Autonomous Vision-based Rotorcraft Landing and Accurate Aerial Terrain Mapping in an Unknown Environment

Todd R. Templeton

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-18
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-18.html

January 22, 2007

Copyright © 2007, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

The author gratefully acknowledges the efforts of Christopher Geyer, David Shim, Shankar Sastry, and Marci Meingast, with whom significant portions of this work were coauthored. This material is used with the permission of the coauthors (and with the permission of the publishers, where applicable).

This work was supported by the following grants: ARO DAAD 19-02-1-0383 and Boeing SEC BAI-Z40705R.

Autonomous Vision-based Rotorcraft Landing and Accurate Aerial Terrain Mapping in an Unknown Environment

by Todd R. Templeton

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor S. Shankar Sastry
Research Advisor

Date

* * * * * *

Professor Ruzena Bajcsy
Second Reader

Date

Contents

Acknowledgments **iv**

Abstract **1**

1 Introduction **2**

2 Vision System Overview **5**
2.1 Introduction 5
2.2 The Recursive Multi-Frame Planar Parallax Algorithm 7
2.3 Motion Stamping 8
2.4 Modular Elevation and Appearance Map, and Landing Site Quality 9
2.5 Target Detector 11
2.6 High-level Planner 13
2.7 Conclusion 15

3 The Recursive Multi-Frame Planar Parallax Algorithm **16**
3.1 Introduction 16
3.2 Analysis of Depth Errors 19
3.3 Multi-Frame Planar Parallax 22
3.4 Nonrecursive Cost Function 24
3.5 Recursive Cost Function 25
3.6 Computation of γ and Flows 29
3.7 Complete Algorithm 30
3.8 Conclusion 32
3.A World Coordinates 33

4 Probabilistic Motion Stamping **34**
4.1 Introduction 34
4.2 Requirements 34
4.3 Notation 35
4.4 System Model 37
4.5 Implementation 39
4.6 Results 47
4.7 Conclusion 48

5 Analytical Motion Stamping **49**
5.1 Introduction . . . 49
5.2 Algorithm . . . 50
5.3 Robustness . . . 55
5.4 Results . . . 57
5.5 Conclusion . . . 58

6 Experimental Platform **59**
6.1 Introduction . . . 59
6.2 Vehicle Setup . . . 59
6.3 Vehicle Control . . . 62
6.4 Extrinsic Calibration . . . 69
6.5 Conclusion . . . 73

7 Results **74**
7.1 Introduction . . . 74
7.2 Results From Synthetic Data . . . 75
7.3 Results From Experimental Data . . . 78
7.4 Conclusion . . . 78

8 Conclusion **81**

Redaction Notice:
Portions of the original document were reprinted from C. Geyer, T. Templeton, M. 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. Permission was obtained from the publisher for limited distribution. However, for mass distribution, these reprinted portions have been marked and removed; please refer to the original paper.

List of Figures

1.1 The Berkeley UAV testbed. 3

2.1 Vision system architecture. 6
2.2 View of the landing target from the air. 12
2.3 An experimental ROC curve of the landing target detector. 13
2.4 Plans for the high-level planner. 14

3.1 Idealized flight for purposes of analyzing range accuracy. 20
3.2 Predicted standard deviations for stereo and multi-baseline as a function of range. 22
3.3 Uncertainty in disparity and range for a translating camera. 26

4.1 Model coordinate frames and transformations. 37
4.2 Iterated extended Kalman filter. 39
4.3 Synthetic experiment: screw trajectory. 46
4.4 Synthetic experiment: first frame. 46
4.5 Synthetic experiment: last frame. 47

5.1 Sensitivity to error in GPS/INS measurement time. 56
5.2 Sensitivity to error in feature equations. 57
5.3 Comparison of true and estimated trajectories. 58

6.1 System architecture of the UAV testbed. 61
6.2 Flight control system architecture with MPC-based trajectory generator. . . 64
6.3 Three-point waypoint specification and MPC-based trajectory generation. . 66
6.4 Experimental trajectory-following result. 68
6.5 Camera to GPS/INS calibration. 70

7.1 Sample rendered image. 75
7.2 Experimental comparison of reconstruction algorithms. 76
7.3 Synthetic images experiment. 77
7.4 Boeing real images experiment. 79
7.5 Berkeley real images experiment. 80

Acknowledgments

The author gratefully acknowledges the efforts of Christopher Geyer, David Shim, Shankar Sastry, and Marci Meingast, with whom significant portions of this work were coauthored. In particular, Chapter 3 and portions of Chapter 7 are based on published work coauthored with Christopher Geyer, Marci Meingast, and Shankar Sastry [1], and Chapter 2 and portions of Chapters 6 and 7 are based on unpublished work coauthored with David Shim, Christopher Geyer, and Shankar Sastry [2]. This material is used with the permission of the coauthors (and with the permission of the publishers, where applicable).

Additional thanks are due to Shankar Sastry and for his roll as advisor, and to Christopher Geyer, Jonathan Sprinkle, and Ruzena Bajcsy for providing additional advice and assistance beyond the call of duty.

Finally, the author wishes to thank his family for putting him on a successful path in life, and his fiancée Cheryl for supporting him on a daily basis.

This work was supported by the following grants: ARO DAAD 19-02-1-0383 and Boeing SEC BAI-Z40705R.

Abstract

In this report, we present research toward a vision-based landing system for unmanned rotorcraft in unknown terrain that is centered around our Recursive Multi-Frame Planar Parallax algorithm [1] for high-accuracy terrain mapping. We give an in-depth description of the vision system, an overview of our experimental platforms, and both synthetic and experimental terrain mapping results.

Chapter 1

Introduction

The research in this report was born from a desire to reconstruct terrain from aerial imagery online in real time, at high altitudes, at an extreme level of accuracy; to integrate this terrain information into a global map that an intelligent UAV could use to make informed decisions; and to be able to efficiently analyze this map as it is created to find safe landing locations for an unmanned rotorcraft. Dissatisfied with the usual sensing paradigms, namely laser scanners (which are easily detected and which require high-powered beams that are energy- and weight-intensive for operation at high altitudes) and stereo cameras (which are highly inaccurate at high altitudes using baselines that can be accommodated on a vehicle), we conceived a novel high-accuracy real-time visual reconstruction paradigm using a single moving camera, namely the Recursive Multi-Frame Planar Parallax algorithm [1].

Experimental validation for the mapping and landing scheme was performed on a full-sized helicopter with the cooperation of Boeing Phantom Works, as well as on a smaller rotorcraft at UC Berkeley (see Figure 1.1). Although inaccuracy in current real-time camera localization algorithms ultimately prohibited us from performing the high-accuracy mapping task online in real time, we were able to demonstrate accurate real-time performance for all other parts of the system by performing the camera localization offline. We believe

Figure 1.1: The Berkeley UAV testbed: an electrically-powered, MPC-controlled rotorcraft for vision-based landing and terrain mapping. *Courtesy of David Shim.*

that accurate real-time camera localization is required for any system that wishes to integrate high-altitude terrain measurements over time and motion into a single global map, and we are confident that further work in this area will produce algorithms that are up to the challenge.

The vision landing problem has been addressed in many previous research projects, although many, including [3], [4], [5], [6], and [7], require an easily-recognizable landing target. No landing target is used in [8], although it is assumed that the visual axis of the camera is perpendicular to the ground and that the image contrast is higher at the boundary of obstacles than anywhere else. The approach most similar to ours is that of A. Johnson et al. at JPL [9], although their use of only two images at a time (a wide-baseline stereo pair over time from a single camera) restricts their 3D reconstruction accuracy at high altitudes.

The remainder of this report is organized as follows: In Chapter 2, we give an overview of the vision system for autonomous rotorcraft landing and mapping. In Chapter 3, we give a more in-depth treatment of the Recursive Multi-Frame Planar Parallax algorithm. In Chapters 4 and 5, we discuss two real-time camera localization techniques that were inves-

tigated during the course of the project, giving an analysis of their accuracy. In Chapter 6 we discuss the vehicle platforms on which the system was tested, and in Chapter 7 we give synthetic and experimental results. Finally, in Chapter 8 we conclude.

Chapter 2

Vision System Overview

Much of the content of this chapter is based on unpublished work coauthored with David Shim, Christopher Geyer, and Shankar Sastry [2]. This material is used with the permission of the coauthors.

2.1 Introduction

The computer vision problem that we address in this project is one of 3D terrain reconstruction and analysis. In particular, we are trying to find suitable landing locations, i.e. regions that are large enough to safely land the helicopter that are flat and free of debris, have a slope of no more than 4 degrees, have been confidently mapped, (possibly) are similar to a desired constant appearance, and (optionally) contain a distinctive landing target[1]. Despite the availability of high-accuracy active technologies such as radar and LIDAR, we use a camera for this task because it is passive (and hence difficult to detect), and because such active technologies require high-powered beams that are energy- and weight-intensive for operation at high altitude.

The vision system (see Figure 2.1) consists of a feature tracking thread, which tracks

[1]The landing target is only used to enable the operator to choose a specific location if many locations are suitable—the vision system always ensures that all other requirements are satisfied.

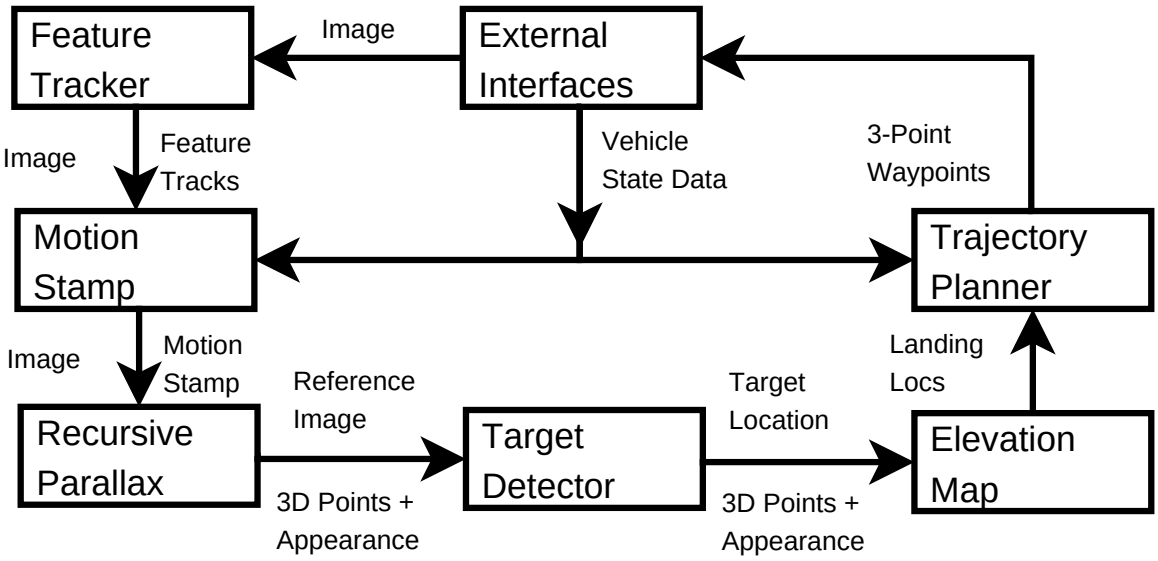


Figure 2.1: Vision system architecture. Note that the target detector is optional and does *not* replace 3D terrain reconstruction using the Recursive Multi-Frame Planar Parallax (RMFPP) algorithm.

distinctive image points through the image sequence and stores them in the feature repository; a motion stamping thread, which uses GPS/INS data and feature tracks to estimate the global position and orientation of the camera when each image was captured and to estimate the 3D locations of the tracked features (the latter of which are used to choose the best reference plane for the Recursive Multi-Frame Planar Parallax algorithm); and the mapping thread, which adds 3D points to its modular elevation and appearance map using the Recursive Multi-Frame Planar Parallax algorithm. The vision system also includes two interchangeable sets of external interfaces: in flight mode, it uses a custom Firewire capture thread, which stores timestamped captured images in a frame repository, and an external communication thread, which receives GPS/INS and other vehicle state data from, and sends desired trajectory information to, the vehicle control computer; in simulation/replay mode the Firewire capture thread is replaced by a custom simulation/replay thread, and all communication through the external communication thread is redirected to the simulation/replay thread.

2.2 The Recursive Multi-Frame Planar Parallax Algorithm

The cornerstone of our approach is our novel Recursive[2] Multi-Frame Planar Parallax (RMFPP) algorithm [1]. The RMFPP algorithm is a direct[3] method for obtaining dense[4] structure (terrain, in our case) estimates with corresponding appearance online in real time by using a single moving camera whose motion has been accurately estimated. We choose to use this single-camera method because of the inaccuracy inherent in estimating distant terrain using a stereo camera pair with a baseline that is attainable on the vehicle (see discussion in Chapter 3), while using multiple images as the camera moves through space allows the RMFPP algorithm to attain expected range error that increases between linearly and with the square root of the range.

Suppose a camera takes images $i = 1, \ldots, m$ of a rigid scene, where image 1 is the reference view in which range will be estimated for each pixel. Then the homographies $\mathsf{H_i}$ that transfer the i-th view to the reference view via a chosen reference plane are given by:

$$\mathsf{H_i} \;=\; \mathsf{K}\left(\mathsf{R_i} - \frac{1}{d}\boldsymbol{T_i}\boldsymbol{N}^T\right)^{-1}\mathsf{K}^{-1} \in \mathbb{R}^{3\times 3}, \tag{2.1}$$

where $(\boldsymbol{N} \in \mathbb{R}^3, d \in \mathbb{R}^3)$ are the unit normal of the reference plane in the coordinate system of the first camera and the perpendicular distance of the first viewpoint from the reference plane, $(\mathsf{R_i} \in SO(3), \boldsymbol{T_i} \in \mathbb{R}^3)$ are the rotation and translation from first camera coordinate system to the i-th one, and $\mathsf{K} \in SL(3)$ is the constant intrinsic calibration matrix of the camera.

Suppose that $\boldsymbol{X} \in \mathbb{R}^3$ is a point in space in the coordinate system of the first camera.

[2]The cost of incorporating measurements from a new image depends only on the number of pixels in the image and does *not* depend on the number of images already seen.

[3]The algorithm expresses a cost function directly in terms of the image rather than depending on feature matching, and gradients of the cost function are calculated by linearization of the brightness constancy constraint (see pro: [10], con: [11]).

[4]The algorithm provides a depth estimate for every pixel that is within a suffi ciently textured region.

Let $\boldsymbol{p_i} = (x_i, y_i)$ for $i = 1, .., m$ be the projection of X into each image, $\pi(x, y, z) = (x/z, y/z)$, and $\pi^*(x, y) = (x, y, 1)$. The quantity

$$\boldsymbol{p_1} \; - \; \underbrace{\pi\,(\,\mathsf{H_i}\,\pi^*(\,\boldsymbol{p_i}\,))}_{\boldsymbol{p_i}'} \tag{2.2}$$

is called planar parallax, and is zero if $\boldsymbol{X}$ lies on the reference plane. The RMFPP algorithm uses planar parallax, which is small for small movements if $\boldsymbol{X}$ is close to the reference plane and increases with increased camera motion, to recursively estimate the quantity $\gamma = h/z$ for each pixel $\boldsymbol{p_1}$ in the reference image, where z is the range of $\boldsymbol{X}$ in the first view and $h = \boldsymbol{N}^T\boldsymbol{X} + d$ is the signed perpendicular distance of $\boldsymbol{X}$ from the reference plane. We then recover the range z using $z = -d/(\boldsymbol{N}^T\,\mathsf{K}^{-1}\,\pi^*(\,\boldsymbol{p_1}\,) - \gamma)$.

The RMFPP algorithm will be discussed in more detail in Chapter 3.

2.3 Motion Stamping

In order to use the RMFPP algorithm, we must first motion stamp each image, i.e. determine the orientation and position of the camera when each image is captured using a combination of (noisy) vehicle state data and (noisy) image feature tracks. We will discuss a probabilistic approach to this problem in Chapter 4 and an analytic approach in Chapter 5, although we will ultimately conclude that the accuracy of these online algorithms has thus far proven to be insufficient for our needs. While we continue to experiment with alternative online motion-stamping methods, for the purposes of the experimental results in this report we use an offline motion stamping method to showcase the performance of the other components: we perform SIFT [12] feature tracking followed by Sparse Bundle Adjustment (SBA) [13], where we initialize the camera orientations and positions using the previous GPS/INS datapoints corrected by a constant coordinate transformation (see Section 6.4). Using this offline method requires the separation of the experiment into three

pieces, but the vision high-level planner still directs the helicopter flight while it collects image data and vehicle state data, the RMFPP algorithm is still run in better than real time on hardware similar to that on the helicopter vision computer, and we successfully execute the closer inspection and landing maneuvers using a premade elevation and appearance map.

2.4 Modular Elevation and Appearance Map, and Landing Site Quality

After it is filtered for outliers (see discussion in Chapter 3), the list of 3D terrain and appearance points produced by the RMFPP algorithm is stored in a modular elevation and appearance map. The map is represented as a 2D (x, y) grid with three layers at each of multiple resolutions: terrain elevation z, terrain elevation variance (expected squared error) $\frac{1}{w}$, and appearance a. The grid is modular in the sense that it is broken into fixed-sized rectangular blocks of real 2D space and that only blocks in which points have been observed are present in the map. All blocks contain all resolutions, and higher resolutions in each block contain more pixels in each layer than lower resolutions; for operations that require only a single resolution, such as calculating landing quality and exporting maps, each map block independently chooses its highest resolution where at least a fixed percentage of pixels have known value (or its lowest resolution if none of its resolutions have enough pixels with known value).

The modular elevation and appearance map is designed to be efficient and robust. To constrain memory usage, only a fixed maximum number of blocks can be present; the least recently accessed block is recycled when a new block is needed and no more blocks are available. Because the scene is likely to change over long periods of time, blocks are reinitialized when they are revisited after no updates for a fixed period of time. To reduce

the update and creation of blocks due to outliers, a fixed number of points from a given RMFPP update must be contained in an existing or potential block for it to be updated or created. To reduce the number of landing candidates generated by a suitable region, only block centers are considered as possible landing sites. To eliminate the trade-off between requiring large landing sites and having many mostly-unexplored blocks, and to allow more dense possible landing sites, the landing quality score of a given block is calculated over itself and a given radius of its neighbor blocks.

To exploit the expectation that a batch of 3D points from the RMFPP algorithm is from a contiguous area, blocks are bidirectionally liked to their immediate neighbors as well as being maintained in a hash table over their (x, y) centers, and all existing blocks that are required for the given list of 3D terrain and appearance points produced by the RMFPP algorithm are retrieved immediately upon determination of the points' 2D bounding box. Adding each 3D point to the map involves creating or locating the proper map block in the prefetched grid and then updating the closest 2D map block pixel at each resolution, i.e. optimally updating the pixel at each layer based on the existing elevation variance at the pixel and the elevation variance for the new 3D point as provided by the RMFPP algorithm.

For maximum efficiency, landing quality scores for each map block are calculated within the map module and the time required to update these scores for a batch of 3D points from the RMFPP algorithm is linear in the number of points and in the size of the points' 2D bounding box. Let S be the set of 2D grid points with known values in a given map block. The following statistics are maintained at each map block at each resolution: the number of 2D grid points whose values are known and the total number of 2D points in the block, from which the fraction of unknown points can be computed; a count of landing target detections and a sum over target qualities as given by the target detector (optional, not used to obtain the experimental results in this report), from which the average target quality can be computed; $\sum_{i \in S} w_i x_i$, $\sum_{i \in S} w_i y_i$, $\sum_{i \in S} w_i z_i$, $\sum_{i \in S} w_i x_i x_i$, $\sum_{i \in S} w_i x_i y_i$, $\sum_{i \in S} w_i x_i z_i$, $\sum_{i \in S} w_i y_i y_i$, $\sum_{i \in S} w_i y_i z_i$, $\sum_{i \in S} w_i z_i z_i$, and $\sum_{i \in S} w_i$, from which the best-

fit plane and plane-fit error can be computed; and $\sum_{i \in S} a_i$ and $\sum_{i \in S} a_i a_i$, from which the average appearance and appearance variance can be computed. When a 2D pixel is updated in a given block and resolution, its previous value is subtracted from, and its new value is added to, each statistic that is maintained for that block and resolution.

To calculate the landing quality score for all modified blocks after an RMFPP map update, an integral image [14] of the maintained statistics is formed over a rectangular area suitably larger than the modified area of the map, and the statistics over a block and its given radius of neighbors are calculated in constant time[5] by adding and subtracting appropriate elements of the integral image. The landing quality score for each modified block (combined with its radius of neighbors) is a linear combination of the angle of the best-fit plane from horizontal, the plane fit error, the percentage of grid squares in the block with unknown value, the difference between the average appearance and a given desired appearance, the appearance variance, and the average target quality (optional). A block is considered to be a landing candidate if it is below given thresholds on each element of the landing quality equation, and landing candidates are maintained in a priority queue so that, all other things being equal, the better (lower 'landing quality' value) candidates are considered first.

2.5 Target Detector

In case multiple safe landing sites are available, we want the vision system to be able to recognize a distinctive target on the ground (see Figure 2.2). The target detector operates directly on each RMFPP reference image, and the 3D output of the RMFPP algorithm for that image is used to determine the 3D location of any detected landing target. As discussed in Section 2.4, target detection counts are used as one of one of several features for determining the quality of a landing site. Hence, target detection does *not* replace 3D

[5]The time required by this operation is independent of the block radius and the radius of the desired landing site.

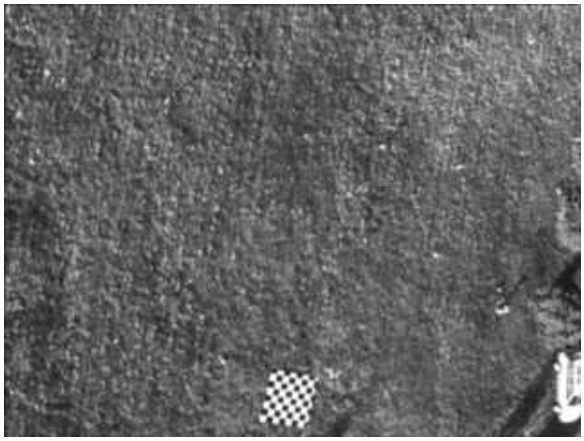


Figure 2.2: View of the landing target from the air.

terrain reconstruction using the RMFPP algorithm.

The target detector first detects corners in the image as local maxima of a filter composed of separable kernels, whose span resembles typical corners. It then determines the homography $h \in \mathbb{H}$, where $\mathbb{H}$ is a discrete set of homographies, that best transforms the n points into points with integer coordinates according to the function $f(h) = \sum_{i=1}^{n} \left(\exp \frac{1}{2\sigma^2} d_i^2\right)^{-1}$ for a given σ, where d_i is the distance of the i-th transformed point from the closest integer coordinates. The best homography must meet or exceed a given function value threshold τ to be considered a positive detection, so many of the homographies can be eliminated without computing the entire sum when the number of points remaining to incorporate into the sum is less than the difference between the threshold and the current value of the sum, or when the number of points remaining to incorporate into the sum is less than the difference between the best final function value so far and the current value of the sum (since each point can contribute at most 1 to the sum). Finally, if a suitable best homography h has been found, the target detector determines the subset of the n points that have transformed coordinates within a given distance q of integer coordinates; if none of them are within this distance, the detector determines that no target is present, otherwise it determines that a target is present at the median x and y image coordinates of

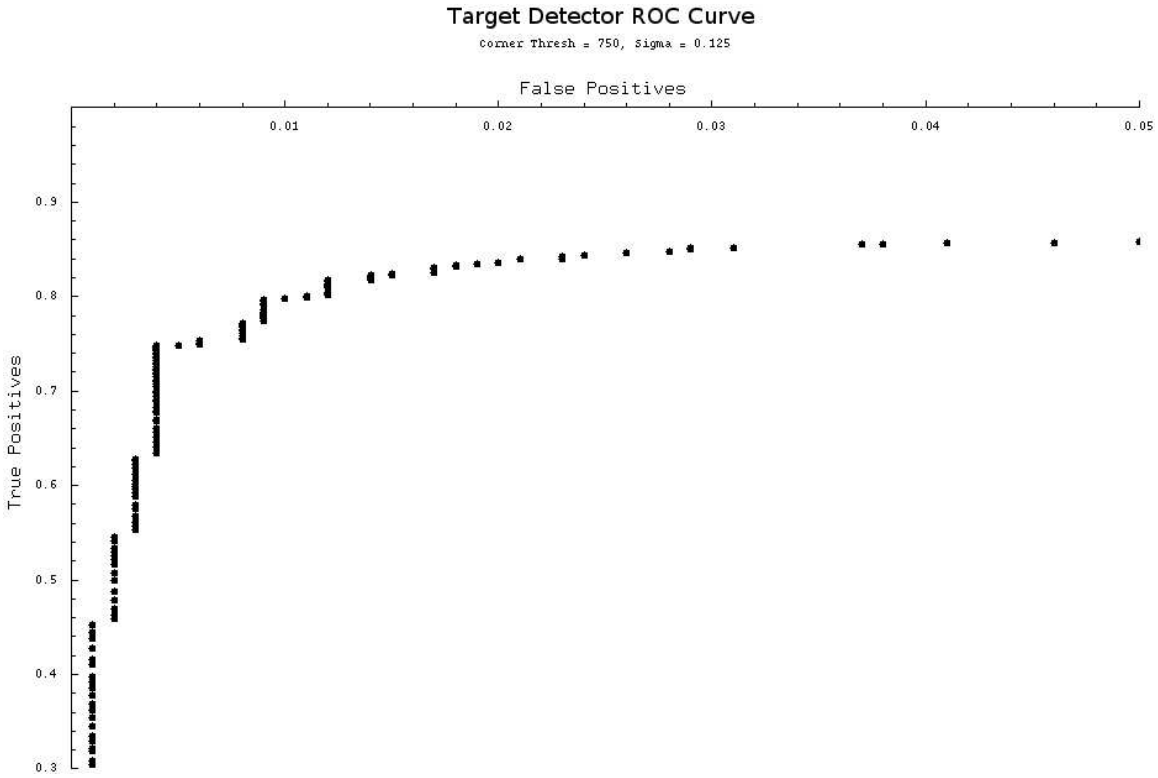


Figure 2.3: An experimental ROC curve of the landing target detector for different values of τ.

the n points, with quality $1 - \exp \frac{1}{2w^2} \left(f(h) - q\right)^2$ for a given w (a lower 'quality' value means a more confident detection).

The target detector was tested on images captured by both the Boeing and Berkeley vehicles at multiple locations, with synthetic landing targets at random scales and orientations inserted at random image locations. Figure 2.3 shows the ROC curve for different values of τ, which gives a result of 80% true detections and 1% false detections on the test data.

2.6 High-level Planner

Concurrently with the above vision algorithms, the vision system executes a high-level planner that operates directly on the vehicle state data. The default plan (when no landing site candidates have been identified within a given maximum radius of the current location)

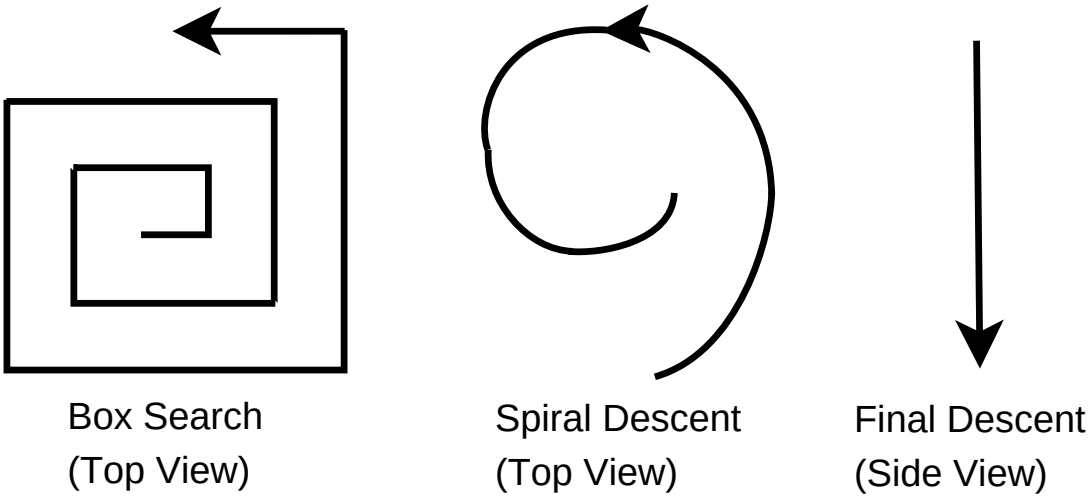


Figure 2.4: Plans for the high-level planner.

is an outwardly-expanding box search centered around the point, and at the altitude of, where the planner is initially enabled (see Figure 2.4). When a landing site candidate is identified that is within the given maximum radius, the planner enters a mode where it directs a descending spiral toward a point a fixed distance directly over the candidate site. The candidate site is examined whenever it is visible during the downward spiral, and all other visible locations are also examined at closer range during this process. At any time during the spiral, the vision system may determine that the site is unsuitable, or the human operator may signal that the site is unsuitable, and the planner will switch to a different candidate site (or return to the default box search plan if there are no nearby candidate sites). Once the helicopter reaches a point a fixed distance directly over the candidate site, the human operator may approve an autonomous landing, at which time the planner directs a constant-speed vertical descent to a fixed lower altitude AGL, followed by a slower constant-speed vertical descent to the ground.

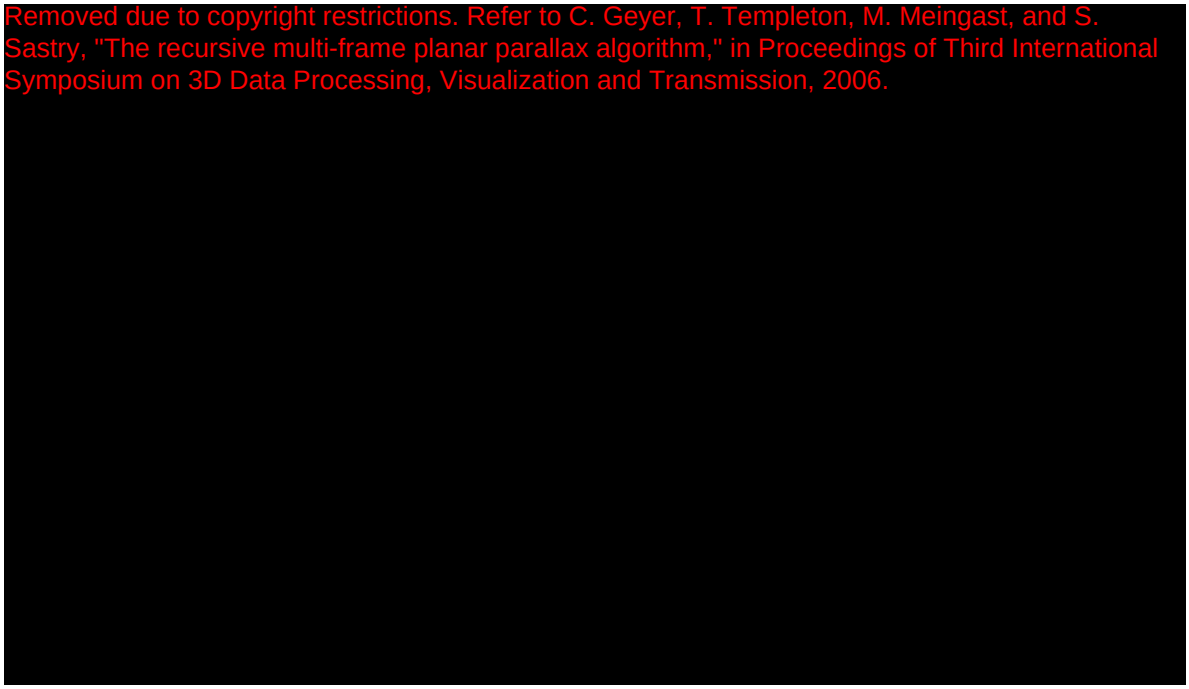
2.7 Conclusion

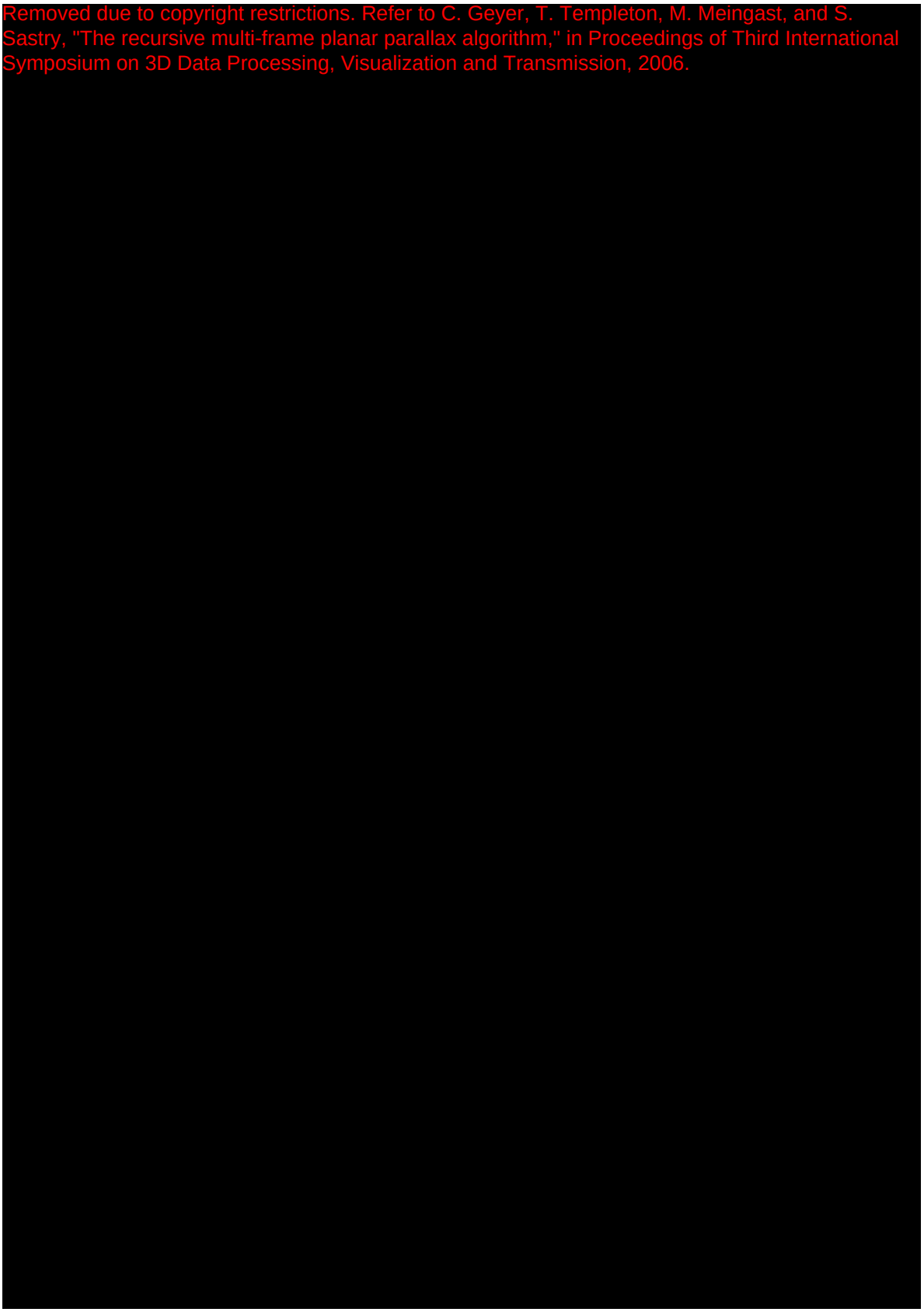
In this chapter, we have given an overview of the vision system used in the autonomous mapping and landing project. We will give more details of the crucial pieces in later chapters: the Recursive Multi-Frame Planar Parallax algorithm in Chapter 3, and the motion stamping problem in Chapters 4 and 5. We will give results from the RMFPP algorithm, both alone and as part of the system, in Chapter 7.

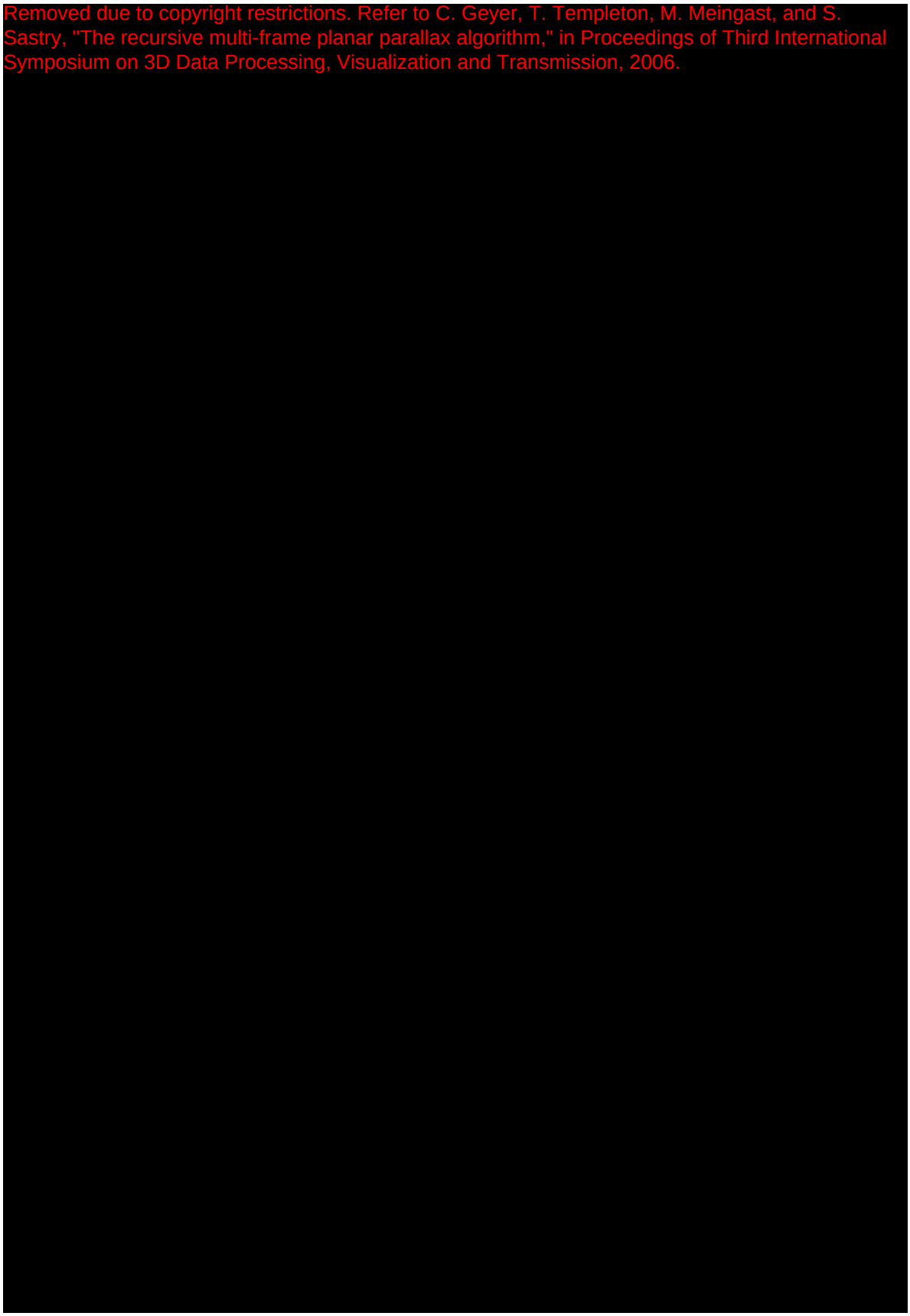
Chapter 3

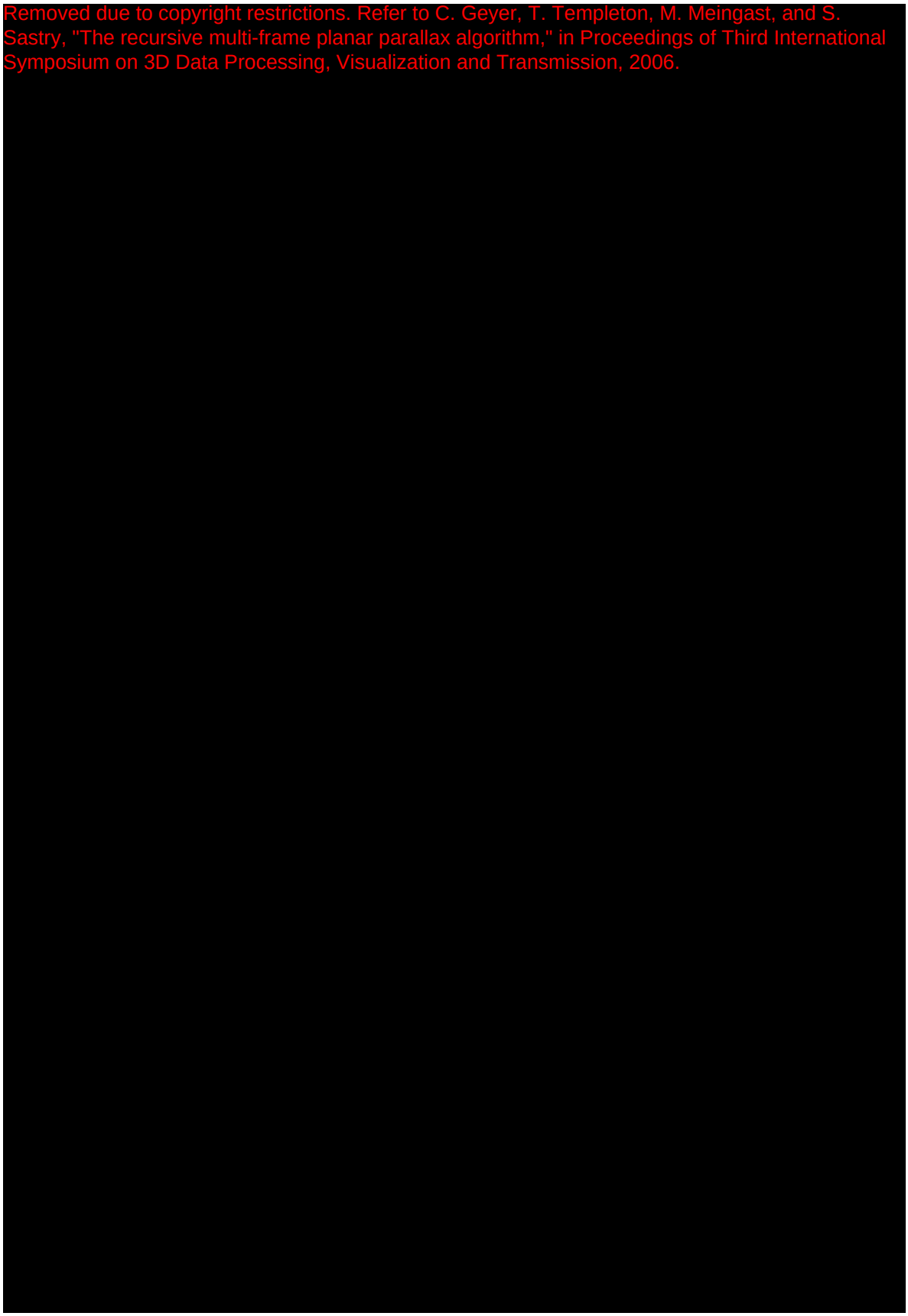
The Recursive Multi-Frame Planar Parallax Algorithm

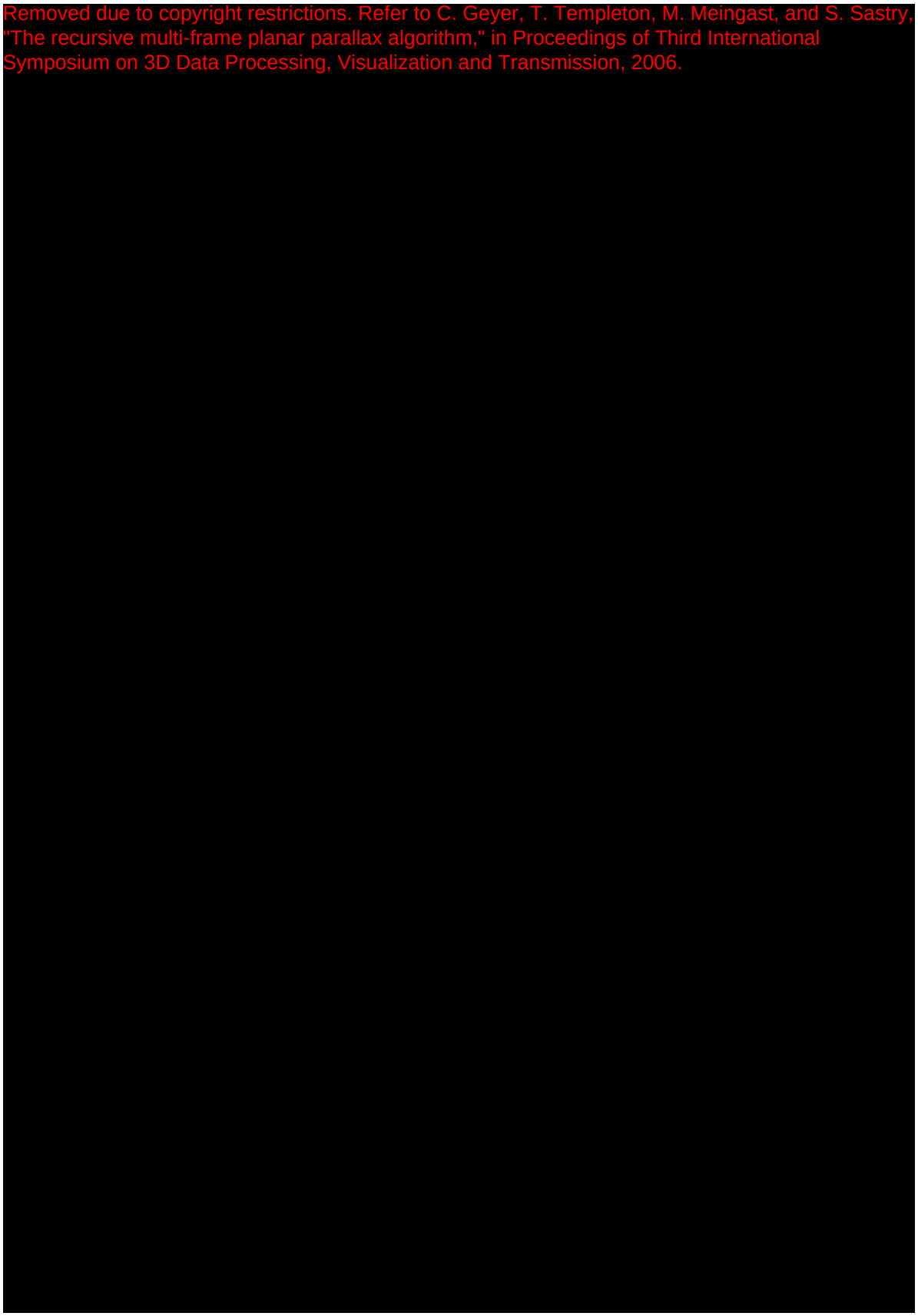
The content of this chapter is based on published work coauthored with Christopher Geyer, Marci Meingast, and Shankar Sastry. This material is used with the permission of the coauthors. ©2006 IEEE. Reprinted, with permission, from [1].

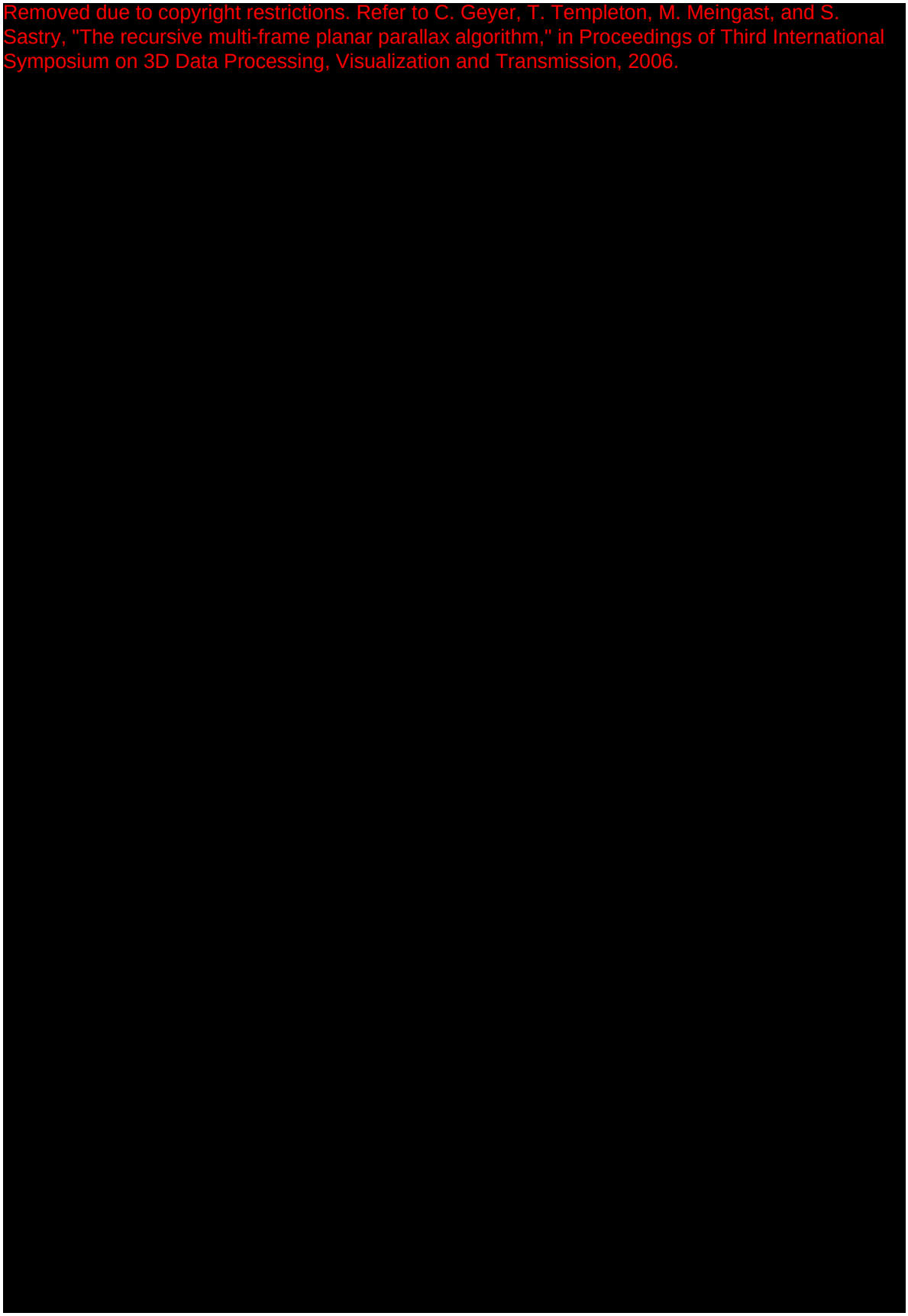


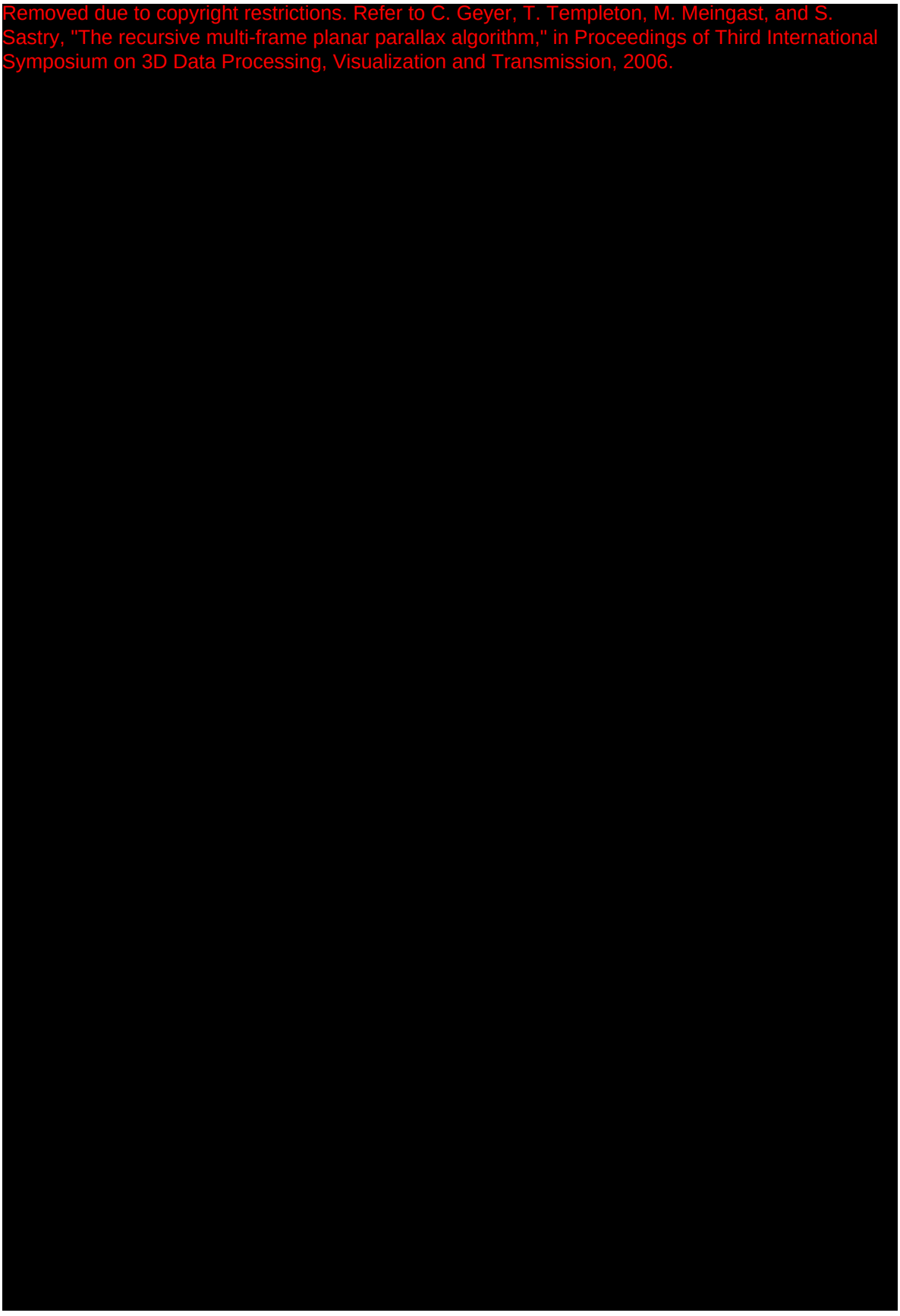

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

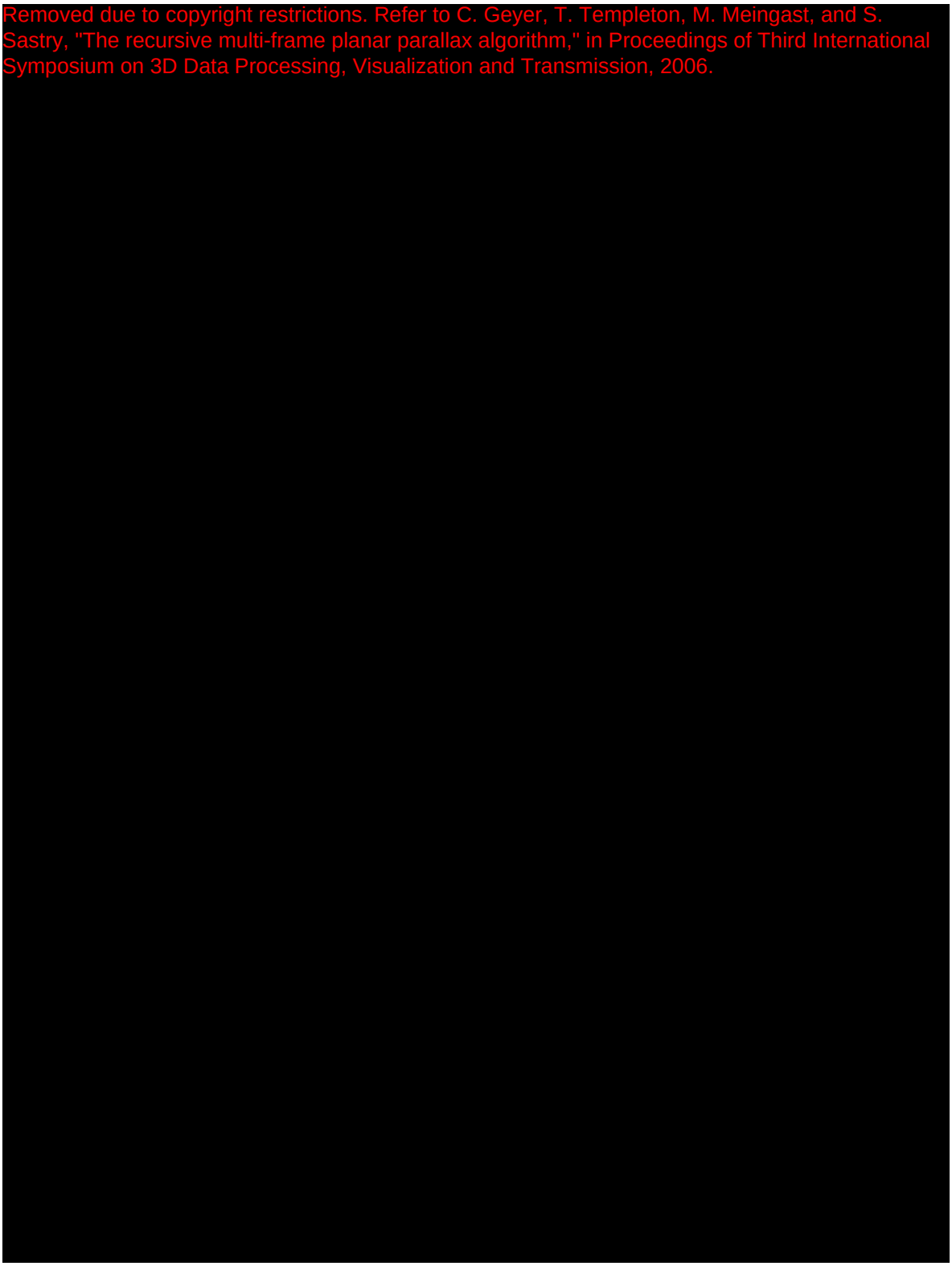

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

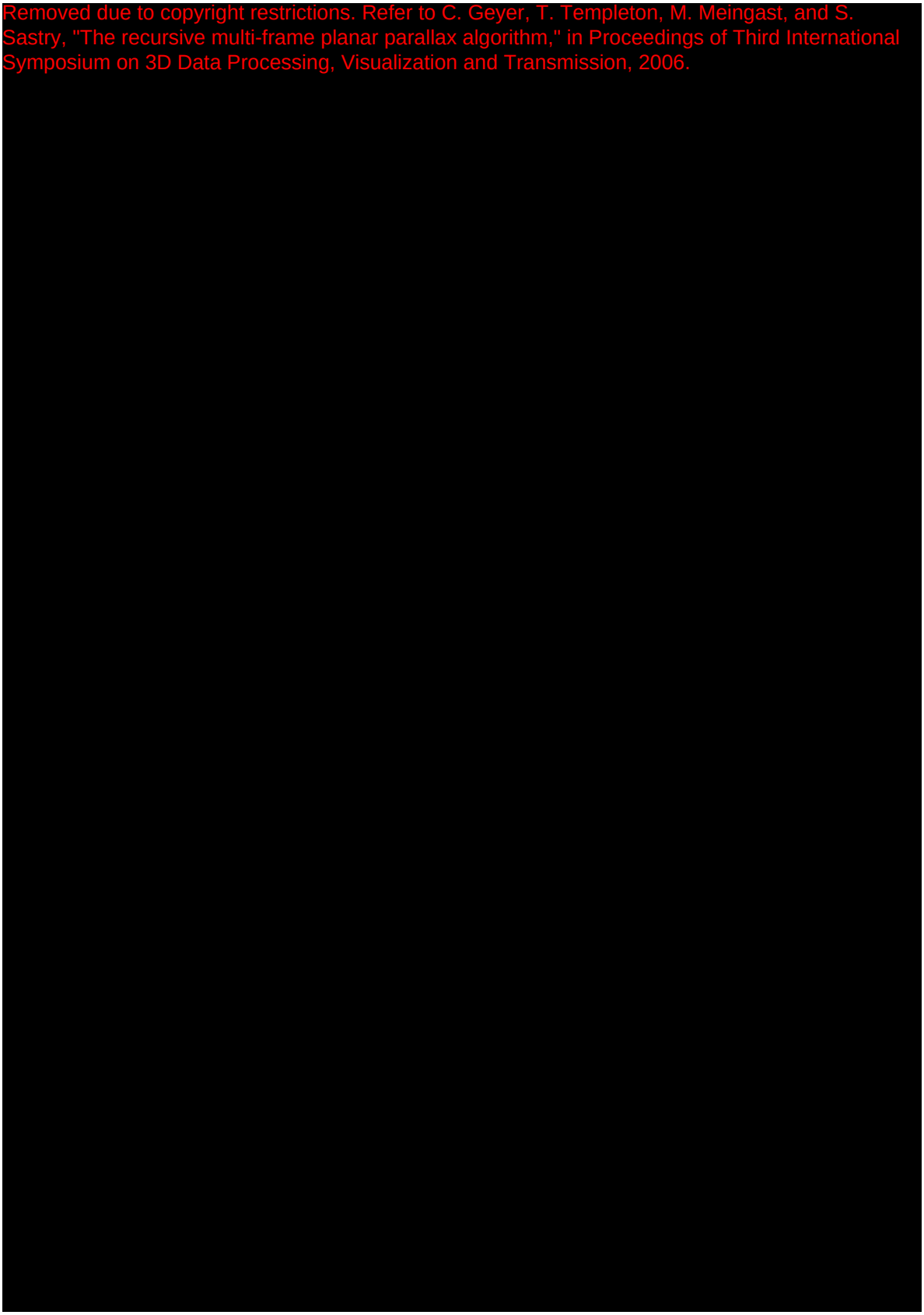

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

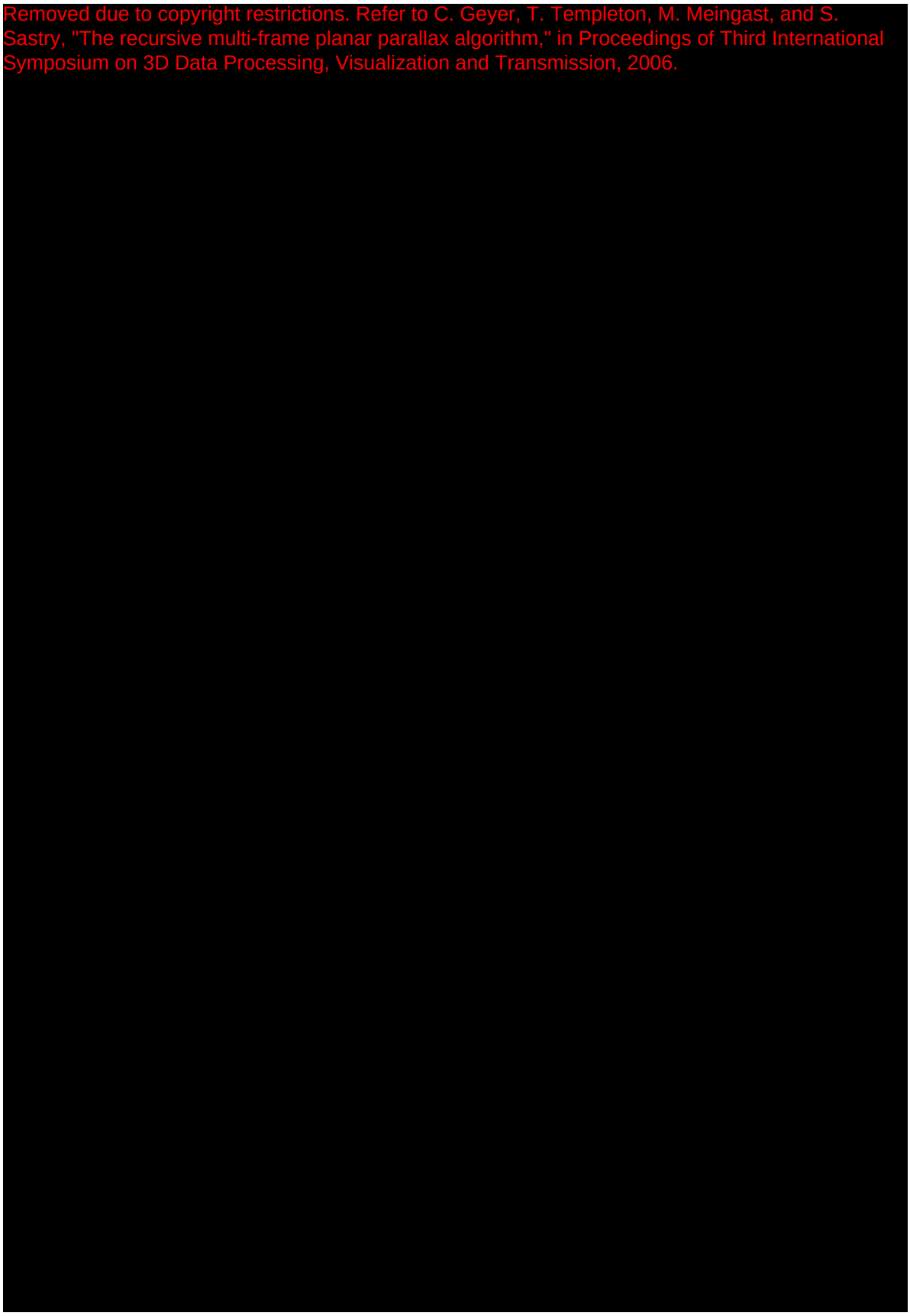

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

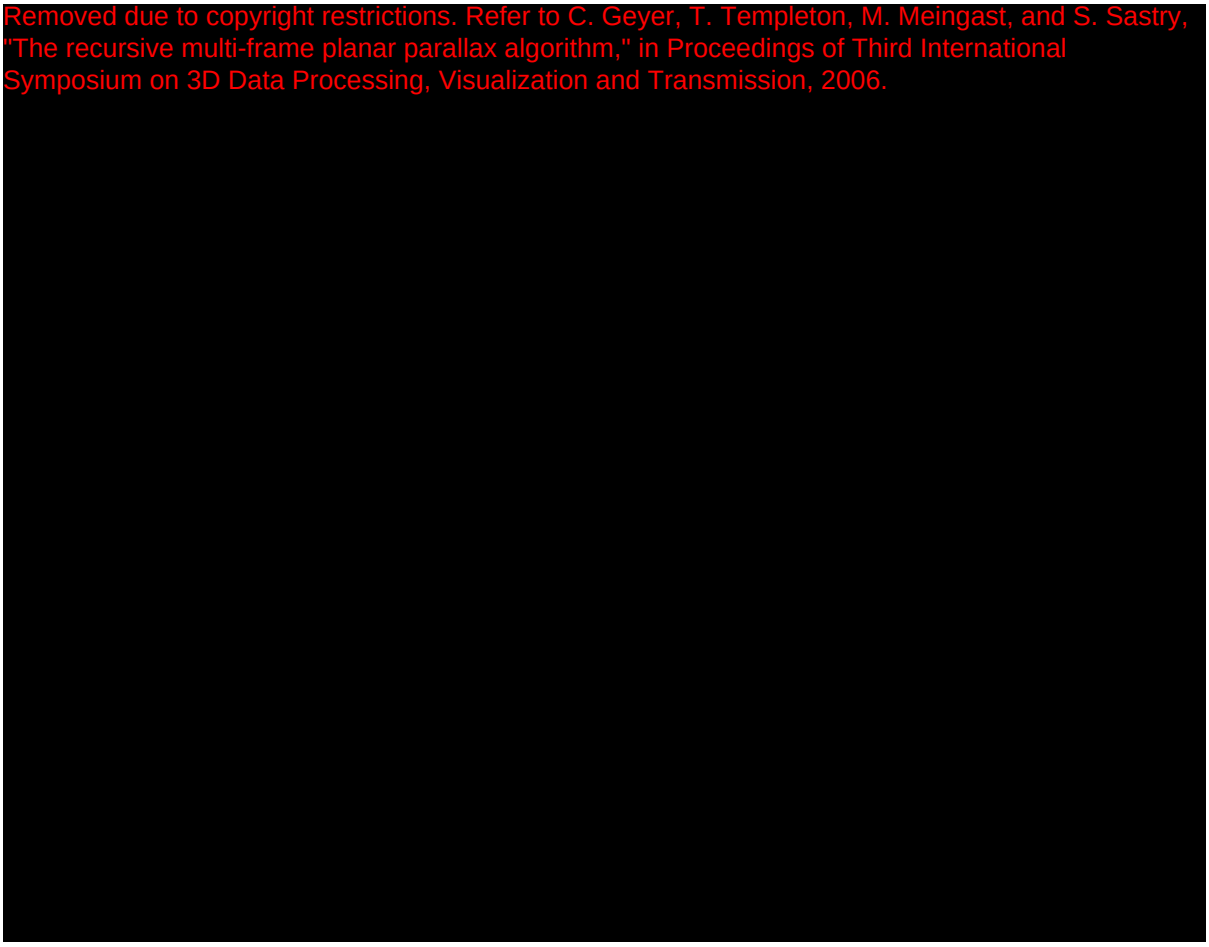

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

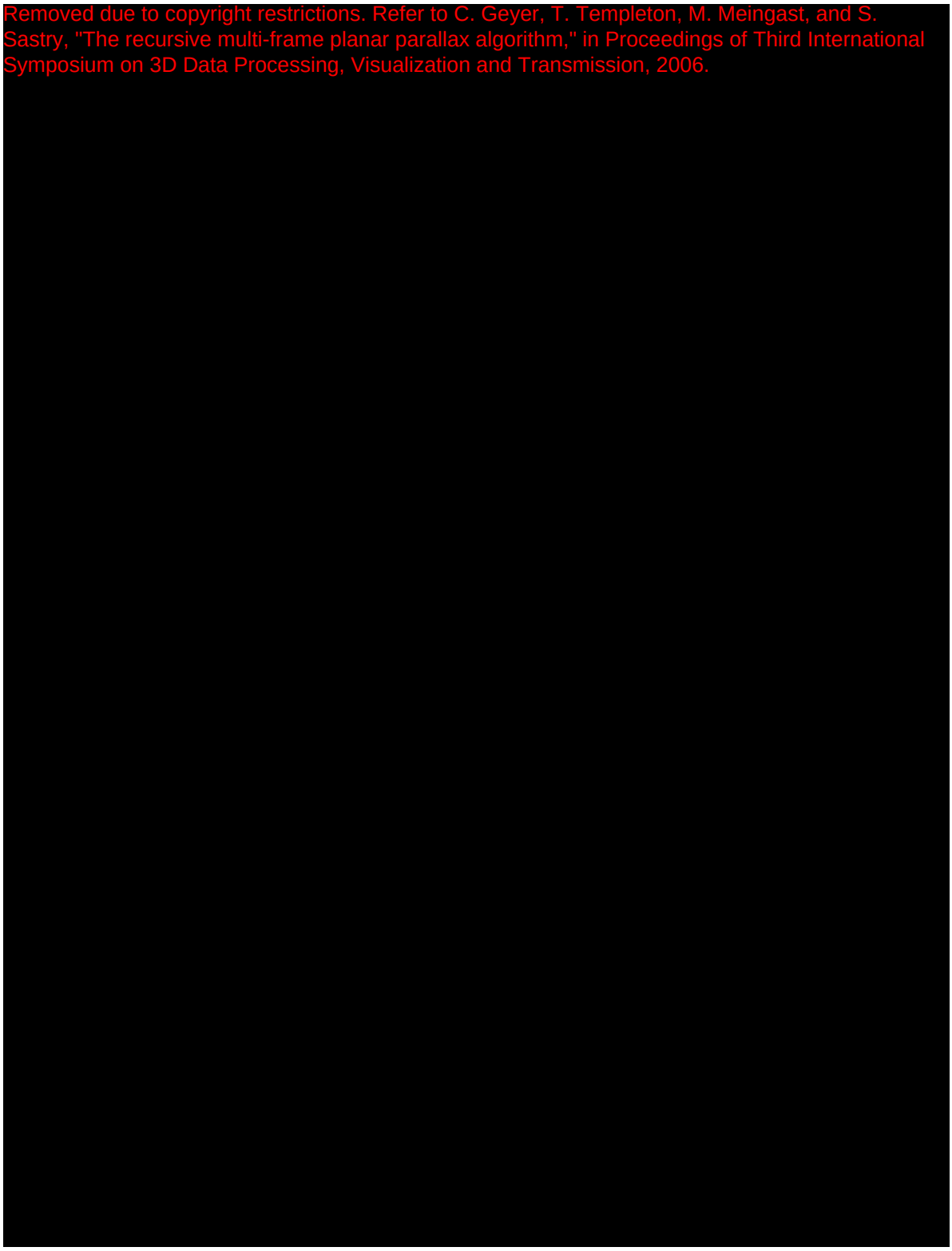

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

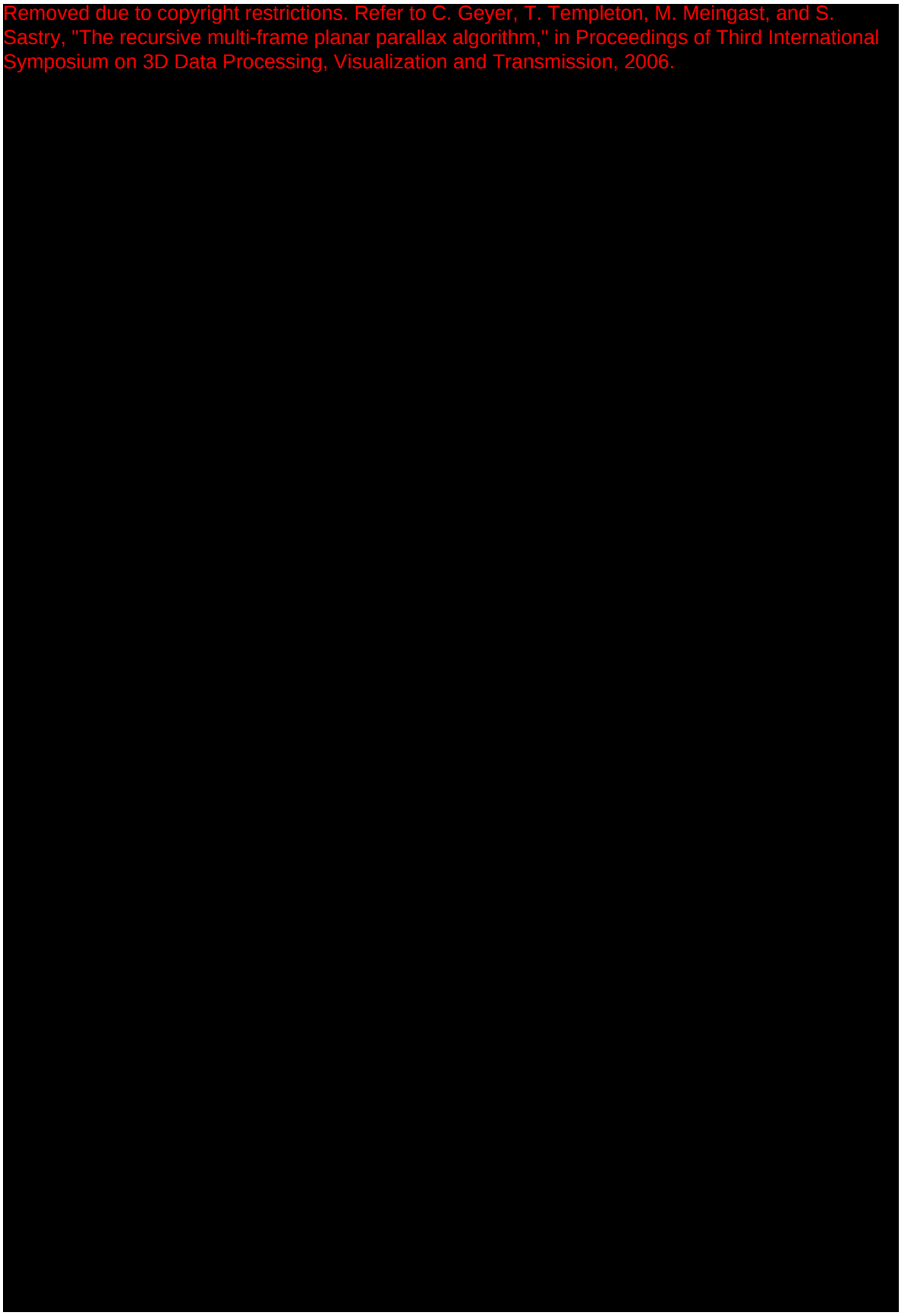

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

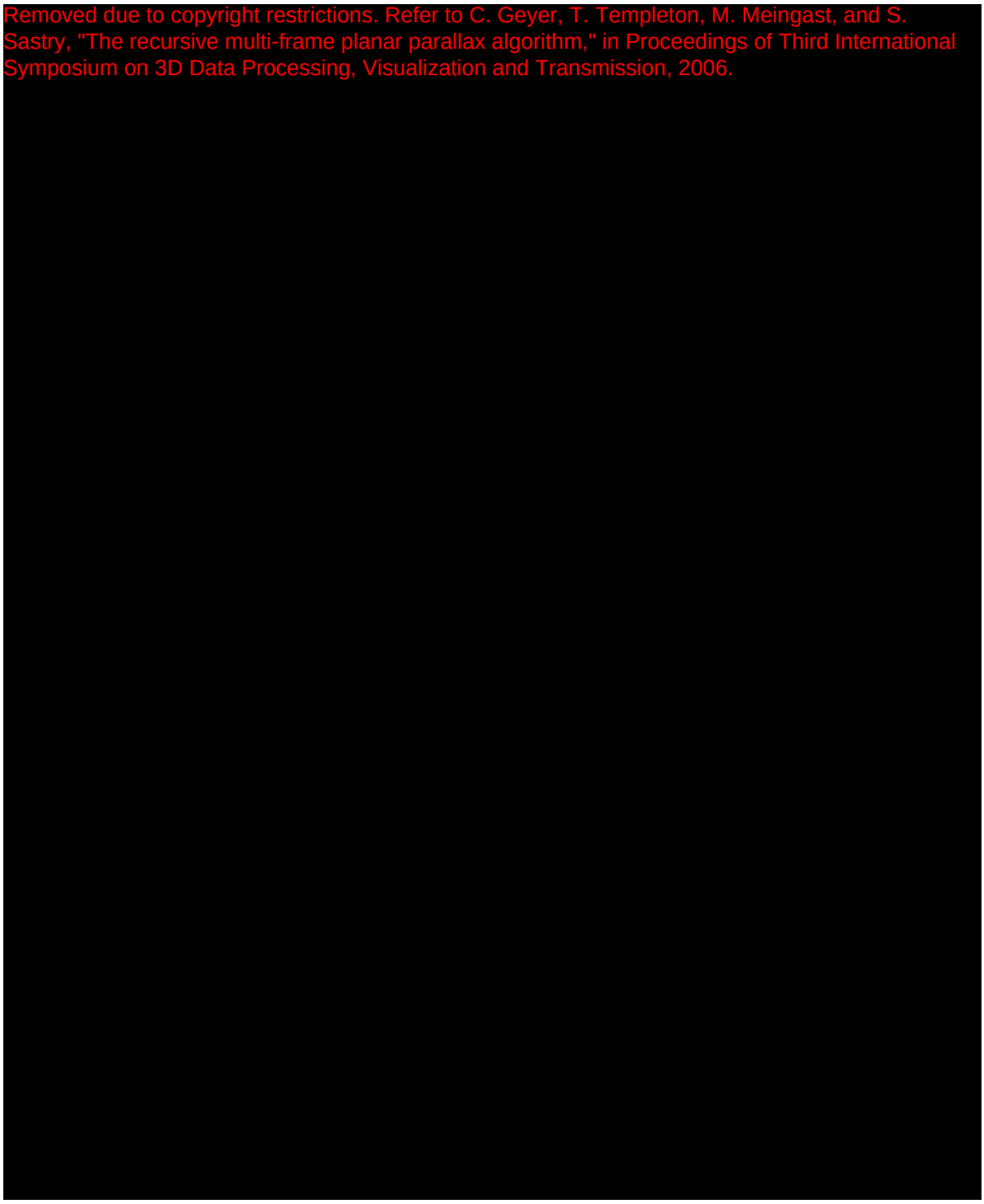

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

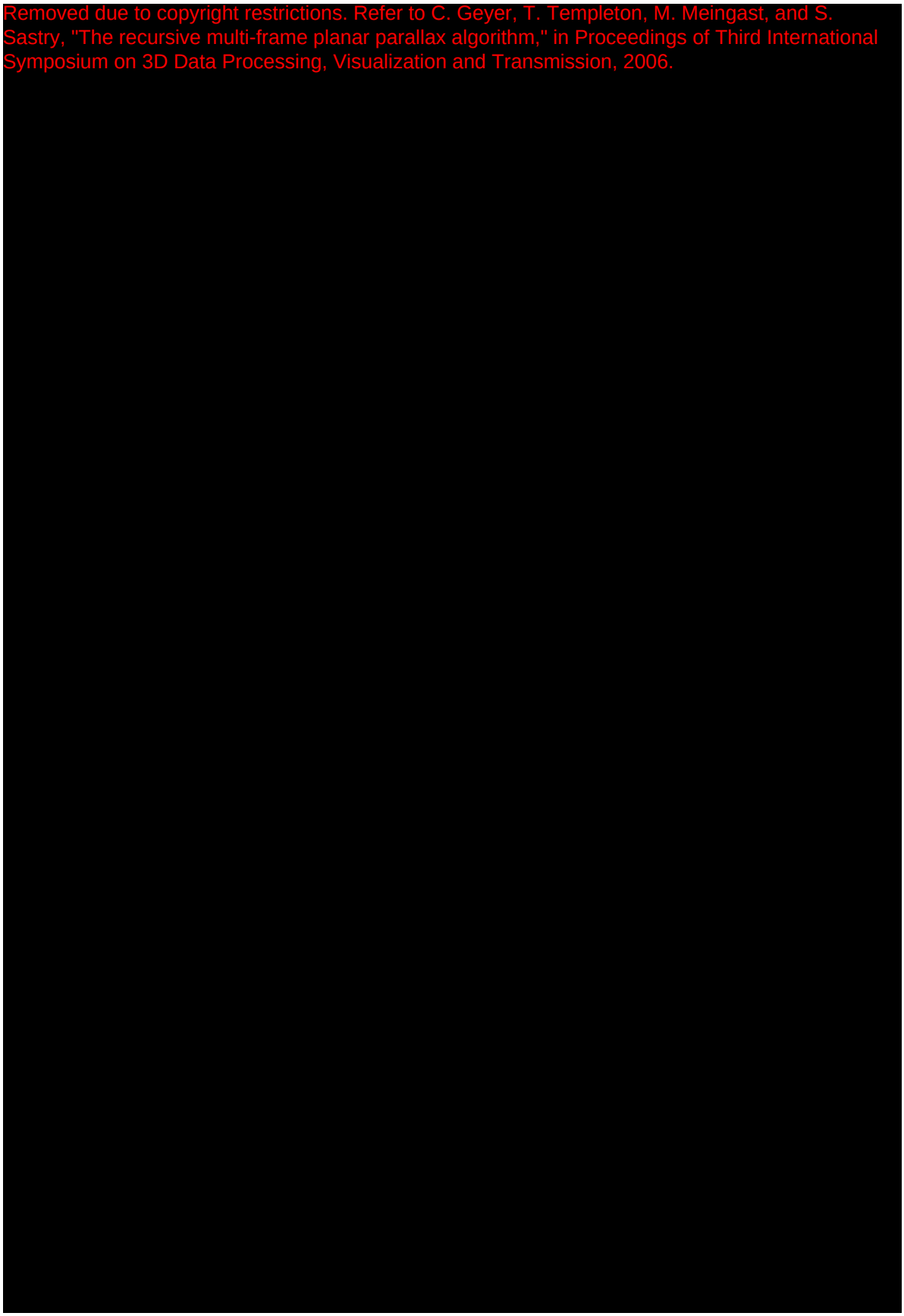

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

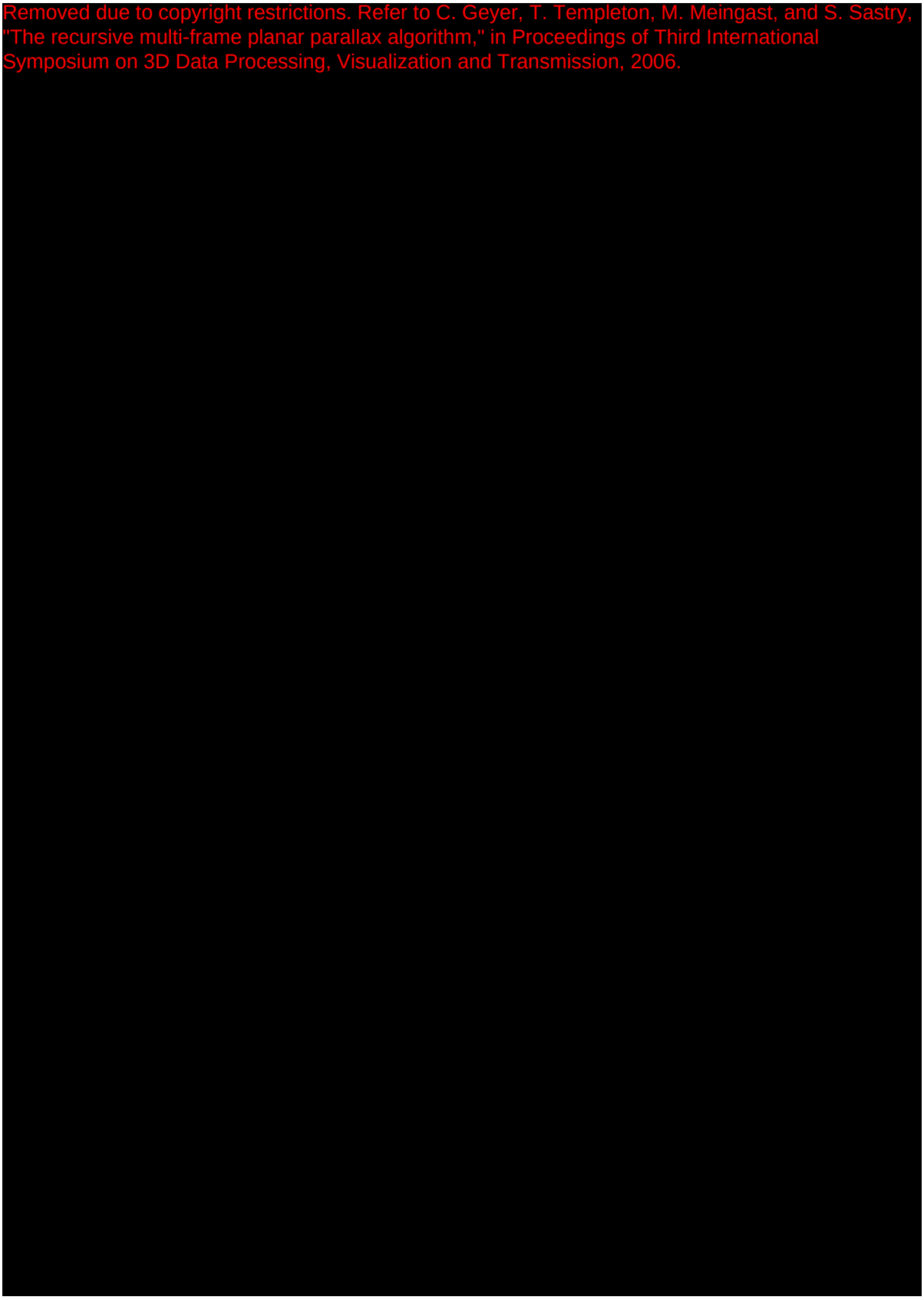

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

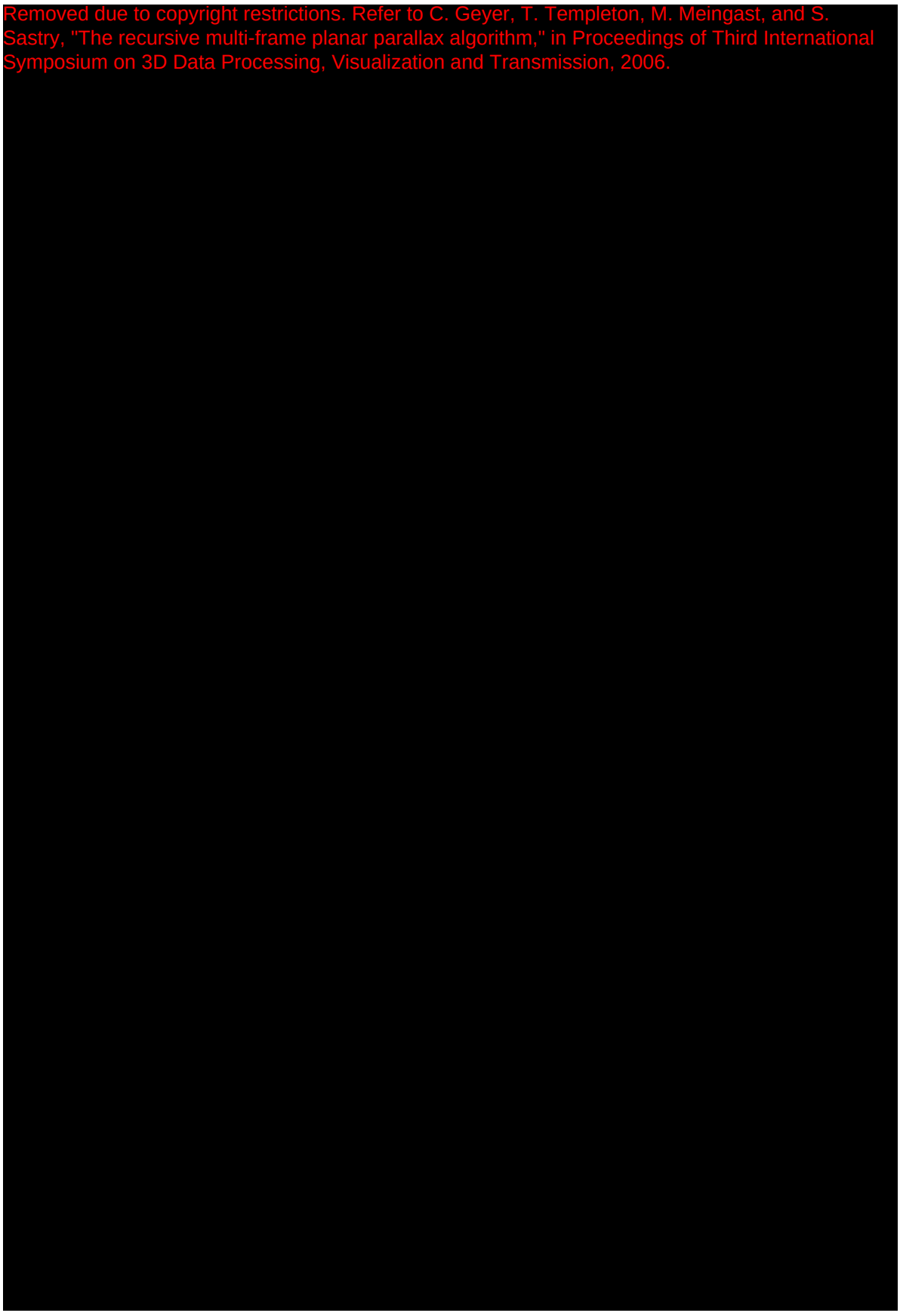

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission, 2006.


Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.


Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.


Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.


Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.


Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.

Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.

3.A World Coordinates

Combining the equation for height above the reference plane in the frame of the reference camera and the definition of γ, the point $\boldsymbol{p} = (x, y)$ in the reference image has z coordinate in the frame of the reference camera

$$z = -\frac{d_1}{\boldsymbol{N}^T \boldsymbol{X'} - \gamma}, \tag{3.20}$$

where $\boldsymbol{X'} = \mathsf{K}^{-1}\, \pi^* (\boldsymbol{p})$. Back-projecting into 3-dimensional space, the point $\boldsymbol{X}$ and its covariance in the frame of the reference camera $\operatorname{cov}(\boldsymbol{X})$ are given by:

$$\boldsymbol{X} = -\frac{d_1}{\boldsymbol{N}^T \boldsymbol{X'} - \gamma} \boldsymbol{X'} \tag{3.21}$$

$$\operatorname{cov}(\boldsymbol{X}) = \mathsf{J} \operatorname{var}(\gamma)\, \mathsf{J}^T \tag{3.22}$$

$$\text{where} \quad \operatorname{var}(\gamma) = \frac{1}{\Sigma A}$$

$$\text{and} \quad \mathsf{J} = \frac{d_1}{\left(\boldsymbol{N}^T \boldsymbol{X'} - \gamma\right)^2} \boldsymbol{X'}. \tag{3.23}$$

To construct an elevation map over multiple reference frames, the point and its covariance can be transformed into the world coordinate system using the known location and orientation of the reference camera.

Chapter 4

Probabilistic Motion Stamping

4.1 Introduction

This chapter describes a motion filter that uses both (noisy) feature image locations from a feature tracker and (noisy) attitude and position measurements, which makes it well-suited for an Unmanned Aerial Vehicle (UAV) that is equipped with a camera and a GPS/INS unit. It is designed for tight integration with a feature tracker and a map builder using the Recursive Multi-Frame Planar Parallax algorithm. It provides an estimate of the current attitude and position of the camera (motion stamp), relative to a fixed world coordinate frame, for each captured image; this estimate is essential to the operation of the map builder. It can also predict the locations of the previous features in the current frame, in order to limit the feature tracker's search space, as it evolves forward by an arbitrary time step to the arrival time of the current frame.

4.2 Requirements

In these systems, image and GPS/INS data may not arrive simultaneously or with the same frequencies, or even with consistent frequencies. It is therefore necessary to process the

two types of measurements separately while still combining them into a single estimate of the underlying system state. The state belief must be able to evolve forward by a variable amount of time to accommodate various time differences between measurements. It is highly desirable to provide predictions of the locations of the currently-tracked features in the current frame so that the feature tracker can limit its search space for each feature, shortening its runtime and increasing its accuracy. The filter also must provide an estimate of the current position of the camera relative to a fixed coordinate frame for the map-builder.

4.3 Notation

The notation used in this chapter is relatively standard but it is defined here for completeness.

Let the function π represent the projection of a pinhole camera, namely

$$\begin{bmatrix} u \\ v \end{bmatrix} = \pi\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \frac{f\begin{bmatrix} x \\ y \end{bmatrix}}{z} \tag{4.1}$$

where $\begin{bmatrix} x & y & z \end{bmatrix}^T$ is a point in 3-dimensional Euclidean space, $\begin{bmatrix} u & v \end{bmatrix}^T$ is the point's projection onto the image plane, and f is the focal length of the camera. The resulting vector $\begin{bmatrix} u & v \end{bmatrix}^T$ is used interchangeably with its homogeneous representation $\begin{bmatrix} u & v & 1 \end{bmatrix}^T$ to write expressions such as $\mathsf{A}\boldsymbol{y}_0$ where A is a matrix with 3 columns and $\boldsymbol{y}_0$ is the projection of a point in space onto the image plane. Similarly, to find the perpendicular distance between a point in space and the camera's focal point, $\rho = z = \boldsymbol{e}_3^T \begin{bmatrix} x & y & z \end{bmatrix}^T$.

This chapter uses the axis-angle parameterization of rotations to represent angular velocities. Define the skew-symmetric matrix $\hat{\omega}$ in $so(3)$ corresponding to the vector $\boldsymbol{\omega} =$

$\begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T$ by

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{4.2}$$

The matrix exponential of $\hat{\omega}$ is a rotation matrix in $SO(3)$ and is easily computed using Rodrigues' formula

$$R(\boldsymbol{\omega}) = e^{\hat{\omega}} = \mathsf{I} + \frac{\hat{\omega}}{\|\boldsymbol{\omega}\|}\sin(\|\boldsymbol{\omega}\|) + \frac{\hat{\omega}^2}{\|\boldsymbol{\omega}\|^2}\left(1 - \cos(\|\boldsymbol{\omega}\|)\right). \tag{4.3}$$

The inverse of Rodrigues' formula is denoted by the function $R^{-1} = Log_{SO(3)}$. Note that in the case of the system state $\boldsymbol{\xi}$, $\hat{\boldsymbol{\xi}}$ is the state estimate not a skew-symmetric matrix.

Although the axis-angle parameterization is convenient because it requires only three parameters to represent any rotation and because it makes it simple to scale the magnitude of the corresponding rotation, it suffers from a singularity in its Jacobian. Therefore, this chapter uses quaternions, which do not suffer from such a singularity, to represent attitudes. Let $\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T$ be a quaternion. A unit quaternion equivalent to $\boldsymbol{q}$ is

$$\begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \frac{1}{\|\boldsymbol{q}\|}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{4.4}$$

and the corresponding rotation matrix is

$$R(\boldsymbol{q}) = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_zq_w & 2q_xq_z + 2q_yq_w \\ 2q_xq_y + 2q_zq_w & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_xq_w \\ 2q_xq_z - 2q_yq_w & 2q_yq_z + 2q_xq_w & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix}. \tag{4.5}$$

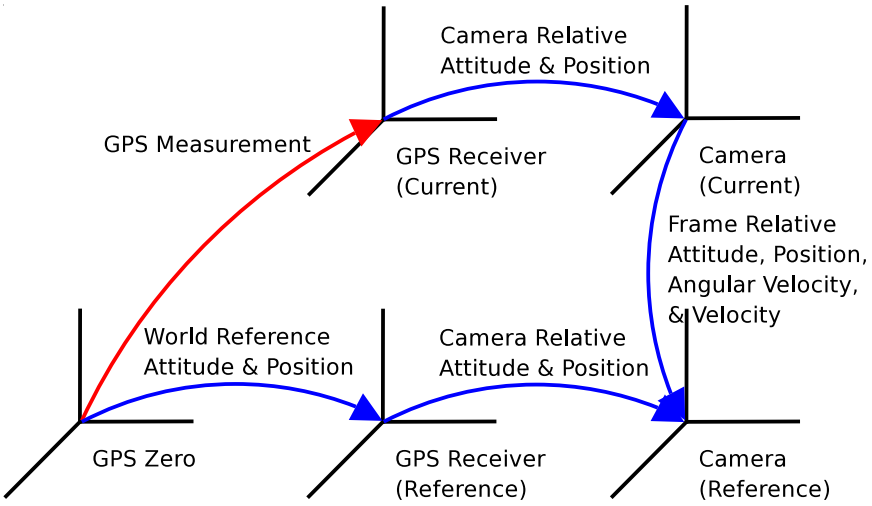


Figure 4.1: Model coordinate frames and transformations.

The inverse of this function results in a unit quaternion and is denoted by R^{-1}. Note that the corresponding functions for the axis-angle parameterization are also denoted by R and R^{-1}; the appropriate one to use is determined by whether the argument, in the case of R, or the result, in the case of R^{-1}, is an angular velocity in the axis-angle representation or an attitude in the quaternion representation.

4.4 System Model

The motion of the system is defined with reference frames and transformations as illustrated in Figure 4.1. $\boldsymbol{\Omega}$, $\boldsymbol{T}$, $\boldsymbol{\omega}$, and $\boldsymbol{V}$ are the attitude, position, angular velocity, and linear velocity of the camera with respect to its current configuration. $\boldsymbol{\Omega}_{rel}$ and $\boldsymbol{T}_{rel}$ are the attitude and position of the camera with respect to the GPS/INS unit. $\boldsymbol{\Omega}_{wr}$ and $\boldsymbol{T}_{wr}$ are the attitude and position of the vehicle (as measured at the GPS/INS unit) with respect to the global GPS zero. The motion of the vehicle is modeled as Brownian motion in $\boldsymbol{\omega}$ and $\boldsymbol{V}$.

The location of the i^{th} feature is represented by its projection $\boldsymbol{y}_0^i$ onto the initial image plane and by its distance ρ^i from the focal point of the initial camera. The focal length f of the camera is also included in the model and is modeled as a random walk if it can change

over time.

The evolution of the model is given by

$$\boldsymbol{y}_0^i(t+\Delta t) = \boldsymbol{y}_0^i(t) \quad i=1,\ldots,N \quad \boldsymbol{y}_0^i(0) = \boldsymbol{y}_0^i \tag{4.6}$$

$$\rho^i(t+\Delta t) = \rho^i(t) \quad i=1,\ldots,N \quad \rho^i(0) = \rho_0^i \tag{4.7}$$

$$\boldsymbol{\Omega}(t+\Delta t) = R^{-1}(R(\boldsymbol{\omega}(t)\,\Delta t)\,R(\boldsymbol{\Omega}(t))) \qquad \boldsymbol{\Omega}(0) = R^{-1}(\mathsf{I}) \tag{4.8}$$

$$\boldsymbol{T}(t+\Delta t) = R(\boldsymbol{\omega}(t)\,\Delta t)\,\boldsymbol{T}(t) + \boldsymbol{V}(t)\,\Delta t \qquad \boldsymbol{T}(0) = 0 \tag{4.9}$$

$$\boldsymbol{\omega}(t+\Delta t) = \boldsymbol{\omega}(t) + \boldsymbol{\alpha}_\omega(t) \qquad \boldsymbol{\omega}(0) = \boldsymbol{\omega}_0 \tag{4.10}$$

$$\boldsymbol{V}(t+\Delta t) = \boldsymbol{V}(t) + \boldsymbol{\alpha}_V(t) \qquad \boldsymbol{V}(0) = \boldsymbol{V}_0 \tag{4.11}$$

$$\boldsymbol{\Omega}_{wr}(t+\Delta t) = \boldsymbol{\Omega}_{wr}(t) \qquad \boldsymbol{\Omega}_{wr}(0) = \boldsymbol{\Omega}_{wr0} \tag{4.12}$$

$$\boldsymbol{T}_{wr}(t+\Delta t) = \boldsymbol{T}_{wr}(t) \qquad \boldsymbol{T}_{wr}(0) = \boldsymbol{T}_{wr0} \tag{4.13}$$

$$\boldsymbol{\Omega}_{rel}(t+\Delta t) = \boldsymbol{\Omega}_{rel}(t) \qquad \boldsymbol{\Omega}_{rel}(0) = \boldsymbol{\Omega}_{rel0} \tag{4.14}$$

$$\boldsymbol{T}_{rel}(t+\Delta t) = \boldsymbol{T}_{rel}(t) \qquad \boldsymbol{T}_{rel}(0) = \boldsymbol{T}_{rel0} \tag{4.15}$$

$$f(t+\Delta t) = f(t) + \alpha_f(t) \qquad f(0) = f_0 \tag{4.16}$$

where $\boldsymbol{\alpha}_V(t)$ and $\boldsymbol{\alpha}_\omega(t)$ represent the Brownian motion in the velocities of the vehicle and $\alpha_f(t)$ represents a random walk in f.

The observable measurements of the system are the projection $\boldsymbol{y}_i(t)$ of the i^{th} feature onto the current image plane (measured by the feature tracker) and the attitude $\boldsymbol{\Omega}_w(t)$ and position $\boldsymbol{T}_w(t)$ of vehicle with respect to the global GPS zero (measured by the GPS/INS unit). These measurements are given by

$$\boldsymbol{y}^i(t) = \pi\big(R(\boldsymbol{\Omega}(t))\,\boldsymbol{y}_0^i(t)\,\rho^i(t) + \boldsymbol{T}(t)\big) + \boldsymbol{n}^i(t)\,,\, i=1,\ldots,N \tag{4.17}$$

$$\boldsymbol{\Omega}_w(t) = R^{-1}\big(R(\boldsymbol{\Omega}_{wr}(t))\,R(\boldsymbol{\Omega}_{rel}(t))\,R(\boldsymbol{\Omega}(t))^{-1}\,R(\boldsymbol{\Omega}_{rel}(t))^{-1}\big) + \boldsymbol{n}^{\Omega_w}(t) \tag{4.18}$$

$$\begin{aligned}\boldsymbol{T}_w(t) &= R(\boldsymbol{\Omega}_{wr}(t))\big(\boldsymbol{T}_{rel}(t) - R(\boldsymbol{\Omega}_{rel}(t))\,R(\boldsymbol{\Omega}(t))^{-1}\,\mathsf{A}(t)\big) + \boldsymbol{T}_{wr}(t) + \boldsymbol{n}^{T_w}(t)\,,\\ &\quad \text{where } \mathsf{A}(t) \triangleq R(\boldsymbol{\Omega}_{rel}(t))^{-1}\,\boldsymbol{T}_{rel}(t) + \boldsymbol{T}(t)\end{aligned} \tag{4.19}$$

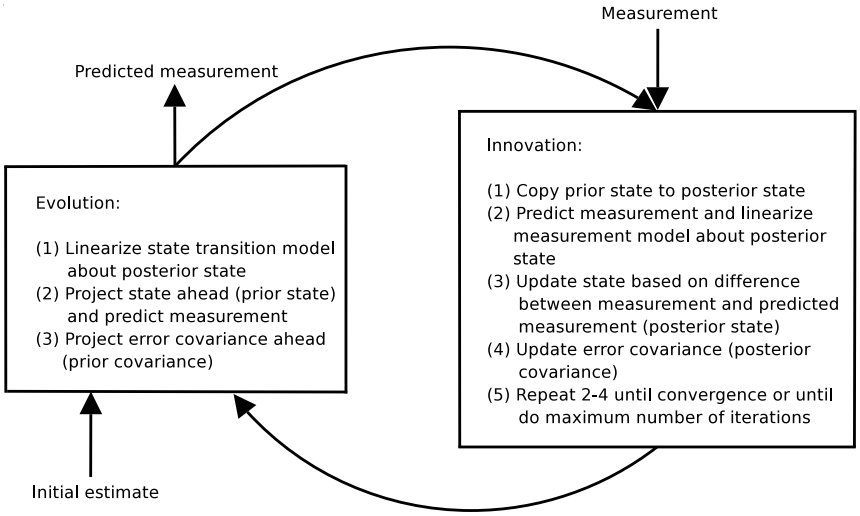


Figure 4.2: Iterated extended Kalman filter.

where $\boldsymbol{n}^{i}(t)$, $\boldsymbol{n}^{\Omega_w}(t)$, and $\boldsymbol{n}^{T_w}(t)$ are additive Gaussian measurement noise.

This is an extension of the model in [29] to include attitude and position measurements ($\boldsymbol{\Omega}_{wr}$, $\boldsymbol{T}_{wr}$, $\boldsymbol{\Omega}_{rel}$, $\boldsymbol{T}_{rel}$, $\boldsymbol{\Omega}_{w}$, $\boldsymbol{T}_{w}$) and to allow evolution by an arbitrary nonnegative Δt. It also eliminates the special features used for scale in [29] and instead makes the system observable using attitude and position measurements.

4.5 Implementation

The filter is implemented as an iterated extended Kalman filter in approximately 4000 lines of C++ code. The extended Kalman filter is an extension of the extended Kalman filter that allows multiple innovation iterations per evolution to achieve better convergence of the nonlinear system. Figure 4.2 illustrates the operation of the iterated extended Kalman filter, where in this case measurements are predicted both in the evolution step (to narrow the search space of the feature tracker) and in the innovation step (to compare to the actual measurements). For full descriptions and derivations of the extended Kalman filter and the iterated extended Kalman filter, see [30] and [31].

4.5.1 Measurements

The fact that feature and GPS/INS measurements are available at different times and with different frequencies requires that the two sets of measurements cannot be evolved and innovated together. Instead, there is one set of evolution and innovation routines for feature measurements and another for GPS/INS measurements. These routines must be run in pairs–a feature innovation must follow a feature evolution and an attitude and position innovation must follow an attitude and position evolution. Separating the evolution from the innovation allows the feature tracker to receive a new frame, evolve the filter to the time when the frame arrived and get predicted feature locations for that frame, locate the features using the predictions to limit its search, and finally innovate the filter using the observed projections of the features onto the image plane.

4.5.2 Appearing and Disappearing Features

Because features appear and disappear due to motion, nonvisible features must be removed from the model and new features must be added. For reasons of practicality, a feature that disappears and then later reappears is treated as two different features. Removing a feature is trivial (it only involves removing the feature from the state estimate), but adding a new feature is problematic because its initial distance from the focal point of the camera is unknown. To get around this problem, as in [29] each new feature is initially assigned an arbitrary distance ρ^i with large variance and placed in its own subfilter so that its large distance variance does not affect the main filter. A new feature is inserted into the main filter from its subfilter when its depth variance becomes comparable to the depth variance of the features in the main filter (implemented as when its depth variance becomes less than or equal to the mean depth variance of the features in the main filter or when the depth variance becomes less than or equal to a fixed threshold).

4.5.3 Initialization

The main filter state

$$\hat{\xi} = \begin{bmatrix} \boldsymbol{y}_0^{1T} & \rho^1 & \dots & \boldsymbol{y}_0^{NT} & \rho^N & \boldsymbol{\Omega}^T & \boldsymbol{T}^T & \boldsymbol{\omega}^T & \boldsymbol{V}^T & \boldsymbol{\Omega}_{wr}^T & \boldsymbol{T}_{wr}^T & \boldsymbol{\Omega}_{rel}^T & \boldsymbol{T}_{rel}^T & f \end{bmatrix}^T \tag{4.20}$$

is initialized using the measurements $\boldsymbol{y}^i(0)$ for $\boldsymbol{y}_0^i(0)$; arbitrary distances for ρ^i; $R^{-1}(\mathsf{I})$ for $\boldsymbol{\Omega}(0)$; 0 for $\boldsymbol{T}(0)$, $\boldsymbol{\omega}(0)$, and $\boldsymbol{V}(0)$; the most recent GPS/INS reading for $\boldsymbol{\Omega}_{wr}(0)$ and $\boldsymbol{T}_{wr}(0)$; a given estimate of $\boldsymbol{\Omega}_{rel}$ and $\boldsymbol{T}_{rel}$ for $\boldsymbol{\Omega}_{rel}(0)$ and $\boldsymbol{T}_{rel}(0)$ (determined by the vehicle); and a given estimate of f for $f(0)$ (determined by the camera). The main filter covariance P is initialized to be block diagonal with the covariance of the blocks corresponding to the $\boldsymbol{y}_0^i$ determined by analysis of the feature tracker, a relatively large number for the blocks corresponding to the ρ^i, 0 for the blocks corresponding to $\boldsymbol{\Omega}$ and $\boldsymbol{T}$, a relatively large number on the diagonal for the blocks corresponding to $\boldsymbol{\omega}$ and $\boldsymbol{V}$, a block corresponding to $\boldsymbol{\Omega}_{wr}$ and $\boldsymbol{T}_{wr}$ that is given by analysis of GPS/INS measurements, a relatively small number on the diagonal for the blocks corresponding to $\boldsymbol{\Omega}_{rel}$ and $\boldsymbol{T}_{rel}$, and a portion such as 0.1 of $f(0)$ as the initial variance of f.

No subfilters are created during initialization, and no features are allowed to move from a subfilter to the main filter until after the first 10 sets of feature evolutions and innovations (of the main filter) to allow the main filter to stabilize.

4.5.4 Main Filter

The system model in equations 4.6-4.19 above can be written in summary form as

$$\boldsymbol{\xi}(t+\Delta t) = f(\boldsymbol{\xi}(t)) + \boldsymbol{w}(t) \quad \boldsymbol{w}(t) \sim \mathcal{N}(0, \Sigma_w) \tag{4.21}$$

$$\begin{bmatrix} \boldsymbol{y}_{feat}(t) \\ \boldsymbol{y}_{body}(t) \end{bmatrix} = \begin{bmatrix} h_{feat}(\boldsymbol{\xi}(t)) \\ h_{body}(\boldsymbol{\xi}(t)) \end{bmatrix} + \boldsymbol{n}(t) \quad \boldsymbol{n}(t) \sim \mathcal{N}(0, \Sigma_n) \tag{4.22}$$

where $\boldsymbol{y}_{feat}(t)$ is the $\boldsymbol{y}^i(t)$ and $\boldsymbol{y}_{body}(t)$ is $\boldsymbol{\Omega}_w(t)$ and $\boldsymbol{T}_w(t)$.

As in [29], modified to iterate the innovation (IEKF) as in [31], the main filter implements the following equations:

Evolution:

$$\hat{\boldsymbol{\xi}}(t+\Delta t|t) = f\Big(\hat{\boldsymbol{\xi}}(t|t)\Big) \tag{4.23}$$

$$\mathsf{P}(t+\Delta t|t) = \mathsf{F}(t)\,\mathsf{P}(t|t)\,\mathsf{F}^T(t) + \Sigma_w \tag{4.24}$$

Innovation:

$$\begin{aligned}\hat{\boldsymbol{\xi}}(t+\Delta t|t+\Delta t) &= \hat{\boldsymbol{\xi}}(t+\Delta t|t) + \mathsf{L}(t+\Delta t)\Big(\boldsymbol{y}(t+\Delta t) - h\Big(\hat{\boldsymbol{\xi}}(t+\Delta t|t)\Big)\Big) \\ &\quad -\mathsf{H}\Big(\hat{\boldsymbol{\xi}}(t+\Delta t|t) - \hat{\boldsymbol{\xi}}(t+\Delta t|t+\Delta t)\Big) \qquad (4.25)\\ \mathsf{P}(t+\Delta t|t+\Delta t) &= \mathsf{\Gamma}(t+\Delta t)\,\mathsf{P}(t+\Delta t|t)\,\mathsf{\Gamma}^T(t+\Delta t) \\ &\quad +\mathsf{L}(t+\Delta t)\,\Sigma_n(t+\Delta t)\,\mathsf{L}^T(t+\Delta t) \qquad (4.26)\end{aligned}$$

For the first iteration of the innovation, $\hat{\boldsymbol{\xi}}(t+\Delta t|t+\Delta t)$ is set equal to $\hat{\boldsymbol{\xi}}(t+\Delta t|t)$.

Gain:

$$\mathsf{\Gamma}(t+\Delta t) \doteq \mathsf{I} - \mathsf{L}(t+\Delta t)\,\mathsf{H}(t+\Delta t) \tag{4.27}$$

$$\mathsf{L}(t+\Delta t) \doteq \mathsf{P}(t+\Delta t|t)\,\mathsf{H}^T(t+\Delta t)\,\mathsf{\Lambda}^{-1}(t+\Delta t) \tag{4.28}$$

$$\mathsf{\Lambda}(t+\Delta t) \doteq \mathsf{H}(t+\Delta t)\,\mathsf{P}(t+\Delta t|t)\,\mathsf{H}^T(t+\Delta t) + \Sigma_n(t+\Delta t) \tag{4.29}$$

Linearization:

$$\mathsf{F}(t) \doteq \frac{\partial f}{\partial \xi}\Big(\hat{\boldsymbol{\xi}}(t|t)\Big) \tag{4.30}$$

$$\mathsf{H}(t+\Delta t) \doteq \frac{\partial h}{\partial \xi}\Big(\hat{\boldsymbol{\xi}}(t+\Delta t|t+\Delta t)\Big) \tag{4.31}$$

Note that f and y in the above equations represent $f_{feat}(t)$ and $y_{feat}(t)$ when a feature measurement is available, and $f_{body}(t)$ and $y_{body}(t)$ when a GPS/INS measurement is available.

4.5.5 Subfilters

A subfilter is created when a new feature appears at time τ and its state $\hat{\boldsymbol{\xi}}_\tau = \begin{bmatrix} \boldsymbol{y}_\tau^{iT} & \rho_\tau^i \end{bmatrix}^T$ is initialized by its projection onto the current image plane ($\boldsymbol{y}_\tau^i$) and an arbitrary depth (ρ_τ^i). Its initial covariance P_τ is block diagonal with the block corresponding to $\boldsymbol{y}_\tau^i$ determined by analysis of the feature tracker, and with the variance of ρ_τ^i a relatively large number. Each subfilter also stores (but never changes except during recentering) the attitude and position estimate from the main filter for the time at which the subfilter is initialized. These stored estimates of $\boldsymbol{\Omega}$ and $\boldsymbol{T}$ at time τ are $\boldsymbol{\Omega}(\tau|\tau)$ and $\boldsymbol{T}(\tau|\tau)$.

As in [29], a subfilter implements the following equations:

Measurement:

$$\boldsymbol{y}^i(t) = \pi\big(R(\boldsymbol{\Omega}(t|t))\, R(\boldsymbol{\Omega}(\tau|\tau))^{-1} \left(\boldsymbol{y}_\tau^i(t)\, \rho_\tau^i(t) - \boldsymbol{T}(\tau|\tau)\right) + \boldsymbol{T}(t|t)\big) + \boldsymbol{n}^i(t) \tag{4.32}$$

where $\boldsymbol{n}^i(t)$ is additive Gaussian measurement noise.

Evolution:

$$\hat{\boldsymbol{\xi}}_\tau^i(t + \Delta t|t) = \hat{\boldsymbol{\xi}}_\tau^i(t|t) \tag{4.33}$$

$$\mathsf{P}_\tau(t + \Delta t|t) = \mathsf{P}_\tau(t|t) + \Sigma_w(t) \tag{4.34}$$

for $t > \tau$.

The subfilter innovation equations are identical to the main filter innovation equations using the above subfilter measurement equation.

When the variance of ρ_τ^i decreases to less than or equal to the mean distance variance

of all features in the main filter or when the variance of ρ_τ^i decreases to less than or equal to a fixed threshold, the subfilter is removed, the feature is projected back into the coordinate frame of the initial camera using

$$\boldsymbol{X}^i = R(-\boldsymbol{\Omega}(\tau|\tau)) - 1 \left(\boldsymbol{y}_\tau^i(t)\, \rho_\tau^i(t) - \boldsymbol{T}(\tau|\tau)\right) \tag{4.35}$$

$$\boldsymbol{y}_0^i(t) = \pi\left(\boldsymbol{X}^i\right) \tag{4.36}$$

$$\rho_0^i(t) = \boldsymbol{e}_3^T \boldsymbol{X}^i, \tag{4.37}$$

and the feature state and covariance are inserted into the main filter. Initially the covariance of the feature with the rest of the main filter state is zero.

Subfilters are not evolved or innovated for (GPS/INS) attitude and position measurement data because they do not include any reference frame state.

4.5.6 Recentering

To reduce feature uncertainty in the main filter, we recenter the reference camera coordinate system when we have no features from the image captured at the reference camera location remaining in the main filter, subject to a given minimum number of feature innovations between recenterings.

We first transform the main filter into the new coordinate system:

$$\begin{aligned}
\boldsymbol{y}_0^i(t|t)' &= \pi\big(R(\boldsymbol{\Omega}(t|t))\,\boldsymbol{y}_0^i(t|t)\,\rho^i(t|t) + \boldsymbol{T}(t|t)\big)\,, i = 1,\ldots,N && (4.38)\\
\rho^i(t|t)' &= \boldsymbol{e}_3^T\,\big(R(\boldsymbol{\Omega}(t|t))\,\boldsymbol{y}_0^i(t|t)\,\rho^i(t|t) + \boldsymbol{T}(t|t)\big)\,, i = 1,\ldots,N && (4.39)\\
\boldsymbol{\Omega}(t|t)' &= R^{-1}(\mathsf{I}) && (4.40)\\
\boldsymbol{T}(t|t)' &= \mathbf{0} && (4.41)\\
\boldsymbol{\Omega}_{wr}(t|t)' &= R^{-1}\big(R(\boldsymbol{\Omega}_{wr}(t|t))\,R(\boldsymbol{\Omega}_{rel}(t|t))\,R(\boldsymbol{\Omega}(t|t))^{-1}\,R(\boldsymbol{\Omega}_{rel}(t|t))^{-1}\big) && (4.42)\\
\boldsymbol{T}_{wr}(t|t)' &= R(\boldsymbol{\Omega}_{wr}(t|t))\big(\boldsymbol{T}_{rel}(t|t) - R(\boldsymbol{\Omega}_{rel}(t|t))\,R(\boldsymbol{\Omega}(t|t))^{-1}\,\mathsf{A}(t|t)\big) + \boldsymbol{T}_{wr}(t|t)\,,\\
&\quad \text{where } \mathsf{A}(t|t) \triangleq R(\boldsymbol{\Omega}_{rel}(t|t))^{-1}\,\boldsymbol{T}_{rel}(t|t) + \boldsymbol{T}(t|t)\,. && (4.43)
\end{aligned}$$

The covariance of the main filter is updated using the Jacobian of this transformation.

We must also correct the camera attitude and position estimate with which each subfilter was initialized by the same change in coordinate system:

$$\boldsymbol{\Omega}(\tau|\tau)' = R^{-1}\big(R(\boldsymbol{\Omega}(\tau|\tau))\,R(\boldsymbol{\Omega}(t|t))^{-1}\big) \quad (4.44)$$

$$\boldsymbol{T}(\tau|\tau)' = \boldsymbol{T}(\tau|\tau) - R(\boldsymbol{\Omega}(\tau|\tau))\,R(\boldsymbol{\Omega}(t|t))^{-1}\,\boldsymbol{T}(t|t)\,. \quad (4.45)$$

Note that no covariance needs to be updated for this transformation, since the covariance of $\boldsymbol{\Omega}(\tau|\tau)$ and $\boldsymbol{T}(\tau|\tau)$ is not maintained in the subfilter.

4.5.7 Tuning

Σ_{w} in the equations above is a tuning parameter. It is chosen to be block diagonal, with zeros corresponding to $\boldsymbol{\Omega}(t)$ and $\boldsymbol{T}(t)$ so that the system is a perfect integrator. Σ_w allows control over the smoothness of the estimate and depends on the amount of noise in the measurement data. In practice, Σ_w is chosen for a unit evolution $\Delta t = 1$ and is scaled by the current Δt during the evolution.

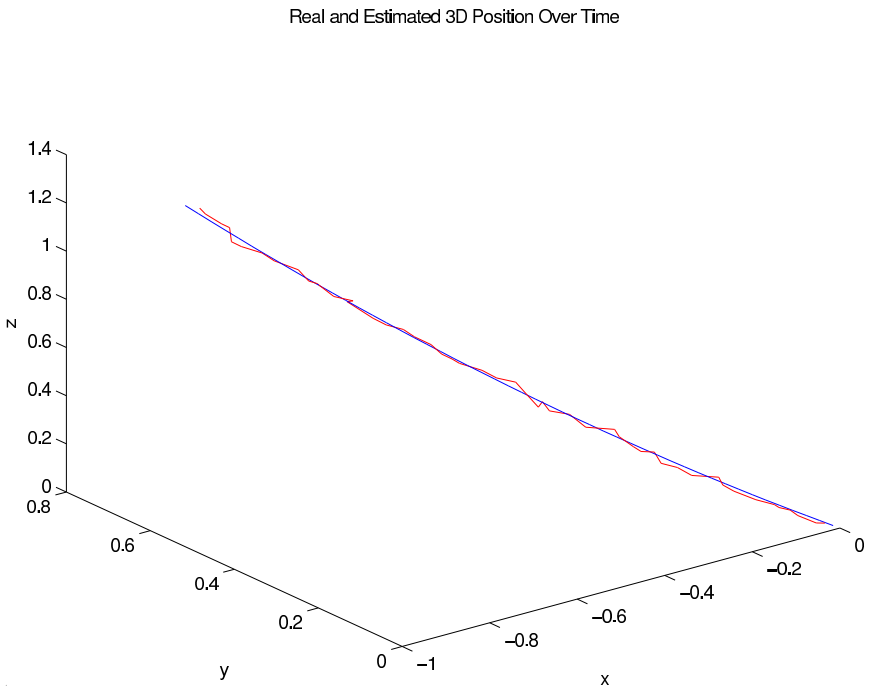


Figure 4.3: Synthetic experiment: screw trajectory (noiseless trajectory blue, filtered trajectory red).

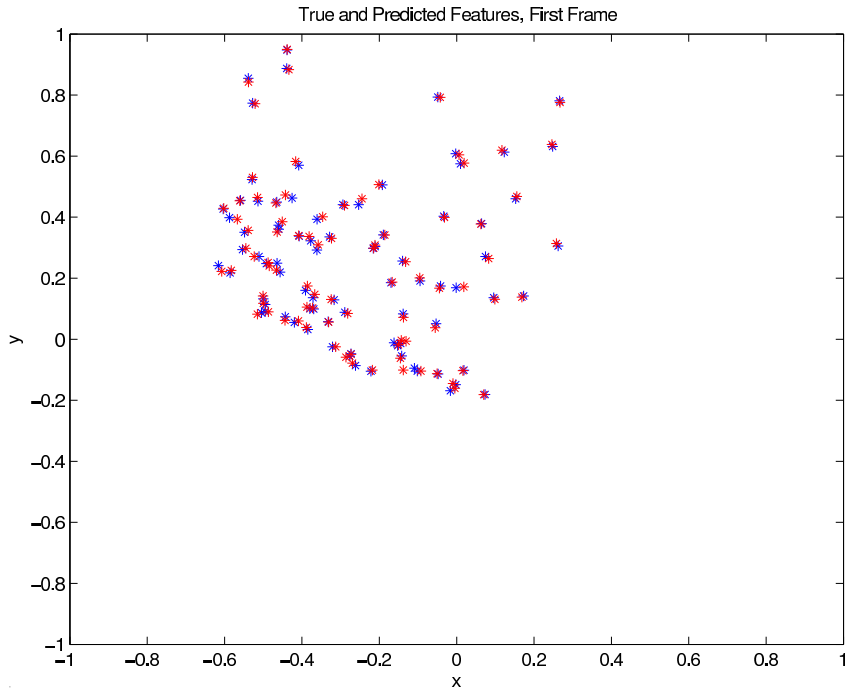


Figure 4.4: Synthetic experiment: first frame (noiseless features blue, filtered features red).

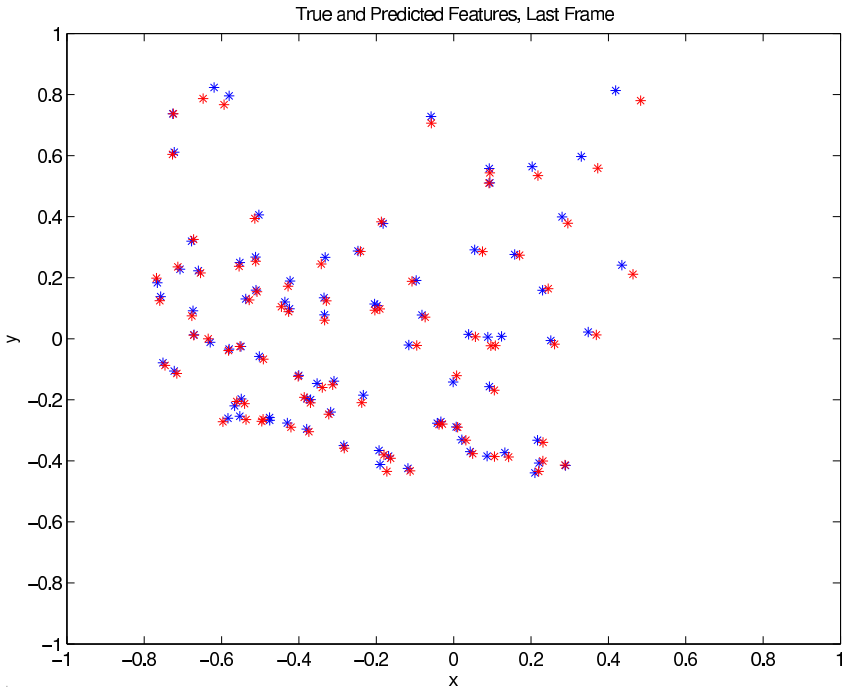


Figure 4.5: Synthetic experiment: last frame (noiseless features blue, filtered features red).

4.6 Results

Figures 4.3-4.5 show the result of running the filter on an interleaved sequence of 50 feature measurements and 50 GPS/INS measurements. The data is synthetic, which allows a comparison to ground truth, with the input to the filter corrupted by additive Gaussian noise with standard deviation 0.01. Figure 4.3 shows the vehicle trajectory, a shallow screw motion in three dimensions. There are 85 features per frame, and the input is constructed such that all features remain in the camera's field of view for the entire motion. Figures 4.4 and 4.5 show the camera image plane with the noiseless (before corruption) features in blue and the filtered features (the output of the filter) in red. Figure 4.4 shows the first usable frame in the sequence (the first frame is used for initialization) and Figure 4.5 shows the last frame in the sequence. Focal length filtering is not used for this sequence because the focal length is not changing over time.

Although the filtered trajectory in Figure 4.3 is not as smooth as the noiseless trajectory,

it does a reasonable job of following the motion considering that the input attitude and position are corrupted by noise with standard deviation 0.01. From Figures 4.4 and 4.5, the results appear to be particularly sensitive to radial noise, which is not surprising since radial noise is highly nonlinear and the filter optimizes based on repeated linearizations. Overall, the results are acceptable but not ideal.

4.7 Conclusion

This chapter has described a filter that integrates feature measurements from a feature tracker with attitude and position measurements from an onboard GPS/INS unit, taking into account all of the necessary coordinate frames and the transformations between them. The filter is also able to evolve the state by an arbitrary nonnegative time and to predict the locations of features in the current frame. This has yielded a filter with acceptable performance, although it is not yet suitably accurate for the adverse conditions of the aerial mapping and landing problem.

Chapter 5

Analytical Motion Stamping

5.1 Introduction

After experimenting with the probabilistic method discussed in the previous chapter, we described an analytical approach. Here we outline, and analyze the robustness of, this analytical motion-stamping algorithm.

The method discussed in this chapter assumes that the intrinsic calibration, i.e. the focal length and center of projection (and other parameters such as skew and radial distortion if they are necessary), of the camera is known. It also assumes that all sensors are colocated in both location and orientation; more realistically, it assumes that all sensors have been calibrated and that their measurements have been corrected to a common reference frame. These calibration algorithms are discussed in Section 6.4.

The remainder of this chapter is organized as follows: We begin by describing the analytical motion stamping algorithm in Section 5.2. Section 5.3 provides a sensitivity analysis and Section 5.4 discusses results on realistic synthetic data. Finally, Section 5.5 concludes.

5.2 Algorithm

To fulfill the RMFPP assumption that the motion is known, we must motion stamp each image (determine the world location and orientation of the camera when it was captured). Although we have location and orientation measurements available from helicopter GPS/INS sensors, they are noisy (particularly the location measurements, which are based on GPS) and do not in general correspond to times when images are captured. They do, however, provide absolute measurements of motion that, intuitively, we can use to correct for drift due to integrating the relative motion information provided by the camera over time.

Because the helicopter GPS/INS sensors are noisy, we (sparsely) track distinctive features between images and use these tracks to improve the accuracy of the online motion estimation. Although feature tracks do not give full orientation and location information, they do give very accurate two-dimensional projections of relative orientation and location. We also assume that the helicopter motion is smooth. Note that the system must run in real time, but that it may delay all frames by a constant amount of time to use 'future' information to estimate the location and orientation of the camera when an image was captured; this only results in a landing site being detected slightly later than it otherwise would be.

The online motion estimation algorithm first computes an initial value for the camera orientation, then combines a polynomial fit on the helicopter location measurements with information from feature tracks using least squares optimization, and finishes with a nonlinear refinement.

5.2.1 Orientation

Because the orientation measurements given by the helicopter sensors are relatively noiseless, because the relative orientation of the camera is independent of its relative position, and because the location estimation requires it, the camera orientation is estimated first.

The orientation estimate is the R that minimizes

$$\sum_i w_i ||\mathsf{R} - \hat{\mathsf{R}}_\mathsf{i}||_F \tag{5.1}$$

as in [32], where the $\hat{\mathsf{R}}_\mathsf{i}$ are preliminary estimates of R from feature tracks and from interpolation of calibrated GPS/INS vehicle orientation measurements, and the w_i are weights that decrease with increasing time between when the current image was captured and when the data used to estimate $\hat{\mathsf{R}}_\mathsf{i}$ was obtained (they can also adjust the overall weight between the two sources of $\hat{\mathsf{R}}_\mathsf{i}$ estimates). The minimum value is achieved when R is the polar factor of the polar decomposition [33] of

$$\mathsf{Q} = \sum_i w_i \hat{\mathsf{R}}_\mathsf{i} \tag{5.2}$$

provided that Q has positive determinant [32].

Each feature track $\hat{\mathsf{R}}_\mathsf{i}$ is obtained by estimating an essential matrix E (assuming that the scene is nonplanar) and a homography H (assuming that the scene is planar) between the current image and a past image, computing $\Delta\hat{\mathsf{R}}_\mathsf{i}$ from whichever of E and H is a better fit, and combining $\Delta\hat{\mathsf{R}}_\mathsf{i}$ with the final orientation estimate of the past image. The procedures for estimating E and H from feature correspondences and for computing $\Delta\hat{\mathsf{R}}_\mathsf{i}$ using these estimates are beyond the scope of this report; see [34] for details.

Each GPS/INS $\hat{\mathsf{R}}_\mathsf{i}$ is obtained by interpolating calibrated vehicle orientation measurements from before and after the current image was captured:

$$\hat{\mathsf{R}}_\mathsf{i} = \left(\mathsf{R}_{\mathsf{i}2}{\mathsf{R}_{\mathsf{i}1}}^T\right)^{\frac{t-t_{i1}}{t_{i2}-t_{i1}}} \mathsf{R}_{\mathsf{i}1} \tag{5.3}$$

where t is the time of the current frame, $\mathsf{R}_{\mathsf{i}1}$ and $\mathsf{R}_{\mathsf{i}2}$ are GPS/INS orientation measurements at times t_{i1} and t_{i2}, and $t_{i1} \leq t \leq t_{i2}$.

5.2.2 Location

The camera location estimate is a least squares estimate that combines a weighted polynomial fit of the calibrated vehicle locations with additional equations based on feature correspondences. Let t be the time of the image that we are currently trying to motion stamp, let $\{(t_i, \boldsymbol{T_i})\}$ be calibrated vehicle location measurements both before and after time t, and define $\Delta t_i = t_i - t$. To calculate a k-th order polynomial fit on the calibrated vehicle location measurements, we require linear equations of the form

$$\begin{bmatrix} \Delta t_i^k & \cdots & \Delta t_i^1 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & \Delta t_i^k & \cdots & \Delta t_i^1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & \Delta t_i^k & \cdots & \Delta t_i^1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{1k} \\ \vdots \\ \alpha_{11} \\ \alpha_{10} \\ \alpha_{2k} \\ \vdots \\ \alpha_{21} \\ \alpha_{20} \\ \alpha_{3k} \\ \vdots \\ \alpha_{31} \\ \alpha_{30} \end{bmatrix} = \boldsymbol{T_i} \quad (5.4)$$

where α_{pq} is the coefficient of the qth order term in the polynomial fit for the p-th dimension of the location. In practice we normalize each row of this equation by the norm of its row in the left-hand matrix or by a small constant, whichever is larger, and scale by $\exp \frac{1}{2}\frac{\Delta t_i^2}{\sigma_1^2}$ to regularize the system and then give relatively more weight to measurements that are closer in time to t. Note that this fit is constructed so that the camera location at time t (the location of interest) is trivially extracted from the parameter vector by $\boldsymbol{T} \leftarrow \begin{bmatrix} \alpha_{10} & \alpha_{20} & \alpha_{30} \end{bmatrix}^T$.

Consider a feature i that is visible in the current image and in a past image j that has been previously motion stamped. Let $\boldsymbol{x_i} = \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}$ and $\boldsymbol{x_{ij}} = \begin{bmatrix} x_{ij} & y_{ij} & 1 \end{bmatrix}$ be the calibrated homogeneous locations of the feature in the current and j-th images respectively. The locations of the feature in the three-dimensional coordinate frames of the current and j-th cameras are $\lambda_i \boldsymbol{x_i}$ and $\lambda_{ij} \boldsymbol{x_{ij}}$ respectively for some λ_i and λ_{ij}. Define the relative orientation and location of the current and j-th camera reference frames by $\mathsf{R_d}, \boldsymbol{T_d}$ such that $\mathsf{R_d}\left(\lambda_i \boldsymbol{x_i}\right) + \boldsymbol{T_d} = \lambda_{ij} \boldsymbol{x_{ij}}$. Then $\mathsf{R_d} = \mathsf{R_j}^T \mathsf{R}$ and $\boldsymbol{T_d} = \mathsf{R_j}^T \left(\boldsymbol{T} - \boldsymbol{T_j}\right)$ where $\mathsf{R}, \boldsymbol{T}$ is the orientation and location of the current frame and $\mathsf{R_j}, \boldsymbol{T_j}$ is the orientation and location of the j-th frame. We can eliminate one unknown from $\mathsf{R_d}\left(\lambda_i \boldsymbol{x_i}\right) + \boldsymbol{T_d} = \lambda_{ij} \boldsymbol{x_{ij}}$ by taking the cross product with $\boldsymbol{x_{ij}}$ on both sides; then $\lambda_i \left(\boldsymbol{x_{ij}} \times \mathsf{R_d} \boldsymbol{x_i}\right) + \boldsymbol{x_{ij}} \times \boldsymbol{T_d} = 0$. Substituting for $\mathsf{R_d}, \boldsymbol{T_d}$ gives $\lambda_i \left(\boldsymbol{x_{ij}} \times \mathsf{R_j}^T \mathsf{R} \boldsymbol{x_i}\right) + \boldsymbol{x_{ij}} \times \mathsf{R_j}^T \boldsymbol{T} = \boldsymbol{x_{ij}} \times \mathsf{R_j}^T \boldsymbol{T_j}$. Note that the only unknowns are $\boldsymbol{T}$ and λ_i. We can write this equation as

$$\begin{bmatrix} \boldsymbol{x_{ij}} \times \mathsf{R_j}^T & \boldsymbol{x_{ij}} \times \mathsf{R_j}^T \mathsf{R} \boldsymbol{x_i} \end{bmatrix} \begin{bmatrix} \alpha_{10} \\ \alpha_{20} \\ \alpha_{30} \\ \lambda_i \end{bmatrix} = \boldsymbol{x_{ij}} \times \mathsf{R_j}^T \boldsymbol{T_j} \tag{5.5}$$

where we use the identity $\boldsymbol{T} = \begin{bmatrix} \alpha_{10} & \alpha_{20} & \alpha_{30} \end{bmatrix}^T$. Note that the three equations given above are not independent, so we only use the first two rows. We normalize each row by the norm of its row in the left-hand matrix and then scale by $\rho \left(1 - \exp \frac{1}{2} \frac{\Delta t^2}{\sigma_2^2}\right)$, where $\Delta t = t_j - t$ and t_j is the time of the j-th frame; this regularizes the system, weights equations for frames that are farther apart in time more heavily, and weights the feature error relative to the polynomial fit error.

By stacking many equations corresponding to calibrated vehicle locations (polynomial fit) and many equations corresponding to feature pairs, we obtain a least squares problem in $\{\alpha_{pq}\}$ and $\{\lambda_i\}$. In practice we use all helicopter location measurements between M_p

frames before the current frame and M_f frames after the current frame (we delay RMFPP reconstruction by M_f frames). We use the N features in the current frame that have been tracked the longest and include equations of the above form for all of the past M_p images in which they appear (this allows a more accurate estimate of each λ_i and also allows us to introduce the fewest additional variables for a given number of feature correspondence equations). Note that $\lambda_i \boldsymbol{x_i}$ gives the location of the i-th feature in the reference frame of the current camera; using these points and the motion stamp for the current image, we can estimate the average height of the terrain for the RMFPP reference plane.

5.2.3 Nonlinear Refinement

Since R and $\boldsymbol{T}$ have been estimated separately, we perform a final nonlinear optimization using Sparse Bundle Adjustment (SBA) [13]. This is a local method based on the Levenberg-Marquardt algorithm that minimizes $||\boldsymbol{y} - f(\boldsymbol{x})||$ over $\boldsymbol{x}$, where $\boldsymbol{x}$ is a vector of the locations of all features in 3D space as well as the camera location and orientation for the current image (note that in general SBA can optimize over the camera locations and orientations for an arbitrary number of frames), $\boldsymbol{y}$ is a vector of all of the feature measurements in the last M_p images and in the current image, and the function f projects the 3D features into the last M_p images and the current image using all camera locations and orientations and the camera intrinsic calibration K. Note that the positions and orientations of the last M_p images are held fixed; only the location and orientation of the current image is allowed to vary.

5.2.4 Remark

Since the first frame that the motion estimation processes will not have any previous motion stamped frames it uses only GPS/INS $\hat{\mathsf{R}}_\mathsf{i}$ and performs only the polynomial fit on this image. It subsequently uses this motion stamp (and then motion stamps derived from this motion

stamp) to calculate new motion estimates. However, it quickly converges to more accurate results as this initial estimate becomes less influential. We turn on the RMFPP after the motion estimation has converged.

5.3 Robustness

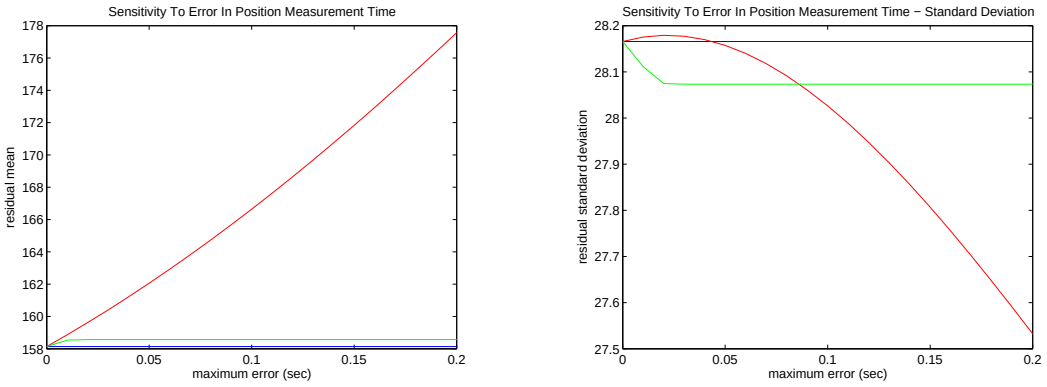
In this section we examine the robustness of the least-squares problem of minimizing $||\bar{\mathsf{A}}\boldsymbol{x} - \boldsymbol{b}||_2$ in the location estimate above. In particular, we consider worst-case minimization over additive noise to elements of $\bar{\mathsf{A}}$. A comparison of $\max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x}^{\boldsymbol{LS}} - \boldsymbol{b}||_2$ (where $\boldsymbol{x}^{\boldsymbol{LS}}$ is the least-squares solution) and $\min_x \max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2$ for different amounts of noise ρ indicates the sensitivity of the solution to the additive noise in $\bar{\mathsf{A}}$.

Consider $\min_x \max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2$ where $A_\rho = \left\{\bar{\mathsf{A}} + \Delta | - v_{ij}(\rho) \leq \delta_{ij} \leq v_{ij}(\rho)\right\}$. Then $\max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x}^{\boldsymbol{LS}} - \boldsymbol{b}||_2$ is equal to $|||\bar{\mathsf{A}}\boldsymbol{x}^{\boldsymbol{LS}} - \boldsymbol{b}| + \mathsf{V}|\boldsymbol{x}^{\boldsymbol{LS}}|||_2$ and $\min_x \max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2$ is equal to the square root of $\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{w},t} t$ s.t. $-\boldsymbol{y} \preceq \bar{\mathsf{A}}\boldsymbol{x} - \boldsymbol{b} \preceq \boldsymbol{y}$, $-\boldsymbol{w} \preceq \boldsymbol{x} \preceq \boldsymbol{w}$, and $||\boldsymbol{y} + \mathsf{V}(\rho)\boldsymbol{w}||_2 \leq t$. The latter is a Second-Order Cone Problem (SOCP), which we solve using the YALMIP [35] optimization framework and the SDPT3 [36] solver.

5.3.1 Error In Position Measurement Time

We first consider the robustness of the solution to error in the time at which the GPS/INS position is measured. Since our vision system periodically receives this data over an Ethernet connection and the clocks on the different computers are not synchronized, our program timestamps the data with the time at which it receives it (after an unknown delay on the other computer, transmission time, and thread- and process-switching time). We model this by an error in Δt_i in Equation 5.4 of at most ρ (and hence an error in Δt_i^j of at most ρ^j). In general this is an approximation that overestimates the sensitivity of the solution, but in this case we are using a linear fit in $\boldsymbol{T_i}$ so there is only one power of Δt_i^j.

Figure 5.1 shows the mean and standard deviation of the solutions to $\max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x}^{\boldsymbol{LS}} -$



(a) **(b)**

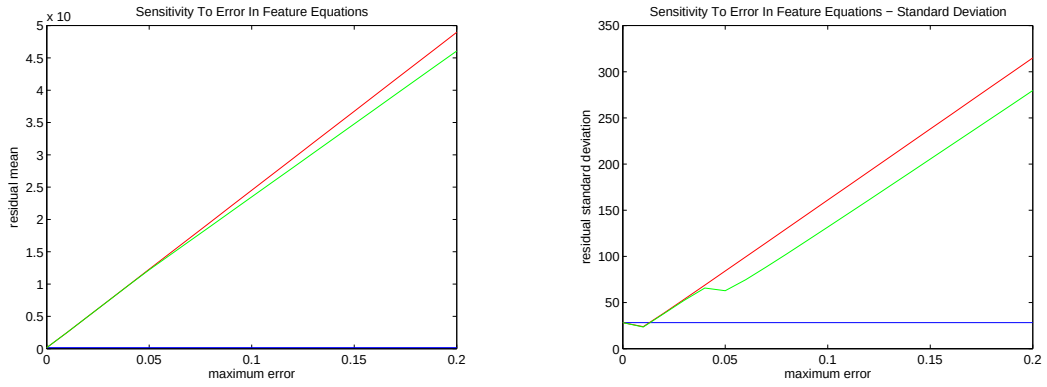
Figure 5.1: Mean (a) and standard deviation (b) of $\min_x ||\bar{\mathsf{A}}\boldsymbol{x} - \boldsymbol{b}||_2$ (blue), $\max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x}^{\boldsymbol{LS}} - \boldsymbol{b}||_2$ (red), and $\min_x \max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2$ (green) for different magnitudes of error in the times of the GPS/INS position measurements. Note that $\min_x ||\bar{\mathsf{A}}\boldsymbol{x} - \boldsymbol{b}||_2$ is independent of the maximum error ρ; it is included for comparison.

$\boldsymbol{b}||_2$ and $\min_x \max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2$ for different values of ρ using 20 different $\bar{\mathsf{A}}$, $\boldsymbol{b}$, and $\boldsymbol{x}^{\boldsymbol{LS}}$ obtained by running the motion stamp algorithm on realistic synthetic images and GPS/INS location and orientation data. The two curves for the mean are similar through $\rho \approx 0.01$, so the method is robust to variability in delay up to $\Delta t_i \approx 0.01$ second.

5.3.2 Error In Feature Equations

We now consider the robustness of the solution to error in the initial orientation estimate R, the past orientation estimates $\mathsf{R_j}$, the feature measurements in the current image $\boldsymbol{x_i}$, and the feature measurements in the past images $\boldsymbol{x_{ij}}$. This is modeled by an error of at most ρ in all elements of $\bar{\mathsf{A}}$ shown in Equation 5.5. In general this is an approximation that overestimates the sensitivity of the solution since the same terms appear in different elements of $\bar{\mathsf{A}}$ and since R and $\mathsf{R_j}$ are constrained to be rotation matrices.

Figure 5.2 shows the mean and standard deviation of the solutions to $\max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x}^{\boldsymbol{LS}} - \boldsymbol{b}||_2$ and $\min_x \max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2$ for different values of ρ using the same data as in the section above. There is no dramatic difference between the two curves like there was in



(a) **(b)**

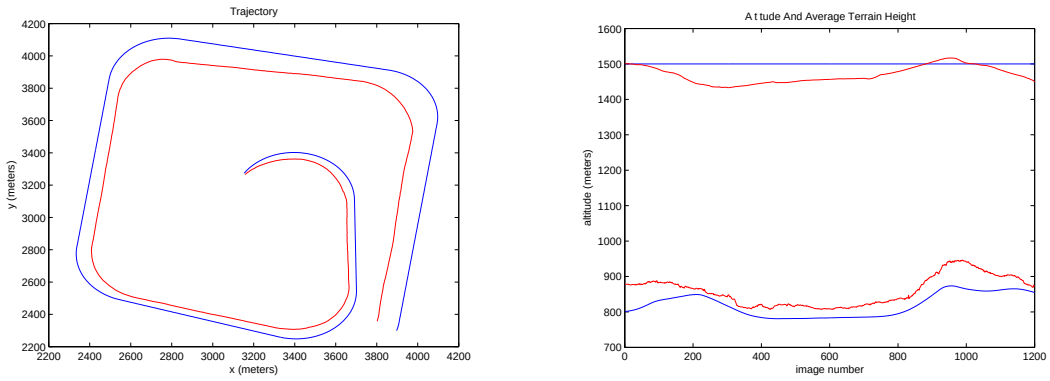
Figure 5.2: Mean (a) and standard deviation (b) of $\min_x ||\bar{\mathsf{A}}\boldsymbol{x} - \boldsymbol{b}||_2$ (blue), $\max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x}^{\boldsymbol{LS}} - \boldsymbol{b}||_2$ (red), and $\min_x \max_{\mathsf{A}\in A_\rho} ||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2$ (green) for different magnitudes of error in the feature equations. Note that $\min_x ||\bar{\mathsf{A}}\boldsymbol{x} - \boldsymbol{b}||_2$ is independent of the maximum error ρ; it is included for comparison.

the last section; the solution is robust to error in the feature equations. This is due to the fact that the feature equations only provide relative information (it is possible for the noisy elements to be consistent with the same solution), the fact that there are many such equations providing similar information, and the fact that the motion is also 'constrained' by the polynomial fit.

5.4 Results

We test the algorithm on a realistic synthetic sequence of images and GPS/INS data, with zero-mean Gaussian noise with standard deviation 3.0 meters added to $\boldsymbol{T_i}$ and a quaternion with mean $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$ and standard deviation $\begin{bmatrix} 0.01 & 0.01 & 0.01 & 0.01 \end{bmatrix}^T$ multiplied onto $\mathsf{R_i}$. The images were rendered from satellite imagery and Digital Terrain Elevation Data (DTED) elevation data (see sample rendered image in Figure 7.1) at 2.5 fps and the synthetic GPS/INS data arrived at 33.3 Hz. The system, including feature tracking and RMFPP, ran in real time on an Intel Pentium M 1.6 GHz processor.

Figure 5.3 shows the true (before noise was added) and estimated trajectory, altitude,



(a) **(b)**

Figure 5.3: True (blue) and estimated (red) trajectory (a), altitude (b, upper), and average terrain height (b, lower) using realistic synthetic imagery and GPS/INS data with zero-mean Gaussian noise with 3.0 meters standard deviation added to $\boldsymbol{T_i}$ and a quaternion with mean $[1, 0, 0, 0]^T$ and standard deviation $[0.01, 0.01, 0.01, 0.01]^T$ multiplied onto $\mathsf{R_i}$.

and average terrain height. Note that the estimated trajectory drifts from the true trajectory, but this is expected because of the high weight put on the feature equations compared to the polynomial (linear in this case) fit in GPS/INS position due to the large amount of noise in the GPS/INS measurements. The altitude and average terrain height remain within 100 meters of their true values. The algorithm tends to maintain the altitude at a constant height above the terrain, evidenced by the the two estimated altitudes being close to parallel. As desired, the estimates are locally consistent even though they drift over time.

5.5 Conclusion

We have motivated, described, and analyzed an analytical algorithm for determining the location and orientation of a camera using images and noisy GPS/INS location data that runs together with RMFPP in real time. We have shown that the trajectory that it estimates on noisy synthetic data drifts but is locally consistent, although its accuracy is not sufficient for the Recursive Multi-Frame Planar Parallax algorithm in practice.

Chapter 6

Experimental Platform

The content of Sections 6.2-6.3 is based on unpublished work coauthored with David Shim, Christopher Geyer, and Shankar Sastry [2]. This material is used with the permission of the coauthors.

6.1 Introduction

Although the final platform target for the aerial mapping landing system was the Boeing Maverick, a modified Robinson R22 (a full-sized 2-passenger helicopter), with vehicle communication using the Open Control Platform (OCP), the Boeing system was largely a black box. In this chapter we discuss the control and architecture of the Berkeley surrogate platform, as well as the sensor calibration that was essential to operation on both platforms.

6.2 Vehicle Setup

The testbed used in this research is based on an electrically powered RC helicopter, whose detailed specifications are given in the Table 6.1. The DC brushless motor with high-capacity Lithium-polymer batteries allows more than 10 minutes of continuous flight with

Base platform	Electric Helicopter (Maxi-Joker 2)
Dimensions	0.26 m (W) x 2.2 m (L) x 0.41 m (H)
Rotor Diameter	1.8 m
Weight	4.5 kg (no onboard electronics) 7.5 kg (fully instrumented)
Powerplant	Actro 32-4 motor (1740W max at 75A) Lithium-Ion-Polymer (10S4P; 40V 8Ah)
Operation Time	15 minutes
Avionics	Navigation: DGPS-aided INS GPS: NovAtel OEM4-MillenRT2 IMU: Inertial Science ISIS-IMU Flight Computer: PC104 Pentium III 700MHz Communication: IEEE 802.11b with RS-232 MUX Vision Computer: PC104 Pentium M 1.6GHz
Autonomy	Waypoint navigation with automatic VTOL Position-tracking servo mode MPC-enabled dynamic path planning with collision avoidance Stability-augmentation system

Table 6.1: Specification of the UAV testbed. *Courtesy of David Shim.*

the ease of fully automatic start-stop operation. The onboard components are designed and integrated with emphasis on weight reduction for longer flight time, reliability, and maneuverability. The vehicle is controlled by a PC104 form factor Pentium III 700MHz CPU with a custom servo interfacing board, an inertial measurement unit (IMU), a high-precision carrier-phase differential global positioning system, and an IEEE 802.11b device (see Figure 6.1). The flight control system communicates with the ground station over the wireless channel for receiving commands and sending the navigation status and system vital signs such as the battery level and the health of onboard components.

The vision-based terrain mapping and analysis system is implemented on a PC104 form factor Pentium M computer running Linux. The CPU is interfaced with a 2GB Compact Flash drive, a Firewire board, and Personal Computer Memory Card International Association (PCMCIA) wireless Ethernet. As shown in Figure 1.1, the vision computer is in-

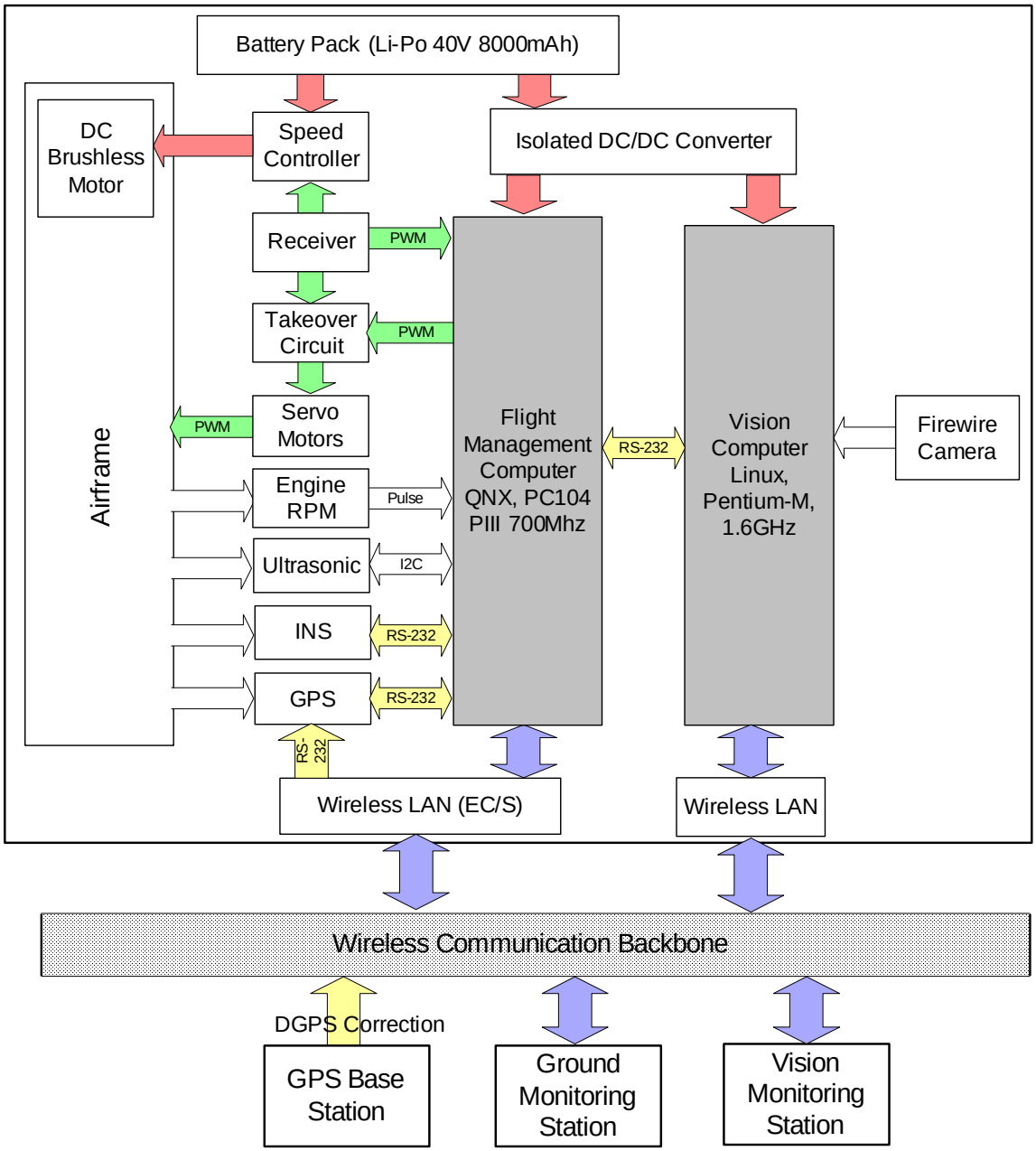


Figure 6.1: System architecture of the UAV testbed for vision-based landing and mapping. *Courtesy of David Shim.*

stalled in the nose of the vehicle in a dedicated aluminum enclosure for modularity and EMI shielding. A Firewire camera is installed in a forward- and downward-looking direction to capture the ground ahead of and below the vehicle. The vision system receives vehicle state data from, and sends trajectory commands to, the flight computer through the RS-232 serial interface.

As described in Section 2.6, the vision system requests appropriate flight patterns as it performs real-time terrain mapping and analysis at varying resolutions. The flight control system is responsible for guiding the vehicle through the requested waypoints with an acceptable accuracy. At a constant rate of 10Hz, the flight control system reports to the vision system the time-stamped navigation status such as position and attitude/heading, which are necessary to reconstruct the terrain with respect to an inertial reference frame (see Figure 6.1).

6.3 Vehicle Control

The flight control system consists of two hierarchical layers: the trajectory generator and the tracking controller. The trajectory generation is based on model predictive control (MPC) [37], which is solved in real time to find a sequence of control inputs that minimizes an appropriate cost function.

6.3.1 MPC-based Trajectory Generation

In [37, 38, 39], it was shown that model predictive control using penalty functions for state constraints and explicit input saturation [40] is a viable approach to address the guidance and control problems of UAVs at a reasonable computation load for real-time operation. In [38], an MPC-based control system was shown to have outstanding tracking control in the presence of coupled dynamic modes and substantial model mismatch. It has also been demonstrated that MPC-based optimization could be formulated to implement a higher

level of autonomy, such as real-time aerial collision avoidance [37], and obstacle avoidance in urban environment using onboard laser scanner [39]. In [41], an MPC-based trajectory planner is implemented as the feedforward control part of a two-degree-of-freedom control system. Here, as in [41], we will use a full vehicle kinodynamic model with input saturation.

Suppose we are given a time invariant nonlinear dynamic system

$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k)\,, \boldsymbol{u}(k))\,, \tag{6.1}$$

where $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}$ and $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^{n_u}$. The optimal control sequence over the finite receding horizon N is found by solving the nonlinear programming problem

$$V(\boldsymbol{x}, k, \boldsymbol{u}) = \sum_{i=k}^{k+N-1} L(\boldsymbol{x}(i), \boldsymbol{u}(i) + F(\boldsymbol{x}(k+N)), \tag{6.2}$$

where L is a positive definite cost function term and F is the terminal cost. When applied to the vision-based landing problem, L contains a term that penalizes the deviation from the desired trajectory. Suppose $\boldsymbol{u}^*(x, k)$ is the optimal control sequence that minimizes $V(\boldsymbol{x}, k, \boldsymbol{u})$ such that $V^*(\boldsymbol{x}, k) = V(\boldsymbol{x}, k, \boldsymbol{u}^*(\boldsymbol{x}, k))$, where $V^*(\boldsymbol{x}, k) \leq V(\boldsymbol{x}, k, \boldsymbol{u})$, $\forall \boldsymbol{u} \in \mathbb{U}$. With $\boldsymbol{u}^*(k)$, one can find $\boldsymbol{x}^*(k)$, $k = i, \cdots, i+N-1$ by solving recursively the given nonlinear dynamics with $\boldsymbol{x}(i) = \boldsymbol{x}_0(i)$ as the initial condition. We propose to use the control law

$$\boldsymbol{u}(k) = \boldsymbol{u}^*(k) + K(\boldsymbol{x}^*(k) - \boldsymbol{x}(k)). \tag{6.3}$$

For the feedback control law K, a control strategy similar to that in [42] is implemented. If the dynamic model used for solving the optimization problem perfectly matches the actual dynamics and the initial condition without any disturbance, there should not be any tracking error. In the real world, such an assumption cannot be satisfied. In this setup, with a tracking feedback controller in the feedback loop, the system can track the given

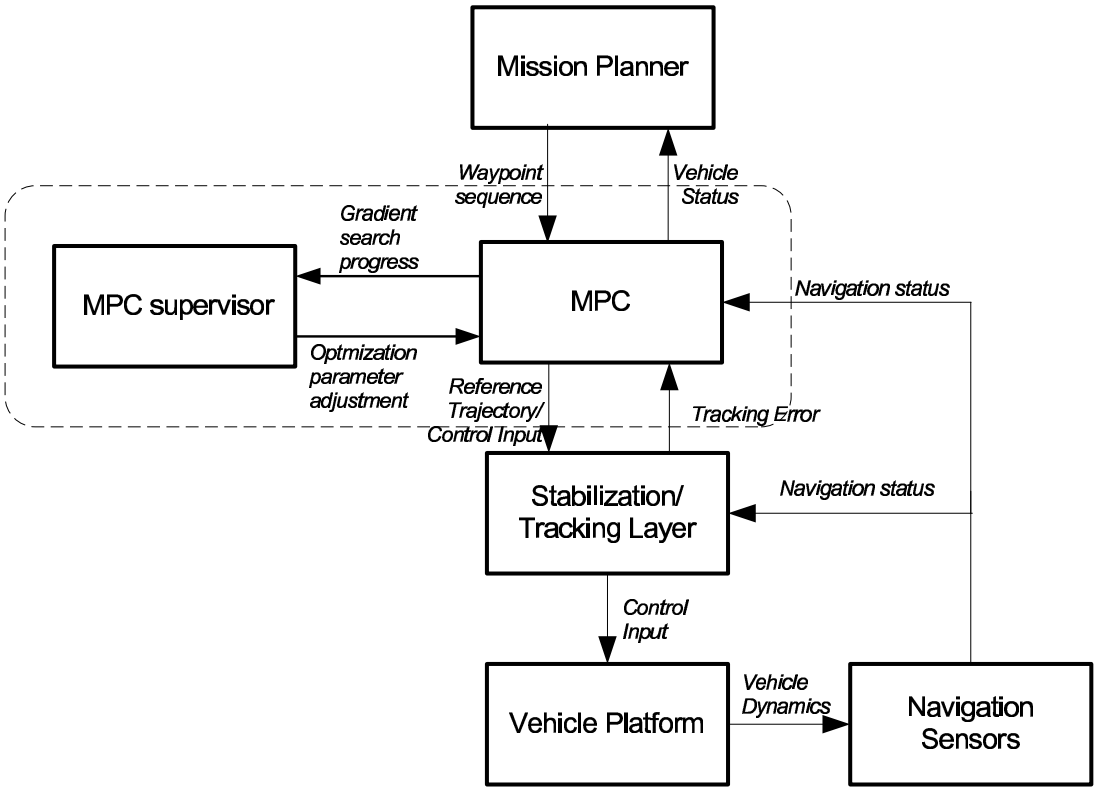


Figure 6.2: Flight control system architecture with MPC-based trajectory generator. *Courtesy of David Shim.*

trajectory reliably in the presence of disturbance or modeling error. The architecture of the proposed flight control system is given in Figure 6.2.

6.3.2 Flight Control Strategy for Autonomous Vision-based Landing

For automatic surveying, the control system should be able to guide the vehicle through the requested waypoints with minimal deviation while keeping the vehicle well within its dynamic performance boundary. The flight control should not induce any excessive vibration or rapid turns that may cause motion blur in the camera image.

The vision system requests one of three types of flight patterns as described in Figure 2.4. During the box search for coarse DEM construction, the vision system sends waypoints that are the vertices of piecewise linear segments. The vehicle is expected to fly along

the linear segments with minimal deviation while making bank-to-turn maneuvers around each vertex at a constant horizontal cruise speed and altitude. During the spiral descent, the vision unit sends a series of waypoints with much finer resolution. In final descent mode, the vision unit sends the landing coordinates to the flight control system, which is solely responsible for the stability and tracking accuracy of the descent maneuver until touchdown. Since helicopters go through substantial perturbation in the dynamics due to the ground effect, wind gusts, and reaction force from the ground, we believe it is best to leave the landing task to the flight control system, which is fine-tuned for the host vehicle.

In order to maintain a given trajectory under environmental disturbance, as opposed to heading straight toward a specific waypoint regardless of the current position, one has to know the previous waypoint[1]. Additionally, in order to achieve a smooth transition between waypoint requests with a constant cruise speed, one has to know the next waypoint *a priori* while the vehicle is approaching the current waypoint so that the flight control system can prepare for the bank-to-turn maneuver without any abrupt changes of heading or cruise velocity. The planning around a waypoint from the current location into a limited portion of future time perfectly fits the fundamental idea of MPC's receding horizon framework. To allow the vehicle flight mode described above, we use the following data structure, which specifies flight mode, past/current/future waypoints (see Figure 6.3), horizontal/vertical speed limit, heading rate limit, and a time tag, for communication from the vision computer to the flight computer over a wired RS-232 channel.

```
typedef struct {
  double GPSTime; // time tag [sec]
  // latitude, longitude, height
  double PastWP[3]; // (rad,rad,m)
  double CurrentWP[3]; // (rad,rad,m)
  double NextWP[3]; // (rad,rad,m)
```

[1]In our implementation, this point is not required to have been the actual previous waypoint; it is merely a point that defines a vector of approach to the current waypoint.

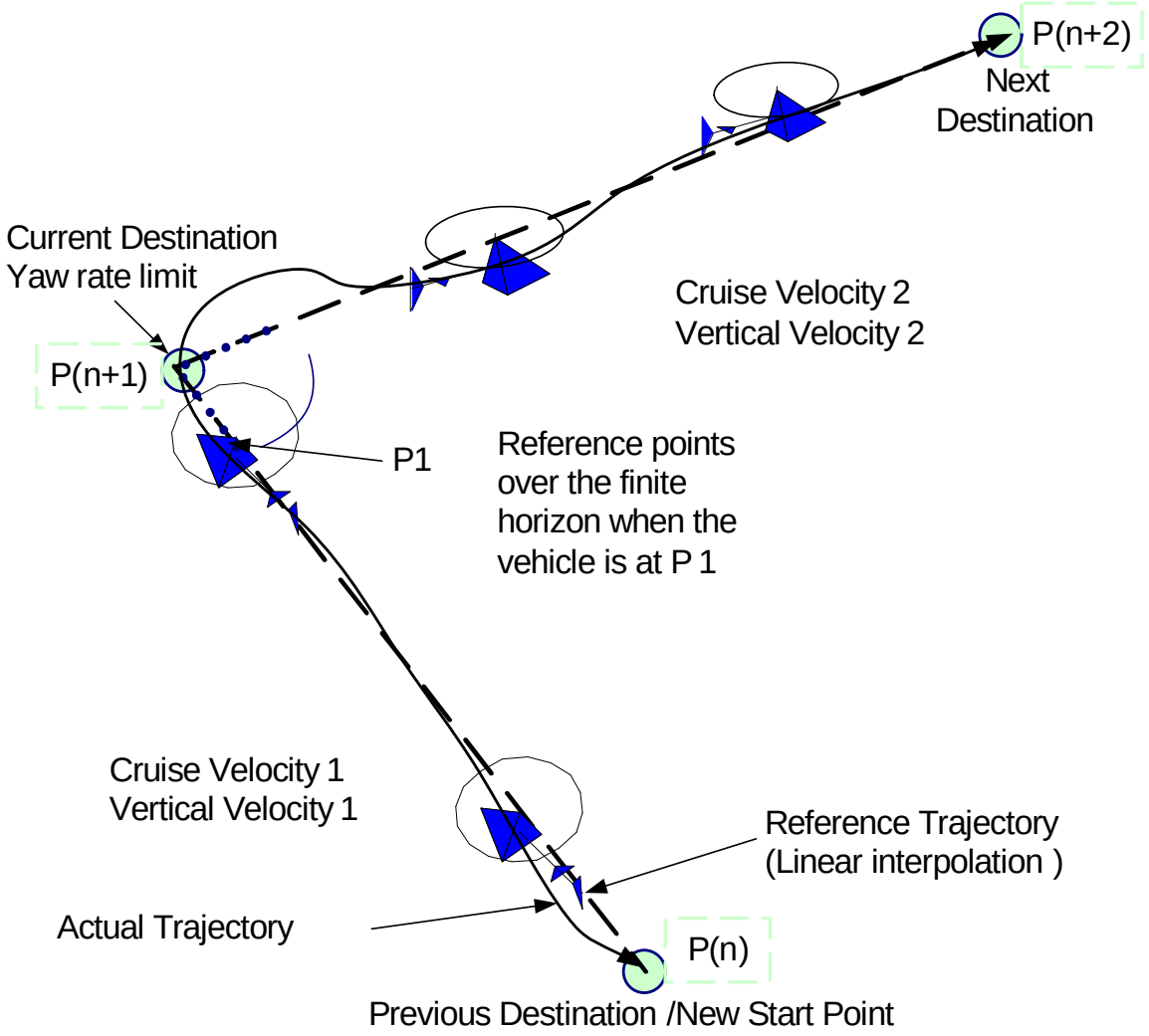


Figure 6.3: Three-point waypoint specification and MPC-based trajectory generation. *Courtesy of David Shim.*

```
  // cruise speed past WP to current WP
  float  Vcruise1; // [m/s]
  // cruise speed current WP to next WP
  float  Vcruise2; // [m/s]
  // reference vertical velocity
  float  Vvert; // [m/s]
  // reference yaw rate
  float  yaw_rate; // [rad/s]
  // ABORT(0), BOX_SEARCH(1),
  // SPIRAL_SEARCH(2), LANDING(3)...
  WORD   FlightMode;
  WORD   DataChecksum;
} MESSAGEBODY, *ptMESSAGEBODY;
```

6.3.3 Results

In order to build the terrain map or investigate a candidate landing site, the vision computer commands the vehicle to perform a box search or closer inspection spiral by requesting waypoints following the format defined above. The actual flight trajectory and other navigation states are presented in Figure 6.4. The vehicle initially follows the waypoints in a box pattern, and then, upon discovering a possible landing zone, the vehicle is commanded to fly in a spiral pattern at a very low velocity. After completing the closer inspection, the vision computer commands the vehicle to abort the spiral and resume the box search at normal flight speed. As shown in the figure, the helicopter follows this flight sequence with acceptable accuracy.

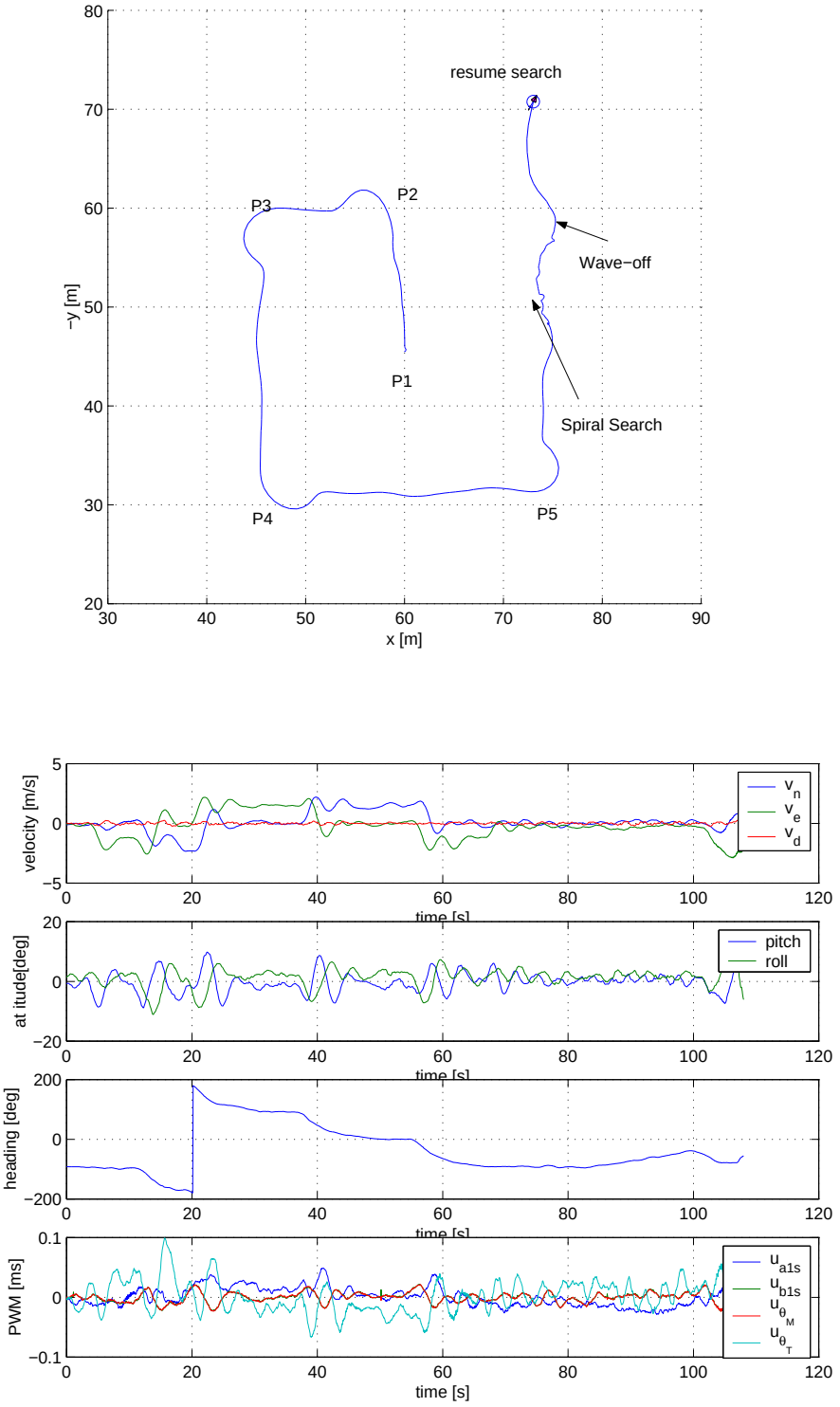


Figure 6.4: Experimental trajectory-following result. *Courtesy of David Shim.*

6.4 Extrinsic Calibration

In order to simplify the motion-stamping problem, we assume that the GPS/INS unit is colocated with the camera in position and orientation, or, more realistically, that its measurements can be corrected so that they are in the reference frame of the camera. Determination of the constant coordinate transformation between the camera and a GPS/INS unit requires data from both sensors during a broad range of positions and orientations, which is possible with the small Berkeley platform by maneuvering it manually (see Figure 6.5) but which is not possible with the full-sized Boeing platform. Calibration of the camera with the GPS/INS unit on the Boeing platform requires the composition of two coordinate transformations: the transformation between the vehicle GPS/INS unit and an additional GPS/INS unit on the (removable) camera platform, and the transformation between the additional GPS/INS unit and the camera. Since the additional GPS/INS unit is on the camera platform, the transformation between this sensor and the camera can be estimated from data collected by manual manipulation of the camera platform alone. The additional calibration between the two GPS/INS units is required to allow the system to use the vehicle GPS/INS unit, which is located near the vehicle's center of gravity and is of higher quality than the one on the camera platform.

6.4.1 GPS/INS to GPS/INS

In this section we assume that the two GPS/INS devices give measurements $(t_1, \mathsf{R}_1, \boldsymbol{T_1})$ and $(t_2, \mathsf{R}_2, \boldsymbol{T_2})$, where t_i is the time of the measurement, $\mathsf{R_i}$ is a 3x3 rotation matrix corresponding to the orientation of the sensor, and $\boldsymbol{T_i}$ is a 3x1 vector corresponding to the location of the sensor. $\mathsf{R_i}$ and $\boldsymbol{T_i}$ are defined such that $\boldsymbol{p_w} = \mathsf{R_i}\boldsymbol{p_i} + \boldsymbol{T_i}$ where $\boldsymbol{p_i}$ is a point in the reference frame of the ith GPS/INS device and $\boldsymbol{p_w}$ are the coordinates of the same point in the world reference frame. We want to find $(R, \boldsymbol{T})$ such that $\boldsymbol{p_1} = \mathsf{R}\boldsymbol{p_2} + \boldsymbol{T}$. Because the location measurements from the GPS/INS devices are very noisy and are gen-

Figure 6.5: Camera to GPS/INS calibration. *Courtesy of David Shim.*

erally inconsistent with each other (particularly in the vertical direction), and because we can find R without using $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$, we directly measure $\boldsymbol{T}$.

If we had pairs of measurements R_1 and R_2 at the same times $t = t_1 = t_2$, we would have direct measurements of R given by ${\mathsf{R}_1}^T\mathsf{R}_2$. In practice, we use pairs of measurements such that $|t_1 - t_2| \leq \delta$. For each such pair we form a linear equation of the form

$$\begin{bmatrix} \mathsf{R}_1 & 0 & 0 \\ 0 & \mathsf{R}_1 & 0 \\ 0 & 0 & \mathsf{R}_1 \end{bmatrix} \begin{bmatrix} r^{11} \\ r^{21} \\ \vdots \\ r^{33} \end{bmatrix} = \begin{bmatrix} r_2^{11} \\ r_2^{21} \\ \vdots \\ r_2^{33} \end{bmatrix} \tag{6.4}$$

where r^{ij} is the i, j element of R and r_2^{ij} is the i, j element of R_2. Stacking many such equations, we have $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$. We would like to find $\boldsymbol{x}$ that minimizes $||\mathsf{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$ subject to $\det \mathsf{R} = 1$ (R is a rotation matrix). If we relax the problem by eliminating the requirement that R be a rotation matrix then the unconstrained optimization problem is a least squares problem, which is easily solved by $\boldsymbol{x} = \mathsf{A}^\dagger \boldsymbol{b}$ where $\mathsf{A}^\dagger$ is the pseudoinverse of A. If $\tilde{\mathsf{R}}$ is the solution found by least squares, the closest orthogonal matrix (by Frobenius norm) is given by

$$\mathsf{R} = \tilde{\mathsf{R}} \left(\tilde{\mathsf{R}}^T \tilde{\mathsf{R}} \right)^{-\frac{1}{2}}, \tag{6.5}$$

the polar factor of the polar decomposition of $\tilde{\mathsf{R}}$ [32],[33].

6.4.2 Camera to GPS/INS

We assume that the GPS/INS device gives measurements $(t_1, \mathsf{R}_1, \boldsymbol{T_1})$ as above and that the camera gives measurements $(t_2, f_1, x_1, y_1, \ldots, f_n, x_n, y_n)$, where f_i indicates whether corner i on the chess board is visible in the image and, if it is visible, x_i, y_i specifies its location in calibrated image coordinates. $\mathsf{R_i}$ and $\boldsymbol{T_i}$ are defined as above. Once again, we want to find $(R, \boldsymbol{T})$ such that $\boldsymbol{p_1} = \mathsf{R}\boldsymbol{p_2} + \boldsymbol{T}$. Because the location measurements from the

GPS/INS device are very noisy (particularly in the vertical direction), and because we can find R without using $\boldsymbol{T_1}$ (or $\boldsymbol{T_2}$ for that matter), we directly measure $\boldsymbol{T}$.

Again we find pairs of measurements such that $|t_1 - t_2| \leq \delta$. We then take random pairs of these pairs, i.e. the pair consisting of $[(t_1, \mathsf{R}_1, \boldsymbol{T_1}), (t_2, f_1, x_1, y_1, \ldots, f_n, x_n, y_n)]$ and $\left[\left(t_1', \mathsf{R}_1', \boldsymbol{T_1'}\right), (t_2', f_1', x_1', y_1', \ldots, f_n', x_n', y_n')\right]$, such that the camera measurements have at least 10 chess board corners in common ($\sum f_i f_i' \geq 10$). Let $\boldsymbol{x_i} = \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T$ and $\boldsymbol{x_i'} = \begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix}^T$. Using the point correspondences $\left\{\boldsymbol{x_i}, \boldsymbol{x_i'} | f_i f_i' = 1\right\}$, we compute a homography H using the normalized Direct Linear Transform (DLT) algorithm [34]. If all of the corresponding world points lie exactly on a plane, $\lambda \boldsymbol{x_i'} = \mathsf{H}\boldsymbol{x_i}$ for some $\lambda \; \forall i$ such that $f_i f_i' = 1$ (H is only defined up to scale). Therefore $\boldsymbol{x_i'} \times \mathsf{H}\boldsymbol{x_i} = 0$. Arranging into a standard linear form, we get

$$\begin{bmatrix} \mathbf{0}^T & -\boldsymbol{x_i}^T & y_i' \boldsymbol{x_i^T} \\ \boldsymbol{x_i}^T & \mathbf{0}^T & -x_i' \boldsymbol{x_i}^T \end{bmatrix} \begin{bmatrix} h^{11} \\ h^{12} \\ \vdots \\ h^{33} \end{bmatrix} = \mathbf{0} \tag{6.6}$$

where h^{ij} is the i, j element of H and we have omitted the third row because it is linearly dependent on the first two. Since the corner detection is not exact (it is discretized to the nearest pixel), we want to find H that minimizes $||\mathsf{A}\boldsymbol{h}||_2^2$ subject to $||\boldsymbol{h}||_2 = 1$, where $\mathsf{A}\boldsymbol{h}$ is at least 10 of the above equations stacked together (theoretically only 8 correspondences are required to estimate a homography, but in practice better results are achieved with more than 8 points). The solution to this optimization problem is given by the least singular vector of A. This is the unnormalized DLT algorithm. The normalized DLT algorithm finds similarity transforms T and T', each consisting of a translation and a scaling (both in the xy plane), such that $\{\mathsf{T}\boldsymbol{x_i} | f_i f_i' = 1\}$ and $\left\{\mathsf{T}'\boldsymbol{x_i'} | f_i f_i' = 1\right\}$ both have mean $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and average distance $\sqrt{2}$ from the origin in the xy plane. It then computes $\tilde{\mathsf{H}}$ using the

unnormalized DLT algorithm on the normalized data; the final result is $\mathsf{H} = \mathsf{T}'^{-1}\tilde{\mathsf{H}}\mathsf{T}$.

Since the scene is planar, the relative rotation between the two cameras in the pair is given, up to scale, by H [34]. Therefore we find ${\mathsf{R}'_2}^T\mathsf{R}_2$ using Equation 6.5, using as input H normalized so that it has positive determinant and one column of unit norm. We find ${\mathsf{R}'_1}^T\mathsf{R}_1$ directly from the GPS/INS data. Since $\left({\mathsf{R}'_2}^T\mathsf{R}_2\right) = \mathsf{R}^T\left({\mathsf{R}'_1}^T\mathsf{R}_1\right)\mathsf{R}$, for each $\left[(t_1, \mathsf{R}_1, \boldsymbol{T_1}), (t_2, f_1, x_1, y_1, \ldots, f_n, x_n, y_n), \left(t'_1, \mathsf{R}'_1, \boldsymbol{T'_1}\right), (t'_2, f'_1, x'_1, y'_1, \ldots, f'_n, x'_n, y'_n)\right]$ we can write

$$\left(\begin{bmatrix} {\mathsf{R}'_1}^T\mathsf{R}_1 & 0 & 0 \\ 0 & {\mathsf{R}'_1}^T\mathsf{R}_1 & 0 \\ 0 & 0 & {\mathsf{R}'_1}^T\mathsf{R}_1 \end{bmatrix} - \begin{bmatrix} q^{11}\mathsf{I} & q^{21}\mathsf{I} & q^{31}\mathsf{I} \\ q^{12}\mathsf{I} & q^{22}\mathsf{I} & q^{32}\mathsf{I} \\ q^{13}\mathsf{I} & q^{23}\mathsf{I} & q^{33}\mathsf{I} \end{bmatrix}\right)\begin{bmatrix} r^{11} \\ r^{21} \\ \vdots \\ r^{33} \end{bmatrix} = \mathbf{0} \qquad (6.7)$$

where r^{ij} is the i, j element of R and q^{ij} is the i, j element of ${\mathsf{R}'_2}^T\mathsf{R}_2$. Stacking many such equations and minimizing gives the optimization problem minimize $||\mathsf{A}\mathsf{x}||_2^2$ subject to $||\boldsymbol{x}||_2 = 1$ and $\det \mathsf{R} = 1$, which we approximate by dropping the second equality constraint, finding the least singular vector of A, and using Equation 6.5 with input normalized to have positive determinant and one column of unit norm.

6.5 Conclusion

In this chapter, we have provided details of the Berkeley UAV testbed, as well as the extrinsic calibration procedure used for both the Boeing and Berkeley vehicles. Experimental mapping results from both vehicles will given in Chapter 7.

Chapter 7

Results

The content of this chapter is based on published work coauthored with Christopher Geyer, Marci Meingast, and Shankar Sastry [1], and on unpublished work coauthored with David Shim, Christopher Geyer, and Shankar Sastry [2]. This material is used with the permission of the coauthors.

7.1 Introduction

In this chapter, we present results of running the mapping system on synthetic sequences, as well as on real sequences from both the Boeing and the Berkeley platforms. Since the accuracy of the online motion stamping algorithms such as those discussed in Chapters 4 and 5 have thus far proven to be insufficient for our needs, for the purposes of the results from real data in this report we separate the experiment into three phases to showcase the performance of the other components: In the first phase, the vision high-level planner directed the helicopter to perform the box search while it collected image data. In the second phase, we performed the offline motion stamping algorithm described in Section 2.3, followed by RMFPP running on similar hardware to the helicopter vision computer in better than real time. In the third phase, we executed the closer inspection and landing

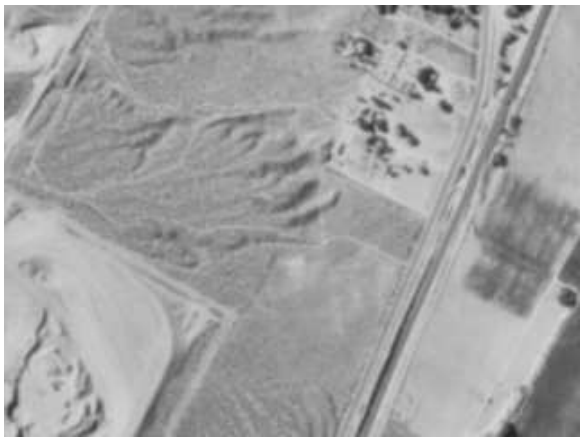
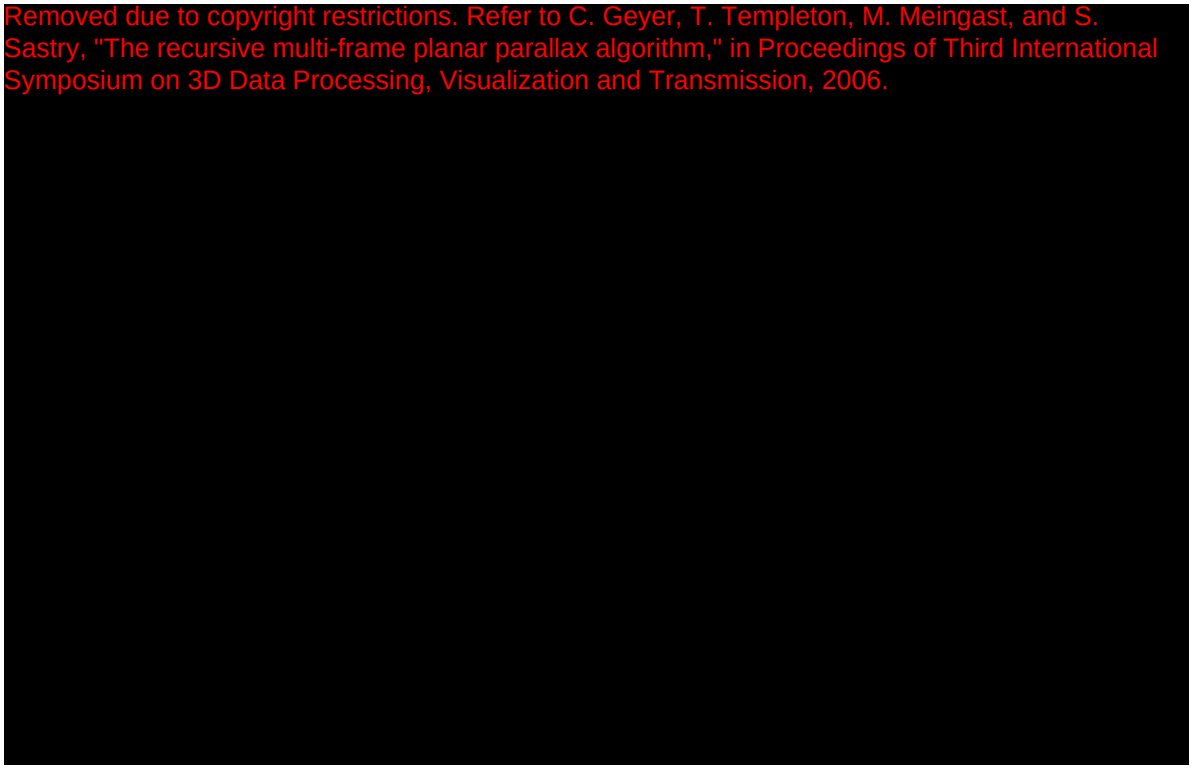


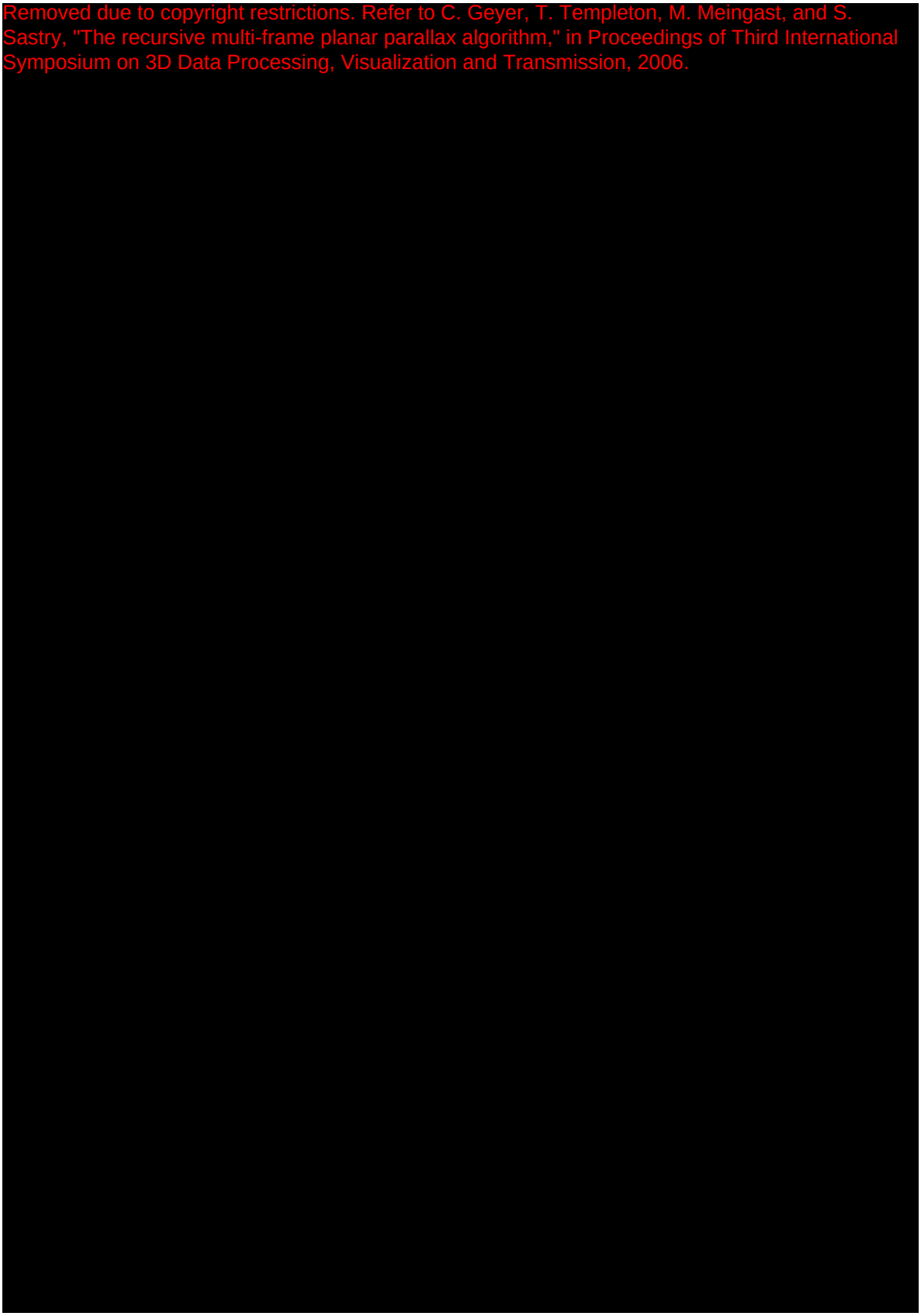
Figure 7.1: Sample synthetic image rendered from satellite imagery and Digital Terrain Elevation Data (DTED) elevation data.

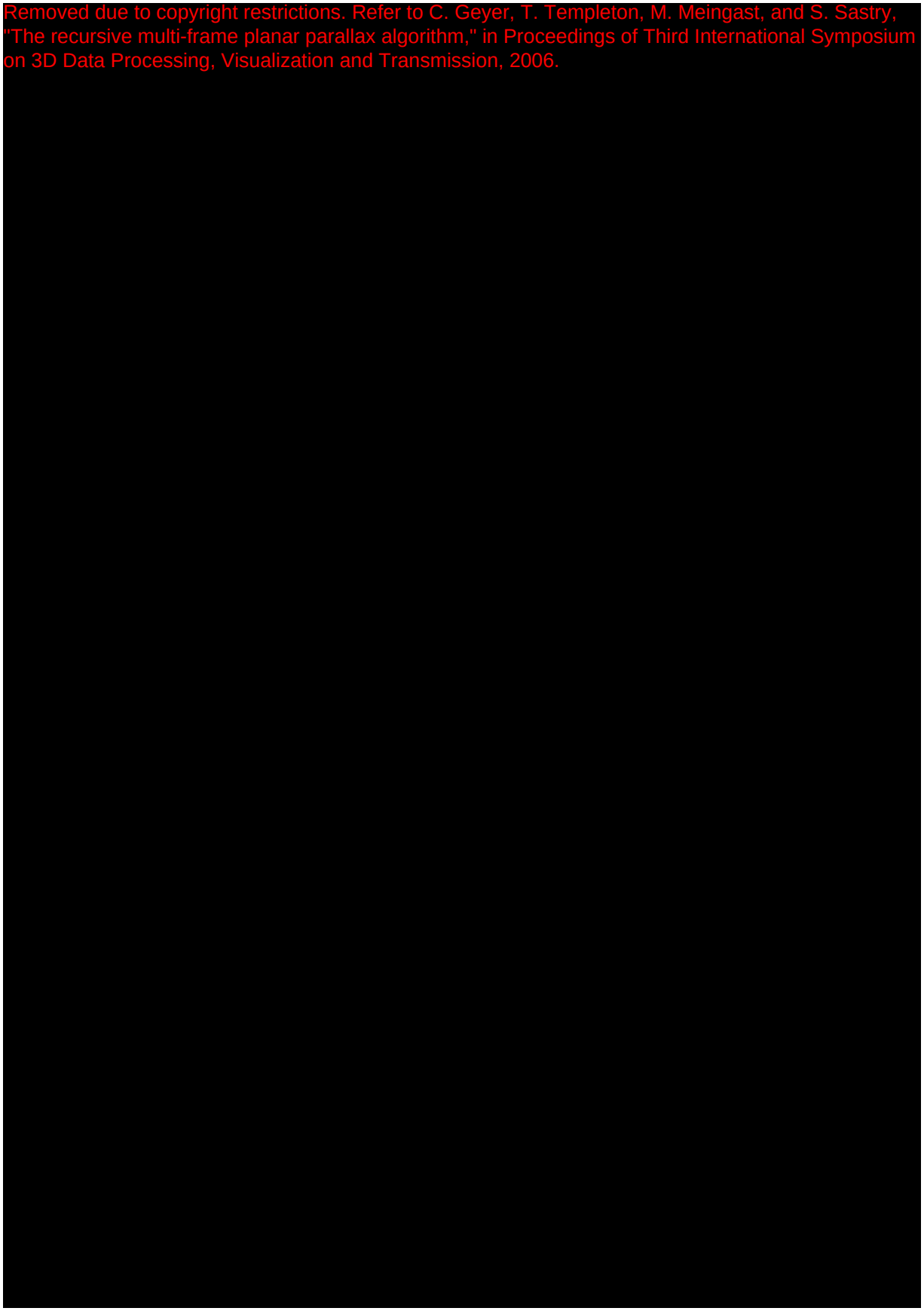
maneuver using a premade elevation and appearance map.

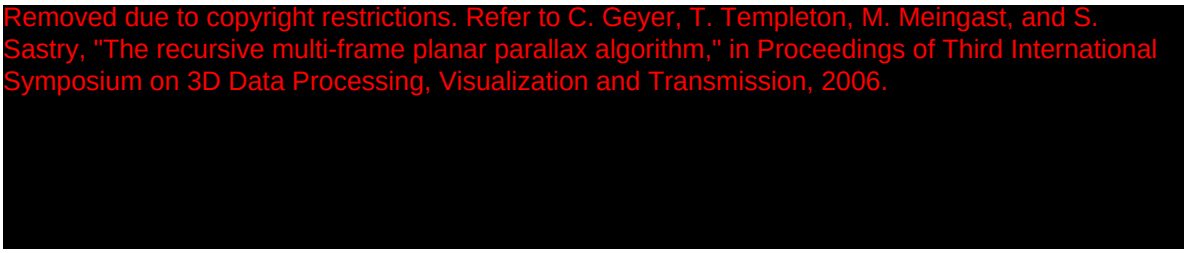
7.2 Results From Synthetic Data

©*2006 IEEE. Reprinted, with permission, from [1].*




Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.


Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.



7.3 Results From Experimental Data

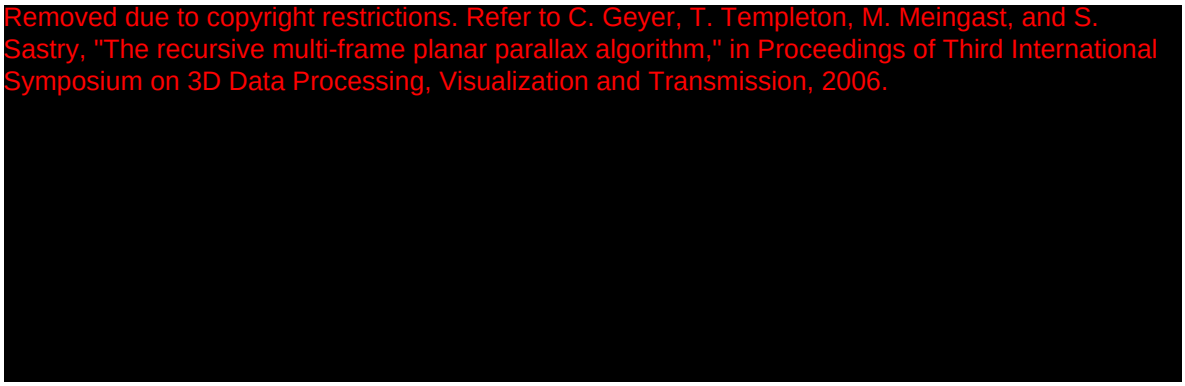
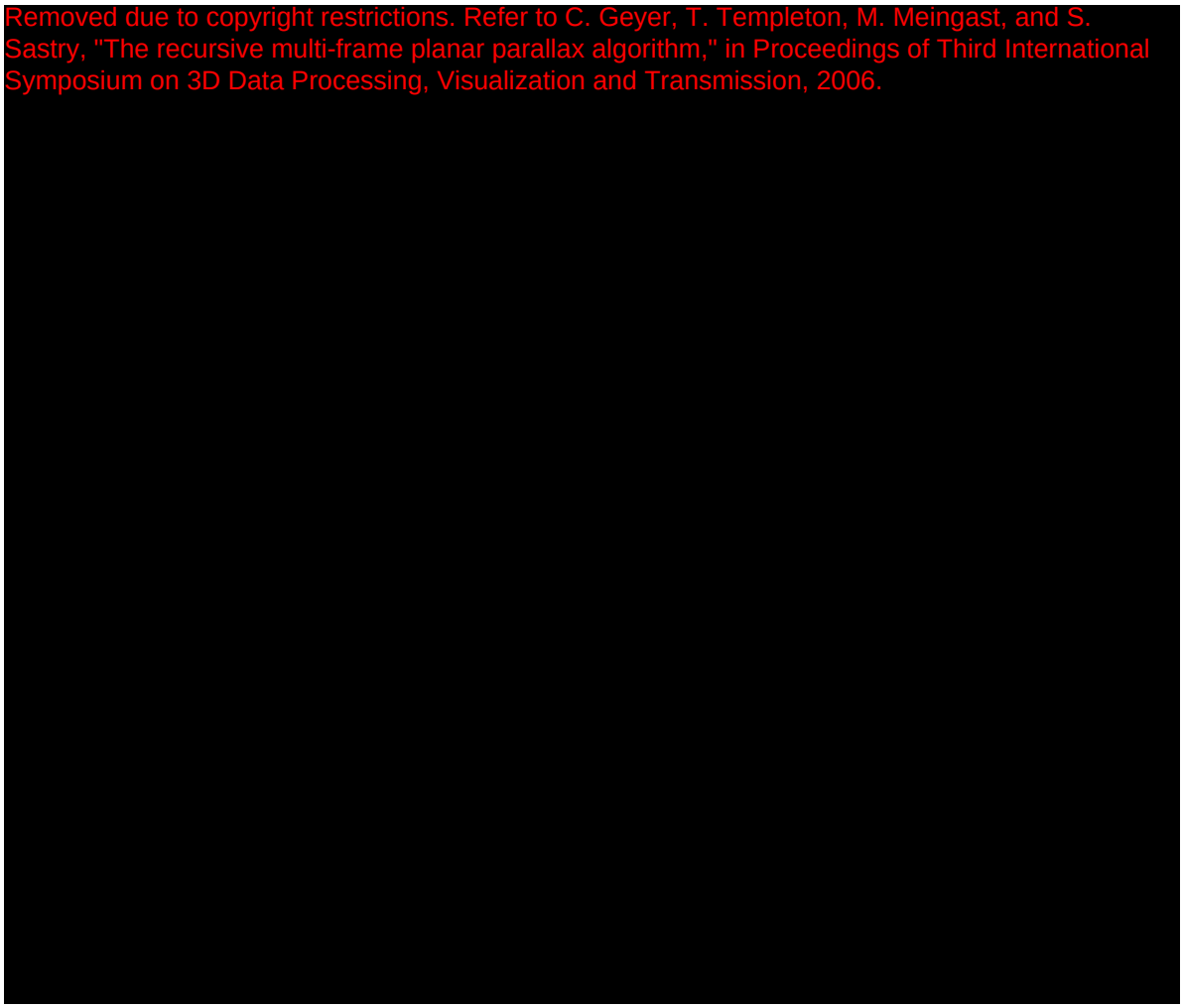


Figure 7.5 shows a similar terrain reconstruction result using experimental aerial imagery from the Berkeley vehicle at Richmond Field Station. Although much of the left side of the view is flat and uneventful (as it should be), the trees in the middle of the view and the road on the right are clearly visible.

7.4 Conclusion

This chapter has given mapping results using synthetic imagery, as well as using experimental imagery captured from both the Boeing and Berkeley platforms. Although we were unable to accurately motion-stamp the images in real time, and hence were unable to build and use the elevation and appearance map in real time, we have shown the abilities of crucial pieces of the mapping and landing system.

[1] ©*2006 IEEE. Reprinted, with permission, from [1].*


Removed due to copyright restrictions. Refer to C. Geyer, T. Templeton, M. 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.

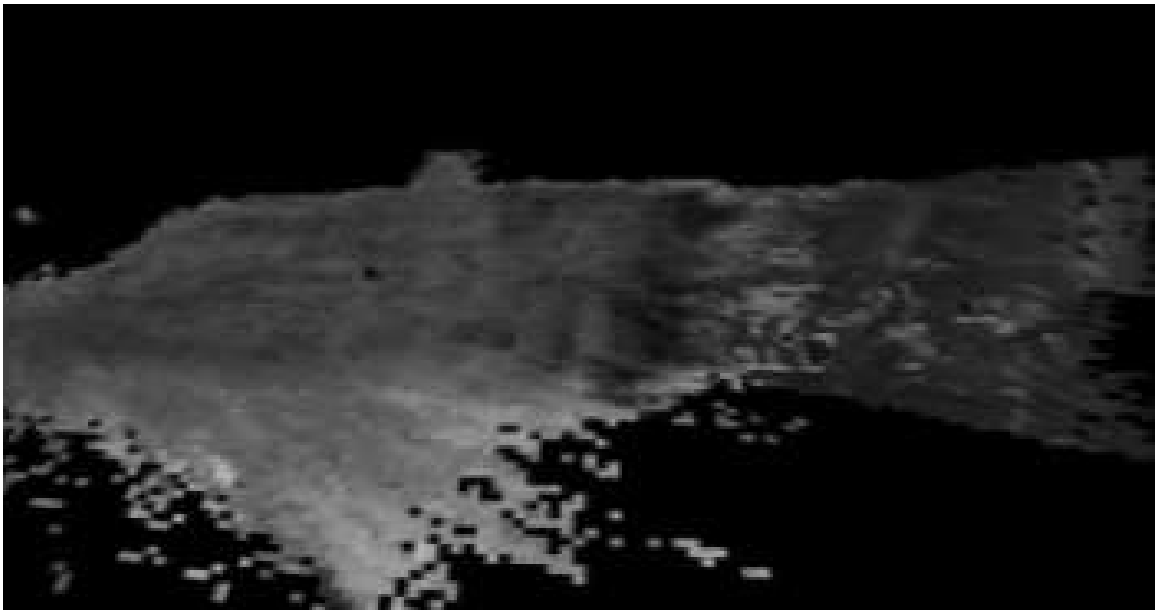


Figure 7.5: Berkeley real images experiment: The reconstructed appearance draped over the reconstructed elevation. *Reproduced, with permission, from [2].*

Chapter 8

Conclusion

In this report, we have presented work toward a vision-based landing system for unmanned rotorcraft in unknown terrain using our new high-accuracy Recursive Multi-Frame Planar Parallax algorithm [1] for terrain mapping. Although we have not yet succeeded in finding a motion-stamping algorithm that performs to the needs of our next-generation approach in the adverse circumstances of high-altitude flight, we are confident that such an algorithm exists, and in the meantime we have given an in-depth description of the vision system, an overview of our experimental platforms, and both synthetic and experimental accurate real-time terrain mapping results.

Bibliography

[1] C. Geyer, T. Templeton, M. 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.

[2] T. Templeton, D. H. Shim, C. Geyer, and S. Sastry, "Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft." Submitted to 2007 IEEE International Conference on Robotics and Automation for consideration, 2006.

[3] O. Shakernia, Y. Ma, T. Koo, and S. Sastry, "Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control," 1999.

[4] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually-guided landing of an unmanned aerial vehicle," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 371–381, Jun 2003.

[5] S. Saripalli and G. S. Sukhatme, "Landing on a moving target using an autonomous helicopter," in *Proceedings of the International Conference on Field and Service Robotics*, Jul 2003.

[6] S. Saripalli, G. S. Sukhatme, and J. F. Montgomery, "An experimental study of the autonomous helicopter landing problem," in *Proceedings, International Symposium on Experimental Robotics*, (Sant'Angelo d'Ischia, Italy), 2002.

[7] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation*, pp. 2799–2804, 2002.

[8] P. J. Garcia-Pardo, G. S. Sukhatme, and J. F. Montgomery, "Towards vision-based safe landing for an autonomous helicopter," *Robotics and Autonomous Systems*, vol. 38, no. 1, pp. 19–29, 2001.

[9] A. Johnson, J. Montgomery, and L. Matthies, "Vision guided landing of an autonomous helicopter in hazardous terrain," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2005.

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

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

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, 2004.

[13] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm," Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[14] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of Computer Vision and Pattern Recognition*, 2001.

[15] E. Sokolowsky, H. Mitchell, and J. de La Beaujardiere, "NASA's scientific visualization studio image server," in *Proc. IEEE Visualization 2005*, p. 103, 2005.

[16] S. K. Jenson and J. O. Domingue, "Extracting topographic structure from digital elevation data for geographic information systems analysis," *Photogrammetric Engineering and Remote Sensing*, vol. 54, no. 11, pp. 1593 – 1600, 1988.

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

[18] J. McMichael and M. Francis, "Micro air vehicles–toward a new dimension in flight," tech. rep., DARPA, 1997.

[19] O. Faugeras, *Three-Dimensional Computer Vision*. The MIT Press, 1993.

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

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

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

[23] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Proc. International Workshop on Vision Algorithms*, September 1999.

[24] M. Okutomi and T. Kanade, "A multiple-baseline stereo method," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 353–363, April 1993.

[25] G. P. Stein and A. Shashua, "Direct estimation of motion and extended scene structure from a moving stereo rig," Tech. Rep. AIM-1621, Massachusetts Institute of Technology, 1997.

[26] Y. 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.

[28] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision*, vol. 3, September 1989.

[29] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "3-d motion and structure from 2-d motion causally integrated over time: Implementation," in *Proceedings of European Conference on Computer Vision*, June 2000.

[30] G. Welch and G. Bishop, "An introduction to the Kalman filter," Tech. Rep. 95-041, UNC - Chapel Hill, Chapel Hill, NC, July 2006. Available from `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`.

[31] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[32] M. Moakher, "Means and averaging in the group of rotations," *SIAM Journal on Matrix Analysis and Applications*, vol. 24, no. 1, 2002.

[33] P. Zielinski and K. Zietak, "The polar decomposition – properties, applications and algorithms," *Matematyka Stosowana*, vol. 38, 1995.

[34] R. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge Univ. Press, 2nd ed., 2000.

[35] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, 2004. Available from `http://control.ee.ethz.ch/~joloef/yalmip.php`.

[36] K. C. Toh, R. H. Tutuncu, and M. J. Todd, "SDPT3 – a MATLAB software for semidefinite-quadratic-linear programming." Available from `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`, December 2002.

[37] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *IEEE Conference on Decision and Control*, December 2003.

[38] D. H. Shim, H. J. Kim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *American Control Conference*, May 2002.

[39] D. H. Shim, H. Chung, and S. Sastry, "Conflict-free navigation in unknown urban environments," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 27–33, September 2006.

[40] G. J. Sutton and R. R. Bitmead, "Computational implementation of NMPC to nonlinear submarine," *Nonlinear Model Predictive Control*, pp. 461–471, 2000.

[41] D. H. Shim and S. Sastry, "A situation-aware flight control system design using real-time model predictive control for unmanned autonomous helicopters," in *AIAA Guidance, Navigation, and Control Conference*, August 2006.

[42] D. H. Shim, *Hierarchical Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles*. PhD thesis, University of California, Berkeley, 2000.