian of Equation (3.5) to obtain var($\mathcal{G}(\boldsymbol{p})$), and through the Jacobian of Equation (2.12) to obtain var(p_z). Finally, we propagate the variance through the Jacobian of the affine transformation $(\mathsf{R}_1, \boldsymbol{T_1})$ to obtain var(Z). Note that this procedure underestimates the variance because the estimates $A_j(\boldsymbol{p})$ (and $B_j(\boldsymbol{p})$) are correlated; however, it underestimates the variance of all pixels in all reference frames, and we find that combining multiple estimates of the same point in the final map using these variances works well in practice.

To allow the calculation of these variances, we maintain the following statistics in addition to $\Sigma A^{(i)}(\boldsymbol{p})$ and $\Sigma B^{(i)}(\boldsymbol{p})$:

$$\begin{aligned}
\Sigma AA^{(i)}(\boldsymbol{p}) &= \sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,A_j(\boldsymbol{p})^2 \\
\Sigma BB^{(i)}(\boldsymbol{p}) &= \sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,B_j(\boldsymbol{p})^2 \\
\Sigma AB^{(i)}(\boldsymbol{p}) &= \sum_{j=2}^{i}\alpha_j(\boldsymbol{p})\,A_j(\boldsymbol{p})\,B_j(\boldsymbol{p}) \\
\Sigma\alpha^{(i)}(\boldsymbol{p}) &= \sum_{j=2}^{i}\alpha_j(\boldsymbol{p}) \\
\Sigma\alpha\alpha^{(i)}(\boldsymbol{p}) &= \sum_{j=2}^{i}\alpha_j(\boldsymbol{p})^2
\end{aligned} \tag{3.13}$$

Because these statistics are maintained similarly to $\Sigma A^{(i)}(\boldsymbol{p})$ and $\Sigma B^{(i)}(\boldsymbol{p})$, we have omitted them from the algorithm above.

The criteria for generating a depth estimate p_z for a pixel $\boldsymbol{p}$ are: it must be valid in at least 5 images since the last reference change (encoded in $\mathsf{numValid}(\boldsymbol{p})$), it must have an average absolute residual (determined by $\mathsf{residual}(\boldsymbol{p}) \,/\, \Sigma\alpha(\boldsymbol{p})$) no larger than 10, and it must be no less than 2 pixels from the edge of the reference image.

The 3D world point outlier filter begins by iteratively rejecting points that are too many standard deviations above the mean in their 3D X or Y coordinates. This is followed by iteratively discarding all points with 3D (X, Y) coordinates in any 25-th (5x5 equal-sized regions) of the portion of the X-Y plane spanned by the data that contains less than 0.1% of the points. Finally, points are again iteratively rejected based on being too many standard deviations above the mean, this time in their 3D Z coordinates or in their 3D Z uncertainty var(Z).

We integrate the 3D world points into a fixed-grid elevation and appearance map. For reconstructing large areas of terrain where the region to be explored is not known a priori, a modular map is best; by modular we mean that we store fixed-resolution, e.g. 16×16, blocks, and add blocks to the map only when data is available in that region. All values contained in the same grid square (not block) are optimally combined using their estimated variances.

Finally, a new reference image $\mathcal{I}_1$ is set when: (*i*) the percentage of pixels that are valid after aligning the next image is below a given threshold (we use 50%); or (*ii*) after a set maximum number of images since the last reference change to reduce mapping latency (we use 20). World points are generated upon every change in reference image.

Chapter 4

Results

In this chapter, we present results of running the RMFPP algorithm on synthetic sequences, as well as on real sequences recorded from a robotic helicopter. The results on synthetic data allow us to evaluate the algorithm analytically; the results on real data allow us to show its applicability to the real world.

4.1 Method

In addition to direct analysis of the RMFPP algorithm, we present comparisons between the RMFPP algorithm and the following algorithms: the MFPP batch algorithm discussed in Chapter 2, and variants of the RMFPP algorithm that only use the first or last frame in the sequence following each reference frame—we refer to these as the Closest and Farthest variants, respectively. The Closest and Farthest variants are particular fixed-baseline and wide-baseline stereo algorithms, respectively, that take advantage of the fact that the scene is distant and known to be close to a reference plane—they are not intended to represent the most advanced stereo algorithms in their respective classes, only the ill-posed nature

of triangulation in the fixed-baseline case with a distant scene (Closest), and the effects of image and geometric noise on reconstruction when using only a single pair (Closest and Farthest), while being directly comparable to the RMFPP algorithm because of their similarity to that method.

All of the experiments in this chapter use our single-threaded C++ implementation of the RMFPP and MFPP algorithms (including the Closest and Farthest variants), which makes extensive use of BLAS [9], LAPACK [1], and OpenCV [2]. They are run on an Intel Core 2 Duo 2.0 GHz CPU with 4 GB of RAM and openSUSE GNU/Linux 10.3 x64-64.

Unless otherwise specified, we use the following parameters for all experiments: Images are 320x240 pixels. Synthetic images (and their ground-truth depth images) are rendered in the 3D modeling program Blender [4]. Virtual cameras are downward-pointing (line of sight perpendicular to the terrain), have a focal length of 350, move at 50 meters/sec, and capture 5 frames/sec—consecutive images are rendered 10 meters apart. For the MFPP algorithm, we use a 4-level image pyramid below, and a 3-level image pyramid at and above, 700 meters above the average terrain elevation; for the Farthest variant of the RMFPP algorithm, we use a 4-level image pyramid below, and a 3-level image pyramid at and above, 1000 meters above the average terrain elevation; for the RMFPP algorithm and its Closest variant, we do not use an image pyramid. We use the following per-image weights: $\alpha_i = 1$ for the MFPP algorithm and $\alpha_i = (i-1)^2$ for the RMFPP algorithm; α_i is irrelevant for the Closest and Farthest variants, so we use $\alpha_i = 1$. These pyramid sizes and per-image weights are experimentally determined to give the best results for each algorithm; we will discuss the effects of both in the following section. The maximum number of iterations $n_{iter} = 20$, and the reference frame is changed only when there is less than 50% overlap between the current frame and the reference frame.

4.2 Translation Experiments

For our initial algorithm comparison and analysis, we use images rendered from a synthetic sinusoidal terrain, shown in Figure 4.1. We render sequences of images and ground-truth depth data from translating downward-looking cameras at the following heights above the mean terrain elevation: 500, 600, 700, 800, 900, 1000, 1250, 1500, 1750, 2000, 2250, and 2500 meters. Each sequence is of the appropriate length so that there is a 50% overlap between the image of the reference plane (horizontal, at the mean terrain elevation) in the first and last frames: at 500 meters, it contains 18 frames; at 1000 meters, 35 frames; and at 2000 meters, 69 frames. We show sample rendered images in Figure 4.2; each rendered texture image is the reference image for our translation experiment at the given height, and the corresponding rendered depth image is the ground truth for the depth image that we reconstruct during the experiment. Note that the RMFPP and MFPP cost functions are equivalent for this experiment, as discussed in Section 3.4; here we are showing the effectiveness of our recursive method while using equivalent cost functions across all algorithms, although our runtime results for the RMFPP algorithm (and its Closest and Farthest variants) still reflect the usage of the more computationally expensive RMFPP cost function.

We first show sample reconstructions obtained by processing the translation image sequences with the RMFPP algorithm. Figure 4.3 contains the (unfiltered) reconstructed depth for each pixel in the reference frame that is visible for at least 5 frames, as well as a histogram of depth reconstruction errors and the number of optimization iterations used for each image, for the sample rendered images in Figure 4.2. We obtain a median absolute error of 0.8 meters at an elevation of 500 meters; 1.8 meters at an elevation of 1000 meters; and 3.7 meters at an elevation of 2000 meters. The runtime is less than 0.17 sec/frame at all elevations—well within the 5 frames/sec rate of the original sequences, due to the small number of optimization iterations that are required after the first frames.

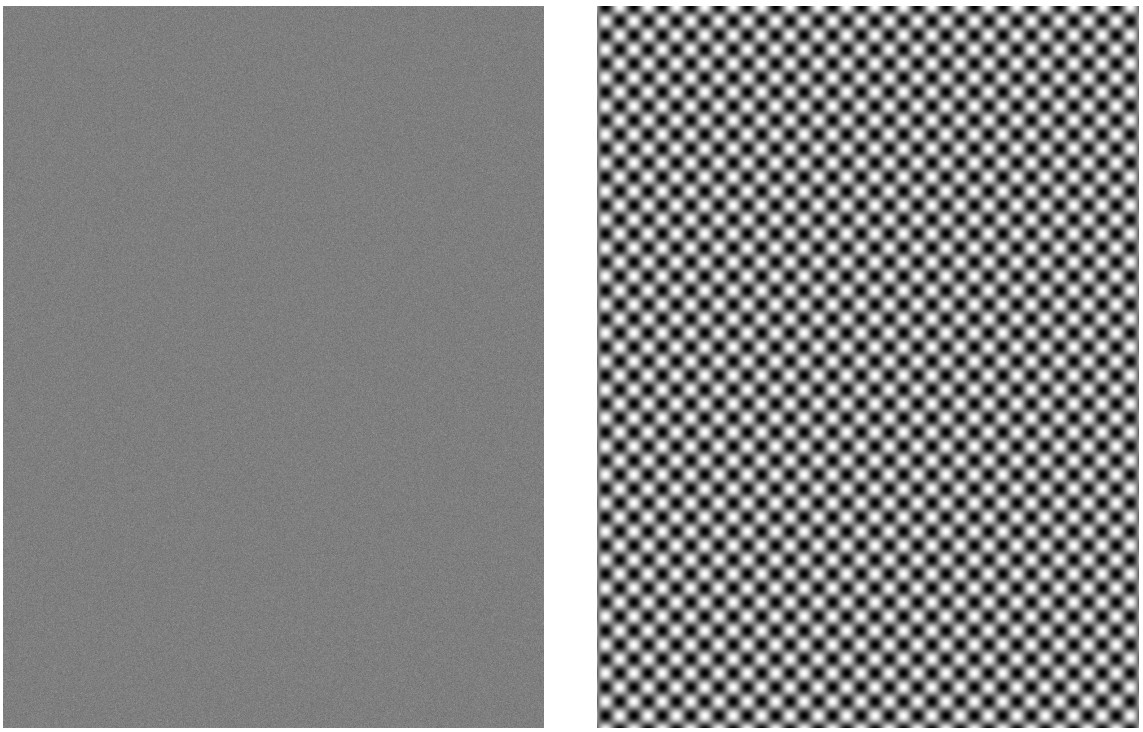
We compare the accuracy and runtime of all four algorithms described above at all twelve rendered elevations. Figure 4.4 shows the results. The Closest algorithm, which only uses the reference frame and the following frame (a fixed 10-meter baseline stereo system), quickly leaves the top of the figure—it reaches a median absolute error of 100 meters at an elevation of 2500 meters as its triangulation becomes ever more ill-posed. The remaining three algorithms, RMFPP, MFPP, and Farthest, all use the full range of the sequence (at least the first and last frames), and have similar median absolute errors in these experiments. However, the RMFPP algorithm is 3 to 4 times faster than the MFPP algorithm per frame. At least as importantly for real-time vision, the RMFPP algorithm returns its result approximately 0.2 sec after the last frame, while the MFPP algorithm does not begin processing until all frames are captured. Hence, we have succeeded in our primary goal for the RMFPP algorithm—replicating the accuracy of the MFPP algorithm, while reducing its runtime to a range that is acceptable for real-time use.

We observe that the similarity and linearity of the reconstruction errors for the RMFPP, MFPP, and Farthest algorithms in the previous experiment correspond to the expected error derived for each algorithm in Section 1.2, with correlation coefficient $\rho \approx 1$ for the RMFPP and MFPP algorithms. To show the advantage of using all frames between the first and the last, as in the RMFPP and MFPP algorithms, instead of just the first and the last as in the Farthest algorithm, we add noise to all frames except the reference frame (thus adding noise to the difference between the appearance of corresponding points from the reference frame to any subsequent frame). Figures 4.5 and 4.6 show that the multiple estimates of $\mathcal{G}$ provided by the RMFPP and MFPP algorithms result in significantly better performance in the presence of image noise.

To justify the per-image weights α_i used in the RMFPP and MFPP algorithms, we show a comparison of different weight degrees, namely $\alpha_i = (i-1)^\beta$ for different values of the exponent β, at different levels of image noise. Note the choice of $(i-1)$, as the images

corresponding to the reference frame $\mathcal{I}_1$ are $\mathcal{I}_2$ through $\mathcal{I}_m$. Figure 4.7 shows the results for both algorithms. We conclude that the MFPP algorithm obtains its best results with $\alpha_i = 1$ (the weights implicitly used by Irani et al. in [7]), and that the RMFPP algorithm degrades significantly below $\alpha_i = (i-1)^2$. We choose to use $\alpha_i = (i-1)^2$ for the RMFPP algorithm to balance the benefit of high weight degree against its effect of diminishing the contribution of earlier frames; we note that the low-elevation result degrades slightly above this point, and that the high-altitude improvement above this point is small, particularly when $\sigma_{img} \leq 5$. If the expected image noise and elevation above the terrain are large, we recommend using a larger weight degree for the RMFPP algorithm.

Lastly, we discuss the use of image pyramids. Figure 4.8 shows the effect of using different numbers of pyramid levels with the four algorithms. We conclude that an image pyramid is unnecessary for the RMFPP algorithm; it achieves the same benefit (that of reducing the size of the disparities between the plane-rectified image $\mathcal{I}_i^r$ and the reference image $\mathcal{I}_1$ via a previous estimate) by using its estimate of $\mathcal{G}$ from the previous frame. In fact, we find in our experiments that using a pyramid slows down the algorithm's optimization considerably because, even though the initial estimate for the coarsest pyramid level is a subsampled version of $\mathcal{G}$ from the previous frame, the process of downscaling the previous frame's final estimate of $\mathcal{G}$ to the coarsest level and then upscaling it to the finest level makes it less accurate than the final estimate of $\mathcal{G}$ from the previous frame at the original image size, even with the refinement optimizations in the coarser pyramid levels. In the case of the Closest algorithm, we find that an image pyramid is not beneficial—the large errors that result from that algorithm are largely due to ill-posed triangulation, not to large disparities between $\mathcal{I}_i^r$ and $\mathcal{I}_1$. Finally, the MFPP and Farthest algorithms both clearly benefit from image pyramids with 4 levels at lower elevations and 3 levels at higher elevations (where the additional subsampling at the fourth level is not necessary and slightly decreases the algorithm's accuracy). All of the previous and subsequent results use the best number of pyramid levels for each algorithm at each elevation.



(a) (b)

Figure 4.1: Synthetic texture (a) and sinusoidal terrain (b) with resolution 1 meter/pixel. The texture is uniform in $[0, 255]$. The terrain has elevation $100 \cdot \sin(0.02X) \cdot \sin(0.02Y)$ where $X \in [-3000, 3000)$ meters and $Y \in (-1000, 7000]$ meters.

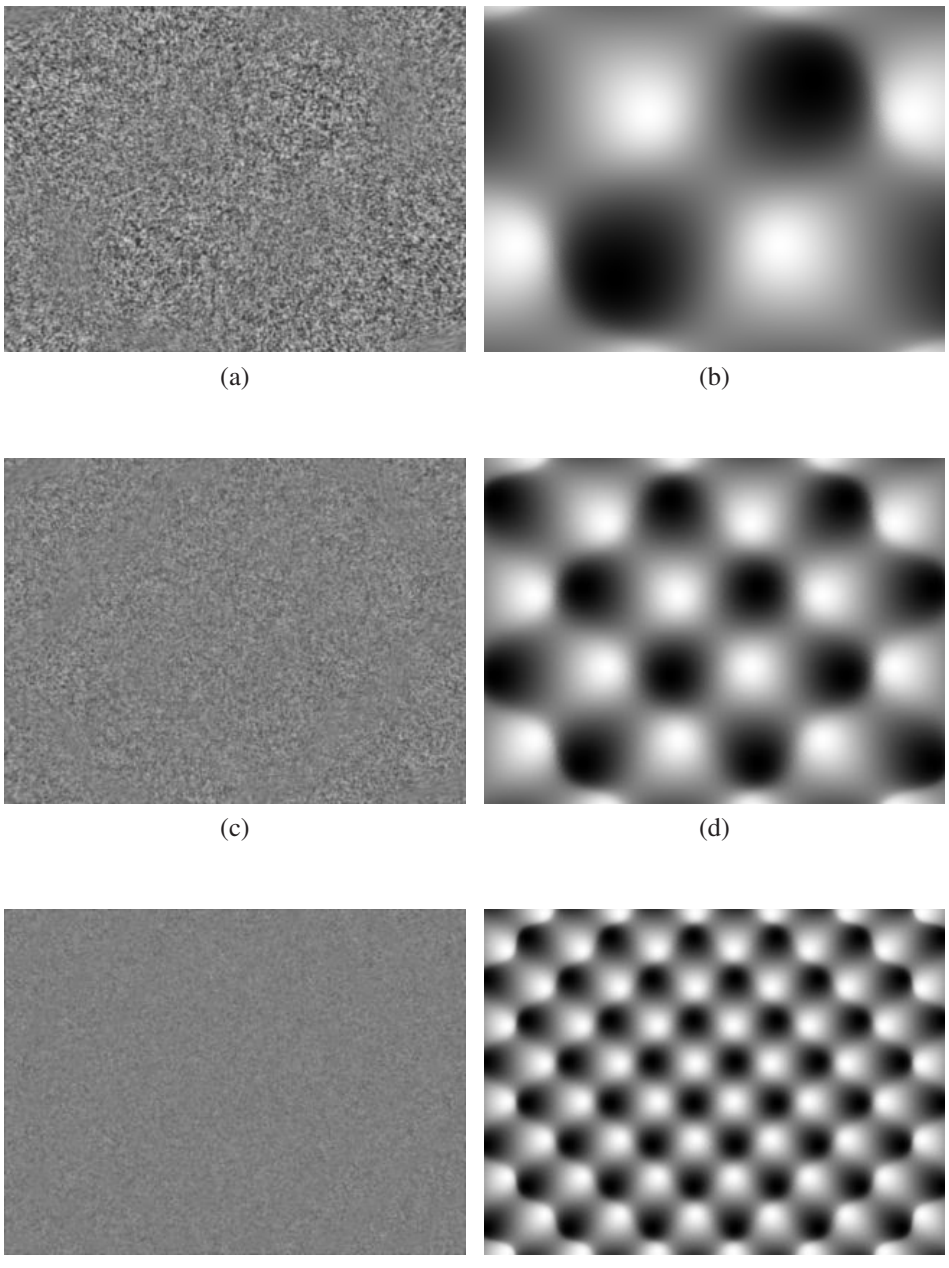
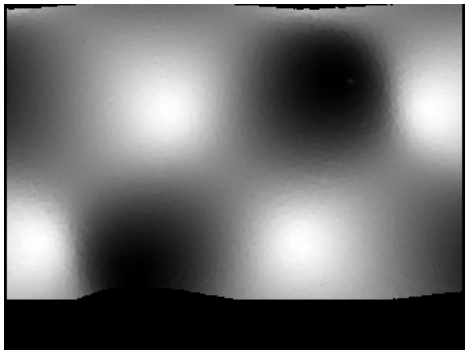
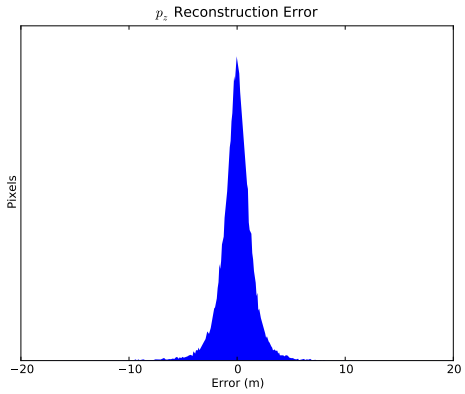


Figure 4.2: Rendered texture and depth from downward-pointing cameras positioned 500 meters (a,b), 1000 meters (c,d), and 2000 meters (e,f) above the mean of the synthetic sinusoidal terrain.

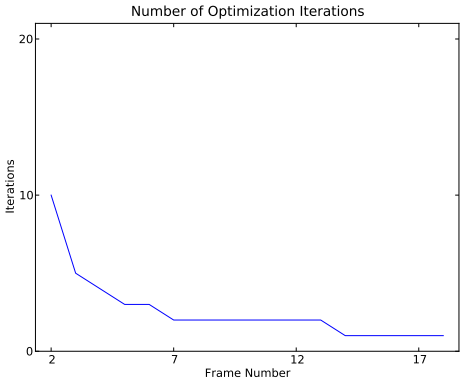


(a)

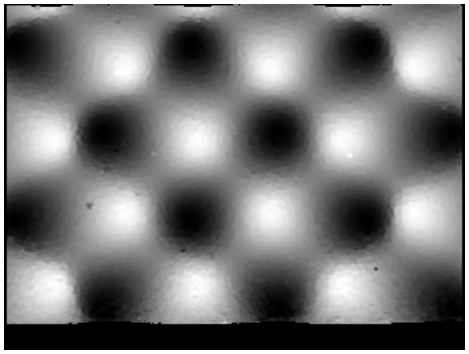
(b)


p_z Reconstruction Error
Pixels
−20
−10
0
10
20
Error (m)

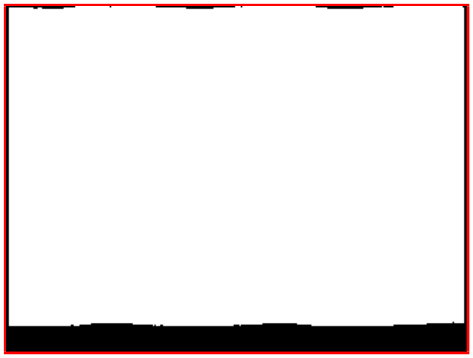
(c)


Number of Optimization Iterations
Iterations
0
10
20
2
7
12
17
Frame Number

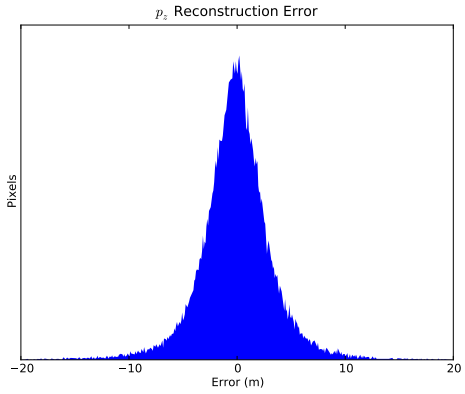
(d)



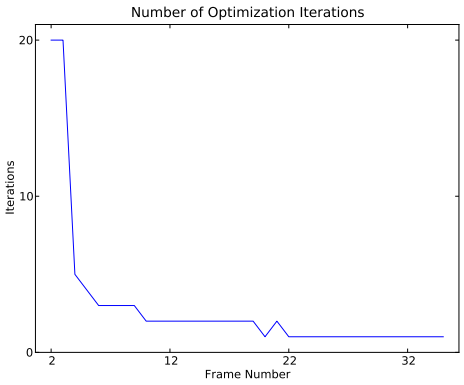
(e)



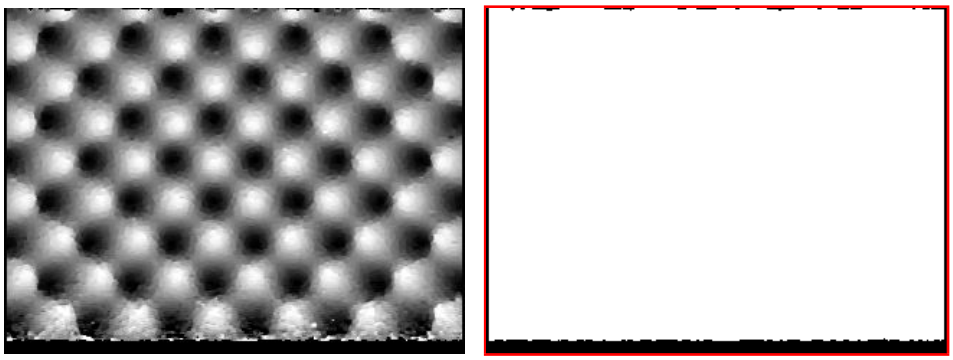
(f)


p_z Reconstruction Error
Pixels
−20
−10
0
10
20
Error (m)

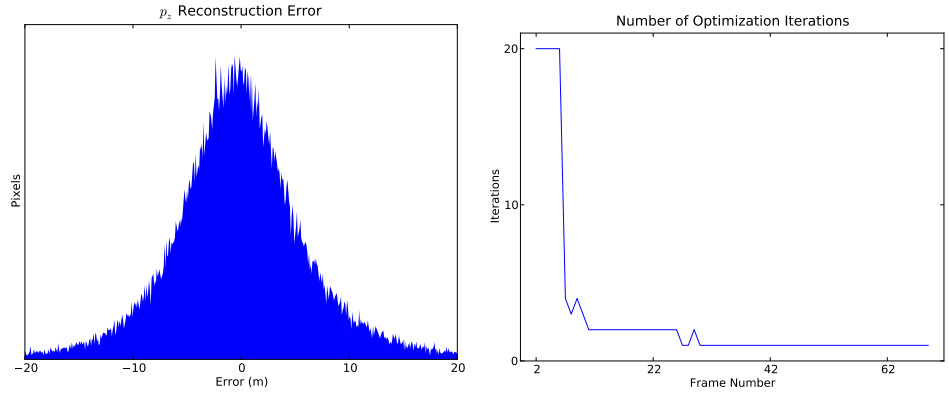
(g)


Number of Optimization Iterations
Iterations
0
10
20
2
12
22
32
Frame Number

(h)

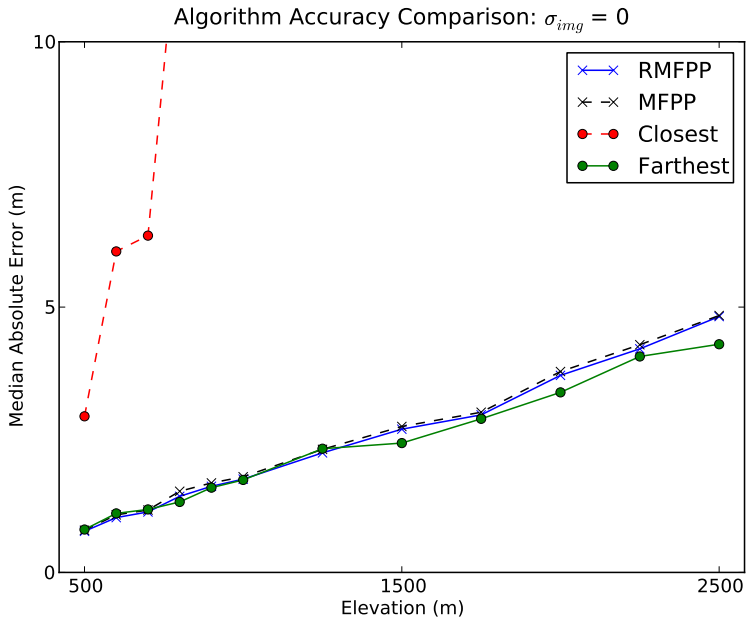


(i) (j)

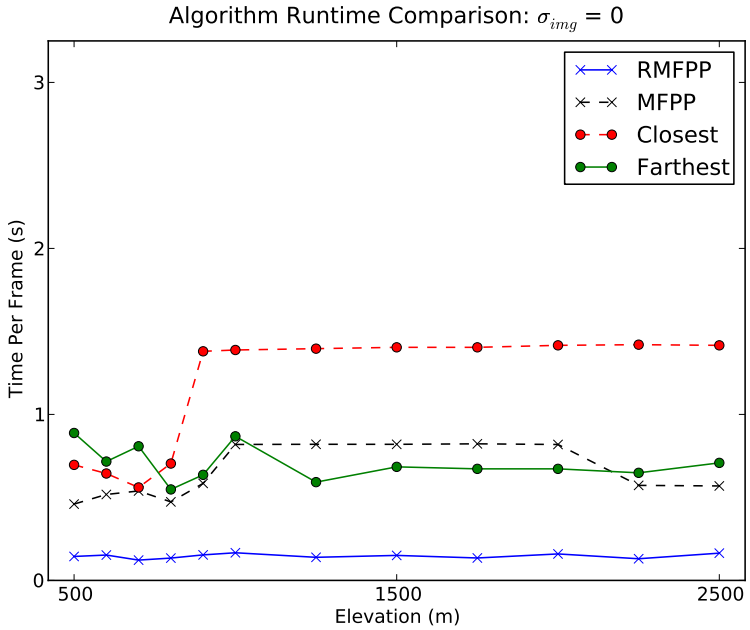


(k) (l)

Figure 4.3: Results of RMFPP reconstruction using image sequences rendered from a translating downward-looking camera 500 meters (a-d), 1000 meters (e-h), and 2000 meters (i-l) above the mean of the sinusoidal terrain. Each result set includes the reconstructed depth (a,e,i), the valid mask for the reconstructed depth (b,f,j), the histogram of depth reconstruction errors (c,g,k), and the number of optimization iterations performed for each image (d,h,l). The reference image and ground-truth depth images are those in Figure 4.2. Note that the depth reconstructions are unfiltered. The red box around the depth valid mask is not part of the mask itself—it serves only to separate the mask image from the surrounding whitespace.

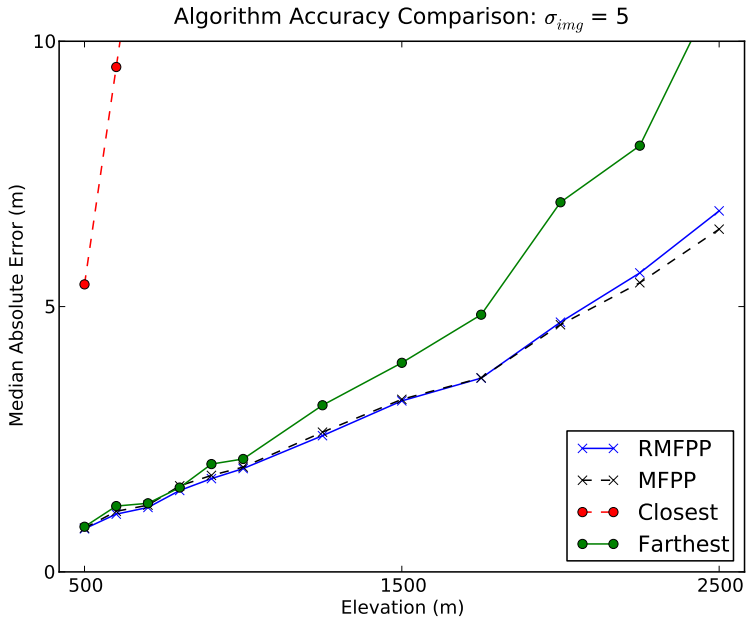


(a)

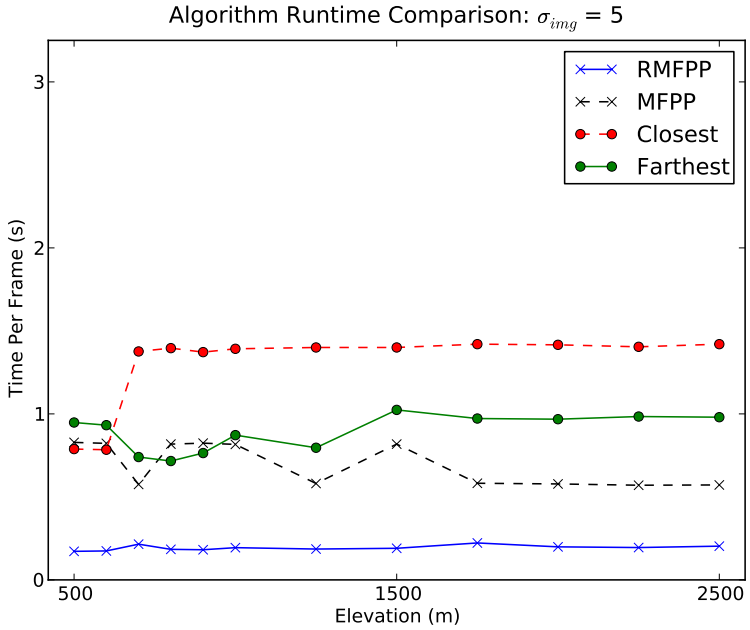


(b)

Figure 4.4: Comparison of algorithm accuracy (a) and runtime (b) based on translation experiments with no noise. The error for Closest continues similarly above (a), but is removed from the plot above 10 meters of median absolute error to allow greater detail for the other algorithms.

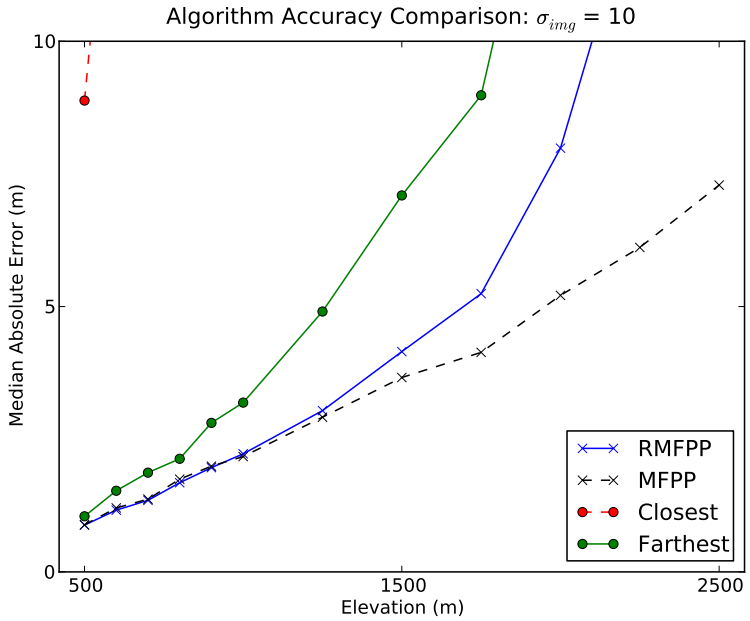


(a)

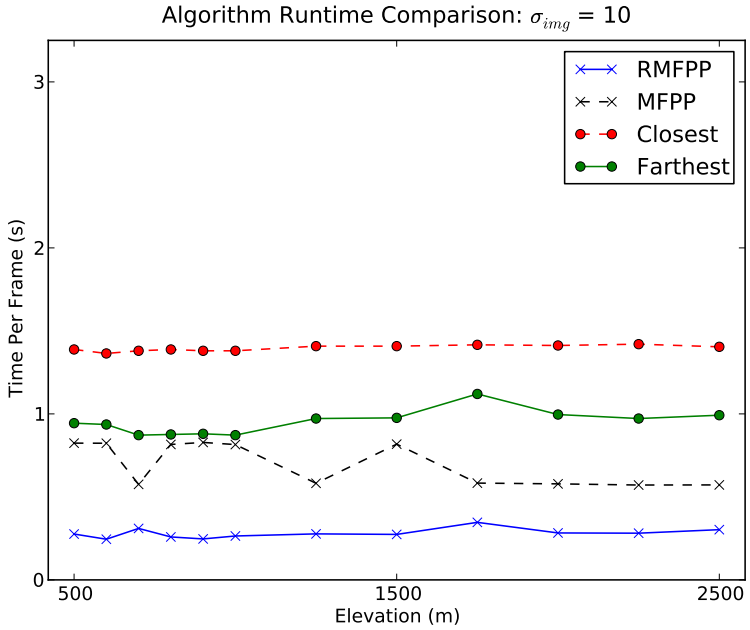


(b)

Figure 4.5: Comparison of algorithm accuracy (a) and runtime (b) based on translation experiments with image noise that is Gaussian with standard deviation $\sigma_{img} = 5$. The error for Closest continues similarly above (a), but is removed from the plot above 10 meters of median absolute error to allow greater detail for the other algorithms. Note the difference in accuracy, in the presence of image noise, between the single-frame algorithm Farthest and the multi-frame algorithms RMFPP and MFPP.



(a)



(b)

Figure 4.6: Comparison of algorithm accuracy (a) and runtime (b) based on translation experiments with image noise that is Gaussian with standard deviation σ_{img} = 10. The error for Closest continues similarly above (a), but is removed from the plot above 10 meters of median absolute error to allow greater detail for the other algorithms. Note the difference in accuracy, in the presence of image noise, between the single-frame algorithm Farthest and the multi-frame algorithms RMFPP and MFPP.

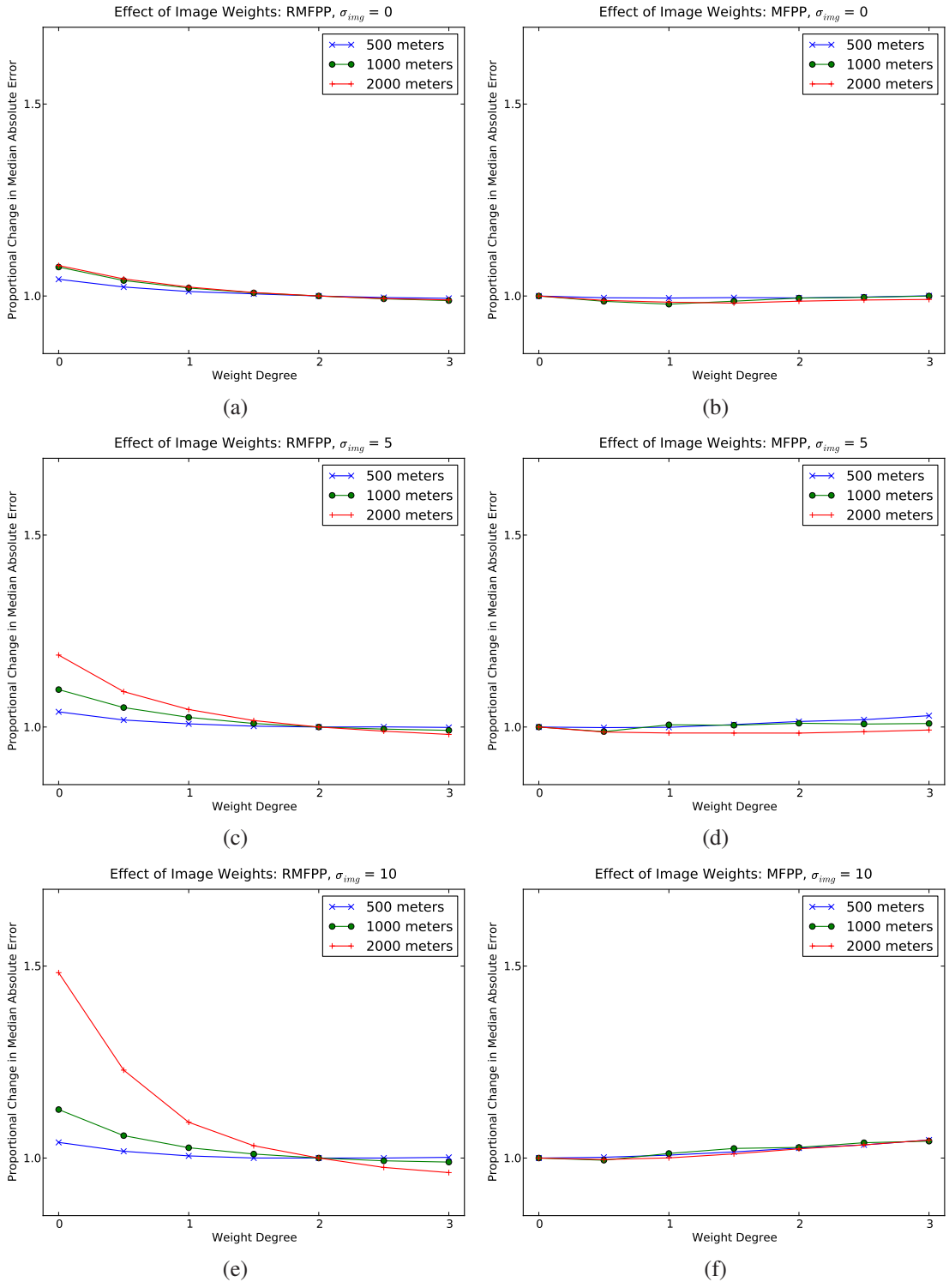


Figure 4.7: Relative accuracy of the RMFPP (a,c,e) and MFPP (b,d,f) algorithms with different degrees of the per-image weight α_i: with $\sigma_{img} = 0$ (a,b), $\sigma_{img} = 5$ (c,d), $\sigma_{img} = 10$ (e,f). Note that the best MFPP results are obtained with $\alpha_i = (i-1)^0$ (degree 0); also note significant degradation of the RMFPP results below $\alpha_i = (i-1)^2$ (degree 2).

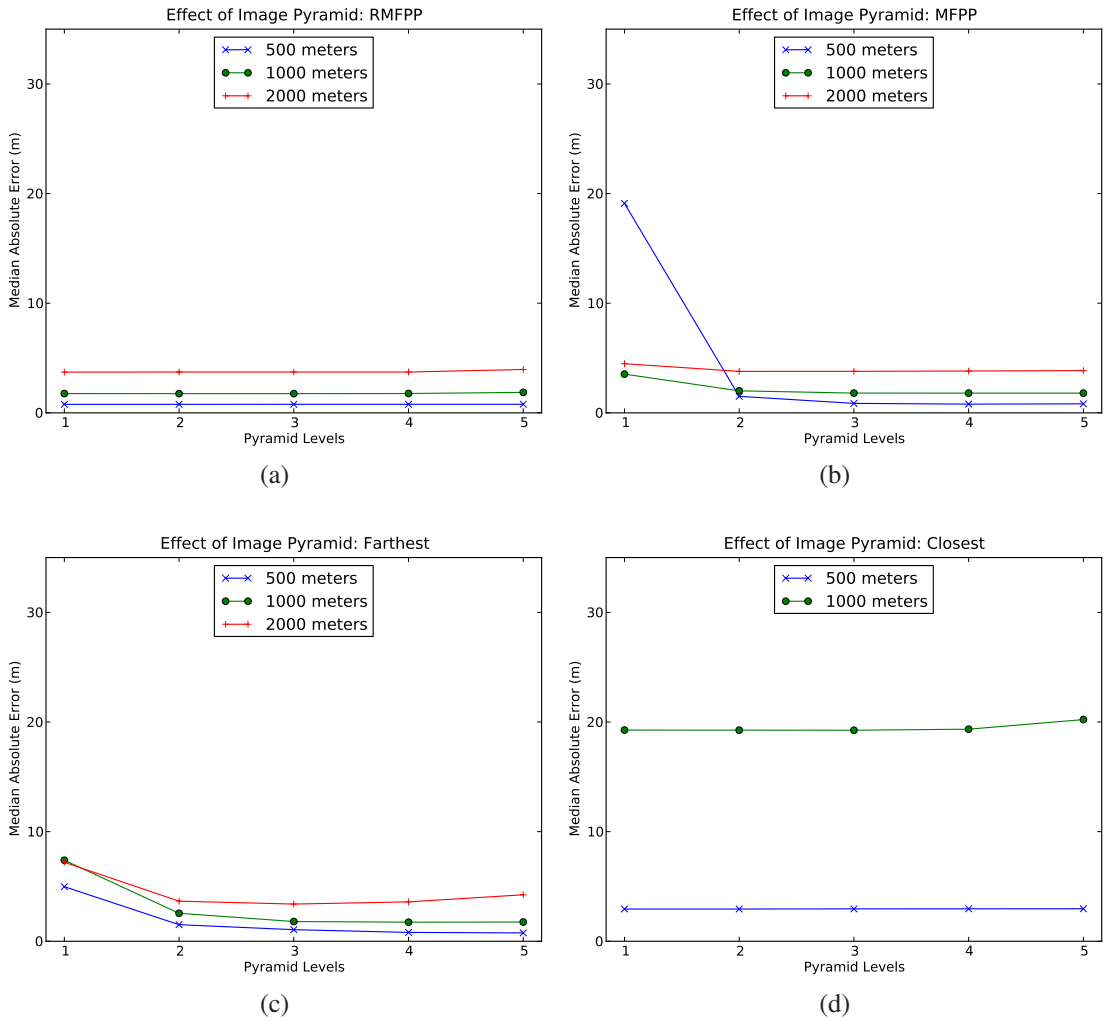


Figure 4.8: Relative accuracy of each algorithm with different numbers of pyramid levels (with one pyramid level equivalent to no image pyramid): RMFPP (a), MFPP (b), Farthest (c), and Closest (d). Note that an image pyramid is not necessary for RMFPP and does not improve Closest, but does improve MFPP and Farthest; the results in the previous and subsequent figures use the optimal number of pyramid levels for each algorithm at each elevation.

4.3 Effect of Geometric Noise

When we derived the RMFPP algorithm from the MFPP algorithm in Chapter 3, we removed the MFPP algorithm's interspersed optimization over the multi-camera and scene geometry. Hence, we have assumed that this geometry is known, or can be estimated well enough that the RMFPP algorithm degrades gracefully instead of failing catastrophically due to a violation of its assumptions. In this section, we explore what range of errors in estimating the required geometric parameters is "good enough": the relative rotation $\mathsf{R_i}$ and translation $\boldsymbol{T_i}$ between the reference frame and each subsequent frame i, and the distance d_1 of the reference frame from the reference plane. Note that the distance between subsequent frames and the reference plane, the d_i, are calculated using the plane normal N_1, d_1, and the relative translation T_i.

The rotation error is parameterized as an axis-angle vector in $\mathbb{R}^3$ with entries that are chosen from the Gaussian distribution with standard deviation $\sigma_{\mathsf{R_i}}$; the rotation matrix that corresponds to this axis-angle vector is multiplied to the right of the relative rotation $\mathsf{R_i}$. The translation error is parameterized as a vector in $\mathbb{R}^3$ with entries that are chosen from the Gaussian distribution with standard deviation $\sigma_{\boldsymbol{T_i}}$; the translation noise vector is added to the relative translation $\boldsymbol{T_i}$. The height error is a scalar in $\mathbb{R}$ that is chosen from the Gaussian distribution with standard deviation σ_{d_1}; the height noise is added to d_1.

We run experiments at elevations of 500, 1000, and 2000 meters, with the same sequences as in the previous section. As shown in Figure 4.9, we find that there is, indeed, a range of errors in $\mathsf{R_i}$, $\boldsymbol{T_i}$, and d_1 in which the RMFPP algorithm degrades gracefully. For $\mathsf{R_i}$, we find that the estimation error $\sigma_{\mathsf{R_i}}$ must be less than 10^{-3}; for $\boldsymbol{T_i}$, the allowable estimation error $\sigma_{\boldsymbol{T_i}}$ scales linearly with elevation and must be less than $d_1/500$. We also find that the algorithm is relatively insensitive to error in d_1—it degrades gracefully through a broad range of σ_{d_1}.

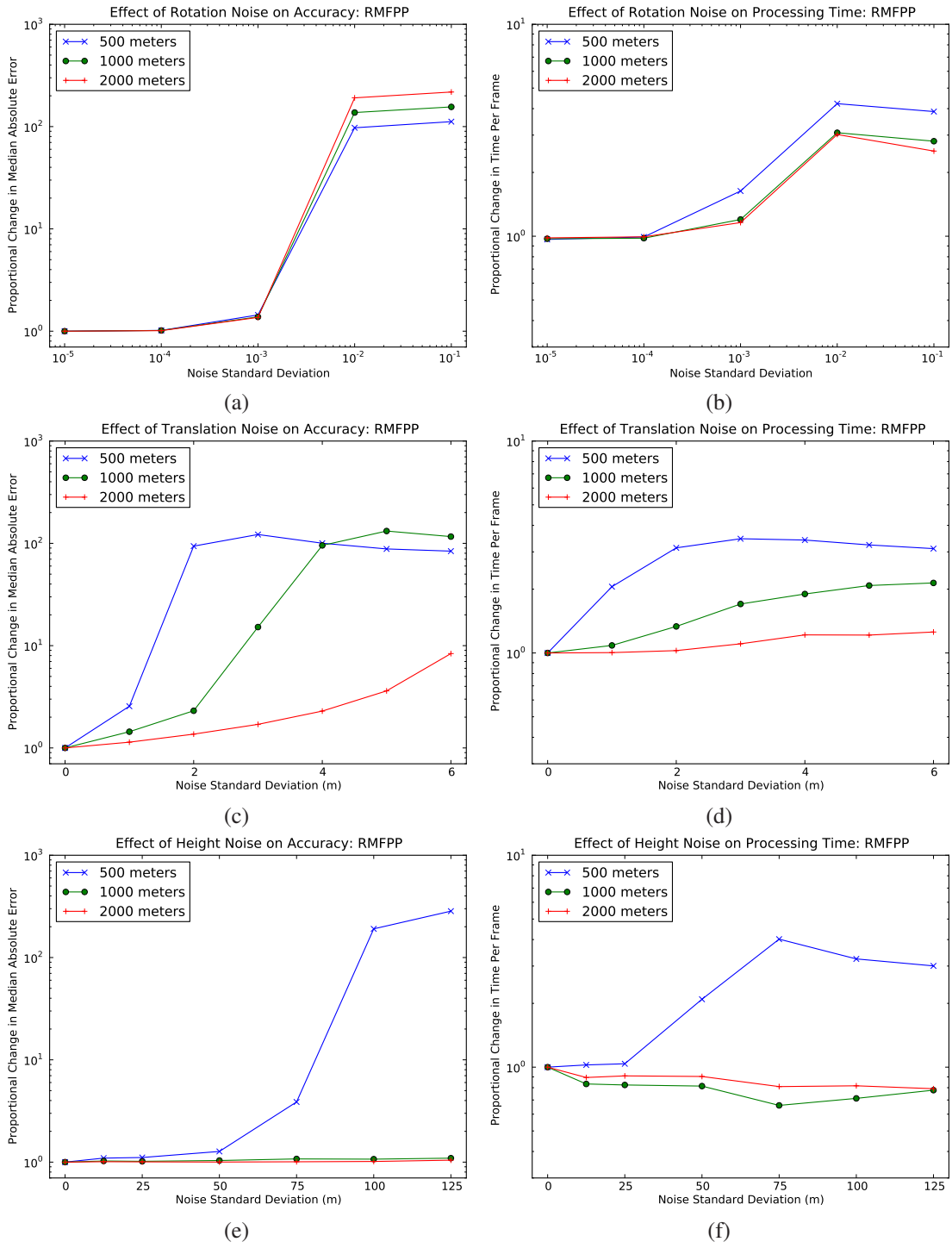


Figure 4.9: Relative accuracy and runtime of the RMFPP algorithm with three types of Gaussian noise on the multi-camera and scene geometry: noise in the relative rotation $\sigma_{\mathrm{R_i}}$ (a,b) and translation $\sigma_{\boldsymbol{T_i}}$ (c,d) between the the current camera and the reference camera, and noise σ_{d_i} in the distance between the reference camera and the mean elevation of the terrain (e,f). Note the graceful degradation of the algorithm's accuracy and runtime through $\sigma_{\mathrm{R_i}} = 10^{-3}$ and $\sigma_{\boldsymbol{T_i}} = d_1/500$, and through a broad range of σ_{d_1}.

4.4 Effect of Forward Camera Tilt

Although we have used a downward-facing camera in the previous experiments to correspond to the theoretical error analysis in Section 1.2, in practice we would like to tilt the camera forward so that we can reconstruct more terrain that is in front of the vehicle. Figure 4.10 shows the result of running the RMFPP and MFPP algorithms on image sequences that are rendered from the sinusoidal terrain in Figure 4.1 using virtual cameras with varying degrees of forward tilt. In order to obtain the best performance from the MFPP algorithm, we compensate for the faster foreground image motion by adding a fifth pyramid level at 500 meters for 40° and above, a fourth pyramid level at 1000 meters for 25° and above, a fifth pyramid level at 1000 meters for 45°, and a fourth pyramid level at 2000 meters for 30° and above. We conclude that both the RMFPP and MFPP algorithms degrade gracefully with a tilted camera.

4.5 Effect of Cost Function

The tilted-camera sequences that we used in the previous section also provide an opportunity to compare the performance of the RMFPP algorithm using the more accurate cost function approximation that we derived in Section 3.2 to the original MFPP cost function from Section 3.4, as e_z (the Z coordinate of the homogeneous epipole) is non-zero in the case of forward camera tilt. We show the results in Figure 4.11. We conclude that using the original MFPP cost function degrades the results slightly at moderate to high camera tilt, but that the difference is never more than 0.2%. Given the higher computational cost of the more accurate RMFPP cost function, in situations where a higher framerate is desired we recommend using the MFPP cost function instead.

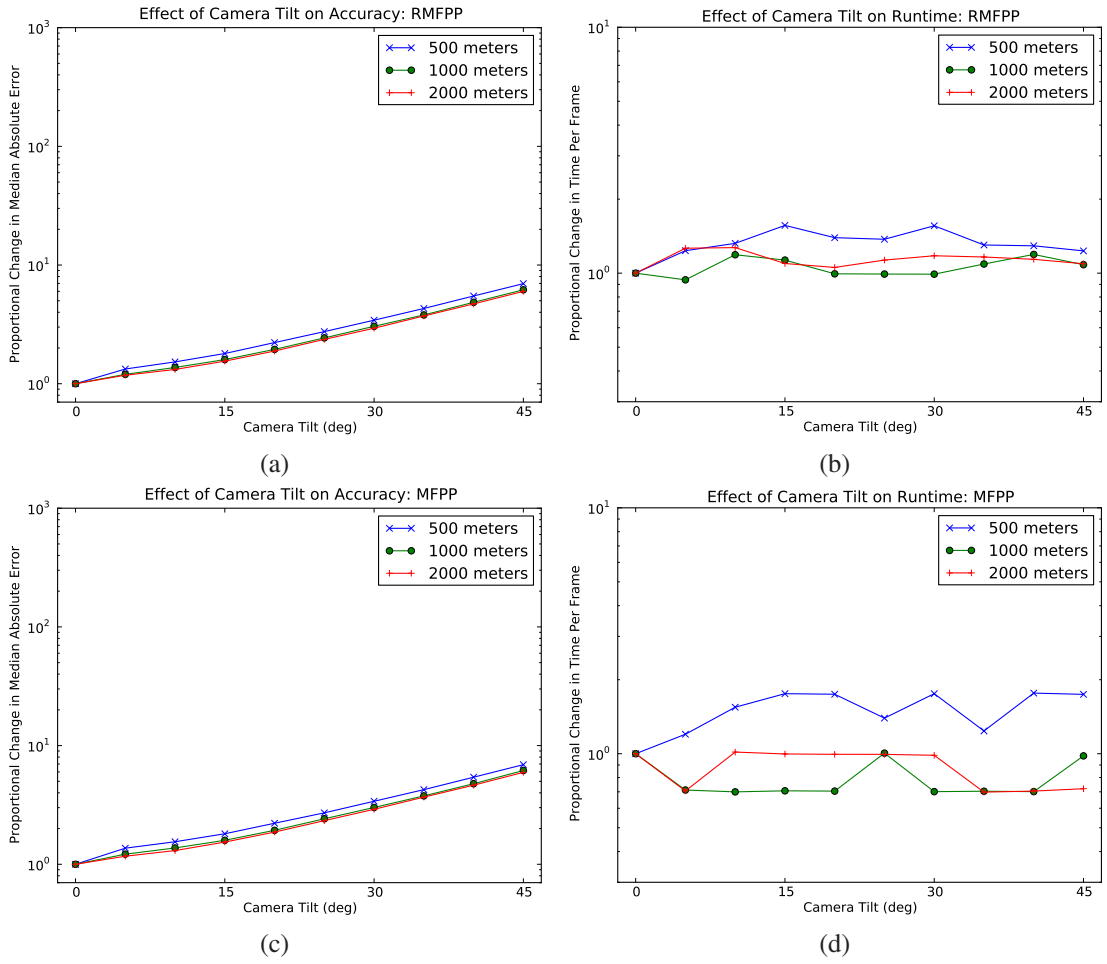
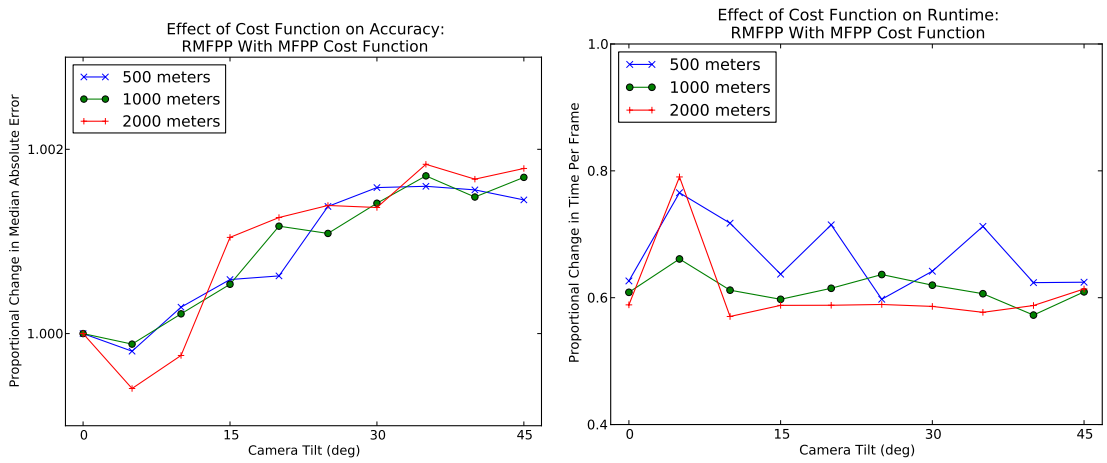


Figure 4.10: Relative accuracy and runtime of the RMFPP (a,b) and MFPP (b,c) algorithms with varying degrees of forward camera tilt. Note that the both algorithms degrade gracefully with camera tilt.



(a) (b)

Figure 4.11: Relative accuracy (a) and runtime (b) of the RMFPP algorithm using the cost function from the MFPP algorithm, compared to using the better cost function approximation derived for the RMFPP algorithm. Note that the algorithm performs better with the RMFPP cost function at moderate to high camera tilt, although the difference in accuracy is never more than 0.2% and the additional complexity of the RMFPP cost function requires more computation time.

4.6 Real Image Experiments

To demonstrate that the RMFPP algorithm is suitable for real-world use, we show results using images recorded from a robotic helicopter north of Southern California Logistics Airport near Victorville, CA. We emphasize that this is a challenging dataset—there is lens flare in the center of every image, which violates the algorithm's assumption that the scene is static, and the vehicle is traveling at only 130-170 meters above ground level, which reduces the number of frames in which a point on the ground is visible (and hence the number of frames $\mathcal{I}_i$ using a reference frame $\mathcal{I}_1$). In these experiments, the camera is tilted forward by 27° and has focal length 342; the images are 320x240 pixels. The vehicle moves at 8-16 meters/sec, and images are captured at 3.75 frames/sec. We determine the location of the camera at each image by refining the vehicle INS data using a combination of SIFT [11] and sparse bundle adjustment [10].

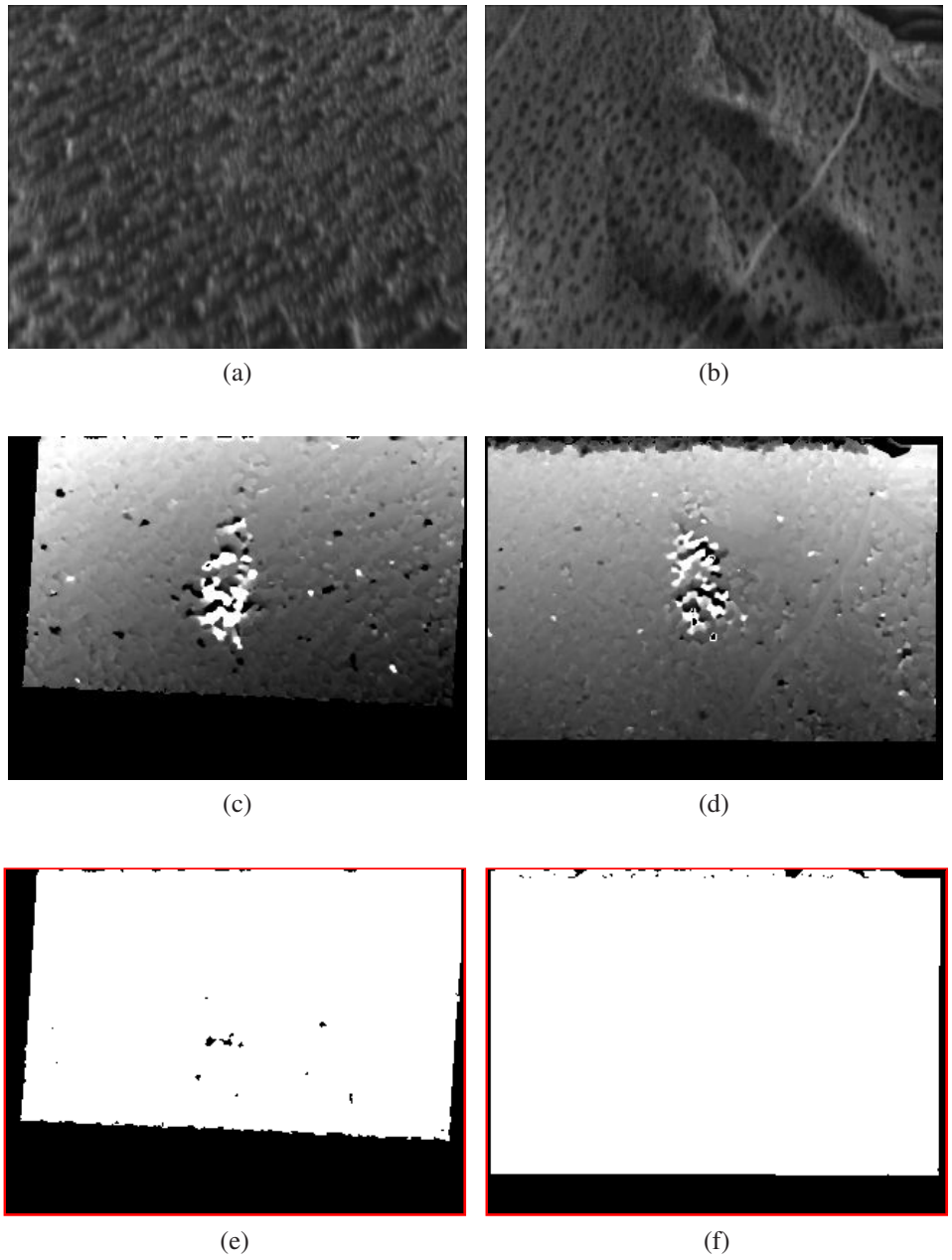


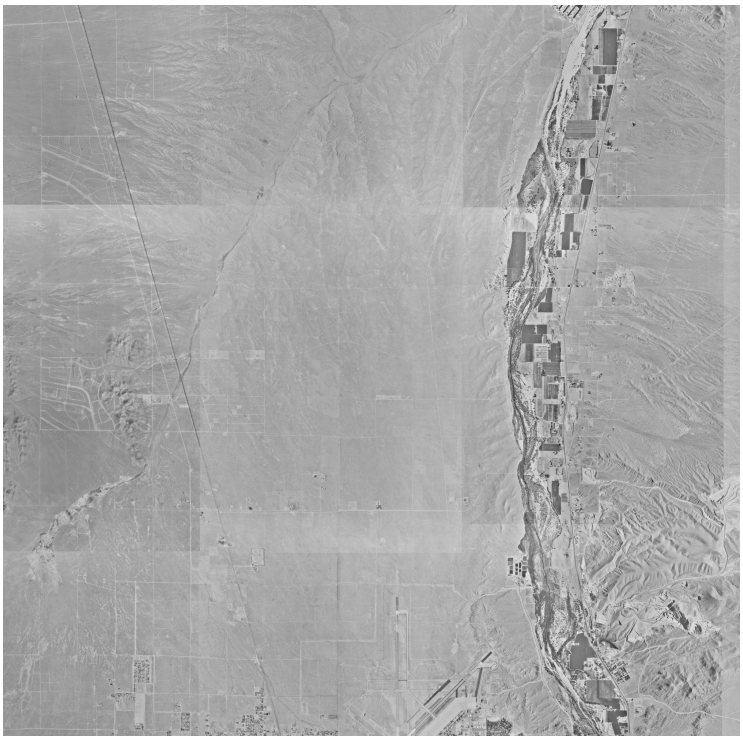
Figure 4.12: RMFPP results using real images recorded from a robotic helicopter north of Southern California Logistics Airport near Victorville, CA: captured image (a,b), reconstructed depth (c,d), and valid mask for reconstructed depth (e,f). Each column is a result set: (a,c,e) and (b,d,f). The error region in the center of the reconstructed depth (c,d) is due to lens flare, which violates the algorithm's assumption that the scene is static. Note that the depth reconstructions are unfiltered. The red box around the depth valid mask is not part of the mask itself—it serves only to separate the mask image from the surrounding whitespace.

Figure 4.12 shows the recorded images, and the (unfiltered) reconstructed depth. The algorithm correctly determines that the top of each image has larger depth than the bottom (due to camera tilt). Although the lens flare leads to an error region in the center of the reconstructed depth, the remainder of the reconstruction is unaffected. The reconstruction runtime is less than 0.5 seconds per frame, despite the lens flare.

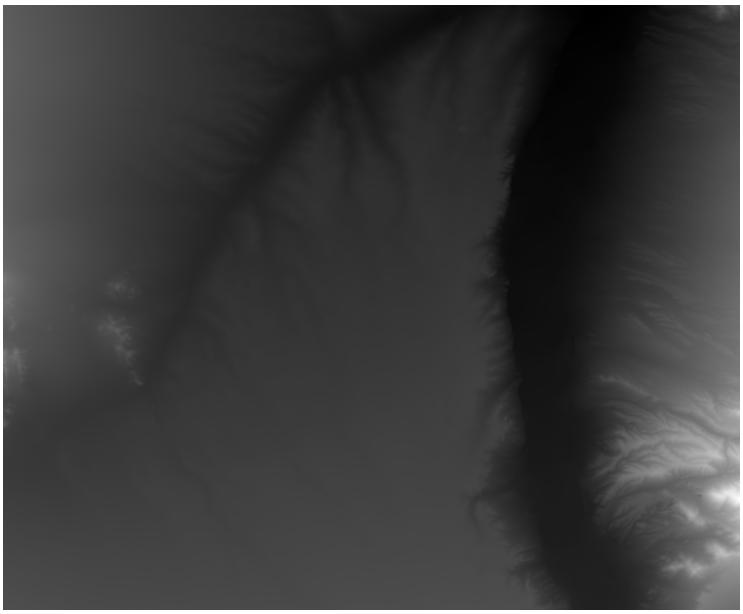
4.7 Full Terrain Reconstruction

As a final demonstration of the RMFPP algorithm, we show a full terrain reconstruction using United States Geological Survey orthoimagery [22] and elevation [21] data, as shown in Figure 4.13. The USGS data includes the region used for the real image experiments above. Using Blender [4] and the same camera parameters as in the sinusoidal terrain experiments above, we render images that are 10 meters apart using an outwardly-spiraling downward-facing camera that is 1000 meters above the terrain's mean elevation. Figure 4.14 shows sample rendered images; note the wide variety of texture and terrain.

The reconstruction uses 4 complete spirals (with 50% overlap), 162 reference frames, 5647 total images, and an average of 0.17 seconds per frame. Figure 4.15 gives a quantitative evaluation of the reconstruction, and Figure 4.16 gives a qualitative view of the final result. The median absolute reconstruction error is 0.87 meters.

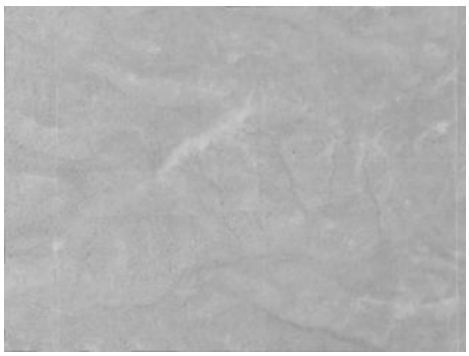


(a)



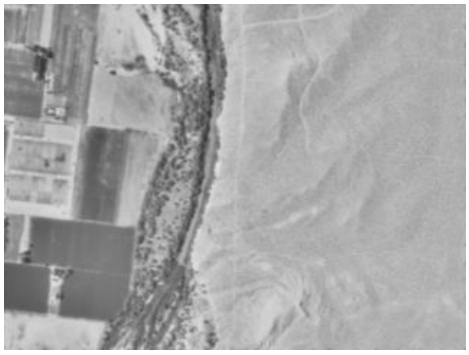
(b)

Figure 4.13: USGS texture (a) and terrain (b) north of Southern California Logistics Airport near Victorville, CA. The texture is comprised of Digital Orthophoto Quarter-Quadrangles [22] with resolution 1 meter/pixel. The terrain is from the National Elevation Dataset [21] and has resolution $1/3$ arc second per pixel. The terrain elevation is 744-1181 meters, with a mean elevation of 840 meters. The difference in aspect ratio is due to the fact that 1/3 arc second of longitude is smaller (in meters) than 1/3 arc second of latitude at this latitude.



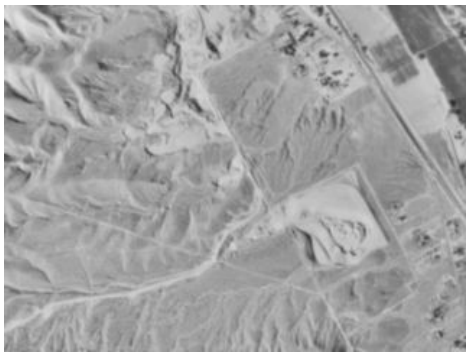
(a)

(b)



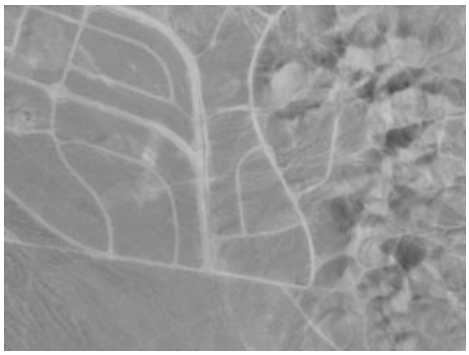
(c)

(d)



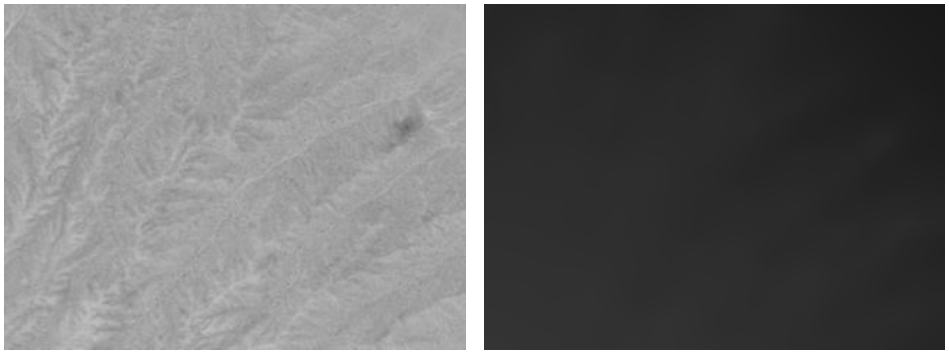
(e)

(f)



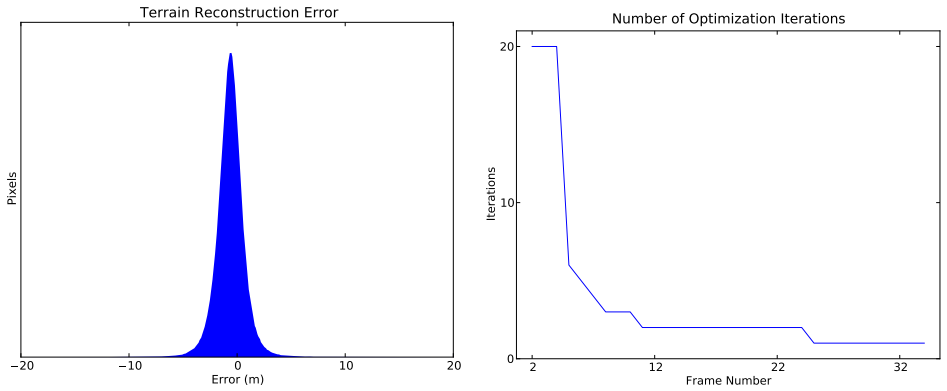
(g)

(h)



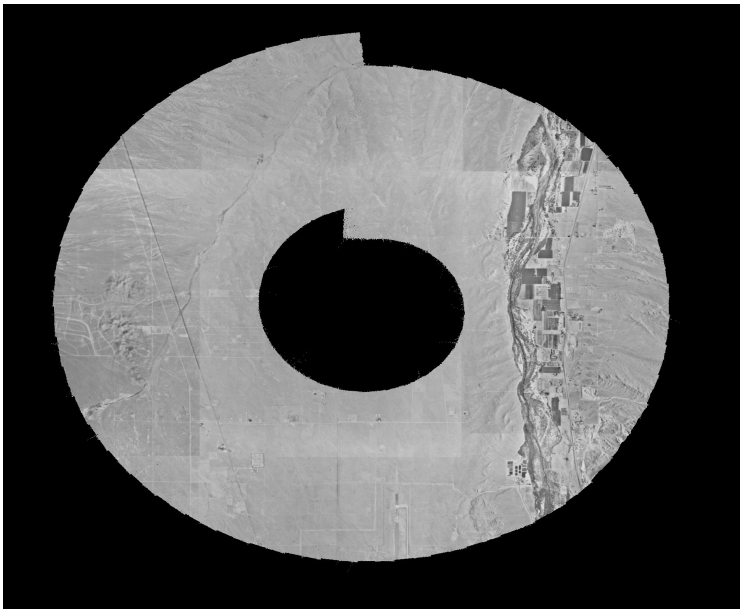
(i) (j)

Figure 4.14: Rendered texture and depth from downward-pointing cameras positioned 1000 meters above the mean of the USGS terrain. Note the wide variety of texture and terrain, from desertous to agricultural, and from flat to mountainous.

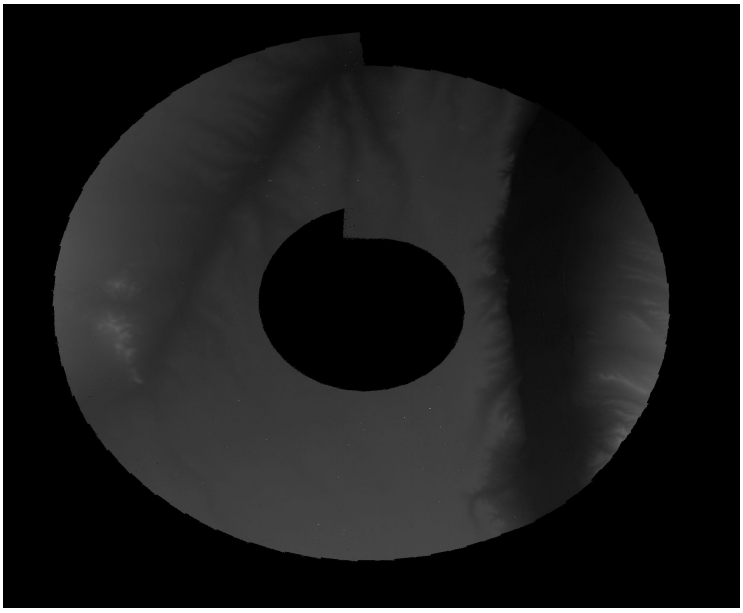


(a) (b)

Figure 4.15: Result of RMFPP reconstruction using image sequences rendered from an outwardly-spiraling downward-looking camera 1000 meters above the mean of the USGS terrain. The histogram of reconstruction errors for the entire area is shown in (a); the number of iterations required for each frame following a representative reference frame is shown in (b).



(a)



(b)

Figure 4.16: The reconstructed USGS texture and terrain, using 4 complete spirals (with 50% overlap) and 5647 frames (including 162 reference frames) spaced 10 meters apart. The resolution is equal to that of the original USGS terrain, $1/3$ arc second per pixel.

Chapter 5

Conclusion

In this dissertation, we have detailed the Recursive Multi-Frame Planar Parallax (RMFPP) algorithm, a recursive extension of Irani et al.'s Multi-Frame Planar Parallax (MFPP) batch algorithm, and shown that its error increases only linearly with depth. We have verified that our recursive cost function, which preserves more of the problem's nonlinearity than does the one in the MFPP algorithm, allows a more accurate recursive procedure. We have also validated the removal of the geometry-refining optimization used in the MFPP algorithm, observing graceful degradation under geometric noise. Finally, we have presented results using both synthetic and real imagery that show that the RMFPP algorithm is at least as accurate as the original MFPP batch algorithm in many circumstances, is preferred to both fixed- and dynamic-baseline two-frame methods, and is suitable for real-time use.

Future directions for this research include the development of a visual egomotion estimation algorithm that is suitable for distant (near-planar) scenes, and that fulfills the accuracy requirements of the RMFPP algorithm as observed in this dissertation. We would also like to combine both the RMFPP algorithm and this egomotion estimation system into a complete real-time robotic reconstruction platform for distant static scenes.

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] Gary Bradski. Programmer's tool chest: The OpenCV library. *Dr. Dobbs Journal*, November 2000.

[3] Olivier Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, 1993.

[4] Blender Foundation. Blender: the free open source 3d content creation suite, 2008. Version 2.48a. Available from `http://www.blender.org`.

[5] Christopher Geyer, Todd Templeton, Marci Meingast, and S. Sastry. The Recursive Multi-Frame Planar Parallax Algorithm. In *Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.

[6] M. Irani and P. Anandan. About direct methods. In *Proc. International Workshop on Vision Algorithms*, September 1999.

[7] M. Irani, P. Anandan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. *Trans. on Pattern Analysis and Machine Intelligence*, 24(11), November 2002.

[8] S. K. Jenson and J. O. Domingue. Extracting topographic structure from digital elevation data for geographic information systems analysis. *Photogrammetric Engineering and Remote Sensing*, 54(11):1593 – 1600, 1988.

[9] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic Linear Algebra Subprograms for FORTRAN usage. *ACM Trans. Math. Soft.*, 5:308–323, 1979.

[10] M.I.A. Lourakis and A.A. Argyros. 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, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[12] Y. Ma, S. Soatto, Jana Košecká, and Shankar Sastry. *An Invitiation to 3D Vision: From Images to Geometric Models*. Springer Verlag, New York, 2003.

[13] L. Matthies and S. A. Shafer. Error modeling in stereo navigation. *J. of Robotics and Automation*, RA-3(3), June 1987.

[14] Larry Matthies, Takeo Kanade, and Richard Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3), September 1989.

[15] J. McMichael and M. Francis. Micro air vehicles–toward a new dimension in flight. Technical report, DARPA, 1997.

[16] M. Okutomi and Takeo Kanade. A multiple-baseline stereo method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):353–363, April 1993.

[17] H. S. Sawhney. 3d geometry from planar parallax. In *Proceedings of Computer Vision and Pattern Recognition*, June 1994.

[18] B. Sofman, J. Bagnell, A. Stentz, and N. Vandapel. Terrain classification from aerial data to support ground vehicle navigation. Technical Report CMU-RI-TR-05-39, Robotics Institute, Carnegie Mellon University, 2006.

[19] E. Sokolowsky, H. Mitchell, and J. de La Beaujardiere. NASA's scientific visualization studio image server. In *Proc. IEEE Visualization 2005*, page 103, 2005.

[20] Gideon P. Stein and Amnon Shashua. Direct estimation of motion and extended scene structure from a moving stereo rig. Technical Report AIM-1621, Massachusetts Institute of Technology, 1997.

[21] United States Geological Survey. 1/3-Arc Second National Elevation Dataset (NED), 2009. Available from `http://seamless.usgs.gov`.

[22] United States Geological Survey. Digital Orthophoto Quarter-Quadrangles (DOQQ), 2009. Available from `http://seamless.usgs.gov`.

[23] Todd Templeton, David Hyunchul Shim, Christopher Geyer, and S. Sastry. Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft. In *Proceedings of 2007 IEEE International Conference on Robotics and Automation*, 2007.

[24] P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In *Proc. International Workshop on Vision Algorithms*, September 1999.

[25] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Proc. International Workshop on Vision Algorithms*, September 1999.

[26] Yalin Xiong and L. Matthies. Error analysis of a real-time stereo system. In *Proceedings of Computer Vision and Pattern Recognition*, June 1997.

[27] M. Zucchelli and H. I. Christensen. Recursive flow based structure from parallax with automatic rescaling. In *British Machine Vision Conference*, September 2001.