9-2018

Satellite Articulation Sensing using Computer Vision

David H. Curtis

Satellite Articulation Sensing using Computer Vision

DISSERTATION

David H. Curtis, Maj, USAF

AFIT-ENY-DS-18-S-060

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-DS-18-S-060

SATELLITE ARTICULATION SENSING USING COMPUTER VISION

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctroal of Philosophy

David H. Curtis, B.S.M.E., M.S.A.E.

Maj, USAF

September 2018

AFIT-ENY-DS-18-S-060

SATELLITE ARTICULATION SENSING USING COMPUTER VISION

DISSERTATION

David H. Curtis, B.S.M.E., M.S.A.E.
Maj, USAF

Committee Membership:

Dr. Richard G. Cobb
Chairman

Maj Scott J. Pierce, Ph.D.
Member

Capt Joshuah A. Hess, Ph.D.
Member

Abstract

Autonomous on-orbit satellite servicing benefits from an inspector satellite that can gain as much information as possible about the primary satellite. This includes performance of articulated objects such as solar arrays, antennas, and sensors. This research develops methods for building an articulated model from monocular imagery using tracked feature points and the known relative inspector route of the inspector satellite with respect to the primary satellite. The articulated model consists of a kinematic chain identifying how components are linked, a point cloud representation of the component shapes, axes of rotation (revolute joints assumed), joint locations, and observed articulation angles. The quality of the articulated model is assessed for inspection routes with various illumination conditions concluding that the illumination angle should be maintained within 60° of the Sun angle and that a quality model can be built without viewing the satellite from all angles. Two methods are also developed for tracking the articulation of a satellite in real-time given an articulated model using either feature points or silhouettes. Performance is evaluated for multiple inspection routes and the effect of inspection route uncertainty is assessed. Additionally, a physical small scale satellite model is built and used to collect stop-motion images simulating articulated motion over a simulated inspection route in a simulated space illumination environment. The images are used as measurements for the silhouette articulation tracking method and successful tracking is demonstrated qualitatively. Next, a human pose tracking algorithm is modified for tracking the satellite articulation demonstrating the applicability of human tracking to satellite articulation tracking methods when an articulated model is available. Lastly, an overview of methods is presented to assist users in determining applicability to specific problems.

Acknowledgements

I would like to thank my wife for all of her support, particularly over the last 3 years. Of the two of us, she certainly is more deserving of the title 'doctor'. I appreciate her continually putting her ambitions on hold so I can pursue mine. I would also like to acknowledge my son, who always makes me smile.

I also would like to thank my committee, particularly Dr. Cobb, for their guidance and advice in completing this research.

Finally, I would like to thank my family; my sisters and brother who have provided encouragement throughout my life, and especially my parents, whose guidance and support have enabled any success I have ever had.

David H. Curtis

Table of Contents

Page

Abstract iv

Acknowledgements v

List of Figures x

I. Introduction 1

1.1 Motivation 1
1.1.1 Building an Articulated Model 2
1.1.2 Articulation Tracking 3
1.2 Research Objectives 4
1.2.1 Objective 1 4
1.2.2 Objective 2 5
1.2.3 Objective 3 5
1.3 Research Overview 5
1.3.1 Building an Articulated Model (Chapter III) 6
1.3.2 Articulation Tracking with Feature Points (Chapter IV) 7
1.3.3 Articulation Tracking with Silhouettes (Chapter V) 7
1.3.4 Satellite Model Image Capture and Processing (Chapter VI) 8
1.4 Expected Contributions 8
1.5 Document Preview 9

II. Background and Literature Review 11

2.1 Chapter Overview 11
2.2 Background 11
2.2.1 Computer Vision 11
2.2.2 Camera Basics 11
2.2.3 Camera System Types 16
2.2.4 Feature Point Detection and Tracking 17
2.2.5 Trajectory Matrix 17
2.2.6 Structure from Motion through Factorization 18
2.2.7 Silhouettes and Edges 21
2.2.8 Simultaneous Localization and Mapping 22
2.2.9 Kalman Filter 23
2.2.10 Extended Kalman Filter 25
2.2.11 Unscented Kalman Filter 26
2.3 Literature Review 28

Page

2.3.1 Computer Vision for Proximity Operations . . . 29
2.3.2 Motion Segmentation . . . 31
2.3.3 Articulated Structure from Motion . . . 38
2.3.4 Real-time Simultaneous Localization and Mapping (SLAM) . . . 46
2.3.5 Human Articulated Motion Tracking . . . 52
2.4 Summary . . . 55

III. Building an Articulated Model . . . 56

3.1 Chapter Overview . . . 56
3.2 Introduction . . . 56
3.3 Method . . . 56
3.3.1 Simulating Data . . . 57
3.3.2 Ray Space Optimization . . . 60
3.3.3 Segmentation . . . 63
3.3.4 Rigid Body Optimization . . . 64
3.3.5 Combine Segments . . . 68
3.3.6 Build Kinematic Chain . . . 69
3.3.7 Articulation Parameter Optimization . . . 70
3.3.8 Evaluation Metrics . . . 75
3.4 Results . . . 77
3.4.1 Initial Assessment . . . 77
3.4.2 Illumination and Percentage of NMC . . . 79
3.4.3 Limitations . . . 85
3.5 Accommodating Main Body Maneuver and Uncertainty . . . 87
3.5.1 Modified Model Development Method . . . 89
3.5.2 Example Results with Moving Main Body and Noise . . . 93
3.5.3 Inspection Route Noise Assessment . . . 93
3.6 Conclusion . . . 95

IV. Articulation Tracking with Feature Points . . . 97

4.1 Chapter Overview . . . 97
4.2 Introduction . . . 97
4.3 Method . . . 97
4.3.1 Simulating Data . . . 98
4.3.2 Filter Framework . . . 98
4.3.3 Propagate Articulation Parameters . . . 100
4.3.4 Point Position EKF Update . . . 102
4.3.5 Update Articulation Parameters . . . 103
4.3.6 Initialization . . . 104
4.3.7 Adding Points to the Model . . . 106

Page

4.3.8 Outlier Rejection . . . 107
4.4 Results . . . 108
4.4.1 Full NMC Demonstration . . . 109
4.4.2 Inspection Route Noise . . . 113
4.4.3 Partial NMC with Articulation Start/Stop . . . 114
4.5 Conclusion . . . 117

V. Articulation Tracking with Silhouettes . . . 119

5.1 Chapter Overview . . . 119
5.2 Introduction . . . 119
5.3 Silhouette Based Tracking Method . . . 121
5.3.1 Simulating Imagery . . . 121
5.3.2 Filter Framework . . . 124
5.3.3 Dynamics . . . 125
5.3.4 Measurement Model . . . 126
5.3.5 Implementation . . . 128
5.3.6 Computational Requirements . . . 130
5.4 Results . . . 130
5.4.1 Full NMC Results . . . 130
5.4.2 Sinusoidal Articulation Movement . . . 131
5.4.3 Start/Stop Scenario . . . 132
5.4.4 Model Sensitivity Assessment . . . 132
5.5 Conclusions . . . 136

VI. Real Imagery . . . 138

6.1 Chapter Overview . . . 138
6.2 Introduction . . . 138
6.3 Setup . . . 139
6.3.1 Satellite Model Specifications . . . 139
6.3.2 Image Collection . . . 139
6.3.3 Camera Calibration . . . 141
6.4 Method . . . 142
6.4.1 Image Processing . . . 142
6.4.2 UKF Settings . . . 143
6.4.3 Computational Time . . . 144
6.5 Results . . . 144
6.6 Modified HumanEva Baseline Algorithm . . . 147
6.6.1 Results . . . 148
6.7 Conclusion . . . 150

Page

VII. Conclusions and Recommendations 152

7.1 Research Objectives 152
7.1.1 Objective 1 152
7.1.2 Objective 2 153
7.1.3 Objective 3 153
7.2 Satellite Articulation Sensing Considerations 154
7.2.1 Building an Articulated Model 154
7.2.2 Articulation Tracking 156
7.2.3 Monocular Vision Ambiguities 157
7.2.4 Articulation Rates 158
7.3 Summary of Methods Presented 159
7.4 Contributions 161
7.5 Recommendations for Future Work 161
Appendices 164

Appendix A. Gradients for Chapter III 165

A.1 Rigid Body Optimization Gradients 165
A.2 Articulation Parameter Gradients 172

Appendix B. Jacobians for Chapter IV 177

B.1 Articulation Parameter Propagation Jacobian 177
B.2 Measurement Model Jacobian 179

Appendix C. AIAA SciTech 2017 Paper 184

Bibliography 200

List of Figures

Figure Page

1. Applicability to inspection mission 6
2. Representation of the pinhole camera model.[73] 13
3. Representation of the orthographic camera model.[73] 15
4. Illustration of the mirror ambiguity 22
5. Overview of SLAM problem.[86] 23
6. Motion segmentation illustration 32
7. Diagram of simulated satellite model 59
8. Demonstration of point visibility 60
9. Example trajectory matrix mask for a complete NMC 63
10. Reference frames 66
11. Example results for building an articulated model 79
12. Illumination offset parameter diagram 80
13. Illumination offset angle effect on model quality for fly-by inspection route 82
14. Percentage of truth model points that are visible during the inspection route 83
15. Illumination offset angle effect on model quality for full NMC inspection route 85
16. Illumination offset angle and percentage of NMC complete effect on articulated model quality 86
17. Results for articulated model building method with noise 89
18. Results for modified articulated model building 94
19. Building an articulated model inspection route noise assessment 95

Figure Page

20. Conditional probabilities for an example point from initial acquisition to assignment 108

21. Articulation tracking full NMC results 110

22. Articulation tracking full NMC final estimated point positions 111

23. Articulation tracking with feature points inspection route noise assessment 114

24. Articulation start/stop scenario diagram 115

25. Articulation tracking with feature points results for start/stop scenario 116

26. Feature point and silhouette resolution effects 122

27. Silhouette method model satellite diagram 123

28. Example silhouette image 124

29. Diagram and sample results for silhouette tracking with linear articulation 131

30. Results for silhouette tracking with linear articulation 132

31. Results for silhouette tracking with sinusoidal articulation 133

32. Silhouette tracking start/stop results 133

33. Silhouette tracking model joint sensitivity with inspection route near panel ambiguity 134

34. Silhouette tracking model joint sensitivity 135

35. Silhouette tracking model size sensitivity 136

36. Satellite model dimensions 140

37. Imagery collection setup 140

38. Real imagery inspection route diagram 141

39. Steps to create a silhouette image 143

Figure Page

40. Example imagery results for UKF tracking method 145

41. Articulation angle results for UKF tracking method 146

42. Rotation and translation results for UKF tracking method 147

43. Example imagery results for APF tracking method 149

44. Articulation angle results for APF tracking method 150

45. Rotation and translation results for APF tracking method 150

46. Illustration of ambiguity in creating a shape from points not seen together 155

47. Monocular camera ambiguities 158

SATELLITE ARTICULATION SENSING USING COMPUTER VISION

I. Introduction

1.1 Motivation

Due to the nature of the space environment, direct human interaction with on-orbit satellites is incredibly dangerous and costly. Therefore, with the exception of a few manned missions, interaction with satellites is traditionally limited to windows of radio communications and observations from telescopes. These methods are inadequate for inspection and monitoring of a satellite to determine performance. To perform these monitoring tasks, an inspector satellite in close proximity to the primary satellite could use sensors, such as a camera, to characterize the primary satellite and verify the performance of articulated objects such as sensors, solar arrays, robotic arms, and communications antennas.

Characterizing the articulation capabilities of a satellite is a portion of the larger satellite inspection and repair research area. The area of satellite inspection and repair has been researched heavily. Large complex satellites cost billions of dollars to build and launch. Having the capability to inspect, repair, or refuel satellites while on-orbit facilitates extension of their useful life cycle, and therefore tremendous cost savings are possible. For instance, the Hubble Space Telescope (HST) has been serviced on-orbit five times.[64]

As the capability of satellite repair improves, satellites could be built with less redundancy allowing additional capabilities or smaller size and cost. Additionally, repairing a satellite removes or delays the need to launch a replacement, reducing

space debris.

The military has had multiple programs focused on researching satellite inspection and repair. The XSS-11 [1] and Automated Navigation and Guidance Experiment for Local Space (ANGELS) [5] experiments conducted by the Air Force Research Laboratory Space Vehicles Directorate were launched in 2005 and 2014 respectively. More recently, the EAGLE satellite experiment has launched with an attached small fly-away satellite to improve situational awareness for space vehicles.[2] These programs investigated/demonstrated many technologies for inspection of objects in space. The Orbital Express mission sponsored by DARPA in 2007 [22] demonstrated the feasibility of on-orbit servicing by performing autonomous proximity operations to include docking and refueling of another satellite. In addition to these space experiments, numerous research efforts have been conducted on the use of computer vision for satellite proximity operations.[32] Some of these efforts will be discussed in Chapter II. In the field of computer vision, there has been significant research on detecting and characterizing articulation of an object from imagery. Some of these methods will also be discussed in Chapter II. To the knowledge of the author, there is no research merging these two fields to sense and characterize articulated motion in space.

The concept of satellite articulation sensing with imagery can be broken into two sub-categories: 1) using imagery to build an articulated model of the satellite, and 2) using imagery and some model of the satellite to track how the satellite is moving.

1.1.1 Building an Articulated Model.

Imagery contains a lot of information, however it is also difficult to transmit to the ground due to the large size of video files. An articulated model could provide the necessary information to users on the ground with a fraction of the data transmission requirement of the raw imagery. This motivates the creation of algorithms that can

characterize the performance (articulated motion analysis, i.e. range and rate of rotations) of the target satellite autonomously. Developing autonomous methods is also important for autonomous repair missions [44] or deep space missions that have little or no ability to communicate with human operators on Earth.

Many satellites, specifically in geosynchronous orbit (GEO), are required to operate at all times. To accommodate these missions, the inspector must be able to perform inspection without interfering with the operations of the primary satellite. Imagery offers a non-invasive method of gathering information about the target satellite. Monocular cameras are also relatively inexpensive with small SWAP (size, weight and power) requirements. Additionally, cameras already exist on a number of on-orbit satellites which could be re-tasked with an inspector mission.

For these reasons, development of a fully autonomous method of sensing and characterizing the articulated motion of a satellite using computer vision is desired. Specifically, this research will investigate the use of a monocular camera on-board an inspector satellite in a close proximity orbit about a primary satellite to determine if the primary satellite has articulating parts, and if so the range of motion of those parts, with the stated goal of creating an articulated model.

1.1.2 Articulation Tracking.

Many algorithms have been developed using computer vision to determine the relative pose between an inspector satellite and a primary satellite in space. Some of these algorithms rely upon markers on the primary satellite that assist the computer vision algorithm in determining pose.[91, 43] Others rely on prior knowledge of the primary satellite's configuration.[65, 39] Some methods rely on stereo vision systems, [92, 29] some on monocular vision systems, [71, 48, 39] and some use laser illumination of reflective markers.[44] They all use a computer vision method that identifies

features in images and matches those features from frame to frame (or in corresponding frames in the case of stereo systems). The relative position of the features is then estimated using some type of estimation filter such as an extended Kalman filter (EKF) or a particle filter. The attitude of the primary satellite is then estimated, with the assumption it is a rigid body, by using feature points to define a primary reference frame, [71, 102] or by using a known model of the primary satellite.[65]

In many of these cases, the primary satellite being investigated is assumed to be a rigid body. If feature points do not move in a manner consistent with rigid body motion, they may be rejected as outliers, however in the case of articulation this method may exclude significant portions of the satellite from the model.

For rendezvous and docking operations in space, it is necessary for an inspecting satellite to understand how the satellite it is rendezvousing with is behaving. In the case of a non-cooperative articulating satellite, this includes tracking and estimating the primary satellite's articulated motion.

1.2 Research Objectives

Three research objectives are listed below. Chapter III describes work toward Objective 1, Chapters IV and V describe work toward Objective 2, and Chapter VI describes work toward Objective 3.

1.2.1 Objective 1.

Develop a method of autonomously building an articulated model of a satellite through inspection with a monocular camera producing resolved imagery. This work will consist of development of an algorithm and testing on simulated feature points from a nominal satellite with articulating panels and an articulating arm. The quality of the model created from inspection routes with various illumination conditions will

be assessed.

1.2.2 Objective 2.

Develop an estimation framework for tracking the articulated motion of the primary satellite sequentially as new imagery becomes available. This work will consist of development of an algorithm that uses an articulated model of the satellite and either simulated feature point locations or image silhouettes to track the articulated motion of the satellite.

1.2.3 Objective 3.

Build a satellite model and collect stop motion imagery mimicking articulation in a simulated space lighting environment. Use captured imagery to validate developed tools as permitted by capture method uncertainties.

1.3 Research Overview

This work can be broken into two parts: 1) Building an articulated model (Chapter III), and 2) Tracking articulation in real-time (Chapters IV and V). Both parts are investigated using a simulated model of a satellite with articulating panels and an articulating appendage. A physical model of a satellite is also created and used in testing of the tracking method from Chapter V. All work assumes revolute joints, a known satellite inspection route, and that feature points or silhouettes can be extracted from monocular imagery. Each of the methods presented addresses some aspect of the overall problem of sensing articulation from an inspector satellite. Figure 1 gives a graphical overview of how the different methods fit into the overall problem and how they relate to the stated research objectives of section 1.2. An overview of each of the methods is presented in the following subsections.

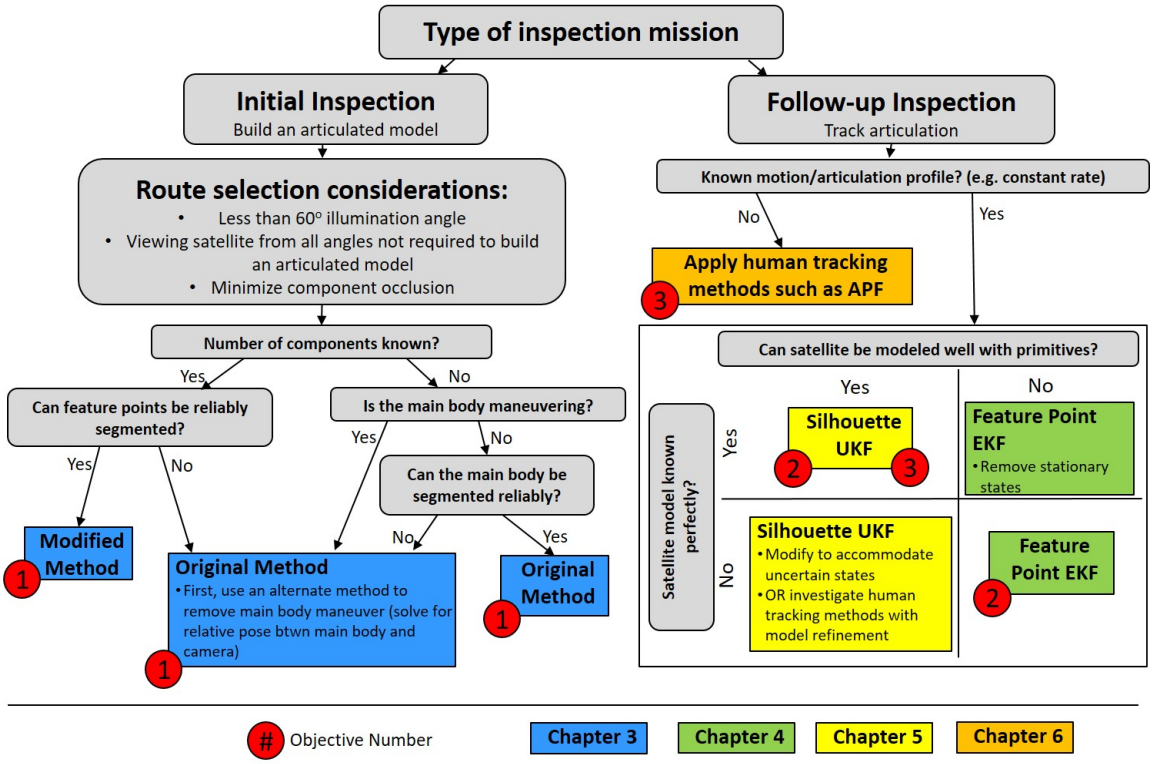

Figure 1. Overview of how each area of research relates to the overall inspection mission and to the research objectives.

1.3.1 Building an Articulated Model (Chapter III).

This work focuses on building an articulated model from a set of monocular images taken from an inspection satellite on a known inspection route. Points are spread over the faces of a model articulated satellite and projected to the image plane of an inspector satellite on a particular inspection route resulting in simulated feature point locations that are consolidated into a trajectory matrix. The points are then segmented into groups that represent rigid body motion and an optimization routine is performed on each group to identify the shape and pose of the rigid body that best describes the feature point locations. Rigid bodies with similar motion are then merged if required. Next a kinematic chain is built that identifies which components are linked to each other. Finally, another optimization routine is run that enforces the articulation constraints on linked components. Results are presented for the

method with multiple active joints on the articulating arm and investigating the effects of various illumination conditions. A modified method allowing for primary satellite maneuver, trajectory matrix uncertainty, and inspection route uncertainty is developed and performance demonstrated.

1.3.2 Articulation Tracking with Feature Points (Chapter IV).

This work uses an articulated model built with the methods of Chapter III to initialize a set of extended Kalman filters (EKFs) to track the pose and articulation angles of the satellite from simulated feature points. The P feature point locations from a frame are used as the measurement in P point position EKFs to estimate the 3D position of the point in its assigned component's body frame. Component rotation and translation, defined by the articulation parameters, are taken as prior knowledge. Next, the 3D body frame positions of each point are used as prior knowledge in an articulation parameter EKF which again uses the 2D feature point locations as measurements to update the articulation parameters. Newly observed feature points are added to the model using a multiple model approach. Results are demonstrated for a full natural motion circumnavigation (NMC)[1] with linear articulation and for a partial NMC with articulation start/stop included in the simulation.

1.3.3 Articulation Tracking with Silhouettes (Chapter V).

This work uses a truth articulation model and an unscented Kalman filter (UKF) to track the pose and articulation angles of the satellite using simulated silhouette images. Silhouette images are binary images with a value of 1 for pixel locations that are occupied by the foreground (the satellite) and a value of zero at pixel locations

[1]An NMC is a 2×1 elliptical route of a satellite (in this case the inspector satellite) with respect to some relative frame (in this case the primary satellite being inspected).[78] It is used heavily in satellite proximity operations, to include inspection/servicing as a minimum fuel method of formation flying, and therefore is used in this research as an example inspection route.

that are occupied by the background. Simulated silhouette images are created by projecting the corner points of a the satellite model components into the image plane and setting the pixels on the interior of the convex hull defined by the corner points to 1. Simulated silhouette images are used as measurements in an UKF framework to track the main body pose and articulation angles. Results are shown for linear articulation and sinusoidal articulation over a complete NMC and for a partial NMC with articulation start/stop motion included in the simulation.

1.3.4 Satellite Model Image Capture and Processing (Chapter VI).

A physical satellite model similar in shape to the one used for numerical simulation was constructed and used to collect imagery in a simulated space illumination environment. An inspection route was simulated by rotating the satellite, moving the camera forward and backward on a track, and moving a single light source to simulate the Sun location. Images were taken at discrete locations on the inspection route with the articulation angles of the five joints on the model moved slightly between frames giving a set of images mimicking those that would be collected of an articulating satellite from an inspector satellite on a nearby inspection route. Results are presented when using these images as the input images for the silhouette method outlined in Chapter V. Additionally, an algorithm created for human tracking (the Annealed Particle Filter from [81, 28]) was modified to track satellite pose/articulation and results are presented.

1.4 Expected Contributions

This work primarily lies in the field of computer vision, specifically as it relates to characterizing and tracking articulated objects. Various methods exist for characterizing articulation from imagery, however no previous work exists in applying

computer vision to characterize articulation in space. Aspects of existing methods have been used to develop a method of building an articulated model and methods of tracking articulation in real-time. Specifically, the contributions from this research are:

1. Introduction of computer vision for the purpose of building an articulated model and tracking satellite articulation in space using monocular imagery.

2. Development and demonstration of a method for building an articulated model of a satellite from monocular imagery taken from a known inspection route.

3. Development and demonstration of methods for sequentially estimating the pose and articulation angles of a satellite using both feature points and silhouette images.

4. Demonstration of tracking articulation angles using real images of a satellite model taken in a simulated space lighting environment.

5. Demonstration that algorithms developed for human articulation tracking can be modified to track the articulated motion of a satellite.

It is anticipated that there will be multiple papers produced from this work, which will be mentioned in the associated chapters.

1.5 Document Preview

Chapter II contains a review of some important topics in the field of computer vision as well as a review of existing work in the field. Chapter III outlines the satellite model development method and results. Chapter IV outlines the articulation tracking with feature points method and results. Chapter V outlines the articulation tracking with silhouettes method and results. Chapter VI outlines the satellite model image

capture process and tracking results from application of the method of Chapter V and a modified human tracking algorithm. Chapter VII outlines the methods presented, the contributions of the current work, and recommendations for future work.

II. Background and Literature Review

2.1 Chapter Overview

This chapter discusses previous works that have informed this research. First, background information is provided on relevant subjects within the computer vision and stochastic estimation fields. Next, recently published work in the following areas is discussed: computer vision for proximity operations, motion segmentation, articulated structure from motion, real-time simultaneous localization and mapping (SLAM), and human articulated motion tracking.

2.2 Background

To provide context for the literature review in the next section, and the research as a whole, a few concepts must be introduced.

2.2.1 Computer Vision.

Computer vision is the concept of programming a computer to do what humans and animals do every day: convert light entering a sensor (the eye) into information about the environment. The ability of a machine to understand its environment is critical to increasing machine autonomy. The field of computer vision has application in a number of areas such as: autonomous driving, facial recognition, machine inspection, medical imaging, surveillance, 3D model building, entertainment, and robotics to name a few.[83]

2.2.2 Camera Basics.

The basic monocular camera captures a representation of a 3D scene on a 2D image plane. The camera model describes how 3D points are translated into the 2D image

plane. There are numerous camera models, but first, the concept of homogeneous coordinates must be introduced.

Homogeneous Coordinates.

Homogeneous coordinates are a method of representing points in n-space using a vector that is $n+1$ in length. The first n elements represent the direction of the vector, while the last element can be used to scale the vector. The transformation between standard Euclidean coordinates and homogeneous coordinates is shown in equation (1) below.[73]

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ w \end{bmatrix}}_{\text{homogeneous}} \leftrightarrow \underbrace{\begin{pmatrix} x_1/\omega \\ x_2/\omega \\ \vdots \\ x_n/\omega \end{pmatrix}}_{\text{Euclidean}} \tag{1}$$

Homogeneous coordinates are scale-invariant, meaning that multiplying by a scalar does not effect the Euclidean equivalent vector. One advantage of homogeneous coordinates is that affine transformations, including rotation and translation, can be applied through matrix multiplication. For instance, applying a rotation R and translation T to a 3D vector in Euclidean space requires separate matrix multiplication and addition, while in homogeneous coordinates the same transformation can be done

with a single matrix multiplication.[73]

$$\underbrace{\left[\begin{array}{c|c} R & T \\ \hline 0\,0\,0 & 1 \end{array}\right]\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ w \end{bmatrix}}_{\text{homogeneous}} \leftrightarrow \underbrace{R\begin{pmatrix} x_1/\omega \\ x_2/\omega \\ x_3/\omega \end{pmatrix} + T}_{\text{Euclidean}} \tag{2}$$

Perspective Camera Models.

Perspective is the concept that parallel lines do not appear parallel to the eye (or camera). For instance, when looking down railroad tracks, the tracks appear to be converging toward each other as they move out into the distance, when in fact they are parallel. Camera models that capture this phenomenon are called perspective camera models. One of the most popular perspective camera models is the pinhole camera. A pinhole camera projects a 3D point to the image plane by drawing a line from

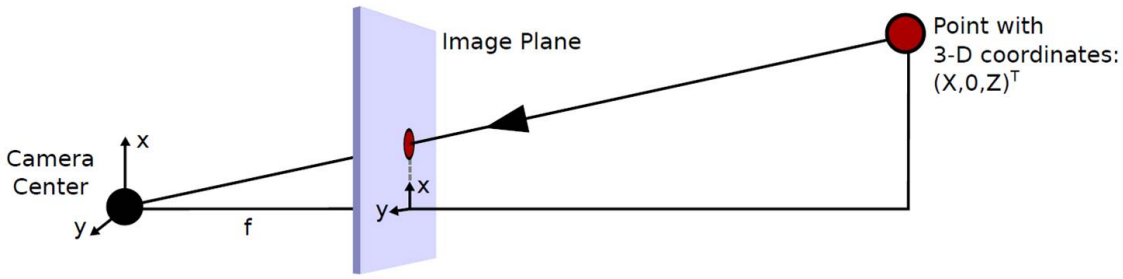

Figure 2. Representation of the pinhole camera model.[73]

the point to the camera center. The place where that line intersects the image plane is the point's 'image coordinate'. From Figure 2 it is evident that using Euclidean coordinates a point $(X, Y, Z)^T$ in 3D space translates to $f\begin{pmatrix} X/Z \\ Y/Z \end{pmatrix}$ in the image plane where f is the focal distance of the camera. This is a non-linear operation, however

using homogeneous coordinates this 'perspective projection' can be written as matrix multiplication.

$$\begin{bmatrix} X \\ Y \\ Z \\ \omega \end{bmatrix} \rightarrow \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ \omega \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} \tag{3}$$

The term on the right side is the 2D image coordinates in homogeneous coordinates with a scale factor of Z.

So far, the camera center has been assumed to be the center of the coordinate frame. To generalize the concept, a rotation R and translation T can be included which converts the 3D point from the world frame to a camera frame (with origin at the center of the image plane and z-axis normal to the image plane). Converting a 3D point in the world frame $(X, Y, Z)^T$ to a 2D point in the image frame $(x, y)^T$ can be expressed as follows where ω_c is used to translate to Euclidean coordinates.

$$\underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{c|c} R & T \\ 0\,0\,0 & 1 \end{array}\right]}_{P_{cam}} \begin{bmatrix} X \\ Y \\ Z \\ \omega_w \end{bmatrix} = \begin{bmatrix} x(\omega_c) \\ y(\omega_c) \\ \omega_c \end{bmatrix} \tag{4}$$

The matrix P_{cam} is the camera's 'projection matrix'. P_{cam} converts 3D points in the world frame to the image frame. The P_{cam} in the equation above represents the pinhole camera model.

Non-perspective Camera Models.

An example of a camera model that does not capture the perspective effect is the orthographic camera model. In the orthographic camera model, all depth information of the 3D point is lost when representing the point in 2D. The image coordinates (x, y) are equal to the X and Y coordinates of the 3D point in the camera frame. Figure 3 shows the projection of points to the image plane. The camera projection matrix is shown below.

$$P_{ortho.} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} R & T \\ 0\,0\,0 & 1 \end{array}\right] \tag{5}$$

This type of model is appropriate when the distance to the object is far greater than the dimension of the object along the optical axis.

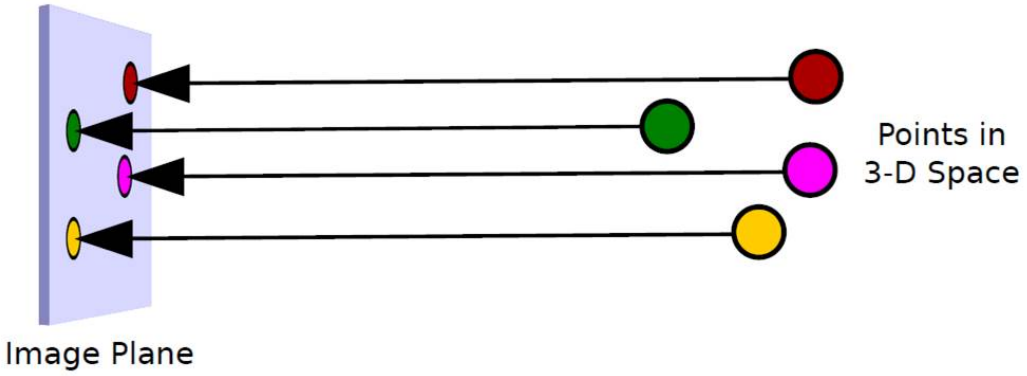

Figure 3. Representation of the orthographic camera model.[73]

A modification to the orthographic camera is the weak perspective camera with a camera projection matrix shown below. This camera model allows the size of the object on the image plane to be scaled. When $\alpha = \beta$ the camera model is referred to

as a 'scaled orthographic camera model'.

$$P_{\text{weak pers.}} = \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} R & T \\ 0\,0\,0 & 1 \end{array}\right] \tag{6}$$

Scaling allows projected image coordinates to more closely represent those seen from an actual camera, which captures perspective. The affine camera model is a further improvement on the orthographic camera model where the orthographic projection is followed by an arbitrary affine transformation.[73]

2.2.3 Camera System Types.

There are many different types of image capture systems that can be used in computer vision. Stereo vision systems consist of two cameras at a known separation that take images at the same time. Having two images at different angles allows depth to be calculated from a single image pair. RGB-D sensors (such as Microsoft Kinect) combine an infrared projector, an infrared camera, and an RGB camera to provide a per pixel depth measurement with the RGB image.[38] A time of flight camera times the return of a pulse of light to estimate depth.[35] A monocular camera system, on the other hand, consists of a single camera. This is the simplest camera system, however each image does not have depth information. Depth information can only be gained by looking at multiple images of the scene taken from different angles. A monocular system is used in this research as it is probably the most likely to be used in space due to SWAP limitations.

2.2.4 Feature Point Detection and Tracking.

Computer vision algorithms can generally be categorized by how they describe the scene from images. A dense representation uses every pixel as a data point (e.g. optic flow), while a sparse representation only uses portions of the image (feature points) to describe the scene. A feature point is a group of pixels in an image that represents a distinct location in 3D space. By identifying corresponding feature points in multiple images more complete information can be gained about the scene, including 3D reconstruction of the objects in the scene. Feature point detection and matching can be broken into four stages: feature detection, feature description, feature matching, and feature tracking.[83]

If feature points have been tracked from frame to frame, they do not need to be matched. Feature point tracking automatically matches feature points from one image to the location in the next image that contains the same feature point. It is best suited for video in which images have been taken sequentially and the amount of motion and appearance change between images is relatively small.[83] From the literature reviewed, the most popular feature point tracker is the Kanade-Lucas-Tomasi (KLT) tracker.[48, 58, 101, 98, 36, 76, 23] Details on the KLT tracker are outlined in [54, 87, 88], and an overview of many aspects of feature points in general is available at [83]. While other search methods exist and are listed outlined in [83], the KLT tracker seems to be the most popular in the literature.

2.2.5 Trajectory Matrix.

The trajectory matrix is a 'bookkeeping' method used in a number of computer vision applications as a way to organize feature point locations. A trajectory matrix is a matrix of the 2D image coordinates of all feature points tracked through all frames. The trajectory matrix, W, is $2F \times P$ in size where F is the number of frames in

the sequence and P is the number of feature points. The make up of the trajectory matrix is shown in equation (7) where $u_{i,j}$ and $v_{i,j}$ are the horizontal and vertical pixel-wise positions of the j-th feature point in the i-th frame in the sequence.

$$W = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,P} \\ v_{1,1} & v_{1,2} & \cdots & v_{1,P} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,P} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ u_{F,1} & u_{F,2} & \cdots & u_{F,P} \\ v_{F,1} & v_{F,2} & \cdots & v_{F,P} \end{bmatrix} \tag{7}$$

Each column of the trajectory matrix represents the motion of a particular feature point, while the combination of two rows represents the location of all feature points in a particular image. This representation is powerful in that it allows linear algebra based methods for segmenting the columns based on different types of motion (motion segmentation) and in determining camera motion and a 3D point cloud of the feature points (structure from motion). Note that the trajectory matrix is alternatively represented as $W = \begin{bmatrix} U \\ V \end{bmatrix}$ where U are all the horizontal image coordinates (u) and V are all the vertical image coordinates (v).

2.2.6 Structure from Motion through Factorization.

A trajectory matrix consisting of points on a rigid body can be factored into a motion matrix (R) and a shape matrix (S) that capture the relative motion of the feature points with respect to the camera and the 3D shape of the rigid body. Tomasi and Kanade's paper [89] outlines the factorization method and is cited by nearly all motion segmentation and structure from motion papers encountered.

The structure from motion method assumes an orthographic camera model is an appropriate representation of the images. It begins by transforming the trajectory matrix W into the registered trajectory matrix $\tilde{W}$ by subtracting the row means. $\tilde{W}$ can then be written as follows where R is the motion matrix which represents the relative rotation between the camera and the object and S is the shape matrix which represents the 3D position of each of the feature points on the object.

$$\tilde{W} = \begin{bmatrix} \mathbf{i}_1^T \\ \vdots \\ \mathbf{i}_F^T \\ \mathbf{j}_1^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{bmatrix} \tag{8}$$

$$\tilde{W} = RS \tag{9}$$

The registered trajectory matrix ($\tilde{W}$) can be decomposed into R and S through singular value decomposition (SVD). Singular value decomposition finds three matrices that when multiplied together yield the original matrix as follows $\tilde{W} = O_1 \, \Sigma \, O_2$. The rows/columns corresponding to the highest three singular values (expressed below using Matlab notation) are then used to construct $\hat{R}$ and $\hat{S}$, which will be used to calculate R and S.

$$\hat{R} = O_{1(1:3,:)}[\Sigma_{(1:3,1:3)}]^{\frac{1}{2}} \tag{10}$$

$$\hat{S} = [\Sigma_{(1:3,1:3)}]^{\frac{1}{2}} O_{2(:,1:3)} \tag{11}$$

$$\tilde{W} = \hat{R}\hat{S} \tag{12}$$

This factorization of $\tilde{W}$ into $\hat{R}$ and $\hat{S}$ is not unique. Inserting any invertible 3×3

matrix (Q) and its inverse yields a different factorization: $\tilde{W} = (\hat{R}Q)(Q^{-1}\hat{S})$. The Q that gives the appropriate factorization is the matrix which enforces the following constraints on the motion matrix: 1) the rows of R are unit vectors, therefore they should have a norm of one 2) the rows of R represent a translation of orthogonal x and y axes, therefore the dot product of corresponding i_f and j_f vectors should be zero. These conditions yield the following 'metric constraints'.

$$\hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{i}}_f = 1 \tag{13}$$

$$\hat{\mathbf{j}}_f^T Q Q^T \hat{\mathbf{j}}_f = 1 \tag{14}$$

$$\hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{j}}_f = 0 \tag{15}$$

The Q that best meets these constraints is then used to calculate R and S.

$$R = \hat{R}Q \tag{16}$$

$$S = Q^{-1}\hat{S} \tag{17}$$

A more in depth look at structure from motion through factorization is available in computer vision text books such as [83, 40] or in the original paper [89]. Work in low-rank matrix factorization methods such as [40, 7, 10, 36, 57] can be used to deal with trajectory matrices that have missing data.

This factorization method is appropriate for a single rigid body. Costeira and Kanade [15] expanded this method for multiple rigid bodies. They performed the SVD of an unsegmented trajectory matrix containing multiple independent motions. Next they used the right singular vectors (V) to create a 'shape interaction matrix' (Q), $Q = VV^T$. Due to the independence of the motions, and therefore the independence of the trajectories of points on different objects, the shape interaction matrix contains zeros in positions where the point represented by the row and the point represented

by the column are on different objects. The permutation of rows and columns that transforms Q into block diagonal form represents the segmentation and can be used to segment the original trajectory matrix. This multi-body structure from motion and segmentation method requires the motions to be fully independent. The motion of a satellite with an articulating component will not result in fully independent motions, but rather articulation results in trajectory matrices that intersect.[98, 90]

One issue with this method of calculating shape is the bas-relief (or mirror) ambiguity.[83] Since the trajectory matrix contains only information about the image coordinates of the points, the correct solution has the same trajectory matrix as would be seen if the feature points were mirrored over a plane of constant depth and the camera moved in the opposite direction. This can be seen in Figure 4 where a simulated satellite point cloud was used to develop a trajectory matrix which was in turn used to calculate a point cloud through factorization. The result demonstrates the mirror ambiguity. The calculated point cloud matches identically with the truth when viewed down the camera axis, however, when viewed perpendicular to the camera axis it is a mirror image of the truth. Numerous solutions to this problem have been suggested, for instance: use information gathered from point occlusions,[83] use perspective effects,[83] or compare the re-projection errors of both solutions.[84]

2.2.7 Silhouettes and Edges.

Feature points are an excellent way to translate 2D image motion into 3D shape, however other methods exist that do not rely on individual feature points. Instead, each image can be translated into a binary silhouette image that contains a value of 1 when the pixel is occupied by the object in question and a value of zero when it is not. Various techniques for image segmenting can be used to build the silhouette image by determining which pixels are in the foreground and which are in the background.[83]

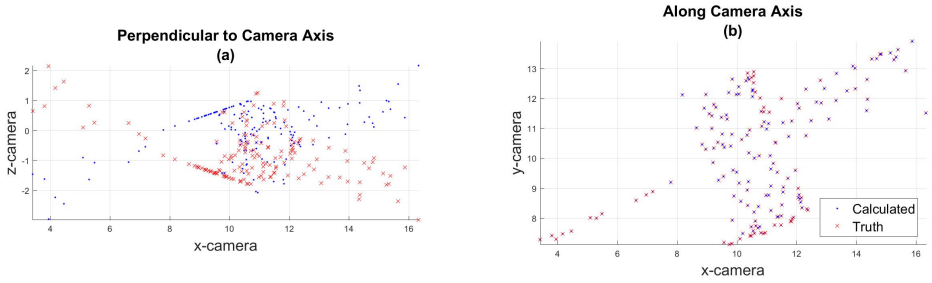

Figure 4. Illustration of the mirror ambiguity issue. (a) Point cloud calculated with factorization and truth point cloud viewed perpendicular to the camera axis (b) Point cloud calculated with factorization and truth point cloud viewed parallel to the camera axis

Silhouettes have an added advantage in space because often the satellite is the only object in view making background extraction trivial.

Edge detection is anther way to process images that can provide useful information. Edges identify where there are large changes in the images. While many edge detection techniques exist[83], most are in some way based on calculating the gradient between adjacent pixels. Detected edges can then be used to build an edge map which quantifies the distance to an edge for each pixel.

2.2.8 Simultaneous Localization and Mapping.

To enable autonomous operation, robots need to be able to understand where they are and what their surroundings look like. This problem is called simultaneous localization and mapping (SLAM), and it is heavily researched in the robotics and computer vision communities. While there are numerous methods of solving the SLAM problem, most of them involve solving for a distribution of the robot pose (x_t) and the feature point locations, or map (m), using some type of measurement (z) and robot control (u). This is expressed mathematically as $p(x_t, m|z_{1:t}, u_{1:t})$.[86] A graphical representation of the SLAM problem is shown in Figure 5. The measurements (z)

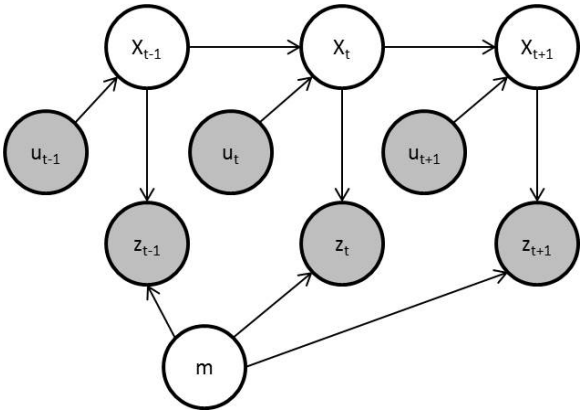

Figure 5. Overview of SLAM problem.[86] Shaded circles are observations and unshaded circles are the states to be estimated.

can be from any sensor such as stereo cameras, monocular cameras, RGB-D cameras, laser scanners, or LIDAR to name a few. Regardless of the sensor type, the measurements are stochastic. Since the measurements are stochastic, the calculated pose and map cannot be calculated deterministically, but must be estimated. The Kalman filter is a tool used to recursively estimate the state using stochastic measurements.

2.2.9 Kalman Filter.

The Kalman filter is an 'optimal, recursive, data processing algorithm'.[59] In its original form, it relies on the assumptions that all measurements are samples from a Gaussian distribution and the system is linear. If these assumptions are met, linear algebra techniques can be used to recursively transform measurements and control inputs into a vector containing the mean of the desired states (the estimate) and a covariance matrix that represents the uncertainty in that estimate.

The Kalman filter basically consists of two steps: a propagation step and an update step. In the propagation step, the discretized equations of motion representing the system dynamics are used to propagate the state forward in time using the previous

state and the applied controls.

$$\hat{\mathbf{x}}_t^- = \Phi\hat{\mathbf{x}}_{t-1} + B\hat{\mathbf{u}}_t \tag{18}$$

$$P_t^- = \Phi P_{t-1}\Phi^T + Q \tag{19}$$

In equations (18) and (19), the state vector ($\hat{\mathbf{x}}$) is propagated from time $t-1$ to time t using the discretized equations of motion (represented by Φ and B) and the applied control u. The covariance matrix (P) is propagated using the state transition matrix (Φ) and a discretization of the uncertainty in the equations of motion (Q). The propagation step is continued until a measurement is available, in which case the update step is performed.

The update step involves incorporating the measurement (z) in a way which optimally accounts for the uncertainty in the propagated state and the uncertainty in the measurement. To do this, the Kalman gain matrix (K) is created using equation (21). The Kalman gain matrix represents how to apply the residual (difference between the measurement and the predicted measurement) to the state estimate. In equation (20), H relates the state to the measurement equation and δ is the noise in the measurement which has a covariance of R.

$$\hat{\mathbf{z}}_t = H\mathbf{x}_t + \delta \tag{20}$$

$$K = P_t H^T (H P_t H^T + R)^{-1} \tag{21}$$

$$\hat{\mathbf{x}}_t^+ = \hat{\mathbf{x}}_t^- + K(\mathbf{z}_t - H\hat{\mathbf{x}}_t^-) \tag{22}$$

$$P_t^+ = (I - KH)P_t^- \tag{23}$$

The Kalman gain matrix is then used to calculate an updated state ($\hat{\mathbf{x}}_t^+$) and an updated covariance matrix (P_t^+).

2.2.10 Extended Kalman Filter.

One of the primary assumptions that enables the Kalman filter to efficiently update the statistics of each state is that the equations relating each state to each measurement and the equations that relate each state to the previous state are linear. The Gaussian random variable transformed by a linear equation remains Gaussian. When the propagation or measurement equations are not linear, the Kalman filter (in its traditional form) cannot be used to estimate the state. For this reason, the extended Kalman filter (EKF) was developed.

The EKF is very similar to the standard Kalman filter, but without the limitation that equations (18) and (19) are linear. Instead, they can be any functions f and h such that $\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \epsilon_t$ and $\mathbf{z}_t = h(\mathbf{x}_t) + \delta_t$ where ϵ_t and δ_t are Gaussian white noise. The linearity requirement in the standard Kalman filter is not a requirement for the propagation of the state itself, but rather it is a requirement to allow the statistics of the state vector to be propagated using the state transition matrix as in equation (19). The state vector statistics can be approximated as linear using a first-order Taylor series expansion of f and h evaluated at the best estimate of the state. To do this, the Jacobian of the function f and g must be taken to give the approximated state transition matrix (Φ) and measurement matrix (H).

$$\Phi \approx \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{t-1}} \tag{24}$$

$$H \approx \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_t} \tag{25}$$

These are used in equations (19), (21)-(23) while equations (18) and (20) are replaced with $\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \epsilon_t$ and $\mathbf{z}_t = h(\mathbf{x}_t) + \delta_t$ respectively.

2.2.11 Unscented Kalman Filter.

The EKF enables the use of nonlinear dynamics and measurement models by linearizing the models to calculate the statistics of the random variables. The unscented Kalman fitler (UKF) is another method of recursive estimation that accommodates non-linear functions. The UKF uses a set of discretely sampled 'sigma points' to parameterize the mean and covariance of the random variables.[47] The following is a brief overview of the UKF framework using notation as consistent as possible with the previous overviews of the KF and EKF.

To begin, a matrix of $2n+1$ sigma points χ_{t-1} is created using the previous estimate and the covariance matrix. γ is a term that indicates how far from the mean the sigma points are spaced.

$$\chi_{t-1} = \left[\hat{\mathbf{x}}^{+}_{t-1} \quad \hat{\mathbf{x}}^{+}_{t-1} + \gamma\sqrt{P^{xx,+}_{t-1}} \quad \hat{\mathbf{x}}^{+}_{t-1} - \gamma\sqrt{P^{xx,+}_{t-1}}\right] \tag{26}$$

Each of these sigma points (the columns of χ_{t-1}) are propogated through the nonlinear dynamics to give χ^{*}_{t-1}. The propagated sigma points are then combined using a vector of weights $\mathbf{w}_m$ to give $\hat{\mathbf{x}}^{-}_{t}$ which is in turn used to determine $P^{xx,-}_{t}$ using a different vector of weights $\mathbf{w}_m$.

$$\hat{\mathbf{x}}^{-}_{t} = \sum_{i=1}^{2n+1} w^{i}_{m}\chi^{*i}_{t-1} \tag{27}$$

$$P^{xx,-}_{t} = \left[\sum_{i=1}^{2n+1} w^{i}_{c}(\chi^{*i}_{t-1} - \hat{\mathbf{x}}^{-}_{t})(\chi^{*i}_{t-1} - \hat{\mathbf{x}}^{-}_{t})^{T}\right] + Q \tag{28}$$

The weight vectors $\mathbf{w}_m$ and $\mathbf{w}_c$ can be used to adjust the filter and match certain aspects of the distribution. The propagated mean and covariance matrices are then

used to create a new set of sigma points.

$$\bar{\chi}_t = \begin{bmatrix} \hat{\mathbf{x}}_t^- & \hat{\mathbf{x}}_t^- + \gamma\sqrt{P_t^{xx,-}} & \hat{\mathbf{x}}_t^- - \gamma\sqrt{P_t^{xx,-}} \end{bmatrix} \tag{29}$$

Measurement sigma points $\mathcal{Z}_t$ are then calculated by evaluating the measurement model for each of the sigma points in $\bar{\chi}_t$. These measurement sigma points can then be used to find a predicted measurement $\hat{\mathbf{z}}_t$ and a measurement covariance P_t^{zz}.

$$\hat{\mathbf{z}}_t = \sum_{i=1}^{2n+1} w_m^i \mathcal{Z}_t \tag{30}$$

$$P_t^{zz} = \left[\sum_{i=1}^{2n+1} w_c^i (\mathcal{Z}_t - \hat{\mathbf{z}}_t)(\mathcal{Z}_t - \hat{\mathbf{z}}_t)^T \right] + R \tag{31}$$

The measurement covariance matrix P_t^{zz} is comparable to the term $HP_tH^T + R$ in equation (21) of the standard Kalman filter equations. The P_tH^T term is comparable to the cross covariance matrix P_t^{xz} calculated from the sigma point matrices, predicted measurement, and predicted state as follows.

$$P_t^{xz} = \left[\sum_{i=1}^{2n+1} w_c^i (\bar{\chi}_t - \hat{\mathbf{x}}_t^-)(\mathcal{Z}_t - \hat{\mathbf{z}}_t)^T \right] \tag{32}$$

Finally, the Kalman gain can be calculated and used to update the state estimate and the covariance matrix. The vector $\mathbf{r}$ is the residual, or the difference between the actual measurement from the sensor $\mathbf{z}_{meas}$ and the predicted measurement $\hat{\mathbf{z}}_t$.

$$K = P_t^{xz}(P_t^{zz})^{-1} \tag{33}$$

$$\mathbf{r} = \mathbf{z}_{meas} - \hat{\mathbf{z}}_t \tag{34}$$

$$\hat{\mathbf{x}}_t^+ = \hat{\mathbf{x}}_t^- + K(\mathbf{r}) \tag{35}$$

$$P_t^{xx,+} = P_t^{xx,-} - KP_t^{zz}K^T \tag{36}$$

Texts such as"Stochastic Models, Estimation and Control"[59] and "Probabilistic Robotics"[86] outline the theory and implementation of the Kalman filter, the extended Kalman Filter (EKF), and the unscented Kalman Filter (UKF) in more detail.

2.3 Literature Review

Many researchers have investigated different aspects of using specifically designed spacecraft to inspect primary satellites for the purposes of maintenance and repair. The requirement for autonomy, precision navigation, and the size, weight, and power (SWAP) constraints in space drives many researchers to computer vision as a solution. Research in the area of computer vision for proximity operations is very diverse, ranging from stereo vision used to estimate target satellite moment of inertia (MOI)[92] to demonstration of fully autonomous rendezvous and capture[44]. Most of these methods use some type of feature point tracking algorithm to match up locations in successive images that represent the same position in 3D space. These feature points are then used to help the inspector satellite navigate around the target satellite or identify properties of the target satellite, or in some cases even dock with the primary satellite. In nearly all cases, the target satellite is assumed to be a rigid body. The author has not discovered previous research attempting to use computer vision to determine if the primary satellite is in-fact a rigid body, or if it is undergoing some type of articulation.

While sensing and characterizing articulation has not been the focus of research in space, there has been extensive research into the area in the computer vision community. The existing work consists of a few sub-categories. Motion segmentation deals specifically with segmenting imagery or feature points based on different observed motions. Articulated structure from motion specifically looks at recovering the kine-

matic chain, structure, and articulation parameters of linked rigid bodies from a full set of imagery. Real-time SLAM solutions use imagery to estimate the kinematic chain, structure, articulation parameters and associated uncertainties as new measurements (imagery) becomes available. Work on tracking human articulated motion can be applicable to track articulation of any object with a known articulated model.

2.3.1 Computer Vision for Proximity Operations.

Many algorithms have been developed using computer vision to determine the relative pose between an inspector satellite and a primary satellite in space. Some of these algorithms rely upon markers on the primary satellite that assist the computer vision algorithm in determining pose.[91, 43] Others rely on prior knowledge of the primary satellites configuration.[65, 39] Some methods rely on stereo vision systems, [92, 29] some on monocular vision systems, [71, 48, 39] some use laser illumination of reflective markers.[44] They all use a computer vision method that identifies features in images and matches those features from frame to frame (or in corresponding frames in the case of stereo systems). The relative position of the features is then estimated using some type of estimation filter such as an extended Kalman filter (EKF) or a particle filter. A reference frame defined with feature points [71, 102] or a known model of the primary satellite [65] can then be used to estimate the attitude of the primary satellite. Alternatively, if the orientation of feature points is known, the attitude can be calculated directly and used as the measurement in the EKF.[91, 71, 102] Feature points can also be used to create a 3D model of the satellite.[33]

Previous work by Tweddle using stereo cameras has demonstrated that mass moments of inertia can be estimated for a spinning satellite using computer vision algorithms.[92] Similar work by Yu estimated mass center and mass moments of inertia for tumbling satellites.[102] If the mass moment of inertia were estimated, changes in

the mass moments of inertia estimates could be sensed,[42] which may indicate some change such as articulation. However, this is a limiting case since the estimation of mass moments of inertia from computer vision is dependent on the angular velocity of the primary satellite and is ambiguous as to the real physical geometry when articulation is possible. A well controlled operational satellite is unlikely to be spinning or tumbling, therefore this method is unlikely to be applicable to sensing articulation in most cases.

While both the inspector and primary satellite are moving, the problem of finding the relative position between the two can be looked at using the assumption that either the primary is moving and the inspector is stationary, or the inspector is moving and the primary is stationary. Ghadiok et al.[33] made the assumption that the primary was stationary and the inspector was moving so that angular velocity measurements from the inspector could be used to improve the accuracy of the relative position of feature points. They used a nonlinear complementary filter to fuse the feature position measurements from the computer vision algorithm with inspector gyro measurements. Similarly, Philip and Ananthasayanam[71] assumed the primary was stationary and used the inspector gyro measurements along with a Kalman filter and extended Kalman filter to estimate primary position and attitude respectively.

Tweedle and Saenz-Otereo [91] developed a method of navigation for a small inspector satellite based on a monocular camera capturing images of four co-planar circular markers. Once the four feature points are detected a nonlinear iterative process is used to solve for the relative pose between the inspector and the co-planar feature points. The result of this process is a relative pose measurement (rotation and translation). This pose calculation is then filtered using a multiplicative extended Kalman filter (MEKF). The MEKF deals with the fact that a quaternion representation of a rotation matrix contains four elements, but only three degrees of freedom.

This means one of the elements is not independent. To deal with this, the MEKF uses a three parameter representation of the change in rotation in the state vector then updates a reference quaternion at each iteration. Alternative solutions, such as the additive EKF (AEKF), are outlined in [16].

Similarily, Philip and Ananthasayanam [71] outline a method of attitude estimation and control for the final phase of rendezvous and docking using monocular imagery to estimate the relative pose between the primary satellite and the inspector. The primary satellite is assumed to have three feature points with known correspondence. These can be used to calculate the relative pose (rotation and translation). The calculated relative pose is used as the measurement in a linear Kalman filter for estimation of the relative distance and velocity, and an EKF for estimation of the relative rotation and angular velocity.

Yu et al. [102] develop a system to determine the relative pose and the moment of inertia ratios of a non-cooperative satellite using stereo-vision. The inputs to the system are 3D feature points (they assume the stereo vision system can provide 3D points directly). Two algorithms (TRIAD and QUEST) are used to estimate the pose of the target from the feature point measurements. The moment of inertia ratios can only be determined because the target satellite is unresponsive and therefore it is in a torque free spin. With the pose information as input measurements, an EKF and an UKF are developed to estimate the relative pose and the moment of inertia ratios. The QUEST algorithm and the UKF resulted in more accurate estimation of the parameters.

2.3.2 Motion Segmentation.

The term motion segmentation is used in the computer vision field for any method that attempts to identify and separate different motions in a video sequence. Zappella

et al.[105] summarize a number of motion segmentation algorithms and classify them based on the method used for segmentation. He also outlines some basic attributes and strategies employed by various motion segmentation algorithms that could be valuable in selecting an approriate method for a particular application. Many of the strategies employ spectral clustering on some type of similarity matrix that quantifies the similarity between each combination of points.

Subspace clustering methods use the fact that for an affine camera model, the trajectory matrix of feature points on a rigid body will have a rank of at most four. Motion segmentation then becomes a problem of clustering the columns of the trajectory matrix into independent subspaces. Figure 6 graphically shows an example of feature point trajectories from the simulated satellite both before segmentation (Figure 6a) and after the trajectory matrix has been segmented (Figure 6b-c) into two different objects. Each line in Figure 6 represents the image coordinates (u and v) of a feature point over time. It is evident in looking at Figure 6a that the trajectories can be grouped into two separate motions; manifold clustering algorithms attempt to make that separation mathematically. Vidal also provides a review of subspace clustering algorithms.[93] In his review, numerous algorithms are outlined and the advantages and disadvantages of each are discussed. Details of a few of the motion segmentation algorithms are outlined in the remainder of this section.

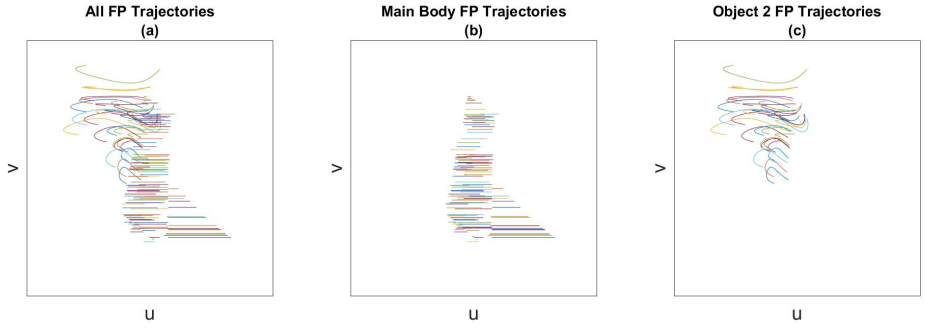

Figure 6. Motion Segmentation Illustration: (a) All Feature Point (FP) Trajectories, (b) Main body FP Trajectories, (c) Object 2 FP Trajectories

Local Subspace Affinity (LSA).

Yan and Pollefeys have a number of papers on the subject of motion segmentation. The method outlined in their 2006 paper [100] has been termed 'Local Subspace Affinity' (LSA). The LSA method is a manifold clustering algorithm, so it is has the trajectory matrix described in equation (7) at its core. To segment the trajectory matrix according to different motions, LSA uses the concept that trajectories of feature points on the same object "lie in a low dimensional linear manifold" and trajectories of feature points on objects with different motions "result in different linear manifolds". LSA also uses the concept of locality which means that the basis of a trajectory and its nearest neighbors will lie in the same linear manifold as other trajectories of the same motion.

The LSA algorithm begins by decomposing the trajectory matrix (W) using a singular value decomposition (SVD) to create U, S, and V. Equation (37) can be used to find the rank (r) of W where λ_i are the ordered singular values of the trajectory matrix (W).

$$r = \underset{r}{\operatorname{argmin}} \left(\frac{\lambda_{r+1}^2}{\sum_{i=1}^{r} \lambda_i^2} + kr \right) \tag{37}$$

The columns of V can now be thought of as the direction of the feature points motion in r-dimensional space, or their location on an r-dimensional sphere. Columns corresponding to feature points with the same motion will lie on lower dimensional cuts of this r-dimensional sphere. The parameter k must be tuned according to the noise in the trajectory matrix. While the LSA algorithm technically does not require prior knowledge regarding the rank of the trajectory matrix, the requirement to 'tune' k means some prior knowledge is required. Zappella provides a fix to this problem by using the entropy of the affinity matrix as a measurement of the quality of the k

parameter [104].

Features will likely have the same motion as features closest to themselves in the transformed space (the r-dimensional sphere). 'Closeness' can be measured by the Euclidean distance or by the principal angle between them. For each feature point, the local subspace is found by first finding its n nearest neighbors (using Euclidean distance or principal angle) and then finding a basis for the combination of those points using SVD. The size of the basis is determined by finding the rank of the local subspace using equation (37) and a lower value of k.

Next, the principal angles between theses subspaces must be determined. To do this, first define two subspaces as $S(\alpha)$ and $S(\beta)$, each with normalized columns. The principal angles (θ_{ss}) between these two subspaces are the angles between each column of $S(\alpha)$ and the closest (largest dot product) column of $S(\beta)$. These principal angles can then be used in equation (38) to calculate an affinity (A) between each subspace.

$$A_{i,j} = e^{-\sum_{k=1}^{M} sin^2(\theta_{ss_k})} \tag{38}$$

In LSA, the number of motions is known *a priori*, so once the affinity matrix is calculated a spectral clustering algorithm is used to determine which columns of the affinity matrix match up to which motion. The normalized cuts method outlined by Shi and Malik [80] is proposed by Yan and Pollefey [100, 101]. The normalized cuts method recursively segments the affinity matrix until the number of segments is met. Numerous cuts (or segmentations) of the affinity matrix are evaluated to determine which one minimizes the 'Ncut', or normalized cuts, criteria shown in equation (39). Shi and Malik [80] prove the normalized cuts criteria measures both the total dissimilarity between the different groups (S_1 and S_2) and the similarity

within the groups.

$$Ncut(S_1, S_2) = \frac{cut(S_1, S_2)}{assoc(S_1, A)} + \frac{cut(S_1, S_2)}{assoc(S_2, A)} \tag{39}$$

In equation (39) the 'cut(S_1,S_2)' operator is the sum of all positions in the affinity matrix where the column index is in group S_1 and the row index is in the group S_2. The 'assoc(S_1,A)' operator is sum of each of the rows of A corresponding to the group S_1.

Shi and Malik [80] also show that the eigenvector corresponding to the second smallest eigenvalue of the symmetric normalized Laplacian matrix (L_{sym}) can be used to segment the affinity matrix. Equation (40) shows how to calculate L_{sym}, where D is a $P \times P$ diagonal matrix containing the sum of each row on the diagonal of the affinity matrix and A is the affinity matrix.

$$\begin{aligned} L &= D - A \\ L_{sym} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \end{aligned} \tag{40}$$

The eigenvector corresponding to the second smallest eigenvalue is then used to segment the affinity matrix. Ideally, the eigenvector should only take on two discrete values [80], however the eigenvector could be continuous. Shi and Malik suggest using multiple thresholds within the range of the eigenvector, calculate the normalized cuts for each, and take the segmentation that minimizes the normalized cut criteria.

Once the affinity matrix is segmented into two groups, each of those groups is then segmented further using the same method until the total number of groups is met. A metric from graph theory is used to determine which of the segments to partition at each step.[80]

While Yan and Pollefey suggest that the rank of the trajectory matrix is not

needed for the LSA algorithm, from equation (37) it is evident that the parameter k is directly related to the rank. The calculation of r from (37) is very sensitive to the right selection of k [104], therefore the LSA algorithm relies on prior knowledge to choose the correct value for k.

Motion Segmentation via Agglomerative Lossy Compression.

Rao et al. [74] developed a method based on Agglomerative Lossy Compression which minimizes a cost function based on the 'coding length', or the number of bits required to describe the segmentation. The method begins by treating each point as a separate subspace. Then it merges subspaces and calculates the effect of the merge on the coding length. Merges that decrease the coding length are kept and the process is repeated until further improvement is not possible.

Generalized Principal Component Analysis.

Vidal et al.[94] proposed a different solution where they first project the trajectory matrix into a 5-dimensional subspace. They prove that different motions, even if the motions are partially dependent (such as articulated motions), will remain different along at least one dimension when projected onto a 5-dimensional subspace. Once projected, an n-degree polynomial (where n is the expected number of independent motions) of five variables is found. The derivative of that polynomial is evaluated for each point yielding a 5 element vector. If the two points are in the same subspace, the angle between these vectors will be zero (or π). The points can then be segmented according to the angle between the vectors. A disadvantage of this method is that the number of trajectories required increases exponentially as the number of different motions increases.[99, 93]

RANSAC with Priors.

Yan and Pollefeys [99] propose another method where they build an affinity matrix ($A = W^T W$) consisting of the inner product of each column with each other column. They use the affinity matrix to calculate a 'normalized spectral representation', y_i of each trajectory in R^k where k is the rank of the affinity matrix. The 'prior matrix' is then created by calculating the probability P that trajectory i belongs to the same motion as trajectory j using the distribution:

$$P_{ij} = \frac{2}{\sqrt{\pi}} \int_0^{y_1 y_j^T} e^{-t^2} dt \tag{41}$$

Next, the probabilites in P are used to randomly select k trajectories that are likely to have the same motion. A model is built from these trajectories and the trajectories that match this model are grouped into a 'consensus set'. This process is repeated numerous times to find the model with the most inliers (largest 'consensus set'). Finally, the whole process is repeated until all the data is in a set or the existing data cannot be satisfactorily fit to a model. This method allows for articulated motion and does not require *a priori* knowledge of the number of motions, however the rank must be estimated to find the appropriate value of k.

Sparse Subspace Clustering.

The SSC algorithm demonstrated the lowest misclassification rate for sequences with articulated motion of the methods reviewed in [93]. This method, proposed by Liu et al.[53], uses the lowest rank representation of the trajectory matrix to build an affinity matrix that can be segmented using the normalized cuts method outline above.[80] The lowest rank representation is found by solving the following

optimization problem:

$$\min_{Z,E}(\|Z\|_* + \lambda\|E\|_{2,1}) \tag{42}$$

$$s.t.,\ X = XZ + E \tag{43}$$

where X is the trajectory matrix, Z is the low-rank representation (LRR), λ is a tuning parameter, and E is an error matrix to account for corrupted data points with the $l_{2,1}$-norm defined as $\|E\|_{2,1} = \sum_{j=1}^{n} \sqrt{\sum_i = 1^n (E_{ij})^2}$. The operator $\|\cdot\|_*$ is the sum of the singular values (the nuclear norm). The optimization problem is solved by augmenting with Lagrange multipliers. Once the low-rank represetion (Z) is found, the affinity matrix is found by $A = |Z| + |Z^T|$. Segmentation using this method produces good results even with up to 60% of columns containing missing data.[53]

2.3.3 Articulated Structure from Motion.

The structure from motion problem for a single rigid body is effectively solved using the factorization method of Tomasi and Kanade.[89] When there are multiple independent motions, a motion segmentation algorithm can be used to separate the feature points into rigid objects which can be reconstructed using factorization. However, when feature points are not rigid, but are not independent alternative methods are needed to solve for the time history of the structure.

Ozden et al.[68] propose a sequential multi-body structure from motion routine that performs initial segmentation of feature points into objects demonstrating distinct motion using three-views: the first, middle, and last views in the sequence. When enough data is available, structure from motion is performed for each object. The algorithm calculates the 3D motion of the detected objects. If the feature points assigned to an object are not in fact part of the same rigid body, either due to mis-segmentation or the object splitting into multiple independent objects, the calculated

camera parameters will not be accurate which will lead to an increase in outliers. A likelihood function is used to determine if the object has split into multiple motions. A similar likelihood function is used to determine if objects have merged.

Yan and Pollefeys [98] focus on recovering the trajectory of the linkage between the two objects in the image. Once the trajectory matrix has been segmented using a subspace clustering algorithm, the intersection of the subspaces representing the two motions can be determined. This intersection of the subspaces (T) contains the trajectory of the link between the two objects. In the case of a hinged linkage (where the objects are connected by multiple points along an axis), the intersection will be two dimensional. To find that subspace, take the SVD of the trajectory matrices of the two objects: $W_1 = U_1\Sigma_1 V_1^T$, $W_2 = U_2\Sigma_2 V_2^T$. The motion subspaces ($\tilde{U}_1$ and $\tilde{U}_2$) are formed by the first four columns of U_1 and U_2. Since the intersection is a 2-dimensional subspace, the matrix $[\tilde{U}_1|\tilde{U}_2]$ will have two singular values equal to zero. Taking the SVD of $[\tilde{U}_1|\tilde{U}_2]$ and setting N equal to the columns of V corresponding to the zero singular values yields,

$$\left[\tilde{U}_1|\tilde{U}_2\right] N = 0 \tag{44}$$

$$\left[\tilde{U}_1|\tilde{U}_2\right] \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} = 0 \tag{45}$$

$$T = \tilde{U}_1 N_1 = -\tilde{U}_2 N_2 \tag{46}$$

where T is the subspace representing the intersection of W_1 and W_2.

Once the subspace of the intersection is found, specific trajectories that lie on the axis can be found. The requirements for a trajectory (m) to lie on the axis are: 1) that it lies in the subspace T and 2) that it does not increase the rank of W_1 (or W_2) when it is appended as an additional column. To meet the first constraint m must

be a linear combination of the columns of T, or

$$m(\alpha, \beta) = T \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{47}$$

where α and β are scalars. The second criteria is met by ensuring the augmented matrix, $[\tilde{W}_1|(m(\alpha,\beta)-\bar{W}_1)]$ has a rank of 3, where $\bar{W}_1$ is the row average of W_1 and $\tilde{W}_1$ is $W_1-\bar{W}_1$. Yan and Pollefeys manipulate this constraint to yield an equation in which the summation of five determinants is equal to zero.[98] Solving these constraints yields a linear equation in α and β. Values of α and β can be selected to give a point on the trajectory at a particular u or v coordinate using equation (47). In the case of a joint linkage (where a single point on the two objects are connected) the intersection is one dimensional. A similar method can be used to find the trajectory of the linkage.

Yan and Pollefeys [101] also present a method for determining the kinematic chain that links multiple articulated objects together. The kinematic chain is the order in which parts are linked. They use the LSA motion segmentation process presented in section 2.3.2 to segment the trajectory matrix into different objects. Next, a graph is created containing nodes for each of the different motion subspaces. The length of the edges connecting the nodes of the graph are the minimum principal angle between the subspaces. This value can be found by looking at the affinity matrix described in equation (38). Since the motion subspace of linked objects are not independent, the minimum value for a principal angle between them will be zero (or near zero). To build the kinematic chain a minimum spanning tree algorithm is run on the graph. This method allows the algorithm to handle multiple independent objects each with articulated motions.

With the kinematic chain known, the rows of the trajectory matrix for the i-th

frame and the points from two linked objects can be expressed as shown in equation (48).

$$W_i = (R_i^1|T_i|(R_i^1 \cdot O_i)|T_i) \begin{bmatrix} S_1 & & \\ \mathbf{1} & & \\ & S_2 & \\ & \mathbf{1} & \end{bmatrix} \tag{48}$$

The rotation matrix O_i represents the articulation of object 2 with respect to object 1 at frame i.

Tresadern and Reid [90] outline a method of detecting articulation, identifying parameters of the linkage, and recovering the shape of the objects in a consistent coordinate frame. The type of articulation is determined by looking at the singular values of the combined trajectory matrix between two objects. The trajectory matrix of two independent motions will have rank eight. A universal joint reduces the rank to seven and a revolute joint reduces the rank to six. To determine the rank, the ratios of the seventh and eight singular values (σ_7/σ_8) and sixth and seventh singular values (σ_6/σ_7) are calculated. For completely independent motion (no articulation) neither of these ratios will be large. For universal joint articulated motion σ_7/σ_8 will be large, and for hinged articulated motion σ_6/σ_7 will be large.

Once the trajectories have been segmented into separate motions, methods are outlined for determining the location of the universal joint and the orientation of the axis for a revolute joint. For a universal joint, the rotations are independent, but the distance between each object center (t_1 and t_2) and the joint location are constant

(d_1 and d_2). Written in terms of the trajectory matrix this means:

$$W = \begin{bmatrix} R_1 & R_2 & t_1 \end{bmatrix} \begin{bmatrix} S_1 & d_1 \\ 0 & S_2 + d_2 \\ 1 & 1 \end{bmatrix} \tag{49}$$

Normalizing the system yields:

$$\tilde{W} = \begin{bmatrix} R_1 & R_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \tag{50}$$

which represents the motion and shape of the two objects.

Similarly, for a revolute joint, the relative orientation of the two objects is also constrained. By assigning the x-axis as the hinge axis, this becomes

$$\tilde{W} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & \mathbf{c}'_2 & \mathbf{c}'_3 \end{bmatrix} \begin{bmatrix} x_1^{(1)} \cdots x_p^{(1)} & x_1^{(2)} \cdots x_p^{(2)} \\ y_1^{(1)} \cdots y_p^{(1)} & \mathbf{0} \\ z_1^{(1)} \cdots z_p^{(1)} & \mathbf{0} \\ \mathbf{0} & y_1^{(2)} \cdots y_p^{(2)} \\ \mathbf{0} & z_1^{(2)} \cdots z_p^{(2)} \end{bmatrix} \tag{51}$$

where $R_1 = [\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3]$ and $R_2 = [\mathbf{c}_1, \mathbf{c}'_2, \mathbf{c}'_3]$. In each case, Tresadern and Reid [90] outline methods of converting the full matrix shape and motion matrices from factorization of the full trajectory matrix using SVD into the form shown in equations (50) and (51).

Yucer et al.[103] developed a method of reconstructing an object with multiple articulated motions using optimization techniques. First, a 'ray-space optimization' method is used to convert each 2D image trajectory into a 3D trajectory. An energy function is developed that measures the frame to frame movement of the feature point.

The 3D trajectory that minimizes the energy function while remaining on the 'ray' between the 3D point and the 2D image coordinates is selected. This method requires the camera motion *a priori*, however this requirement is not an issue if the relative inspection route between the inspector and the primary satellite is known. The 3D trajectories of each point are then used to segment the motion into N rigid bodies based on the idea that points on the same rigid body will remain near each other, and the same distance from each other, throughout all frames. Once segmented, equation (52) is used in an optimization routine to find a rotation matrix (R_f^t), point cloud for each object (Ω_n), and translation (T_f^n) that minimizes the error in re-projection to image coordinates (W_f^n). $P_{cam,f}$ is the calibration matrix at frame f.

$$\min_{\Omega,R,T} \sum_{f=1}^{F} \sum_{n=1}^{N} \|W_f^n - P_{cam,f}(R_f^n \Omega^n + T_f^n)\|^2 \tag{52}$$

Next, the kinematic chain, or 'skeleton', is estimated as a minimum spanning tree of a graph created with the rigid components at the nodes and edges based on the minimum distance between objects. Finally, the location of the joints between the rigid components is estimated and the optimization is repeated with the added constraint that the 3D positions of joints must be maintained. This method works with a perspective camera and demonstrated excellent results, however the camera motion is required *a priori* and the multiple optimization routines are computationally expensive.

Paladini et al. [69] present a method in which the motion matrix and shape matrix are solved in an alternating fashion with least squares. First, an initial solution to the motion matrix ($M^{(0)}$) is found using the method outline by Tresadern and Reid[90]. This is then refined by optimizing a cost function that includes the metric constraints and the articulation constraints. The motion matrix and the trajectory matrix are

then used to solve the shape matrix ($S^{(t)}$). The shape matrix is then used with the trajectory matrix to solve again for the motion matrix ($M^{(t+1)}$). This process is continued until convergence. Additionally, Paladini et al. [69] 'wrap' this method in an outer loop that allows the method to handle missing data. The method outlined in [57] is used to provide an initial fit of the rigid data, then the steps outlined above are used to calculate the motion matrix and the shape matrix. These are then multiplied to calculate the missing entries. This process is continued until convergence. Results suggest the algorithm is capable of handling over 60% missing data entries without detriment to accuracy of the recovered shape.[69]

Zhang and Hung [106] approach the problem of recovering articulated structure from motion as an ellipsoid fitting problem. They begin by noting that an ellipsoid is defined as a set of points P such that

$$P = \begin{bmatrix} r_1\sigma_1 & r_2\sigma_2 & r_3\sigma_3 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \tag{53}$$

where $\mathbf{r}$ is a unit vector and V_1, V_2, and V_2 are orthogonal vectors. The factorization of $\tilde{W}_f$ can be represented by $\tilde{W}_f = R_f S$. Taking the SVD of the shape matrix ($S = U\Sigma V^T$) yields, $\tilde{W}_f = R_f U\Sigma V^T$. Incorporating the U and Σ into R_f gives,

$$\tilde{W}_f = \begin{bmatrix} r_1\sigma_1 & r_2\sigma_2 & r_3\sigma_3 \\ & \vdots & \\ r_1\sigma_1 & r_2\sigma_2 & r_3\sigma_3 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \tag{54}$$

This means that each row of $\tilde{W}^i$, where i represents the points on a rigid body, can be represented as a 3D ellipsoid. Since points corresponding to the same motion will lie on the same ellipsoid, this can be used to segment the points. They use the error

in fitting a point to an ellipsoid as a metric with which to segment the feature points into separate motions. Once segmented, they build the kinematic chain by finding the minimum spanning tree of a graph containing each object as a node connected by a edges determined by a metric measuring the distance between two objects over all frames. The mirror ambiguities are reduced to a single ambiguity for the entire structure by picking a solution for the first part and propagating the solution to the other parts by assuming motions of linked parts are similar.

Much of the literature on recovery of articulated structure from motion is focused on capturing the motion of a human. Some of this research can be applied to articulated motion of any kind. For instance, Fayad et al. [30] present a method of automatically recovering the 3D shape and structure of an articulated body. Their method uses an optimization routine to assign feature points to the motion that minimizes the total re-projection error of the solution. The method allows points to belong to more than one motion. This overlap of points on multiple motions defines the joint between the two objects and allows the kinematic chain to be built within the optimization framework. The number of motions does not need to be known, but each motion must have at least three points, and there must be at least one point that is contained in both linked objects. While applied primarily to human motion, the method is capable of performing on imagery of any articulated object. However, this method may fail if there are not points on each object that overlap the joint, which may be the case for an articulated solar array on a satellite.

Russell et al. [77] expand on the method of Fayad et al. to include the capability to segment motion into independent objects and further segment independent objects into multiple parts based on dependent motions such as articulation. They do this by accounting for the fact that an object may be in front of another object, in which case feature points that are nearest to one another are not necessarily part of the

same object. Another improvement to [30] is in the cost function. Rather than using the re-projection error as the metric that enforces the geometric constraint in the cost function, they use the summation of the Sampson errors (an approximation of the geometric distance between image coordinates in one frame projected to the next frame) for each point between every pair of consecutive frames as well as a saliency term. The saliency term measures the difference between the salience at each feature point location in each image and the average saliency of all the points in the model. Saliency is a measure of how 'interesting' areas of an image are.[45]

Understanding articulated motion is also important to allow robots to learn about how objects move in order to allow the robots to manipulate the objects in the future. With this motivation, Pillai et al. [72] developed a training process by which the parameters of everyday articulate objects, such as doors and drawers are determined off-line from video of a person using those objects. RGB-D imagery is used along with a custom method of feature point tracking to create 3D feature point trajectories for the scene. To segment features into different motions a likelihood function is built based on the relative motion between each pair of feature points. After segmentation, optimization routines are used to determine the pose of each object and to build the kinematic chain. Once the robot has 'learned' about how objects articulate it uses that knowledge to predict how the objects will respond to manipulation.

2.3.4 Real-time Simultaneous Localization and Mapping (SLAM).

The Kalman filter is a powerful tool for estimation that is often applied to the SLAM problem. Thrun et al. [86] extensively discuss the use of the EKF in solving the SLAM problem. In many implementations the measurements supplied to the EFK are from measurement systems capable of providing both bearing measurements as well as depth measurements, such as laser scanners, stereo systems, or RGB-D systems.

In the case of a monocular camera, no depth information is available from a single measurements.

Davison et al. [24, 25] present a method of using a monocular camera to solve the SLAM problem in real-time (at 30 Hz) using an EKF. The state vector for the EKF consists of the camera state and (x,y,z) coordinates of feature points in the map. The propagation (or prediction) step of the EKF requires a model of how each of the states is changing. A constant rate model was selected for the camera motion, requiring inclusion of the camera velocity and angular velocity in the state vector. The primary goal of Davison et al.'s implementation is to accurately and efficiently localize the camera, therefore, each point that is sensed is not tracked. For efficiency, only quality points that can be reliably tracked are included in the state vector. The feature point tracking method used is interesting in that the predicted point location and covariance are used to inform the search area in the next image. Also of note is the method of initializing points. Since depth information is not available from a single image, new points cannot be added to the state vector immediately. Instead, a line from the camera center to the observed point is added to the state vector. Particles are added to the line at different depths. With each new measurement, the likelihood of each particle is updated based on the location of the feature point. When the particle likelihoods are sufficiently Gaussian, the feature point is added to the map. Civera [14] extended this work by developing a parameterization method which allows feature points to be added to the map immediately by coding the point locations using the camera location in the frame where the feature was first detected, the angles to the feature from camera center and the inverse of the depth.

The EKF is one of the most popular methods of solving the SLAM problem, however it can be intractable when there are a large number of points in the map. Since the covariance matrix is $K \times K$ (where K is the number of feature points)

and must be updated with each measurement, the computational requirement scales $\mathcal{O}(K^2)$. To combat this problem, and relying on the conditional independence of points in the map, Montemerlo et al. [62, 61] developed the FastSLAM approach which combines a particle filter and an EKF. The FastSLAM method reduces the computational requirement to $\mathcal{O}(MlogK)$ where M is the number of particles used.

Most solutions to the SLAM problem assume that the feature points in the images are stationary with respect to one another. Inclusion of moving points in the state vector will introduce errors in the solution of both the camera motion and the map. Traditionally, points that do not remain in their predicted locations (moving points) are rejected as outliers, however in dynamic environments, or when an accurate motion model of the camera is not known, moving points can cause problems. Wangsiripitak and Murray [97] propose a solution by incorporating a moving object tracker into their implementation of EKF monocular SLAM. By tracking moving objects they are able to avoid inclusion of feature points that are on the moving object, or caused by the edge of the moving object interacting with the background, in the state vector. They are also able to use the motion of the tracked object to determine when a feature point in the scene should be occluded by the moving object. This prevents them from dropping and reacquiring feature points that are only temporarily occluded from the state vector.

Wang et al. [96] present a method of combining the tasks of SLAM and detection and tracking of moving objects (DATMO). When a robot moves through a dynamic environment, the objects in the environment can be categorized as moving objects or stationary objects. The SLAM with DATMO problem is an attempt to find the statistics for the robot pose (x_k), the state of the moving objects ($\mathbf{O}_k$), and the location of the stationary objects (M_k) given the measurements (Z_k) and the applied

controls (U_k).

$$p(x_k, \mathbf{O}_k, M_k | Z_k, U_k) \tag{55}$$

They prove these states can be decomposed into separate components represented by equation (57) using three assumptions: 1) the measurement can be decomposed into measurements of moving and stationary objects ($Z_k = [Z_k^o, Z_k^m]$), 2) the measurements of the moving objects carry no information about the stationary landmarks, and 3) there is no interaction between the robot and the moving object. In the case of articulated motion, the second assumption is not true. The measurements of the articulated object are not independent of the stationary object.

$$\begin{aligned} p(x_k, \mathbf{O}_k, \mathbf{M}_k | Z_k, U_k) \quad \propto \quad & \underbrace{p(z_k^o | \mathbf{O}_k, x_k)}_{\text{DATMO Update}} \\ \cdot \quad & \underbrace{\int p(\mathbf{O}_k | \mathbf{O}_{k-1}) p(\mathbf{O}_{k-1} | Z_{k-1}^o, U_{k-1}) d\mathbf{O}_{k-1}}_{\text{DATMO Prediction}} \\ \cdot \quad & \underbrace{p(z_k^m | \mathbf{M}_k, x_k)}_{\text{SLAM Update}} \\ \cdot \quad & \underbrace{\int p(x_k | u_k, x_{k-1}) p(x_{k-1}, \mathbf{M}_{k-1} | Z_{k-1}^m, U_{k-1}) dx_{k-1}}_{\text{SLAM Prediction}} \end{aligned} \tag{56}$$

In a follow-on work, Wang et al. [95] addressed the case where moving object motion models are strongly correlated. They create a scene interaction model to represent the long-term interactions between a moving object and its surroundings; for instance multiple vehicles that interact with a scene in the same way due to traffic laws. Additionally, they create a neighboring object interaction model which uses the motion of near by objects to inform the motion of an object that may be occluded; for instance people walking in a crowd. Both of these interaction models

take advantage of multiple objects moving in the same way. Wang et al. [95] also present a method detecting if a moving object has stopped by using an interactive multiple model technique in which the object is tracked using multiple motion models, including the stationary model, and the uncertainty in the state estimate is used to determining which model is most likely correct.

Kundu and Jawahar [51, 50] build a framework which incorporates feature detection, motion segmentation, SLAM, and moving object tracking to reconstruct a unified dynamic 3D map of the scene which includes the reconstruction of moving rigid bodies. Each of the modules inform one another so all the information available can be used to constrain the possible solutions in each step. Initial feature point matching is done over a large range, however once the relative camera motion is known from the SLAM step it is used to refine the feature point matching. The initial motion segmentation is done using a two-view motion segmentation algorithm outlined in [75]. In subsequent frames, the relative camera motions from the SLAM module is used along with epipolar constraints to determine the probability that feature points are part of a the stationary background or part of a previously modeled moving object. The visual SLAM module estimates the translation and rotation between the camera frame and the world frame using the stationary feature points and the translation and rotation between the camera and the object frame using the feature points associated with each object. Iterative bundle adjustments are used for each independent motion to optimally update the relative camera pose and the feature point locations. To track the motion of the independent objects, a bearing only tracking (BOT) particle filter is used. Particles are placed on the ray from the camera to the object at various depths. Information from the SLAM module is used to put bounds on the possible depths and velocities of the particles, improving the performance of the particle filter. In the unification, module information from SLAM and the BOT is used to eliminate

ambiguities in the full reconstruction of the scene.

Kumar et al. [49] incorporates articulated objects into an EKF SLAM solution. Instead of separating stationary map points and moving map points as is done in SLAM with DATMO [96], each feature point is tracked using the parameters of a motion model. The motion models represent the point as static, belonging to an object with a prismatic joint, or belonging to an object with a revolute joint. Each feature point is assigned to an articulated motion model M when the probability of the model M given the measurements $\mathbf{Z}_{0:t}$ is over a threshold τ: $P(M|\mathbf{Z}_{0:t}) > \tau$. Once assigned to a model, the parameters assigned to the state vector for the feature point are not the world frame 3D position of the points, but rather parameters that define the articulated structure and the articulated motion. In the case of a revolute joint, these parameters are a plane of motion P, the center of the articulation axis in that plane (x_0, y_0), the radius from the center of that circle to the point r, and the motion parameter ϕ_t that represents the articulation angle at time t. Similar representations are provided for the prismatic and the static case. Note that the measurements in this work are 3D positions taken from and RGB-D camera, this simplifies the Jacobians for the EKF significantly as compared to the monocular EKF implementation in [24, 25, 14].

Martin and Brock [58] also used RGB-D measurements from a stationary camera to estimate the characteristics of articulate motion in real-time. A three level recursive filter is used where each level estimates the measurements for the next level up. The first level takes in the measurements of the feature points from the camera and estimates the 3D position of all the feature points using the rigid body motion to inform the estimation. The second level consists of separate EKFs for each rigid body which take in 3D feature point locations from the first level as measurements and estimate the pose and velocity of the rigid object. The third level consists of three

EKFs for every pair of rigid bodies. For each pair, an EKF is maintained for each of three relationships between the rigid bodies: a prismatic joint, a revolute joint, and a rigid joint. If none of these filters adequately explain the motion, they are assumed to be disconnected.

Alternatively, a method from Hausman et al. [41] attempts to actively reduce the uncertainty in assigning an articulation model by manipulating the object. The relative pose between objects is used along with a particle filter to choose an 'action' the robot can perform on the object that is most likely to decrease model assignment uncertainties.

2.3.5 Human Articulated Motion Tracking.

Perhaps the biggest application for tracking articulated motion is in tracking the movement of human beings. In these cases, the articulated model and its capabilities/limitations is known; it is the human skeletal system. This is akin to a situation in which the articulated model of a satellite is known and we are attempting to estimate its motion. Due to the similarity with the objectives of the research herein a very brief discussion of the work in human tracking is presented here. For a more thorough discussion on the research in the area the reader is directed to [60] and [70].

Much of the literature on human articulated motion tracking first converts images to silhouettes. The concept of using silhouette images to reconstruct the shape of objects dates back to 1978 when Baumgart [6] used four silhouette images to estimate the shape of numerous objects. Since then, various aspects of Shape-from-Silhouette (also known as Visual Hull construction) have been investigated.[12] In a two part work Cheung and Kanade [12, 13] first outline their method of shape reconstruction using silhouettes from multiple synchronized camera locations [12], then they apply these methods to build a kinematic model of an individual and track its articulated

motion.[13, 11]

Many of the works for human motion tracking use the generic human skeletal model along with limb lengths specific to the individual as the basis for determining the joint articulations (pose).[60] The number of estimated parameters (degrees of freedom, DOF) can vary widely from 10 DOF for methods that track upper body only to over 50 DOF for full body pose tracking.[70] Often a single image (or a set of images) could be explained by multiple sets of articulation angles leading to multiple local minimums. This led many researches to investigate tracking techniques such as the particle filter in which multiple sets of articulation angles are propagated and used to represent the estimated states distribution. The number of particles required (and therefore the computational cost), however, increases exponentially as the number of states to be estimated increases. To combat this curse of dimensionality Deutscher and Reid [28] implement an annealed particle filter (APF) in which the particle weighting function is smoothed and evaluated over multiple layers. At each layer, the smoothing is decreased until the final layer at which point the original weighting function is used. Implementation of this method on a 30 degree of freedom (DOF) model with 400 particles and 10 layers (requiring 4,000 weighting function evaluations) outperformed a standard particle filter evaluation with 40,000 particles. For each particle the articulated model (using truncated cones to represent each limb) is projected to the image plane for each available camera and points are sampled along the edges and within the interior of each limb. The sampled edge points are compared to the edge map and the sampled interior points are compared to the silhouette map to determine the quality (weight) of the particle. Sigal et al. [81] include an implementation of the APF as the baseline algorithm in the comprehensive 'HumanEva' dataset. The data set contains approximately 40,000 frames of synchronized motion capture data and multi-view imagery providing a method of testing human motion

capture methods.

John et al. [46] introduce a method they term hierarchical particle swarm optimization (HPSO) in which they solve for the pose at each instance using a particle swarm optimization technique. Using a truncated cone model with 31 DOF they limit the number of required particles by optimizing in 12 steps. First the body position and orientation are optimized, then the angles of five chains are optimized independently, one joint at a time. No motion model is used, however the results of the previous frame are used to inform the selection of particles for the next time frame. This allows the tracker to self-initialize and recover from local minima.

Chang and Lin [9] introduce the progressive particle filter (PPF) which uses the hierarchical search to decrease the number of required particles to 80 for a 32 DOF model. First they sample and evaluate the weights for the global position of the body, then they perform a iterative mean shift algorithm on each particle to shift it to its local maximum probability and evaluate the particle weights using the measurement. Next they repeat the process for the upper extremity parameters and the lower extremity parameters in turn.

Many of the methods for tracking human pose use images from multiple synchronized cameras. Tracking human pose with a single camera is more challenging [46, 81] in part due to the depth ambiguity associated with monocular vision. Sminchisescu and Triggs [82] proposed a solution that takes this ambiguity into account while employing a particle filter to track human pose. Alternatively, Agarwal and Triggs [4] use known image/pose pairs to train regressors that translate a 100D vector representing the image to a 60D vector representing the pose. More recently, deep learning and neural networks have been used in many computer vision tasks, including human pose prediction.[85]

2.4 Summary

Significant work has been done in recent years in the field of computer vision. It is at the forefront in the effort to increase the autonomy of robots. In general, efforts can be categorized into those that work on the video sequence as a whole and those that incrementally build information about the scene in real-time as images become available. Work has been done in both of these categories to sense and characterize articulated motion.

Batch processing methods use all available data to calculate the articulated structure, camera motion, and kinematic chain. Many of the existing methods use optimization techniques to find the solution that minimizes a cost function, such as the re-projection error.[103, 69]

Real-time methods stem from solutions to the SLAM problem for robots, most of which use some type of Bayesian filter to estimate the statistics of the robot position and the feature point locations. Traditional real-time SLAM methods assume feature points are stationary and discard those that do not meet that criteria. In dynamic environments many of the feature points can be moving, necessitating the inclusion of methods for detection and tracking of moving objects.[96] The two works outlined that estimate articulation parameters in real time use a sensor capable of providing depth information.[49, 58]

While significant of research exist in the area of articulation sensing using computer vision, work has not been done in investigating the applicability of these methods to the space rendezvous and proximity operations (RPO) articulation problem. Based on the literature, the approaches herein leverage existing techniques, particularly [103, 25, 14, 58, 62, 81], and investigate application for space articulation sensing.

III. Building an Articulated Model

3.1 Chapter Overview

This chapter consists of a methodology and results for building an articulated model from simulated feature point locations to meet Objective 1 of this research. The work presented is also available in two conference papers [18, 17]. This chapter consists of an introduction (section 3.2), the method for building an articulated model (section 3.3) with results (section 3.4), a modified method accomodating uncertainty and manuever with results (section 3.5), and a conclusion (section 3.6).

3.2 Introduction

This chapter develops and demonstrates a method of using feature point locations and the satellite inspection trajectory to build an articulated model of the inspected satellite. Feature point locations are simulated by projecting surface points from a nominal articulating satellite to the image plane of an inspecting satellite. These simulated feature points are used along with the known inspection route to build an articulated model consisting of component shapes, joint locations, axes, and range of motion. Performance is assessed using eight evaluation metrics for up to eight articulating joints and under various illumination conditions. A modified method that accommodates primary satellite maneuver, trajectory matrix uncertainty, and inspection route uncertainty is also presented.

3.3 Method

This work focuses on building an articulated model from a set of monocular images taken from an inspection satellite on a known inspection route. Points are spread over the faces of a simulated articulated satellite model and projected to the image plane

of an inspector satellite on a particular inspection route resulting in simulated feature points locations that are consolidated into a trajectory matrix. The points are then segmented into groups that represent rigid body motion and an optimization routine is performed on each group to identify the shape and pose of the rigid body that best describes the feature point locations. Rigid bodies with similar motion are then merged if required. Next a kinematic chain is built that identifies which components are linked to each other. Finally another optimization routine is run that enforces the articulation constraints on linked components. Results are presented for the method with multiple active joints on the articulating arm and investigating the effects of various illumination conditions. Finally, a modified method allowing for primary satellite maneuver, trajectory matrix uncertainty, and inspection route uncertainty is developed and performance demonstrated.

3.3.1 Simulating Data.

To simulate feature point locations, a point cloud that represents the inspected satellite is used. Each of the points in the point cloud is projected to the image frame of the inspector satellite. The 2D location of the projected point in the image plane simulates the location of a feature point in an image. This assumes a feature point tracker, such as a KLT tracker (see section 2.2.4), could acquire and track feature points reliably. Simulating data in this way allows the method to be easily tested under a number of different articulation profiles and inspection routes. The simulated data consists of a trajectory matrix containing the feature point locations of each visible point in the point cloud and the inspection route in the world frame.

The basic steps to building a simulated trajectory matrix are as follows:

1. Create a model of the satellite that includes the desired components and articulated joint information (e.g. polyhedrons with axis orientations and spatial

layout defined).

2. Randomly spread points on the surface of the satellite resulting in a 3D point cloud that represents the satellite.

3. Assign motion to the articulation joint resulting in a 3D point cloud that is time dependent.

4. Select points that are visible to the camera. Certain points will be on faces not visible by the camera while other will be occluded by other parts of the satellite. Additionally, points may not be visible because they are not illuminated.

5. Project the visible 3D points to image coordinates using the camera pose and a pinhole camera model.

To develop the satellite model, a cursory review of a few existing space robotic arms was conducted. Flores-Abad et al.'s [32] review provided an overview of various robotic arms used in space. The Canadarm2 and the European Robotic Arm (ERA) each have seven degrees of freedom, while the Japanese Experiment Module Remote Manipulator System (JEMRMS) has six degrees of freedom. A further review of these examples revealed that none of the joints are prismatic. All the arms consist of numerous single degree of freedom (revolute) joints arranged in series. All of these things were taken into account when developing the model to be used in simulation which is shown in Figure 7. The model has six revolute joints on a robotic arm and two solar arrays, each connected to the main body with a revolute joint. These joints can be actuated or left rigid during simulation, allowing various configurations to be tested.

Points are randomly spread over every face of the satellite model to simulate feature points. To determine if a point p is visible to the camera three vectors are needed: the vector from the Sun to the point $\vec{v}_{s,p}$, the vector from the point to the

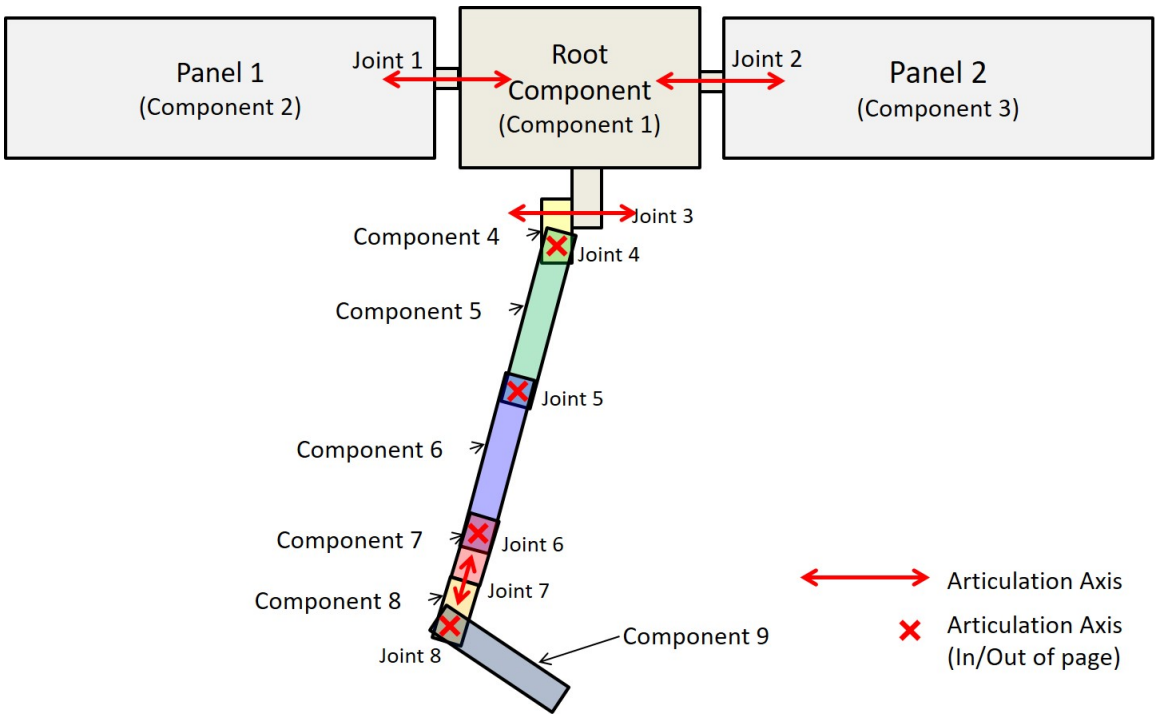

Figure 7. Diagram of simulated satellite model.

camera $\vec{v}_{p,c}$ and the outward facing normal vector for the face $\vec{n}$. The diagram in Figure 8 shows the geometric relationships for three potential situations. The first criteria for a point to be visible is the angle between $\vec{n}$ and $\vec{v}_{p,c}$, termed ϕ, must be less than 90°. This determines if the point is visible to the camera. The second criteria is the angle between n and $-\vec{v}_{s,p}$, termed θ must be less than 90°. This determines if the point is illuminated. Finally, to be visible the vectors $\vec{v}_{s,p}$ and $v_{p,c}$ must not be occluded; in other words they must not pass through any other solid object (such as another part of the satellite).

Since the model consists of planar faces, points on a particular face will appear and disappear together as the angle between the normal vector and the Sun or camera cross the 90° mark. In reality, feature points would likely fade in/out as the illumination or viewing angles approach 90°. To accommodate this, an error function is used to assign a probability of illumination $P(illum)$ or a probability of visibility $P(vis)$ based on θ and ϕ respectively. This probability allows a gradual appearance/disap-

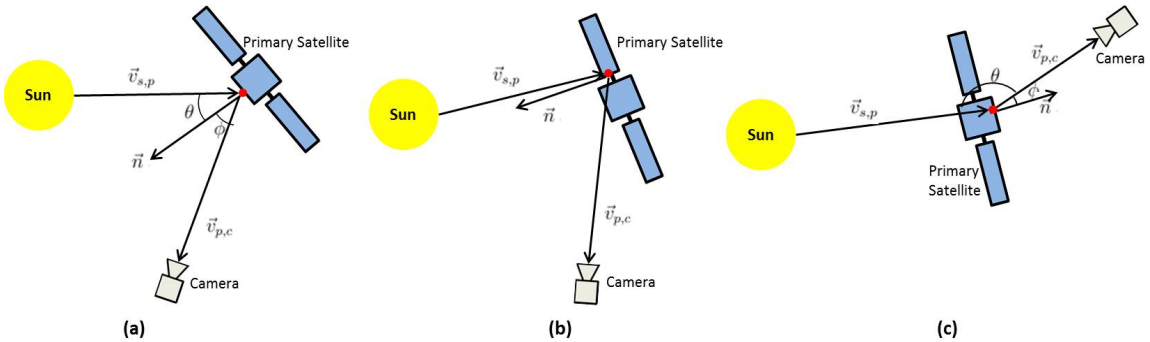

Figure 8. Demonstration of point visibility. (a) Point is visible and illuminated. (b) Point on the solar array is occluded by the main body. (c) Point is visible to the camera, but is not illuminated.

pearance of points as θ and ϕ cross the 90° mark.

For simulation purposes, it is assumed that throughout the trajectory the camera is at a distance from the primary satellite that produces resolved imagery of appropriate resolution for feature point extraction and tracking. In all cases, the primary satellite is simulated to be in geostationary orbit at equinox. The effect of Earth shadowing is not simulated. Note that while the nominal satellite contains panels that resemble solar arrays, no effort was taken to ensure their motion is consistent with a solar array. Both sides of the array may be illuminated at different portions of the route.

3.3.2 Ray Space Optimization.

The ray space optimization technique outlined in Yucer et al. [103] provides an excellent method of estimating 3D shape from a 2D image coordinates when the camera motion is known. The method parametrizes the 3D location of each point (p) in a given frame (i) using the camera center (C_i), the direction vector from the camera center to the point (D_i^p) and the distance along the ray from the camera center to

the point (μ_i^p) using the following equation:

$$S_i^p = C_i + \mu_i^p D_i^p \tag{57}$$

where S_i^p is the 3D location of point p in frame i. Since the points are not stationary there are many valid 3D paths that can result in the same 2D image coordinates, therefore, an added assumption is made that the points track smoothly from frame to frame. This leads to a cost function (equation (58)) that is only a function of the depth μ_i^p at each frame which can be optimized for each point using Matlab's 'fminunc'.

$$\min_{\vec{\mu}^p} \sum_{i=1}^{F-1} \omega_i^p \|(C_i + \mu_i^p D_i^p) - (C_{i+1} + \mu_{i+1}^p D_{i+1}^p)\|^2 \tag{58}$$

The term ω_i^p is a weighting term that is based on the 2D distance of the point in frame $i+1$ from the epipolar line corresponding to the point location in frame i (refer to [83, 40] for more information on epipolar lines) . This weighting method effectively keeps the 3D location of the point when it is moving near its location when it is static.

The ray space optimization technique has a limitation with regard to speed of articulation. Since the ray space optimization routine attempts to find the shortest path that models the observed motion, when the camera motion is less than the articulated motion the solution tends toward a path in which the point follows a trajectory similar to the camera motion rather than the articulated motion. If the ray space optimization results are used for initialization later in the process inaccuracies are compounded.

The distance from the epipolar line can also be used to determine if the point is stationary. With the assumption that the known camera motion is with respect to the main body of the satellite, points that are stationary are points on the main body (or

components that are rigidly attached to the main body throughout the inspection). Points that have an average distance from the epipolar line below a threshold (γ_s) are considered stationary and are segmented to the main body.

Knowing that a point is stationary also allows the calculation of its position to be simplified to a triangulation problem that can be solved using linear least squares. For a stationary point $S_1^p = S_2^p = ... = S_F^p$, therefore equation (57) becomes $S^p = C_i + \mu_i^p D_i^p$. When written for each of the F frames there are $3F$ equations. Since S is no longer different for each frame there are only $F + 3$ unknowns. This set of equations, in equation (59), can be written in the form of $Ax = b$ (equation (60)) and solved with linear least squares (I is a 3×3 identity matrix).

$$\begin{aligned} S^p - \mu_1^p D_1^p &= C_1 \\ S^p - \mu_2^p D_2^p &= C_2 \\ \vdots &= \vdots \\ S^p - \mu_F^p D_F^p &= C_F \end{aligned} \tag{59}$$

$$\begin{bmatrix} I & -D_1^p & 0 & \cdots & 0 \\ I & 0 & -D_2^p & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I & 0 & 0 & \cdots & -D_F^p \end{bmatrix} \begin{bmatrix} S^p \\ \mu_1^p \\ \mu_2^p \\ \vdots \\ \mu_F^p \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_F \end{bmatrix} \tag{60}$$

This method works well in the case where there is no uncertainty in the feature point locations or when the main body is not maneuvering in the world frame. In the presence of noise, or if the main body is maneuvering, however, a simple threshold may not be capable of segmenting points that are stationary.

3.3.3 Segmentation.

One of the most challenging aspects of this problem is segmenting points into groups that represent distinct rigid bodies. The nature of a circumnavigation inspection route (or NMC) yields a trajectory matrix with a significant amount of missing data. Figure 9 shows an example trajectory matrix for a complete NMC where empty elements are shown in black and elements containing data are shown in white. Due to the sparsity of the trajectory matrix there are many points on the same rigid component that do not share any common frames. This makes it difficult to segment them onto the same rigid component. When added to the fact that the number of rigid components is not assumed to be known *a priori*, over segmentation (choosing more point groups than rigid bodies) is necessary.

Figure 9. Example trajectory matrix mask for a complete NMC.

Spectral clustering is a popular method of segmenting data.[93] While there are multiple methods of spectral clustering, they all operate on some type of similarity matrix. For P points, a similarity matrix is $P \times P$ in dimension with each element representing how similar the point corresponding to its column is to the point corresponding to its row. The type of similarity metric used varies widely. Some, such as LSA [100] use a metric based on the angles between the subspaces of nearest neighbors, while others use a metric based on the range and velocity between points.[106] Once the similarity metric is choosen, there are numerous ways of segmenting the data. For this work we investigated both spectral clustering using k-means [67] as well as recursive 2-way spectral clustering.[80, 103] Both methods involve solving for the eigenvectors of the Laplacian matrix. The Laplacian matrix is the similarity ma-

trix minus a diagonal matrix of the row sums of the similarity matrix. The reader is directed to [55] for additional information on spectral clustering, and [93] for an overview of its use in motion segmentation.

The k-means clustering technique requires the number of segments (k) to be given while the 2-way spectral clustering technique used by Yucer et al. [103] only requires an estimate for the number of segments. In testing both methods produced similar results with approximately 90% of points segmented correctly. For this work a similarity matrix based on the range and 2D velocity [30] between image points was used with spectral clustering using k-means [67, 8] and a value of k ranging from 15-30 based on the number of active joints.

3.3.4 Rigid Body Optimization.

Once the points are segmented, the next step is to find how all the rigid bodies are moving. Similar to Yucer et al.[103] we sought the translation, rotation, and shape that minimized the reprojection error and a smoothness penalty.

$$\min_{R,T,\Omega} \underbrace{\sum_{i=1}^{F} \|W_i^n - P_{cam,i}(R_i^{w,b_n}\mathbf{\Omega}^n + \tilde{T}_i^{w,b_n})\|^2}_{\text{Reprojection Error}} + \underbrace{\lambda_{rb} \sum_{i=2}^{F} \left[\arccos(.5(trc(R_i^{w,b_n}(R_{i-1}^{w,b_n})^T) - 1)) + \|T_i^{w,b_n} - T_{i-1}^{w,b_n}\|\right]}_{\text{Smoothness Penalty}} \quad (61)$$

This optimization is performed for each segment. W^n are the columns in the trajectory matrix for the n^{th} segment. $P_{cam,i}$ is the camera matrix that projects points in the world frame to the camera image plane for frame i. Ω^n is the 3D location of the points in the body frame. R^{w,b_n} is the rotation matrix that rotates body frame for segment n to the world frame. $\tilde{T}^n$ is the translation from the origin of the world

frame to the center of the points in Ω^n while T^{w,b_n} is the translation from the origin of the world frame to the origin of the body frame. They are related by:

$$\tilde{T}_i^{w,b_n} = T_i^{w,b_n} - R_i^{w,b_n}\bar{\Omega}^n \tag{62}$$

Using $\tilde{T}^{w,b_n}$ in the reprojection error portion of the cost function helps to decouple the rotation from translation.

There are multiple rigid body shape and motion combinations that can minimize the reprojection error. Since the rigid bodies are most likely to follow smooth paths, the smoothness penalty is added to encourage solutions that follow smooth rotational and translational paths. λ_{rb} is used to weight the smoothness penalty. The appropriate value of λ_{rb} will be dependent on the scaling of the trajectory matrix. Setting it too low will not enforce smoothness and may lead to a jittery path while setting it too high will minimize the rigid body motion at the expense of reprojection error. To allow the method to be robust to scaling, the value of λ_{rb} is chosen adaptively. Before optimization begins, and after every 100 iterations during optimization, the value of λ_{rb} is adjusted so that the smoothness penalty is no more than 25% and no less than 5% of the reprojection error.

The rotation matrix (R^{w,b_n}) is parameterized using an Euler axis ($\vec{a}$) and an Euler angle (ϕ) and represents the rotation matrix from the component's body frame (b_n) to the world frame (w). The reference frames used in this work are shown in Figure 10. The Euler axis is not constrained to be unit length, instead it is normalized before being used to calculate $R^{w,n}$.

This cost function is highly non-linear and contains $(7 \times F) + (3 \times P)$ optimization variables where F is the number for frames and P is the number of points in the component. With 100 frames and 50 points (which is typical for simulations run) this translates to 850 optimization variables. Many optimizers, including Matlab's fmi-

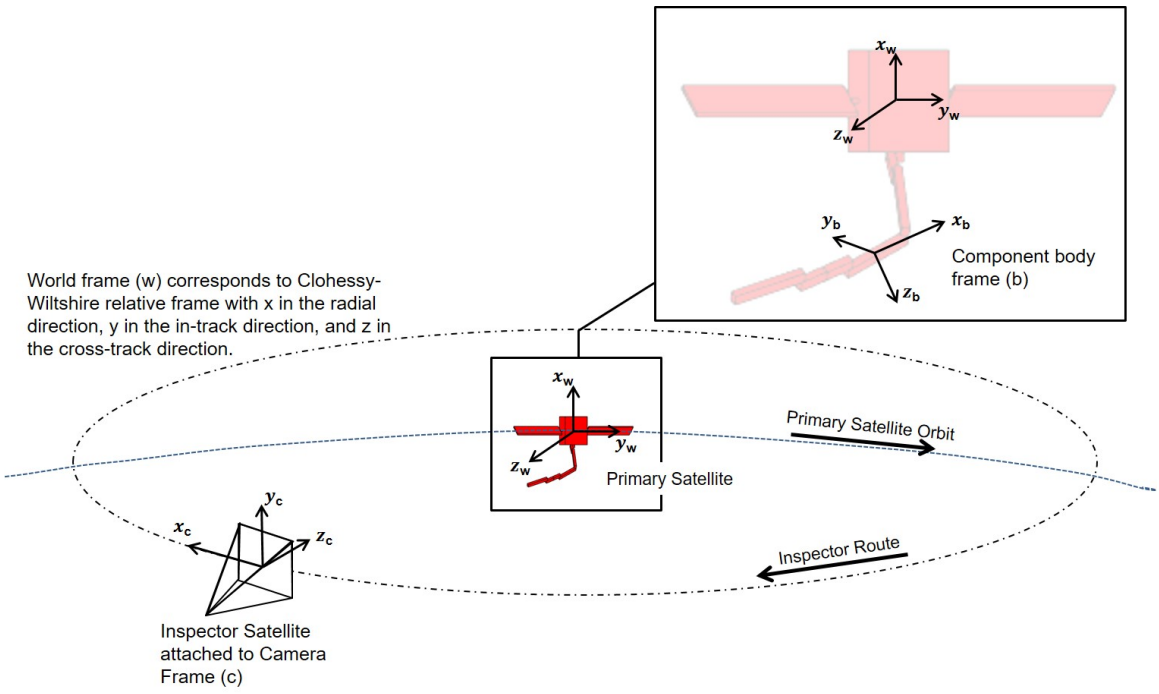

Figure 10. Reference frames used in this work. The 'world frame' (w) corresponds to the CW frame and moves with the primary satellite in its orbit. Each component has a 'body frame' (b) attached to it which is arbitrarily assigned. The 'camera frame' (c) is attached to the inspector camera with the positive z axis along the optical axis and the positive x axis aligned with the camera's direction of travel in the world frame.

nunc which was used in this work, calculate derivatives via finite differences if they are not supplied. This method was used initially, however it was computationally slow since the optimizer needs to compute finite differences onces for each optimization variable to calculate the gradient at each iteration. To speed up computation, analytical derivatives were calculated and supplied to the optimizer. The analytical derivatives corresponding to the work herein are available in Appendix A. When the analytical derivatives are supplied the computational time decreases by approximately 99.7% (the optimization was done ≈ 360 times faster).

To begin optimization, an initial guess must be supplied for all the optimization variables. In many cases, the ray space optimization results provide a good method of initializing the shape (Ω^n) and the translation (T^n). However, as mentioned in section 3.3.2, when the camera rotation is less than the articulated motion there are

issues. If inaccurate ray space optimization results are used, the error propagates through the process resulting in an inaccurate model.

To resolve this issue, another method was developed to initialize the shape and rotation parameters using the scaled orthographic projection model [63]. Using a scaled orthographic camera model, the trajectory matrix can be written as $W = \alpha(R\Omega + T_{2D})$ where α is a scaling based on the distance from the camera to satellite ($\alpha = \frac{f}{|r^{WC}|}$), R is the $2F \times 3$ motion matrix containing the first two rows of the rotation matrix between the camera frame and the body frame at each time step, Ω is the $3 \times P$ shape matrix in the body frame, and T_{2D} is the $2F \times 1$ translation from the world frame origin to the body frame origin projected into the 2D image plane. Since the scale factor (α) and the rotation matrix between the camera frame and the world frame are known from the inspection route, the scaled orthographic projection equation can be used to solve for the resulting shape from any body rotation and translation as follows.

$$\tilde{W} = R\Omega = \tfrac{1}{\alpha}W - \vec{T_{2D}} \tag{63}$$

$$\Omega = R^{\dagger}W \tag{64}$$

$$W_{calc} = \alpha(R\Omega + \vec{T_{2D}}) \tag{65}$$

$$E = \|W - W_{calc}\| \tag{66}$$

The motion matrix R is determined for 5,000 seeds consisting of a randomly selected constant Euler axis $\vec{a}$ and a linear set of Euler angles $\vec{\phi}$ of a random slope. For each R and T_{2D} (calculated from ray space optimizaiton results), the error (E) between W_{calc} and W is calculated. The seed with the lowest error is used to initialize $\vec{a}$, $\vec{\phi}$, and T^{w,b_n}; the resulting Ω is used to initialize shape. This is a highly non-linear optimization problem with numerous local minima. While this method of initialization worked

well for the cases run, alternatives, such as using a heuristic optimization solver for initialization or a global optimization algorithm may produce superior results.

Outlier rejection is conducted every 100 iterations during rigid body optimization. Points with a reprojection error higher than some multiple (γ_f) of the average per point reprojection error and points that have an average range to all other points higher than γ_r standard deviations from the mean range between points are rejected as outliers.

3.3.5 Combine Segments.

Since the data has been over segmented, the next step is to combine (merge) any segments that are part of the same rigid body. To do this, four adjacency matrices that compare the location and rotation of each segment to each other segment are created. If two segments are in fact the same rigid body, the rotations should be the same, therefore the rotation between their body frames ($R_i^{b_1,b_2}$) should be constant for all common image frames.

$$R_i^{b_1,b_2} = (R_i^{w,b_1})^T R_i^{w,b_2} \tag{67}$$

The average variance of the elements of $R_i^{b_1,b_2}$ is used to define the $(1,2)$ position of an $N \times N$ adjacency matrix where N is the number of segments.

It is assumed that segments that are part of the same rigid body are close to one another, therefore an adjacency matrix is created that compares the range between translations in common frames. Segments that are the same rigid body will also remain at the same distance from one another over common frames so a third adjacency matrix is created comparing the distance between components' average location, allowing comparison of segments that do not appear in common frames. A fourth adjacency matrix is created to compare the variance of the range between

the translations. These four adjacency matrices are used to determine which pairs of segments to test for merging.

For each segment, the closest two segments from each of the four adjacency matrices are selected for testing to determine if they should be merged. Rigid body optimization is performed on the combined segments as outlined in the previous section. If the function value is below 150% (γ_c) of the average function values of the segments independently, the two segments are merged and the adjacency matrices are recalculated. This is continued until all segments have been checked without triggering a merge.

3.3.6 Build Kinematic Chain.

The next step is to determine the kinematic chain that best describes which components are linked to each other. This is done by evaluating the average range adjacency matrix mentioned in the previous section. The kinematic chain is the minimum spanning tree of a graph (G) with each component as a node and edges defined by the average distance between components in common frames (CF).

$$G(j,k) = \frac{1}{CF} \sum_{i=1}^{CF} \|T_i^j - T_i^k\| \tag{68}$$

To ensure the root of the kinematic chain corresponds to the main body, the segments are first ordered by their average translation.

The kinematic chain can be expressed as a table of dimension $M \times 2$ where M is

the number of joints ($M = N - 1$) containing parent-child relationships.

$$H = \begin{bmatrix} parent_1 & child_1 \\ parent_2 & child_2 \\ \vdots & \vdots \\ parent_M & child_M \end{bmatrix} \tag{69}$$

H is re-ordered as required to ensure that each segment appears as a child before it appears as a parent. Components that do not share common frames with their parent component are eliminated. Note that this method is not robust to cycles in the kinematic chain.

3.3.7 Articulation Parameter Optimization.

Once the kinematic chain is known, the articulation constraints can be applied. For this application the joints are assumed to be revolute. Yucer et al. [103] use a similar method to the one presented here, however they assume universal joints and do not include the joint locations as optimization variables. The articulation parameters are the articulation axes between linked parts ($\hat{a}$), the angle of articulation (ϕ), and the joint locations (J^p, J^c). For N components, there are $N - 1$ joints. For each joint, p identifies the parent component and c identifies the child component. The w superscript represents the world frame defined by the center of the main body of the primary satellite. The first term of equation (70) is the same as the reprojection error in equation (61), however the rotation and translations are now dependent on the higher components in the kinematic chain. The last term is also similar to the smoothness penalty in equation (61), however due to the articulation constraints the

articulation angle is the only temporally dependent variable.

$$\min_{R,J,\Omega} \sum_{j=1}^{N-1} \Bigg[\underbrace{\sum_{i=1}^{CF} \left[\| W_i - P_{cam,i}(R_i^{w,c}\Omega^c + T^{w,c}) \|^2 \right]}_{\text{Reprojection Error}} + \underbrace{\lambda_{jp} \sum_{k=1}^{12} \ln(1 + e^{d_k \eta})}_{\text{Joint Penalty}} + \underbrace{\lambda_{sc} \sum_{i=2}^{CF} (1 - \cos(\phi_i - \phi_{i-1}))^2}_{\text{Smoothness Penalty}} \Bigg] \tag{70}$$

The rotation matrices and translations are calculated based on the hierarchy of the kinematic chain and the articulation constraints as follows. Define $X = [x_1, x_2, \cdots, x_l]$ as a single chain starting at the root and ending with a component that has no children where each element of X identifies a component and l is the length of the chain. The rotations and translations can be written as follows:

$$R_i^{w,x_k} = \begin{cases} I & , \ k = 1 \\ R_i^{w,x_{k-1}} R_i^{x_{k-1},x_k} & , \text{ otherwise} \end{cases} \tag{71}$$

$$T_i^{w,x_k} = \begin{cases} \mathbf{0} & , \ k = 1 \\ R_i^{w,x_{k-1}} J^p + T_i^{w,x_{k-1}} - R_i^{w,x_k} J^c & , \text{ otherwise} \end{cases} \tag{72}$$

J^p and J^c are the location of the joint in the parent's body frame and child's body frame respectively for the joint between x_k and x_{k-1}. Parameterizing using the joint locations forces linkage between the components and reduces the number of optimization variables.

The rotation between linked frames $R_i^{x_{k-1},x_k}$ is parameterized by two Euler axes, one for the parent (x_{k-1}) and one for the child (x_k), and an Euler angle ϕ_i. Since body frames are chosen arbitrarily during rigid body optimization of each component,

there is no reason the articulation axis would be the same between the body frame of the parent and the body frame of the child. This means the frames must be aligned before the articulation angle is applied. An intermediate frame (aa) is used to align the frames and then a rotation matrix is formed using the articulation axis of the parent ($\hat{a}_p^{x_k}$) and the articulation angle (ϕ_i).

$$\begin{aligned}
R_i^{x_{k-1},x_k} &= R_i^{x_{k-1},aa} R^{aa,x_k} \\
R^{aa,x_k} &= \cos(\psi_a) I + (1 - \cos(\psi_a)) \hat{a}_a (\hat{a}_a)^T - \sin(\psi_a) \hat{a}_a^{\times} \\
\vec{a}_a &= \hat{a}_{x_{k-1}} \times \hat{a}_{x_k} \\
\hat{a}_a &= \frac{\vec{a}_a}{\|\vec{a}_a\|} \\
\psi_a &= \arccos(\hat{a}_{x_{k-1}} \cdot \hat{a}_{x_k}) \qquad (73) \\
R_i^{x_{k-1},aa} &= \cos(\phi_i) I + (1 - \cos(\phi_i)) \hat{a}_{x_{k-1}} (\hat{a}_{x_{k-1}})^T - \sin(\phi_i) \hat{a}_{x_{k-1}}^{\times} \qquad (74)
\end{aligned}$$

The joint penalty in equation (70) 'encourages' the joints to remain close to the point cloud. For each component, a bounding box (BB) containing the points in Ω^n is created and scaled by a factor of 1.5 (γ_{BB}). A soft plus function uses the difference between joint location and the bounding box (d) and a multiplier (η) to penalize the joints (J_p and J_c) if they are outside of the bounding box.

To initialize the optimization for each joint, the information from the rigid body optimization is used. Since each joint is revolute it consists of rotation about a constant axis. A rotation matrix relating the body frame of each component to the world frame is available from rigid body optimization. To find the articulation parameters ($\hat{a}_p$, $\hat{a}_c$, ϕ_i) that most closely match the calculated rotation between components ($R_i^{p,c}$) over common frames, the following cost function is optimized:

$$\min_{M,\phi} \sum_{i=1}^{CF} \arccos(0.5 trace(R_i^{p,c}(M_i^{p,c})^T) - 1) + \lambda_a \sum_{i=2}^{CF-1} (\phi_{i+1} - 2\phi_i + \phi_{i+1})^2 \qquad (75)$$

Algorithm 1 Incremental Joint Addition

1: **procedure** IJA
2: H is table of joints from equation (69)
3: $\mathbf{\Omega} = \{\Omega^1, \Omega^2, \cdots, \Omega^N\}$
4: $f(x)$ =Value of equation (70) at x
5: **while** i $\leq$ M **do**
6: $p = H(i, 1)$
7: $c = H(i, 2)$
8: $x_0 = \left[x_0, \hat{a}^i_{p_0}, \hat{a}^i_{c_0}, \vec{\phi}^i_0, J^i_{p_0}, J^i_{c_0}\right]$
9: Optimize equation (70) for x using constant shape $\mathbf{\Omega}$
10: $x^s_0 = [x, \Omega^c]$
11: Optimize equation (70) for x^s using constant shape for all except component c
12: **if** $var(\vec{\phi}^i) > \gamma_a$ **then**
13: $f(x)_{org} = f(x)\gamma_m$
14: Combine Ω^c and $\Omega^p \rightarrow \Omega^{pc}$
15: $i = i - 1$
16: Run lines 6-11 with Ω^{pc} as c
17: **if** $f(x) < f(x)_{org}(\gamma_m)$ **then**
18: $\Omega_c = \Omega_{pc}$
19: $H(i, 2) = H(i + 1, 2)$
20: Delete $H(i + 1, :)$
21: **end if**
22: **end if**
23: Split $x^s = \left[x, \Omega^{c^*}\right]$
24: $x_0 = x$, $\Omega^c = \Omega^{c^*}$
25: $i = i + 1$
26: **end while**
27: **end procedure**

The first term enforces the single axis of rotation constraint while the second term encourages smooth motion by only penalizing articulation that is not at a constant rate. $R^{p,c}_i$ is calculated from $R^{w,p}_i$ and $R^{w,c}_i$. $M^{p,c}_i$ is calculated from the optimization variables ($\hat{a}_p$, $\hat{a}_c$, ϕ_i) by first aligning the axes and then rotating by ϕ_i as outlined in equation set (74).

As expressed in Yucer et al. [103], the joint location translated into the world frame will coincide for two linked parts.

$$R^p_i J^p + T^p_i = R^c_i J^c + T^c_i, \; \forall i \in CF$$

For components linked by a universal joint there is a single J_p and J_c that best satisfy this constraint. For a revolute joint, any point on the axis will satisfy this constraint.

To narrow in on the joint locations that are closest to each other, the norm of the joint locations is also minimized giving the following cost function.

$$\min_{J_p,J_c} \sum_{i=1}^{CF} \|(R_i^p J^p + T_i^p) - (R_i^c J^c + T_i^c)\|^2 + \lambda_b \left[\|J_p\| + \|J_c\|\right] \tag{76}$$

As the kinematic chain grows, the complexity of the problem increases, as does the number of variables. The number of optimization variables in equation (70) is $((12 + CF) \times (N - 1)) + (3 \times P)$. Each articulation parameter ($\hat{a}_p$, $\hat{a}_c$, ϕ, J_p, J_c) for each joint has an influence on each joint below it in the kinematic chain. This means the optimization variables are highly interdependent resulting in many local minima for equation (70). However, the articulation parameters do not have an influence on the components above the joint in the kinematic chain. Using this relationship, a method termed Incremental Joint Addition (IJA), similar to hierarchical methods in [79, 9, 46]) was developed (see Alg. 1). IJA optimizes for the articulation parameters in order, according to the kinematic chain adding each joint to the problem one at a time and using the results to initialize the next step. Equation (70) is optimized $N-1$ times, each time another joint is added to the chain. This decreases the complexity since each time a set of parameters is introduced into the optimization they only influence the reprojection error for one component. It also provides a method of further combining components that are likely to be the same rigid body. If the variance of the Euler angle between components is low, they are tested and possibly merged. More details of the IJA implementation are shown in Alg. 1. Results using the IJA method as well as optimizing over all joint parameters together (AJP) are presented in section 3.4.1. The IJA method was used for all results in section 3.4.2.

3.3.8 Evaluation Metrics.

Multiple metrics are used to evaluate the results. Because the trajectory matrix is simulated from a known point cloud, the position of each point at each time step is known. This allows calculation of the normalized reconstruction error [30] (E_{3D}) which is direct comparison of the calculated point cloud to the known point cloud in the world frame at each time step. E_{3D} is calculated using equation (116) where $\mathbf{S}_i^{truth}$ are the true world frame point locations of all points calculated in image frame i and $\mathbf{S}_i^{calc}$ are the calculated world frame point locations in image frame i.

$$E_{3D} = \frac{\sum_{i=1}^{F} \|\mathbf{S}_i^{truth} - \mathbf{S}_i^{calc}\|}{\sum_{i=1}^{F} \|\mathbf{S}_i^{truth}\|} \times 100\% \tag{77}$$

The calculated kinematic chain may contain more or fewer joints than the truth model. Once the calculated joints are matched to the most appropriate truth joint the articulation parameters can be compared. However, the articulation parameters of the calculated model are in a body frame that is unrelated to the body frame used to create the truth model. Therefore, the articulation axis error ($E_{\hat{a}}$) and joint error (E_J) are calculated in the world frame (w) as follows, where N is the number of joints, and F is the number of frames with data. Since the joints are revolute, any location on the axis is correct, so the joint error is measured as the distance from the joint to the line represented by the true joint location and articulation axis in the world frame.

$$E_{\hat{a}} = \frac{1}{N} \sum_{j=1}^{N} \left[\frac{1}{F_j} \sum_{i=1}^{F_j} \arccos(|(\hat{a}_{truth,w}^j)^T \hat{a}_{calc,w}^j|) \right] \tag{78}$$

$$E_J = \frac{1}{N} \sum_{j=1}^{N} \left[\frac{1}{F_j} \sum_{i=1}^{F_j} \|\hat{a}_{truth,w}^j \times (J_{calc,w}^j - J_{truth,w}^j)\| \right] \tag{79}$$

The articulation angle is evaluated using the error in the range of angles covered

$E_{\phi_{range}}$ and the rate of change of the angle $E_{\dot{\phi}}$.

$$E_{\phi_{range}} = \frac{1}{N}\sum_{j=1}^{N} |\Delta^j_{truth} - \Delta^j_{calc}| \tag{80}$$

$$\Delta^j = max(\vec{\phi^j}) - min(\vec{\phi^j}) \tag{81}$$

$$E_{\dot{\phi}} = \frac{1}{N}\sum_{j=1}^{N}\left[\frac{1}{F_j}\sum_{i=2}^{F_j} |\dot{\phi}^i_{truth} - \dot{\phi}^i_{calc}|\right] \tag{82}$$

$$\dot{\phi}^i = \phi^{i-1} - \phi^i \tag{83}$$

In addition to evaluating the results directly, it may be interesting to understand the satellite appendage's range of motion, or more generally, what area around the satellite are the appendages capable of reaching. This can be termed the satellite's workspace. To calculate the satellite's workspace, the space around the satellite is discretized into a grid of $m \times m \times m = M$ points. Each joint is moved through its range of articulation angles at Y increments. A convex hull is created encompassing each shape in the world frame in the current increment and the previous increment. The grid points in M that fall within the convex hull [27] are annotated as covered. The collection of points in M that are covered represents the workspace of the satellite. The complexity of this calculation scales exponentially with the number of increments and the length of the longest portion of the kinematic chain K. If the computation time for a single increment of a 2 link chain is Q then Q scales as Y^K. This means a 6 link chain incremented 10 times requires computation time of $10^6 Q$. The computational requirement can be large, but can be managed by using only a handful of increments. Both the calculated workspace and the true workspace are found in this manner. They are compared using three percentages. P_{wc} is the percentage of the truth workspace covered by the calculated model, P_{woc} is the percentage of additional space covered by the calculated model, and P_{gc} is the percentage of the grid points where the true workspace matches the calculated workspace. These values are calcu-

lated as follows where B is an $M \times 1$ vector with $B_i = 1$ when grid point i is covered and $B_i = 0$ when grid point i is not covered. $\odot$ denotes element-wise multiplication.

$$P_{wc} = \frac{\sum B^{calc} \odot B^{truth}}{\sum B^{truth}} \tag{84}$$

$$P_{woc} = \frac{\sum B^{calc} \odot |B^{truth} - 1|}{\sum B^{truth}} \tag{85}$$

$$P_{gc} = \frac{\sum 1 - |B^{calc} - B^{truth}|}{M} \tag{86}$$

Note that the workspace calculation does not consider physical limits to motion imposed by other components. In other words, two components may occupy the same physical space during some combination of motions.

3.4 Results

3.4.1 Initial Assessment.

The algorithm performance was measured with various joint activity using a 2×1 NMC [78] in geostationary orbit as the inspection route. The selected NMC was 2,000 units (in-track) by 1,000 units (radial) when projected into the orbital plane with ± 50 units of cross track and was phased for the best illumination conditions (see Figure 12). The results presented were conducted under winter solstice illumination conditions, however trials were also conducted under summer solstice and equinox illumination with similar results. Figure 7 shows a schematic of the simulated satellite and its articulating joints. In all cases, the panels articulate linearly from -1.3 radians to 1.3 radians. Joints in the appendage that are active articulate linearly from -0.94 radians to 0.94 radians.

Two methods of articulation parameter optimization were evaluated: 1) optimizing over all joints together (AJP) and 2) incrementally adding joints (IJA). AJP results and IJA results are comparable for most metrics and for most trials shown

Table 1. Results for Single NMC at winter solstice.

Input/Truth			Results									
AP method	# of Components	# of Comp. in arm	# of Components	# of Comp. in arm	E_{3D} (%)	P_{wc} (%)	P_{woc} (%)	$E_{\hat{a}}$ (deg.)	E_J (units)	$E_{\Delta\phi}$ (deg.)	$E_{\dot{\phi}}$ (deg. per frame)	Time for AP Opt (s)
AJP	3	1	8	3	0.29%	80%	9.4%	0.038	0.0015	0.057	0.0036	678
IJA	3	1	6	1	0.30%	94%	8.2%	0.21	0.0044	0.015	0.0028	6,213
AJP	4	2	10	5	0.72%	97%	8.9%	1.66	0.059	1.32	0.063	646
IJA	4	2	7	2	0.31%	97%	0.82%	0.26	0.0044	0.38	0.0056	7,287
AJP	5	3	10	4	0.21%	96%	0.87%	0.48	0.013	2.35	0.028	969
IJA	5	3	8	3	0.25%	97%	0.9%	0.33	0.014	2.46	0.030	1,163
AJP	6	4	11	4	0.39%	97%	1.4%	0.74	0.024	2.92	0.044	1,280
IJA	6	4	9	4	0.32%	97%	0.87%	0.57	0.016	2.29	0.039	1,000
AJP	7	5	13	6	0.81%	92%	1.6%	2.46	0.046	4.29	0.086	1,881
IJA	7	5	11	5	0.89%	93%	1.5%	3.55	0.047	3.61	0.080	2,466
AJP	8	6	15	9	1.2%	63%	5.4%	8.59	0.051	9.17	0.15	4,427
IJA	8	6	11	6	0.82%	93%	0.70%	4.24	0.031	8.02	0.092	3,949

in Table 1. While there are cases when each performs better, there are no instances when AJP performs significantly better than IJA. There are instances, such as when all 6 joints are active, when the IJA performs significantly better than IJA. The ability of IJA to merge components contributes to its improved performance and allows the number of components to be closer to truth than AJP in nearly all cases. These advantages may be worth the added computational cost, particularly as the number of joints increases.

The articulation parameter metrics ($E_{\hat{a}}$, E_J, $E_{\Delta\phi}$, and $E_{\dot{\phi}}$) shown in Table 1 are calculated for only the joints on the appendage. The algorithm is able to characterize the articulation parameters when all 6 joints are active, however the average error per joint increases as the length of the kinematic chain increases. Detailed plots of each trial are not presented, however the results with four active joints on the articulating appendage are presented in Figure 11.

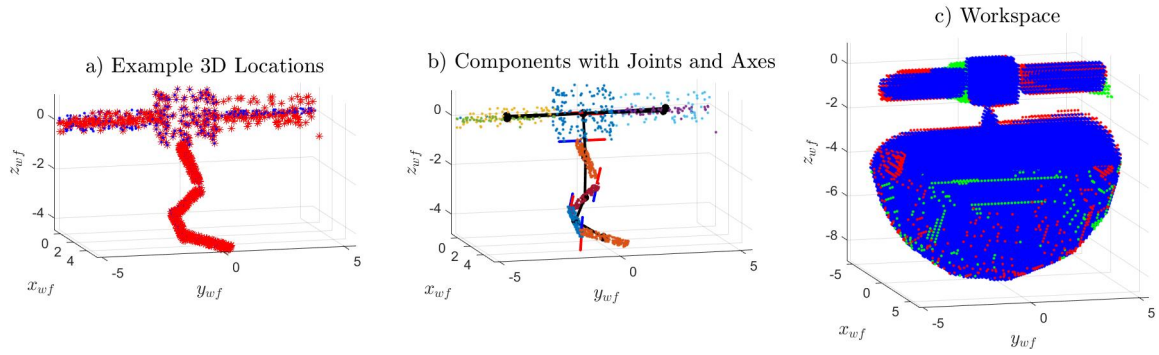

Figure 11. Example results. a) Truth (blue ·) and calculated (red ∗) 3D position of points for an example frame. b) Calculated points, kinematic chain, joint locations, and axes for an example frame. Points in each component are a different color. c) Grid positions in both true and calculated workspace (blue ·), Grid positions in only true workspace (red ·), Grid positions in only calculated workspace (green ·).

Table 2 shows the approximate computational time for this method. This is for a simulation with 100 frames and 1,100 points with 7 active joints for a total of 8 components. The majority of the computational time is spent in the combining segments section of the algorithm.

3.4.2 Illumination and Percentage of NMC.

For the simulations in this section the two panels and joints 3, 5, and 6 on the articulating arm (see Figure 7) were activated. Inspection routes with sub-optimal lighting conditions were investigated. To minimize variation, the motion of the satel-

Table 2. Computational Time.

Section	Approximate Computational Time
Simulate Data	16 sec
Ray space optimization	3.5 min
Segmentation	1 min
Rigid body optimization	31 min
Combine Segments	4.3 hrs
AP optimization (AJP)	6 min
AP optimization (IJA)	35 min
Total	5-5.5 hrs

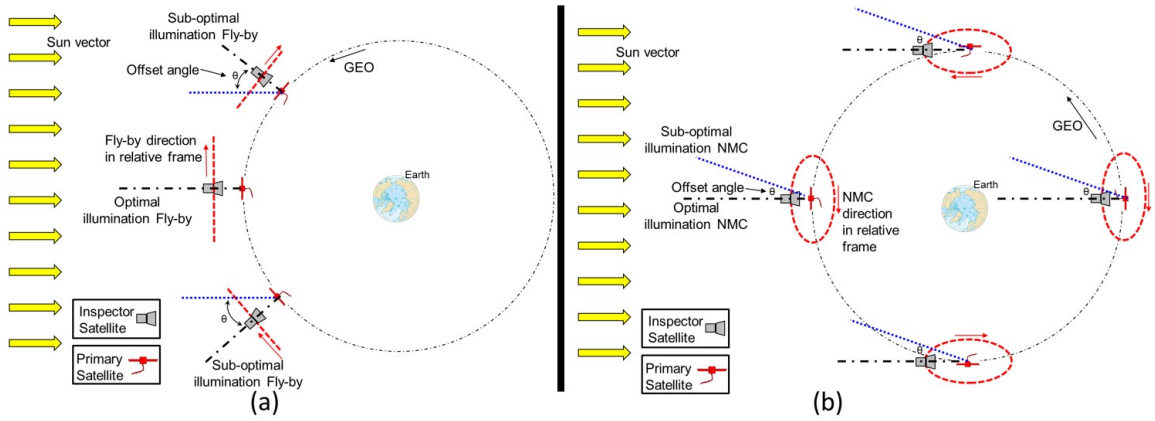

Figure 12. Inspection route diagrams illustrating the illumination offset parameter (θ). a) Fly-by inspection route. b) NMC inspection route.

lite is the same in all cases. Over the length of the simulation, joints 1 and 2 articulate linearly from -1.26 radians to 1.26 radians while joints 3, 5, and 6 articulate linearly from -0.94 radians to 0.94 radians.

Two types of inspection routes are investigated in this work: a 2×1 NMC,[78] and a fly-by in which the inspector route consists of a straight line path in the relative frame. In each case, the route can be phased for the 'best' illumination conditions or it can be offset by some illumination offset angle (θ). The 'best' illumination conditions are assumed to be those in which the primary satellite is most illuminated as viewed by the inspector satellite, however, due to the nature of the space environment, even these illumination conditions are sub-optimal as compared to a terrestrial application in which an object can be lit from all sides. Figure 12 shows the meaning of θ for fly-by routes and NMCs. Fly-by routes consist of the inspector satellite moving past the primary satellite on a linear path in the relative frame that is parallel with the in-track direction. The route is simulated to occur over 1 hour (assuming the primary satellite is in a 24 hour circular orbit). In this case, θ is related to the location in the orbit where the fly-by occurs. For an NMC the zero illumination offset condition ($\theta = 0$) represents a route in which the phasing of the NMC is such that the inspector

is always between the primary satellite and the Sun. In this case, θ represents sub-optimal phasing of the NMC with the primary satellite orbit.

Figure 11 shows an example of the results available from each trial. There are numerous aspects of this method that are non-deterministic such as the placement of points in the truth model, the segmentation algorithm, and the rigid body optimization initialization method. Therefore, numerous simulations were conducted to evaluate the effect of the illumination offset angle on the evaluation metrics. Results for the fly-by inspection route are shown in Figure 13. Each plot shows a fourth-order polynomial fit along with bounds containing at least 50% of predicted values calculated using Matlab's polyfit command. While some metrics ($E_{\phi_{range}}$, $E_{\dot{\phi}}$, and P_{woc}) do not seem to be effected by the illumination offset angle, most metrics begin to degrade around $\theta = \pm 60°$. This is consistent with when the percentage of visible points begins to decrease as shown in Figure 14.

Trials were also conducted for full NMCs with various illumination offsets. The results are shown in Figure 15. As with the fly-by, the results begin to diminish as the illumination offset approaches $\theta = \pm 60°$. This is again consistent with the decrease in the percentage of visible points as shown in Figure 14. This suggests that for these two types of inspection routes the percentage of visible points corresponds to the quality of the results. However, the overall results for the fly-by are better than the overall results of the NMC for most metrics even though the NMC views a much higher percentage of points. This suggests that generally there is a point at which a higher percentage of visible points does not produce better results. This is likely because more points being visible results in a trajectory matrix with more missing data which adds complexity to the 'Combine Segments' portion of the method (section 3.3.5).

To further investigate this, additional simulations were run in which the percent-

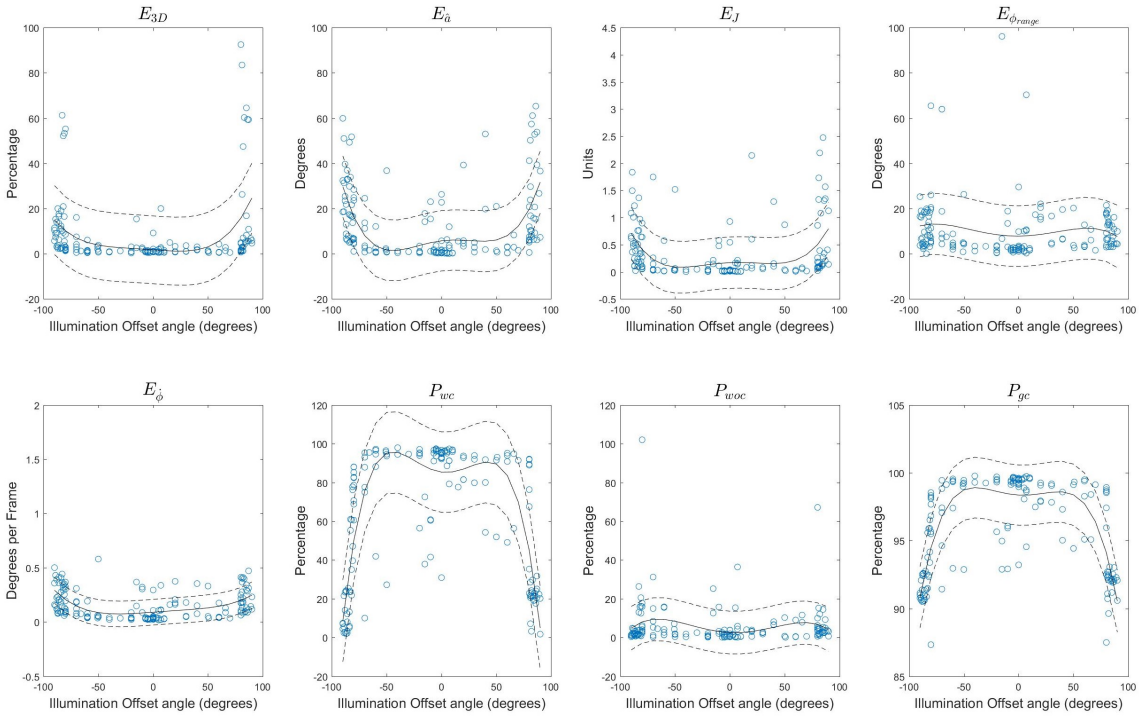

Figure 13. Fly-by trial results for each evaluation metric (outlined in section 3.3.8). Blue dots represent the value for each trial, black lines are a 4th order polynomial fit to the data, and dashed lines contain 50% of future predictions

age of the NMC completed was varied along with the illumination offset angle. As the percentage of the NMC increases, more points will be visible, but the amount of missing data in the trajectory matrix will also increase. More visible points means more information is available about the overall shape of the primary satellite, however, more missing data complicates the segmentation process and the combination of segments. This suggests there is limit at which more data is not beneficial to the success of the algorithm.

Results are shown in Figure 16. For ease of evaluation and presentation, a single metric (Q_{all}) is used to quantify the results from each simulation. Q_{all} is the summation of the normalized metrics described in section 3.3.8. The percentage metrics (P_{wc}, P_{woc}, P_{gc}) are normalized by their maximum while the error metrics (E_{3D}, $E_{\hat{a}}$, E_J, $E_{\phi_{range}}$, $E_{\dot{\phi}}$) are normalized by their maximum and subtracted from 1 so a better

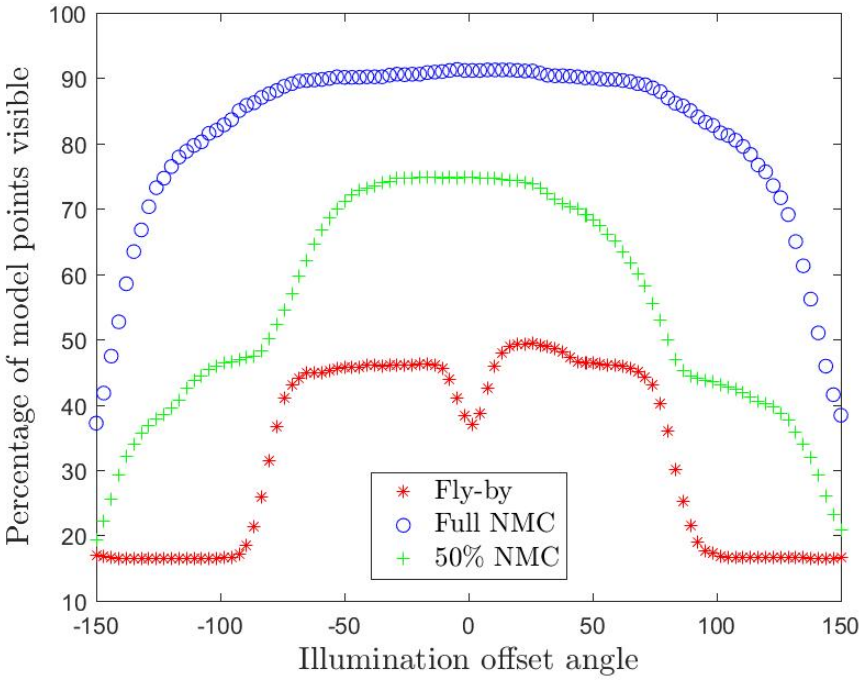

Figure 14. Percentage of truth model points that are visible during the inspection route.

model is represented by a larger number. The contour represents a surface created by a fourth-order polynomial in each of the test variables fit to the Q_{all} metric. The points represent each of the test points with a color corresponding to the value of Q_{all} for that simulation.

After 50% of the NMC is completed, the results suggest that additional data causes an overall decreasing trend in the quality of the resulting model. This suggests that if the inspection route covers more than 50% of the NMC, only the data from the first half of the NMC should be used to create the model. Any additional data could be processed using some type of model refinement method in which new points and frames are appropriately added to the existing model. The results also suggest that an inspection route limited to 50% of an NMC may be as suitable as a longer route when creating a model.

Table 3. Parameters. Method of determination: NA = Noise analysis; PPK=Prior problem knowledge; CFR=Cost function ratios; PD=Point density; RA=Risk assessment

Parameter	Name/description	Value	Method of determination
γ_s	Stationary threshold: determines when a point is considered stationary	1^{-6}	NA
k	Number of segments: determines the number of segments for spectral clustering. (sec. 3.3.3)	10 or 22	PPK
$\gamma_{sc+}, \gamma_{sc-}$	Ratio of smoothness penalty to reprojection error penalty. (sec. 3.3.4)	0.25, 0.05	NA
γ_f	Reprojection error fair share multiplier for outlier rejection. (sec. 3.3.4)	8	PD
γ_r	Range rejection multiplier for outlier rejection. (sec. 3.3.4)	4	PD, PPK
γ_c	Segment merge criteria: allowable increase in function value for acceptable merge. (sec. 3.3.5)	2	RA
γ_{BB}	Joint bounding box expansion: determines the size of the bounding box around the component shape. (sec. 42)	0.5	PPK
η	Coefficient in joint penalty. (eqn. (70))	100	CFR
λ_a	Rotation parameter initialization smoothness weight. (eqn. (75))	5	CFR
λ_b	Joint location initialization closeness weight. (eqn. (76))	5	CFR
γ_a	Minimum articulation angle mean rate. (IJA algorithm)	0.007 rad. per frame	RA
γ_m	Segment merge criteria: allowable increase in function value for acceptable merge. (IJA algorithm)	5	RA

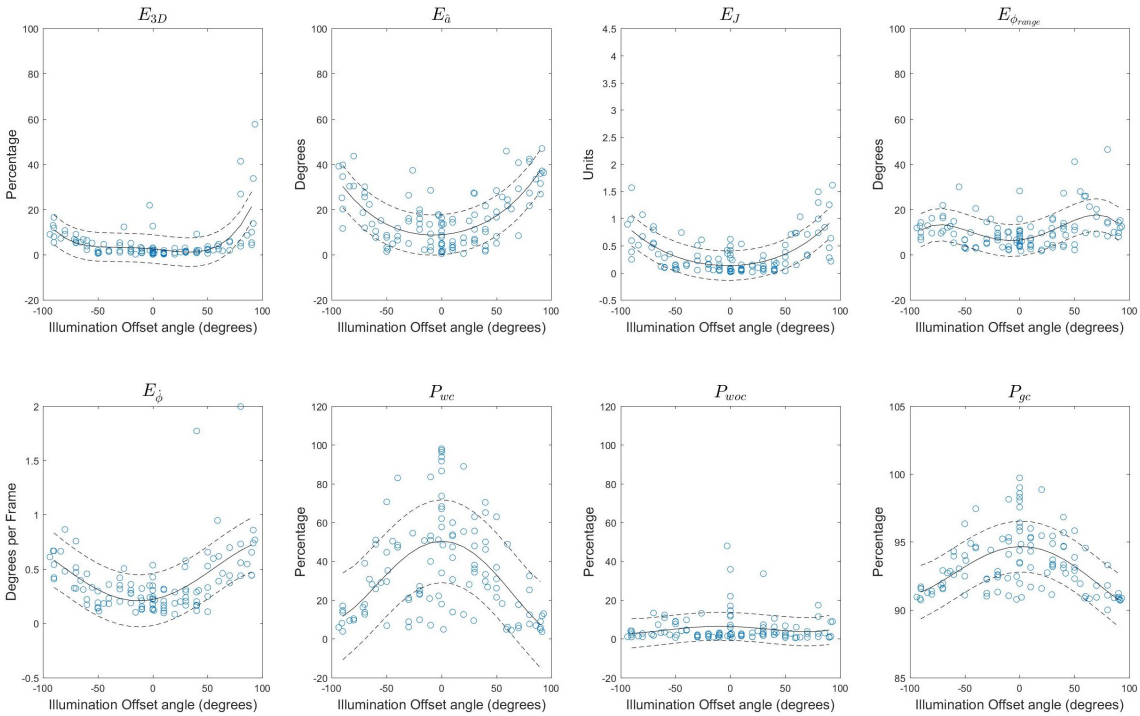

Figure 15. Full NMC trial results for each evaluation metric (outlined in section 3.3.8). Blue dots represent the value for each trial, black lines are a 4th order polynomial fit to the data, and dashed lines contain 50% of future predictions

3.4.3 Limitations.

This method is capable of characterizing satellite articulation, however limitations exist.

- There are a number of parameters that require adjustment based on the specifics of the problem. These parameters are shown in Table 3 along with the method of determining an appropriate setting. Noise analysis (NA) means that the best parameter setting is based on the amount of uncertainty in the trajectory matrix and the inspector satellite route. Prior problem knowledge (PPK) means that some information regarding the particular problem, such as an estimate on the number of components, should be used to select an appropriate value. Cost function ratios (CFR) means that the ratio of the different terms in the cost function should be used to scale the value appropriately. Point density

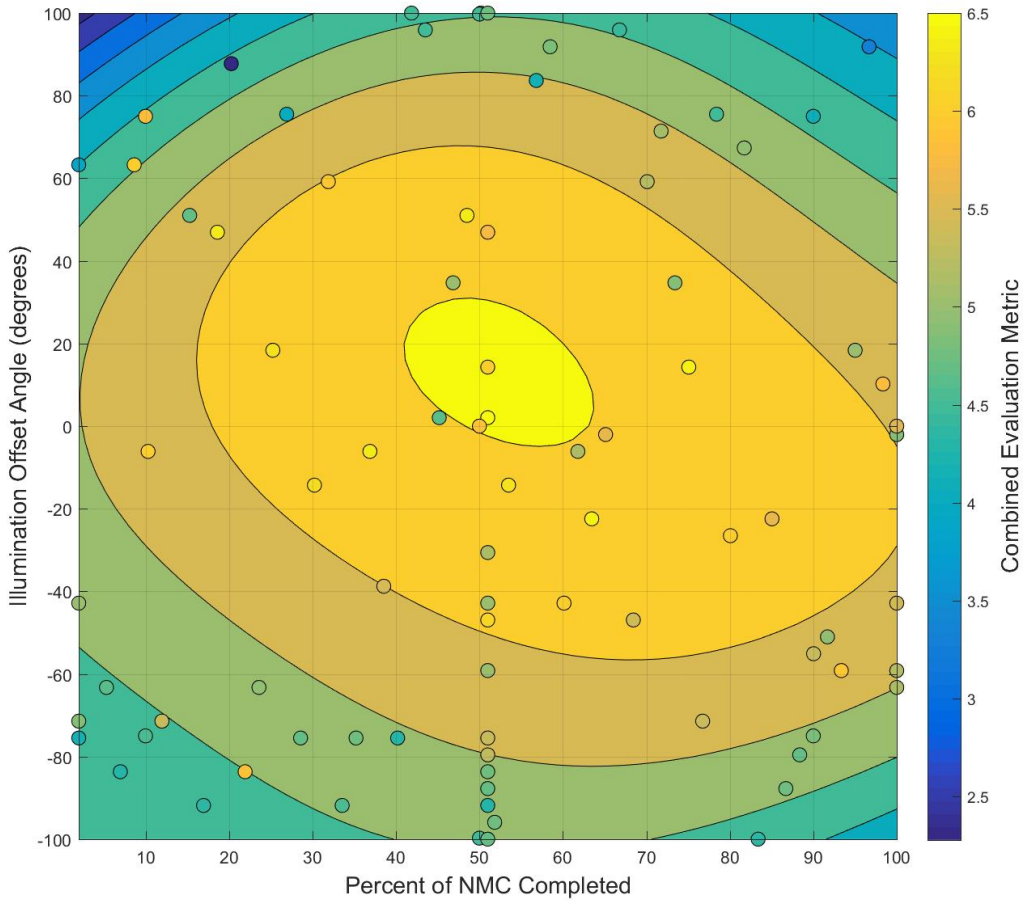

Figure 16. Results from Illumination offset angle and Percentage of completed NMC testing. Contour plot represents a 4th order polynomial fit of both experimentation variables to the equally weighted combination of all evaluation metrics. Dots represent each trial colored by the value of the combined metric at that point.

(PD) means the number of points found/tracked should dictate the criteria for removing points. Risk assessment (RA) means the risk in having more or less than the correct number of components should be assessed.

- The current implementation only calculates parameters for a component when the component is visible and when all components above it in the kinematic chain are visible. This could be highly limiting for certain cases in which a component is shadowed or occluded for a significant number of frames.

- As currently implemented, this method requires the main body of the satellite to be motionless in the world frame. This limitation is investigated in the next section.

- It is assumed that the motion of the inspector satellite is known. While this is likely to be the case generally, the exact relative position may not be known. Accommodating uncertainty in the inspection route is investigated in the next section.

- Real imagery and the challenges of feature point recognition and tracking in representative space illumination conditions have not been investigated in this work.

3.5 Accommodating Main Body Maneuver and Uncertainty

Two of the main limitations of the method outlined above are that the main body must remain stationary in the world frame and that the inspection route must be known perfectly. Since the location of the world frame is relative, these two limitations are related. At first glance, it seems as though removing this limitation would be straight forward; simply add additional optimization variables that account for the rotation and translation of the main body with respect to the world frame. Attempts to implementing this solution however were unsuccessful, likely because of two issues: 1) inaccuracies in segmenting points on the main body, and 2) inaccuracies in calculating the maneuver of the main body. In the method outlined above, the main body points are segmented to the main body because they have small distances to the epipolar line between temporally adjacent frames. With trajectory matrix uncertainty, inspection route uncertainty, or a maneuver of the main body it is difficult to segment points in this way, therefore they must be segmented with the remaining

points using spectral clustering. None of the similarity matrices and spectral clustering techniques tested were able to segment the full trajectory matrix (including the main body) with enough accuracy for success.

Additionally, a maneuver of the main body means that the main body shape and pose must be solved using the rigid body optimization process which contains many local minima. This issue can be overcome in the previous method because inaccurate shape/pose solutions can be rectified further in the process when they are constrained to the main body. If, however, the main body is maneuvering, and therefore its motion must be solved using rigid body optimization, there is no way to stop a poor solution for the main body pose from adversely effecting everything below it in the kinematic chain.

Removing these two issues, the method as previously outlined is capable of building an articulated model. To demonstrate this, a 30% NMC test case was run in which noise was added to the trajectory matrix and to the inspection route. Zero mean white Gaussian noise sampled from $\mathcal{N}(0, \eta_{TM}^2)$ where η_{TM} was set to 0.5% of the overall standard deviation of all elements in the trajectory matrix was added to each element of the trajectory matrix. For the inspection route, noise was added to the pointing angle of the camera by multiplying R_i^{CW} by a rotation matrix created from a randomly sampled Euler axis and a Euler angle sampled from $\mathcal{N}(0, \eta_\theta^2)$ where $\eta_\theta = 1.6e-5$, and noise was added to the camera location r_i^{CW} by adding a 3×1 translation sampled from $\mathcal{N}(0, \eta_r^2)$ where $\eta_r = 0.01$. The points belonging to the main body were removed (using the true segmentation) and their shape was solved using linear least squares as described in section 3.3.2. The rest of the algorithm was run as described in section 3.3. Results are shown in Figure 17 and Table 4. These results are comparable to those shown in the section 3.4.2.

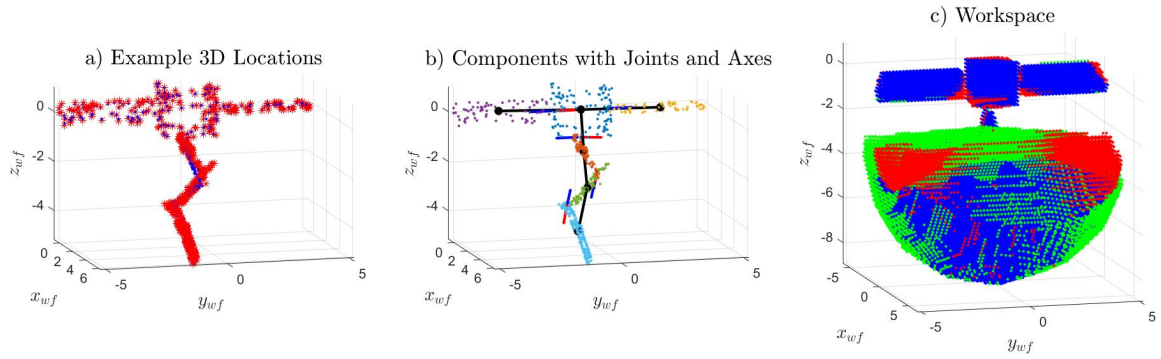

Figure 17. Results with trajectory matrix and inspection route noise. a) Truth (blue ·) and calculated (red ∗) 3D position of points for first frame. b) Calculated points, kinematic chain, joint locations, and axes (truth in blue calculated in red) for first frame. Points in each component are a different color. c) Grid positions in both true and calculated workspace (blue ·), Grid positions in only true workspace (red ·), Grid positions in only calculated workspace (green ·).

Table 4. Results for 30% NMC with known main body segmentation, trajectory matrix noise, and inspection route noise.

E_{3D}	P_{wc}	P_{woc}	P_{gc}	$E_{\hat{a}}$ (deg.)	E_J (units)	$E_{\Delta\phi}$ (deg.)	$E_{\dot{\phi}}$ (deg./frame)
1.3%	95.0%	7.6%	98.8%	4.6	0.12	9.8	0.16

3.5.1 Modified Model Development Method.

In developing a method that works with main body maneuver, a few things were taken into account. First off, a reliable segmentation method was desired. The LSA method of [100] provides a reliable segmentation, however it requires a full TM with no missing data. Secondly, a more realistic articulation scenario was desired. Information available on the Canadarm2 [66] aboard the International Space Station suggests that operation over the articulation angle range in section 3.4.1 would occur over approximately 28 minutes, which equates to 2% of an NMC at GEO. Therefore, for development and testing of this method an inspection route consisting of 2% of an NMC was used. Both panels and three joints on the arm were articulated linearly as described in section 3.4.2.

The first step of the modified method was to employ motion segmentation on the trajectory matrix which was simulated in the same manner as outlined in section

3.3.1. The LSA method of motion segmentation builds a $P \times P$ affinity matrix based on the angle between the local subspaces of each point combination as described in section 2.3.2. This method requires a full trajectory matrix, with no missing data. The trajectory matrix generated from a 2% NMC route consists of 22% missing data, so to use the LSA segmentation method points viewed in less than 95% of frames were eliminated. Empty entries within the remaining columns were filled with the nearest value. This produced a full trajectory matrix that was segmented using LSA. Note that the number of components was also supplied to the LSA algorithm. This provided segmentation into six components with approximately 97% of points assigned to the correct component.

Since the satellite was viewed at an angle from which the components are not overlapping, the trajectory matrix itself provides enough information to build a kinematic chain. An adjacency matrix was created by comparing the average distance between the points in corresponding groups. The adjacency matrix H entries are calculated as follows using Matlab notation where W' is the trajectory matrix, i_n are the indices of the n^{th} component, and p_n is the number of points in the n^{th} component. The minimum spanning tree of H is the kinematic chain.

$$H(j,k) = \frac{1}{p_j p_k} \sum_{q=1}^{p_i} \sum_{r=1}^{p_j} \|W(:, i_j(q)) - W(:, i_k(r))\| \tag{87}$$

With a kinematic chain and segmentation, the root component can be identified. Rigid body optimization as described in section 3.3.4 may produce an incorrect local minimum. This is exacerbated since the camera's angular motion relative to the world frame is small in the 2% NMC scenario. To combat this problem, the rigid body optimization of the root component (containing the maneuver of the primary satellite) was paired with the solution of the articulation parameters attached to the

root component. The root component's rigid body motion was initialized with 20,000 seeds using the scaled orthographic camera structure from motion method described in section 3.3.4. For the 30 seeds with the lowest error, rigid body optimization was performed along with articulation parameter optimization for each of the components attached to the root component using the IJA method outline in section 42 and a particle swarm optimization (PSO) routine to initialize the articulation parameters. The IJA method was then continued for the remaining joints using the best solution of the 30 seeds. An overview of the method is shown in Algorithm 2.

To accommodate noise in the trajectory matrix, a change was made to the overall reprojection error cost functions (equations (61) and (70)). Adding noise to the trajectory matrix means that the feature point locations no longer represent rigid body motion. The optimizer, however, still attempts to drive the reprojection error to zero. Since zero reprojection error no longer represents rigid body motion, errors can occur. To fix this issue, the reprojection error portion of equations (61) and (70), D in the equations below, was adjusted to set the cost function penalty to zero when the error is below the anticipated uncertainty in the trajectory matrix (σ_{TM}).

$$G_{i,j} = W_{i,j} - P_{cam}(R_{i,j}\mathbf{\Omega_j} + T_{i,j}) \tag{88}$$

$$D = \sum_{i=1}^{F}\sum_{j=1}^{M}(I_{i,j}(|G_{i,j}| - 3\sigma_{TM}))^2 \tag{89}$$

$$\frac{\partial D}{\partial G_{i,j}} = 2(I_{i,j}(|G_{i,j}| - 3\sigma_{TM}))sgn(G_{i,j}) \tag{90}$$

In the equation above, F is the number of frames, M is the number of points, the subscript i, j on R and T refers to the appropriate rotation and translation, and I is an indicator function that is set to zero when $|G_{i,j}|$ is less than $3\sigma_{TM}$ and one when it is greater than $3\sigma_{TM}$. The gradient $\frac{\partial D}{\partial G_{i,j}}$ is also shown above. Note that

Algorithm 2 Modified Model Development Method

procedure MODIFIED MODEL DEVELOPMENT METHOD
H is table of joints representing the kinematic chain
for i=1:20,000 **do**
x_i=random selction of an angular rate, starting translation, and translational rate
$[f_i, \Omega_i^{rc}]$=orthographicSfM(x_i)
end for
idx=indices of 30 lowest values in f
jts=rows of H attached to root component
for i=1:30 **do**
$\tilde{x}_i$=RBOpt(x_i,Ω_i^{rc})
for j=jts **do**
$\tilde{x}_i$=[$\tilde{x}_i$, initializeAP($\tilde{x}_i$,H(j,:))]
$[\tilde{x}_i, \tilde{f}_i]$=articulationParameterOptimization($\tilde{x}_i$)
end for
end for
idx=index for minimum value of $\tilde{f}$
$x_{ap} = \tilde{x}_{idx}$
for i=remaining joints in H **do**
x_{ap}=[x_{ap}, initializeAP(x_{ap},H(i))]
x_{ap}=APOptimization(x_{ap})
end for
Extract articulation parameters $(R^{wn}, T^{wn}, \hat{a}^m, J_p^m, J_c^m, \phi^m, \Omega^n)$ from x_{ap}
end procedure

---

procedure ORTHOGRAPHICSFM
Translate x to a rotation R and a translation T at each frame
Use R and T to solve for the error using equations (63) - (66)
end procedure

---

procedure RBOPT
Translate x and Ω to state vector as shown in section 3.3.4
Minimize equation (61) with Matlab's fminunc
end procedure

---

procedure INITIALIZEAP
$x = \begin{bmatrix} \hat{a}_p, & J_p, & J_c, & \dot{\phi} \end{bmatrix}$
Use Matlab's PSO to minimize error from orthographicSfM(x)
Calculate appropriate articulation parameters and child shape for joint
end procedure

---

procedure APOPT
Translate x and Ω to state vector as shown in section
Minimize equation (70) with Matlab's fminunc
end procedure

---

this function has a continuous, but no longer smooth gradient, and therefore does not have a continuous second derivative. The gradient is supplied to the optimization solver (Matlab's fminunc) however the Hessian is approximated numerically at each

iteration using the built-in BFGS method. The discontinuity in the second derivative of the cost function did not cause issues with this implementation, however care should be taken using this cost function with other solvers, particularly if the Hessian is calculated in a different way.

3.5.2 Example Results with Moving Main Body and Noise.

As discussed previously a 2% NMC inspection route was used for development of this method. Since the world frame is arbitrary and noise was added to the inspection route that defines the world frame with respect to the camera frame, it proved difficult to accurately solve for the specific rotation and translation involved in the maneuver. However, if the goal is to build an articulated model of the satellite, the maneuver itself (rigid body motion in the world frame) is unimportant. For this reason, the results are presented with the effects of the maneuver removed by aligning the first frame to the truth data. For this example, noise was added to the trajectory matrix as described in section 3.5 with η_{TM} equal to 0.08% of the overall standard deviation of the trajectory matrix; this is approximately equivalent to a $3\sigma = 1pixel$ if the image were 2000×2000 pixels in size. Noise was added to the inspection route at a baseline level of $\eta_\theta = \eta_\theta^b = 1.6e-5$ (equivalent to $3\sigma = 10arcsec$) and $\eta_r = \eta_r^b = 0.01$.

Table 5. Results for 2% NMC with LSA segmentation, main body manuever, trajectory matrix uncertainty, and inspection route uncertainty. Terms with * are calculated for the first frame only.

E_{3D}^*	P_{wc}	P_{woc}	P_{gc}	$E_{\hat{a}}^*$ (deg.)	E_J^* (units)	$E_{\Delta\phi}$ (deg.)	$E_{\dot{\phi}}$ (deg./frame)
3.1%	90.2%	10.6%	98.1%	4.8	0.12	6.8	.32

3.5.3 Inspection Route Noise Assessment.

To evaluate the effect of inspection route uncertainty the baseline values (η_θ^b and η_r^b) were multiplied by integer values until performance decreased. Figure 19a shows

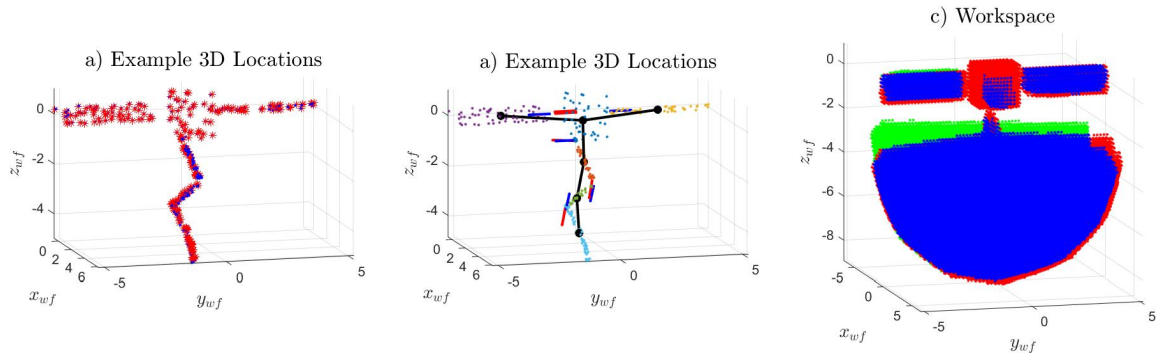

Figure 18. Results with a maneuver, trajectory matrix noise, and inspection route noise. a) Truth (blue ·) and calculated (red ∗) 3D position of points for first frame. b) Calculated points, kinematic chain, joint locations, and axes (truth in blue calculated in red) for first frame. Points in each component are a different color. c) Grid positions in both true and calculated workspace (blue ·), Grid positions in only true workspace (red ·), Grid positions in only calculated workspace (green ·).

the results when both noise values were increased together. The error metrics shown were selected for ease of plotting and the units are those appropriate to the error metric (see Table 5). Performance remains relatively consistent until a level of $\eta_\theta = 4\eta_\theta^b$ and $\eta_r = 4\eta_r^b$. When the noise levels were increased independently (Figure 19b and c) performance remained consistent until the noise was 5 times the baseline. To further investigate the limits, the minimum ratio for the smoothness penalty (γ_{sc-} from Table 3) was decreased to 0.01. This decreases the smoothness penalty, allowing the results to better accommodate inspection route noise. Making this slight adjustment enabled the algorithm to build an accurate model with up to 10 times the baseline inspection route noise as shown in Figure 19d. Notice that the $1 - P_{gc}$ metric seems to decrease at the highest noise level. This is because the results in this trial were so far from the truth model that they were almost entirely outside the discretized space used to calculate the workspace metrics. The average computational time (using a desktop PC) for all of the trials in Figure 19 was 6.1 hrs.

Robustness to noise could be further improved by evaluating more seeds (line 9 of Algorithm 2). For instance, increasing to 50 seeds allowed the algorithm to build an accurate model with $\eta_\theta = 30\eta_\theta^b$ and $\eta_r = 30\eta_r^b$.

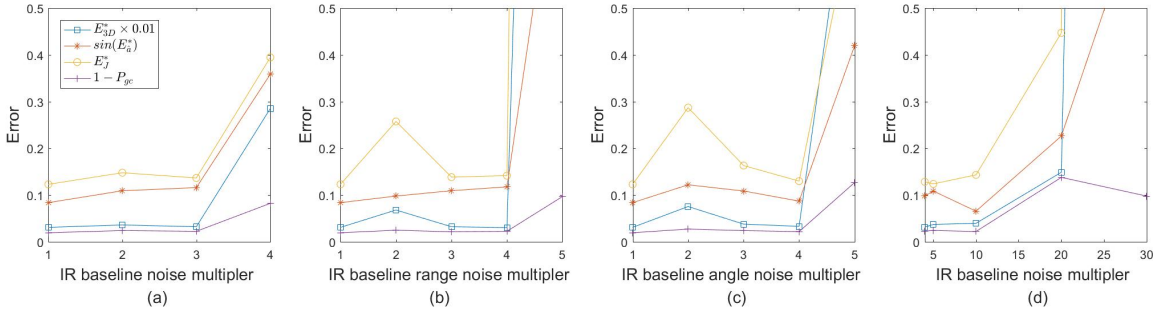

Figure 19. Inspection route (IR) noise assessment. a) η_θ and η_r increased together; b) η_r increased, η_θ held at the baseline; c) η_θ increased, η_r held at the baseline; b) minimum smoothness ratio reduced, η_θ and η_r increased together. Note that when the marker is not shown in the plot that means the error was higher than the y-axis scale.

3.6 Conclusion

The methods presented in this chapter demonstrate that an articulated model can be created from a trajectory matrix of feature points and the inspection route. The original method demonstrates how a model could be built from a sparse trajectory matrix gathered from a full NMC inspection route. The capability of this method to produce an accurate articulated model was demonstrated with up to 6 active joints in the appendage and with inspection routes illumination offset angle of approximately 60°. It was also found that performance was better for a partial NMC. This method has limitations, however these limitations could be mitigated with prior knowledge or with human input.

A modification of the method was also developed that allows the main body to maneuver and accommodates uncertainty in the trajectory matrix and inspection route. Example results are presented, and the performance with increasing levels of inspection route noise was assessed. This method relies on having a known number of components and reduces the trajectory matrix so that it does not contain any missing elements, allowing for accurate segmentation. With a full trajectory matrix, alternative methods employing structure from motion techniques could be used to build the articulated model.[100, 101] As a test case, the BALM algorithm outlined

in [26] was used to solve for the structure and motion of two linked parts from the 2% NMC trajectory matrix outlined in section 3.5.1. When the algorithm was initiated with the true rotation and shape as described in [26] the algorithm took 21 hours to converge to the solution. These methods have not been investigated further, however they have the potential advantage of being able to build the articulated model from the trajectory matrix alone, without knowledge of the inspection route. An example of employing these type of methods for a simple satellite with a single articulating panel is shown in Appendix C.

IV. Articulation Tracking with Feature Points

4.1 Chapter Overview

This chapter consists of method development and results for an estimation framework to track the articulated motion of a primary satellite sequentially to meet Objective 2 of this research. The work presented is also available in a conference paper [21]. This chapter consists of an introduction (section 4.2), the method for tracking articulation with feature points (section 4.3), results (section 4.4), and a conclusion (section 4.5).

4.2 Introduction

Autonomous on-orbit satellite servicing and inspection benefits from an inspector satellite that can track the motion of a primary satellite, including the motion of appendages such as solar arrays, antennas, and sensors. This chapter presents a method of estimating the articulation parameters and shape of a satellite using resolved monocular imagery. A simulated point cloud representing a nominal satellite with articulating solar panels and a complex articulating appendage is developed and projected to the image coordinates that would be seen from an inspector following a given inspection route. A model created using the method in Chapter III is used to initialize a set of extended Kalman filters to track the satellite's motion and articulation from the image coordinates.

4.3 Method

The method begins with an articulated model consisting of the following: N components each containing a set of a 3D points representing the shape of the component in its body frame; a kinematic chain outlining which components are connected to

each other; parameters for $N-1$ joints each consisting of two joint locations (J_p, J_c) and two axes $(\hat{a}_p, \hat{a}_c)$. For this work, the model used was developed using the method outlined in Chapter III, however any model with the required parameters could be used. The filters are initialized using the model, the first frame in the simulated sequence, and the camera's pose. The P feature point locations from a frame are used as the measurement in P seperate point position EKFs (ppEKF) to estimate the 3D position of the point in its assigned component's body frame with component rotation and translation, defined by the articulation parameters, taken as prior knowledge. Next the 3D body frame positions of each point are used as prior knowledge in an articulation parameter EKF (apEKF) with the 2D feature point locations as measurements to update the articulation parameters. Newly seen points are added to the model using a multiple model approach. Results are demonstrated for a full NMC with linear articulation and for a partial NMC with articulation start/stop included in the simulation.

4.3.1 Simulating Data.

Data is simulated for this method in the same way as described in section 3.3.1 with articulation of both panels and joints 3, 5, and 6. The model used is shown in Figure 7.

4.3.2 Filter Framework.

A feature point represents a specific position in 3D space projected to a 2D measurement of the point's location in an image. In this work, the 3D position of each point $\mathbf{s}^j$ is measured in the body frame of the component to which it is assigned. A diagram of the coordinate frames used in this work is shown in Figure 10. The relationship between each component's body frame and the world frame is captured in

the articulated model which can be represented by the state vector $\mathbf{x}^{AP}$. The articulated model is required to relate a 3D point $\mathbf{s}^j$ to the corresponding 2D measurement $\mathbf{z}_t^j$, however, given an articulated model each point can be estimated separately, rendering them conditionally independent.[86] This allows development of a framework, inspired by the FastSLAM method of Montemerlo et al.[62, 61] and the multi-level recursive estimation method of Martin Martin and Brock [58], consisting of a set of EKFs to estimate $\mathbf{s}^j$ and a separate EKF to estimate $\mathbf{x}^{AP}$.

With each measurement, the articulation parameters are propagated to the current time step t to give $\mathbf{x}_t^{AP-}$. Next, 3D positions of each point are updated independently with an EKF, termed point position EKF (ppEKF), that estimates the 3D position of the point $\mathbf{s}^{j+}$ given the 2D image coordinates as measurements ($\mathbf{z}_t^j$) and the propagated articulation states $\mathbf{x}_t^{AP-}$. Finally, the articulation parameters are updated with an EKF (apEKF) that estimates $\mathbf{x}_t^{AP+}$ given the measurement $\mathbf{z}_t^j$ and the 3D point positions $\mathbf{s}^{j+}$. These steps are discussed in more detail in the following sections.

The state vector for the articulation parameter EKF (apEKF) is as follows:

$$\mathbf{x}^{AP} = \begin{bmatrix} (\mathbf{q}^{wr})^T & (\mathbf{T}^{wr})^T & (\omega^{wr})^T & (\dot{\mathbf{T}}^{wr})^T & \nu^{1:M} \end{bmatrix} \tag{91}$$

The first 13 states capture the orientation and velocities of the root component with respect to the world frame. The rotation of the root component is expressed in quaternion form as $\mathbf{q}^{wr}$ while its translation is expressed as $\mathbf{T}^{wr}$. Quaternions in this work are expressed as in Markley [56] with $\mathbf{q} = \begin{bmatrix} \mathbf{e}\sin(\phi/2) \\ \cos(\phi/2) \end{bmatrix}$. The variable $\nu^{1:M}$ contains the articulation parameters for each of the $M = N - 1$ joints as follows:

$$\nu^i = \begin{bmatrix} \theta^i & \psi^i & (\mathbf{J_p}^i)^T & (\mathbf{J_c}^i)^T & \phi^i & \dot{\phi}^i \end{bmatrix} \tag{92}$$

The kinematic chain can be expressed as a table (see section 3.3.6) with each row

containing the parent/child relationship for a joint. The states θ^i and ψ^i are the azimuth and elevation, respectively, defining the articulation axis for the joint in the parent's body frame. During initialization, the body frame of the child is rotated so that the articulation axes of the parent and child are aligned. This allows the axis of the joint to be represented by a single set of angles. The states $\mathbf{J}_p^i$ and $\mathbf{J}_c^i$ are the locations of the joint in the body frames of the parent and the child respectively. The states ϕ^i and $\dot{\phi}^i$ are the articulation angle and the articulation rate respectively.

The measurement ($\mathbf{z}_t$) is the 2D location of feature points in the image frame where $\mathbf{z}_t^j = [u_t^j, v_t^j]^T$ and P is the number of observed feature points in frame t.

$$\mathbf{z}_t = \begin{bmatrix} (\mathbf{z}_t^{c_1})^T & (\mathbf{z}_t^{c_2})^T & \cdots & (\mathbf{z}_t^{c_P})^T \end{bmatrix}^T \tag{93}$$

Since simulated feature point data is being used in this work, it is assumed that points can be accurately tracked and the point correspondence between views is known until the points drop out of view for a significant number of frames. The vector c provides the correspondence between the feature point locations in frame t and the point identifier j.

The quaternion was allowed to update unconstrained and was normalized when used to calculate the rotation matrix. After each update, the quaternion normalization method outlined in Civera [14] was applied to the state vector and the covariance matrix. While other methods of handling the quaternion, such as in references [37, 56, 16], were investigated, this method worked well in the situations tested.

4.3.3 Propagate Articulation Parameters.

Using this framework, most of the states are expected to remain constant over time. The only states expected to change with time are the root component motion ($\mathbf{q}^{wr}$ and $\mathbf{T}^{wr}$) and the articulation angles (ϕ^i). The root component quaternion is

propagated as in Civera [14] assuming a constant angular rate with noise η_ω and t indexing the time step.

$$\mathbf{q}_t^{wr} = \mathbf{q}_{t-1}^{wr} \otimes q((\omega_t + \eta_\omega \Delta t)\Delta t) \tag{94}$$

$$\omega_t^{wr} = \omega_{t-1}^{wr} + \eta_\omega \Delta t \tag{95}$$

The $\otimes$ represents quaternion multiplication and $q(*)$ is the conversion from a rotation vector to a quaternion $q(*) = \begin{bmatrix} \frac{*}{\|*\|}\sin(\frac{\|*\|}{2}) \\ \cos(\frac{\|*\|}{2}) \end{bmatrix}$. The translation is propagated assuming a constant rate with noise $\eta_{\dot{T}}$.

$$\mathbf{T}_t^{wr} = \mathbf{T}_{t-1}^{wr} + (\dot{\mathbf{T}}_{t-1}^{wr} + \eta_{\dot{T}}\Delta t)\Delta t \tag{96}$$

$$\dot{\mathbf{T}}_t^{wr} = \dot{\mathbf{T}}_{t-1}^{wr} + \eta_{\dot{T}}\Delta t \tag{97}$$

The articulation angle ϕ^i propagates with a constant rate $\dot{\phi}^i$ with noise $\eta_{\dot{\phi}}$.

$$\phi_t^i = \phi_{t-1}^i + (\dot{\phi}_{t-1}^i + \eta_{\dot{\phi}}\Delta t)\Delta t \tag{98}$$

$$\dot{\phi}_t^i = \dot{\phi}_{t-1}^i + \eta_{\dot{\phi}}\Delta t \tag{99}$$

The remaining states in $\mathbf{x}^{AP}$ are constant and are propagated generally as $x_t = x_{t-1} + \eta_x \Delta t$ where η_x is the estimated process noise for the state x. Propagation of these states results in $\mathbf{x}_t^{AP-}$. The covariance matrix P^{AP} is propagated by $P_t^{AP-} = F_t P_{t-1}^{AP+}(F_t)^T + G_t Q (G_t)^T$ where F_t is the Jacobian of the propagation equations, G_t is the derivative of the propagation equations with respect to the noise channels, and Q is the noise strengths. The derivatives required for F and G are shown in Appendix B.

4.3.4 Point Position EKF Update.

For every point j in component n an EKF is used to estimate the state $\mathbf{s}_t^{n,j+}$ using the corresponding measurement $\mathbf{z}_t^{n,j}$ and the propagated state $\mathbf{x}_t^{AP-}$. Since each point is assumed constant in the body frame, the propagation step is not required ($\mathbf{s}_t^{n,j-} = \mathbf{s}_{t-1}^{n,j+}$, $P_t^{s^{n,j}-} = P_{t-1}^{s^{n,j}+}$). The measurement model for the ppEKF is as follows:

$$\begin{bmatrix} u' \\ v' \\ \alpha \end{bmatrix} = RO_t^n \begin{bmatrix} \mathbf{s}_t^{n,j-} \\ 1 \end{bmatrix} \tag{100}$$

$$RO_t^n = P_{cam} \left[\begin{array}{c|c} R^{CW} & \mathbf{r}^{CW} \end{array}\right] \left[\begin{array}{c|c} R^{wn} & \mathbf{T}^{wn} \\ 0\,0\,0 & 1 \end{array}\right] \tag{101}$$

$$\hat{\mathbf{z}}_t^{n,j} = \begin{bmatrix} \frac{u'}{\alpha} \\ \frac{v'}{\alpha} \end{bmatrix} \tag{102}$$

The reprojection operator (RO_t^n) for the n^{th} component is a function of the known camera orientation (R_t^{CW}, $\mathbf{r}_t^{CW}$), the camera calibration matrix (P_{cam}), and the articulation parameters $\mathbf{x}_t^{AP-}$. Since the articulation parameter states are not in the state vector for the ppEKF the linearized measurement model $H_t^{n,j}$ can be written as follows using Matlab notation to index RO_t^n.

$$H_t^{n,j} = \frac{1}{\alpha^2} \begin{bmatrix} \alpha RO_t^{n,(1,1:3)} - u' RO_t^{n,(3,1:3)} \\ \alpha RO_t^{n,(2,1:3)} - v' RO_t^{n,(3,1:3)} \end{bmatrix} \tag{103}$$

Equation (103) can be used in the standard Kalman filter update equations to update $P_t^{s^{n,j}-}$ and estimate $\mathbf{s}_t^{n,j+}$ where R is the measurement noise.

$$A^{s^{n,j}} = H_t^{n,j} P_t^{s^{n,j}-} (H_t^{n,j})^T + R \tag{104}$$

$$K = P_t^{s^{n,j}-} (H_t^{n,j})^T (A^{s^{n,j}})^{-1} \tag{105}$$

$$P_t^{s^{n,j}+} = (I - K H_t^{n,j}) P_t^{s^{n,j}-} \tag{106}$$

$$\mathbf{r}^{n,j} = \mathbf{z}_t^{n,j} - \hat{\mathbf{z}}_t^{n,j} \tag{107}$$

$$\mathbf{s}_t^{n,j+} = \mathbf{s}_t^{n,j-} + K\mathbf{r}^{n,j} \tag{108}$$

The measurement covariance $A^{s^{n,j}}$ is stored for each filter and is used in the update of the articulation parameters outlined in the next section.

4.3.5 Update Articulation Parameters.

Next, the 3D positions of the points are used as prior knowledge to update the articulation parameters. The measurement model is the same as outlined in equations (100)-(102), but using the updated positions $\mathbf{s}_t^{n,j+}$. The articulation parameters determine the rotation matrix R^{wn} and translation $\mathbf{T}^{wn}$ for each component. For the root component $\mathbf{T}^{wn} = \mathbf{T}^{wr}$ is known directly from the state vector and R^{wr} can be easily calculated from $\mathbf{q}^{wr}$. For the remaining components, rotation matrices and translations are calculated based on the hierarchy of the kinematic chain and the articulation parameters. Define $X = [x_1, x_2, \cdots, x_l]$ as a single chain starting at the root and ending with a component that has no children where each element of X identifies a component and l is the length of the chain. The rotations and translations

can be written as follows for the joint between x_k and x_{k-1} defined as joint i.

$$R^{w,x_k} = \begin{cases} R^{wr} & , \ k = 1 \\ R^{w,x_{k-1}} R^{x_{k-1},x_k} & , \text{ otherwise} \end{cases} \tag{109}$$

$$\mathbf{T}^{w,x_k} = \begin{cases} \mathbf{T}^{wr} & , \ k = 1 \\ R^{w,x_{k-1}} \mathbf{J}_p^i + \mathbf{T}^{w,x_{k-1}} - R^{w,x_k} \mathbf{J}_c^i & , \text{ otherwise} \end{cases} \tag{110}$$

The rotation matrices between linked parts (R^{x_{k-1},x_k}) can be found by using the azimuth (θ^i), elevation (ψ^i), and articulation angle (ϕ^i) for the joint. The term $\hat{a}^\times$ is the skew-symmetric representation of $\hat{a}$.

$$\hat{a} = \begin{bmatrix} \cos(\theta^i)\cos(\psi^i), & \sin(\theta^i)\cos(\psi^i), & \sin(\psi^i) \end{bmatrix}^T \tag{111}$$

$$R^{x_{k-1},x_k} = \cos(\phi^i) I + (1 - \cos(\phi^i)) \hat{a}\hat{a}^T - \sin(\phi^i)\hat{a}^\times \tag{112}$$

In this case, the linearization of the measurement model is much more complex since each measurement is dependent on the articulation parameters for every joint above it in the kinematic chain. The derivatives required to calculate H_t^{AP} are shown in Appendix B. The state $\mathbf{x}_t^{AP-}$ and the articulation parameter covariance P_t^{AP-} can then be updated using standard Kalman filter update equations similar to equations (105)-(108). Note that the measurement noise covariance R is built at each update with the appropriate $A^{s^{n,j}}$ matrices from the ppEKF update along the diagonal. This brings the uncertainties from the ppEKFs to the apEKF.

4.3.6 Initialization.

An initial state vector is required to begin the apEKF and the ppEKFs. Due to the nature of the problem, convergence to the correct solution is not guaranteed,

therefore the initial state must be chosen judiciously. The given model is used to find a state vector that most closely projects points matching the first image. As mentioned previously, the model from Chapter III was used in this work. In Chapter III the articulation is parameterized by two axes for each joint, one defines the axis in the parent's frame and the other defines the axis in the child's frame. Rotating the child's frame to align its axis with the parent's axis allows one axis to be eliminated. Defining the axis with θ^i and ψ^i as in equation (111) further reduces the number of states, however it introduces the potential for a singularity when $\psi = \pm\frac{\pi}{2}$. To avoid this, the body frame is rotated if an axis is near the singularity. Since the axis should be stationary, it is unlikely to move to the singularity. In this way, the model is used to initialize θ^i and ψ^i. The joint locations $\mathbf{J_p}^i$ and $\mathbf{J_c}^i$ from the model, rotated appropriately for frame alignment, are used directly to initialize the filter.

Once the body frames have been aligned, the next step is to initiate the root component rotation $\mathbf{q}^{wr}$ and translation $\mathbf{T}^{wr}$. Defining the world frame at the origin of the root component and assuming the inspection route is defined with respect to the world frame allows these to be set to $\mathbf{q}^{wr} = [0, 0, 0, 1]^T$ and $\mathbf{T}^{wr} = [0, 0, 0]^T$.

Next, the articulation angles ϕ can be determined. This is done by randomly selecting 5,000 sets of $\phi's$ and projecting the model points to the image plane assuming those articulations. No point correspondence is known between the model and the first image feature points, so the distance from each reprojected point (Z) to each first image feature point (W) is used to quantify the quality of the selected set of $\phi's$. Specifically, the summed distance between each point in Z and its nearest neighbor in W added to the summed distance between each point in W and its nearest neighbor in Z is used as the cost function (J) measuring the quality of a set of $\phi's$.

$$J = \sum_{i \in Z} \min_j D_{i,j} + \sum_{j \in W} \min_i D_{i,j} \tag{113}$$

D is a matrix containing the Euclidean distances between each point in Z and each point in W. The 15 best sets of $\phi's$ are then used as starting points for minimizing J with Matlab's *fminunc*. The best solution is used as the initial set of $\phi's$.

The rates (ω^{wr}, $\dot{\mathbf{T}}^{wr}$, $\dot{\phi}^i$) are initialized at zero leaving only the shapes ($\mathbf{\Omega}^r$, $\mathbf{\Omega}_c^i$). These are initialized by comparing the locations in the reprojected model Z and the first image feature point W. For each point in W the closest 12 points in Z are found. Of these closest 12 points, the 3 that are closest to the camera (lowest z component in the camera frame) are selected to determine the component assignment and location for the point. The point is assigned to the most common component of these three points in the model and the location of the point within that component is taken as the mean location of those points. This method could have difficulties in cases where multiple components are stacked up in-line with the camera's optical axis at initiation. In that case other methods such as triangulation could be used to assist in assigning points to the appropriate component.

4.3.7 Adding Points to the Model.

In many inspection routes, such as a natural motion circumnavigation (NMC), the inspector satellite moves around the primary satellite. This means that during inspection, feature points will drop out of view and new feature points will be visible. Therefore, a method is necessary to add new feature points to the state vector for tracking. In EKF SLAM applications such as Davison [24] and Civera [14], an alternative representation for the point location is used to initialize the point in the filter until the distance can be estimated with sufficient certainty. In this application, the points need to be assigned to an appropriate component and their position within that component must be estimated. To do this, a multiple model framework is employed. For each newly seen point a bank of N ppEKFs is created, one for

each of the components. Each filter estimates the points 3D position in the body frame of component n. The reprojection operators (RO^n) calculated from $\mathbf{x}_t^{AP+}$ are taken as prior knowledge in each filter which is propagated and updated as outlined in the ppEKF section. The measurement covariance $A^{n,j} = H^{n,j}P^{n,j}(H^{n,j})^T + R$ and the measurement residual $\mathbf{r}_j^n = \mathbf{z}_k - \hat{\mathbf{z}}_k$ can then be used to calculate the posterior probability $p(\mathbf{z}_t|\mathbf{s}_j^{n-}, RO_t^n)$ for the n^{th} filter.[42, 31, 86]

$$p(\mathbf{z}_t|\mathbf{s}_j^{n-}, RO_t^n) = \frac{1}{det(2\pi A_j^n)^{\frac{1}{2}}} e^{-1/2((\mathbf{r}^n)^T(A^{n,j})^{-1}\mathbf{r}^n)} \tag{114}$$

The conditional probability of each component given the measurement $\mathbf{z}$ can then be calculated.

$$p(n|\mathbf{z}_t) = \frac{p(\mathbf{z}_t|\mathbf{s}_j^{n-}, RO_t^n)p(n|\mathbf{z}_{t-1})}{\sum_{l=1}^{N}\left[p(\mathbf{z}_t|\mathbf{s}_j^{l-}, RO_t^l)p(l|\mathbf{z}_{t-1})\right]} \tag{115}$$

The initial conditional probabilities for each component are set to $1/N$. The initial state is set to zero and the initial covariance matrix is set to the identity matrix times σ_s^2. The filter banks for each point are updated until the conditional probability of a particular model remains above a threshold for some number of frames, at which point the point $\mathbf{s}_j$ is added to the model. An example of the conditional probabilities for a point as it is assigned to a component is shown in Figure 20. Notice the highest probability varies between multiple components before it settles to component 1.

4.3.8 Outlier Rejection.

Points are removed from the model when their RMS residual is more than four times (judiciously chosen) the average RMS value for all the residuals. This method works well to remove a few points that may have been assigned to the wrong com-

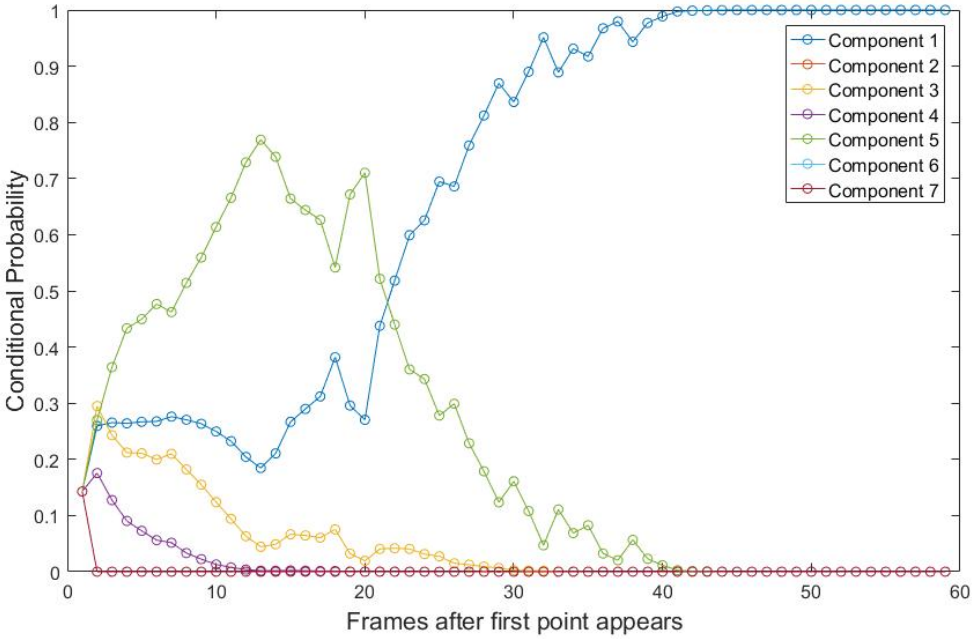

Figure 20. Conditional probabilities for an example point from initial acquisition to assignment.

ponent, however it has limitations. For instance, if the articulation parameters are incorrect for a particular component, all the points corresponding to that component could be rejected as outliers before the apEKF converges to the correct articulation parameters.

4.4 Results

To test this method an articulated model resulting from the model building method of Chapter III was used as the input model. The truth model used for simulating the data consisted of 5 joints, one between each panel and the main body and three on the appendage. However, the articulated model calculated from the method in Chapter III identified an extra component and joint on one of the solar arrays. This inaccuracy in the developed model was retained in this work to investigate performance with an inaccurate initial model. A diagram of the nominal

satellite used is shown in Figure 7. A 2×1 NMC inspection route phased with zero illumination offset is the basis for the results presented. In each of the scenarios white Gaussian noise $\mathcal{N}(0,\, \sigma_{u,v}^2)$ is added to the image coordinates where $\sigma_{u,v}$ is set to 0.2% of the overall standard deviation of image coordinates. At each frame noise was also added to R^{CW} in the form of a random rotation of ρ radians where ρ is sampled from $\mathcal{N}(0,\, \sigma_\rho^2)$. Additionally, noise is added to $\mathbf{r}^{CW}$ by multiplying each component by random noise sampled from $\mathcal{N}(1, \sigma_\gamma^2)$.

4.4.1 Full NMC Demonstration.

As with any EKF, this framework requires noise values to be tuned to improve performance. Table 6 shows the noise values used.

Table 6. Noise settings

Initial covariance values								
σ_q^2	σ_T^2	σ_ω^2	$\sigma_{\dot{T}}^2$	σ_s^2	$\sigma_{\theta,\psi}^2$	σ_J^2	σ_ϕ^2	$\sigma_{\dot{\phi}}^2$
$1e^{-9}$	$1e^{-7}$	$1e^{-8}$	$1e^{-6}$	$1e^{-2}$	$5e^{-6}$	$5e^{-5}$	$8e^{-8}$	$1e^{-3}$

Process noise values					
η_ω	$\eta_{\dot{T}}$	η_s	$\eta_{\theta,\psi}$	η_J	$\eta_{\dot{\phi}}$
$1e^{-12}$	$1e^{-10}$	0	0	0	0
Measurement noise					
$\eta_{u,v} = (\sigma_{u,v} + \sigma_\rho + \sigma_\gamma)^2$					
Inspection route noise					
$\sigma_\rho = 3e^{-4}$			$\sigma_\gamma = 2e^{-4}$		

To demonstrate the capability of the filter to track the model long term, a complete NMC is performed with the panels articulating at a constant rate between -1.26 radians to 1.26 radians and each of the active joints (joints 3, 5, and 6 from Figure 7) articulating at a constant rate between -0.94 radians to 0.94 radians. The root component remains stationary in the world frame in this simulation. The simulation consists of 2880 updates over the full NMC. This translates to 2 updates per minute if the primary satellite is in geosynchronous orbit. The average computational time

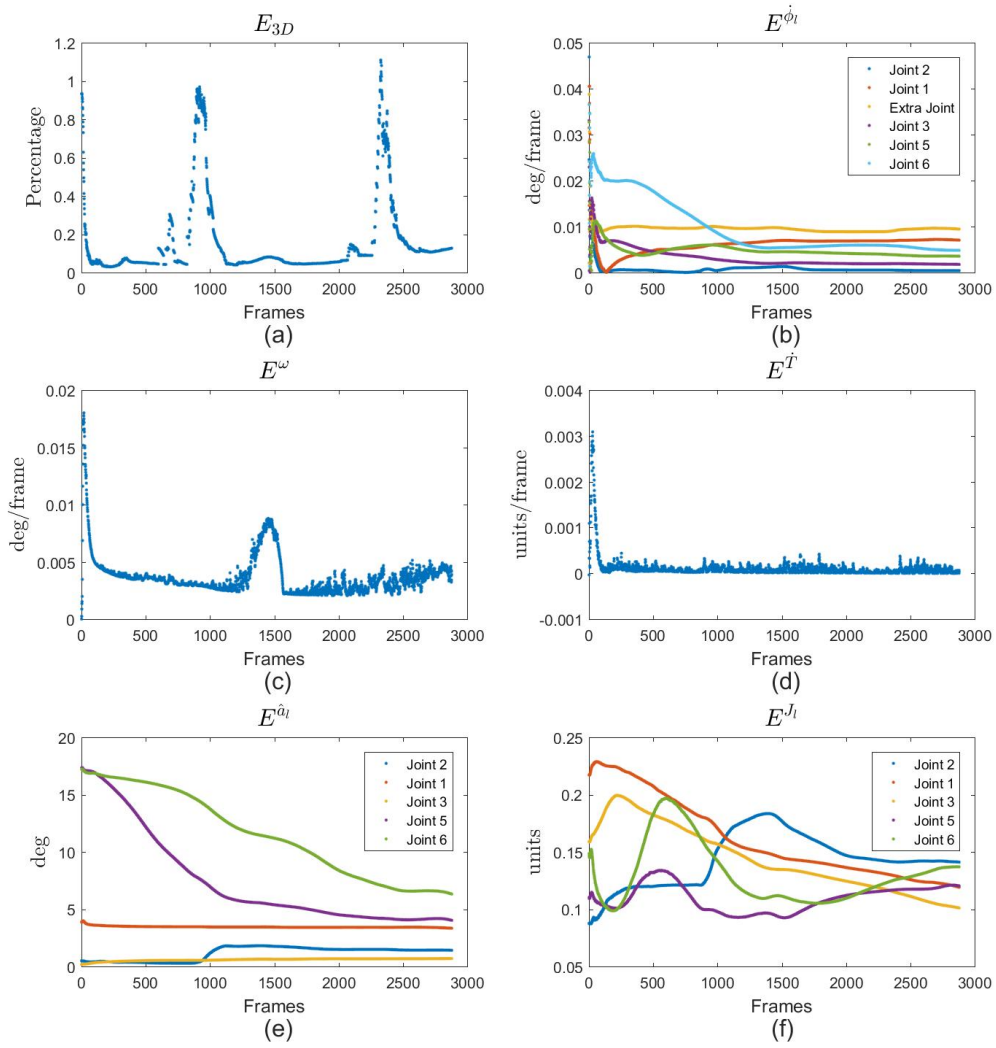

Figure 21. Full NMC results.

using a desktop computer for updating the filter is approximately 0.3 sec, leaving plenty of time for image processing and feature point tracking. As more points are added to the model the computational time increases. In this work, there are 1365 points in the model at the completion of the NMC; 341 of these are duplicates that were reacquired after disappearing from view. A method of assigning reacquired points to the appropriate ppEKF could be implemented to limit the number of points in the model.

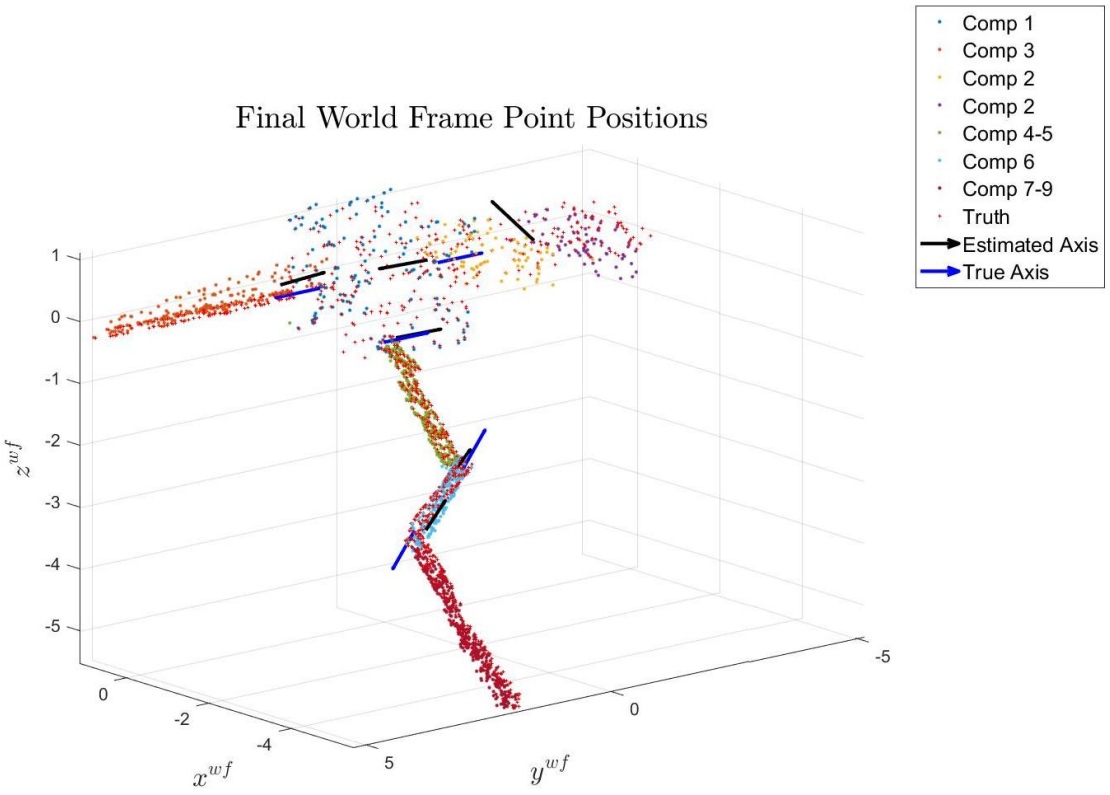

Figure 22. Full NMC Final estimated point positions in world frame. Component color labels correspond to labels in Figure 7.

The percent error of the 3D points in the model is shown in Figure 21a. The percent error, or normalized reconstruction error as defined in [30] is calculated as follows where $\mathbf{S}_t^{truth}$ is the true world frame location of each point in the model at the t^{th} time step and $\mathbf{S}_t^{calc}$ is the estimation of each point $\mathbf{s}_t^{n,j+}$ translated into the world frame at the t^{th} time step.

$$E_t^{3D} = \frac{\|\mathbf{S}_t^{truth} - \mathbf{S}_t^{calc}\|}{\|\mathbf{S}_t^{truth}\|} \times 100\% \tag{116}$$

Notice the spikes in error around frames 900 and 2400. These correspond to the times after the camera views the satellite down the axis of the panels so that one of the panels is completely occluded. When the panel comes back into view, many points

are added to the model causing the temporary increase in error.

The error between the calculated articulation rate and the true articulation rate ($E_t^{\dot{\phi}_l}$) is shown in Figure 21b.

$$E_t^{\dot{\phi}_l} = |(|\dot{\phi}_t^{calc_l}| - |\dot{\phi}_t^{true_l}|)| \tag{117}$$

Since the truth model and the calculated model do not share the same body frames for the components, comparing angular rate gives a better indication of filter performance than comparing the articulation angles directly. The data set labeled 'Extra Joint' is the joint that does not exist in the truth model, therefore it is compared to zero. Figure 21c shows the error between the magnitude of the calculated angular rate and the true angular rate of the root component (E_ω) and Figure 21d shows the error between the magnitude of the calculated translational rate and the truth translation rate of the root component ($E^{\dot{\mathbf{T}}}$).

$$E_t^{\omega} = |\|\omega_t^{calc}\| - \|\omega_t^{true}\||\tag{118}$$

$$E_t^{\dot{\mathbf{T}}} = |\|\dot{\mathbf{T}}_t^{calc}\| - \|\dot{\mathbf{T}}_t^{true}\||\tag{119}$$

The articulation axes error and the joint errors are shown in Figure 21e-f. The articulation axis and joint errors are calculated in the world frame (w) as defined in equations (121), where N is the number of joints, and F is the number of frames. Since the joints are revolute, any location on the axis is acceptable, therefore for the joints the error is measured as the distance from the joint to the line represented by

the true joint location and articulation axis in the world frame.

$$E_t^{\hat{a}} = \frac{1}{N}\sum_{l=1}^{N}\left[\arccos(|(\hat{a}_t^{true_l,w})^T \hat{a}_t^{calc_l,w}|)\right] \tag{120}$$

$$E_t^{J} = \frac{1}{N}\sum_{l=1}^{N}\left[\|\hat{a}_t^{true_l,w} \times (J_t^{calc_l,w} - J_t^{true_l,w})\|\right] \tag{121}$$

Figure 22 shows the final estimated point cloud, the truth point cloud, the truth axes and the estimated axes locations in the world frame. Each of the axes directions align well with the truth axis directions in the world frame. The point locations also are near the truth locations. 94.8% of points are assigned to the correct component. Errors in shape are visible in the solar arrays. This is likely due to the fact that the NMC trajectory results in complete occlusion of both panels for a portion of the NMC. Therefore, when the panel reappears, the points are added to the body frame of the component that has been modified by any articulation rate error present at the beginning of the occlusion.

4.4.2 Inspection Route Noise.

To investigate the effect of the inspection route noise, the filter was run at 64 combinations of σ_ρ and σ_γ representing all combinations of 8 values spaced logarithmically between $5e-5$ and $5e-4$. Each combination was run 5 times and the results were averaged. Since inspection route noise causes the reprojected feature points location to be different from the true feature point location, the impact of the inspection route noise can be captured in the measurement noise. For each run, the measurement noise ($\eta_{u,v}$) was adjusted by summing the image noise ($\sigma_{u,v}$), the inspection route angle noise (σ_ρ), and the inspection route position noise (σ_γ) then squaring the result. For each combination, the metrics E_t^{3D}, $E_t^{\dot{\phi}_l}$, E_t^{ω}, $E_t^{\dot{\mathbf{T}}}$, $E_t^{\hat{a}}$, and E_t^{J} for the individual runs are averaged. Figure 23 shows the results.

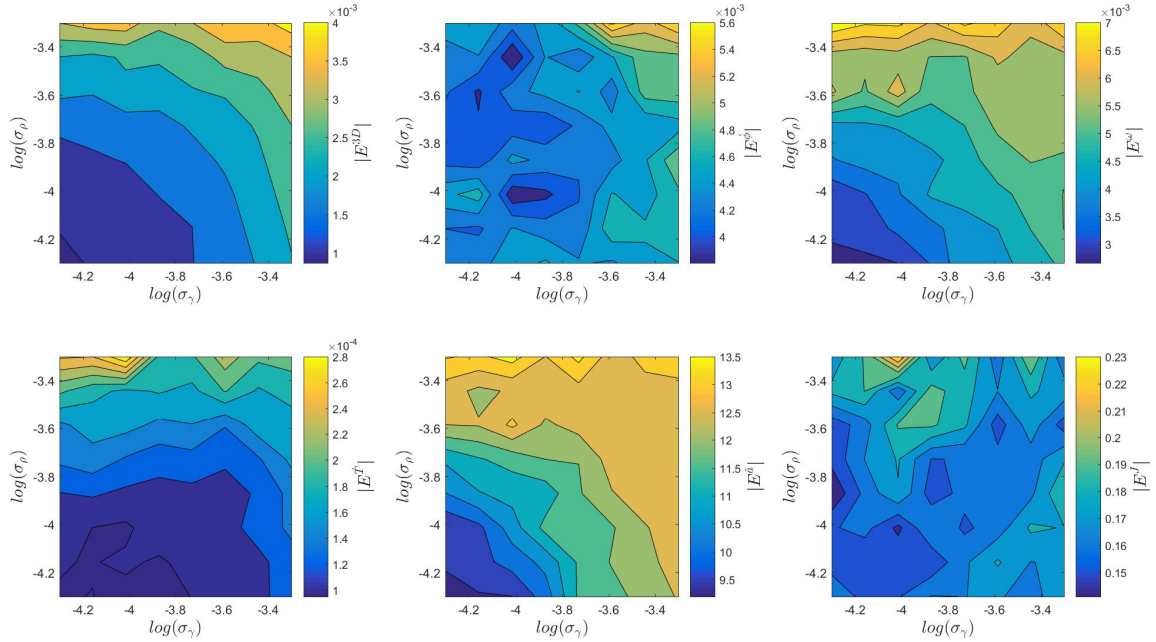

Figure 23. Mean evaluation metrics for various inspection noise combinations

As expected, the error metrics generally increase as the noise increases. This information could be used to select relative navigation systems such that the error in relative position and angle can be determined with enough accuracy to give the desired filter performance. For instance, if the performance demonstrated in Figures 21 and 22 is acceptable, position accuracy known to within $\pm 0.06\%$ (3σ) and relative pointing accuracy to within $\pm 0.05^\circ$ (3σ) would be sufficient. This is within the capabilities of current systems.[52]

4.4.3 Partial NMC with Articulation Start/Stop.

Perhaps a more interesting scenario would involve an inspection route in which the primary satellite begins and stops articulation within the inspection time. Using the comparison to an existing robotic arm mentioned in section 3.5.1 the articulation should occur over 2% of an NMC. This faster articulation rate requires the update rate to be increased. For this simulation the update rate is set to 0.33 Hz (one update

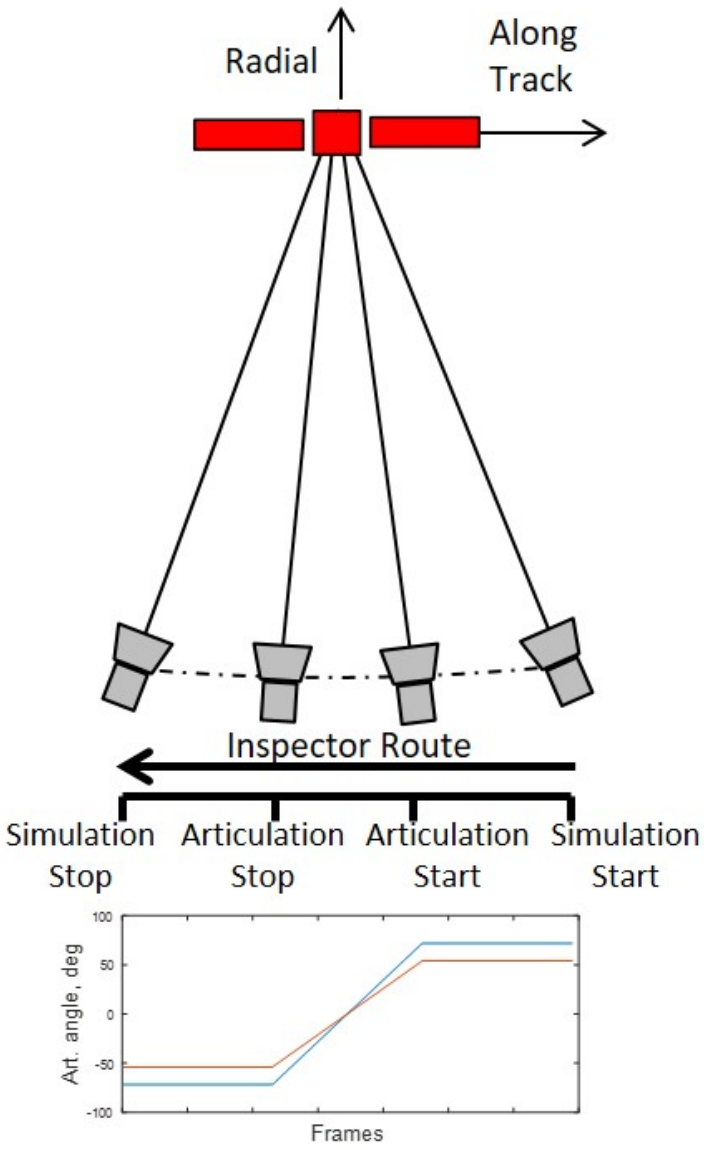

Figure 24. Diagram of scenario consisting of 6% of an NMC with articulation occurring over the middle third of the simulation only.

every 3 seconds). With a 0.3 second demonstrated computational time per update this could still be done in real-time, particularly since the ppEKF updates can be done in parallel. The articulation time period was then centered between two equally sized time sets of zero articulation. A diagram illustrating the scenario is shown in Figure 24. Throughout the simulation, the primary satellite was maneuvering in the world frame with small translational and angular rates.

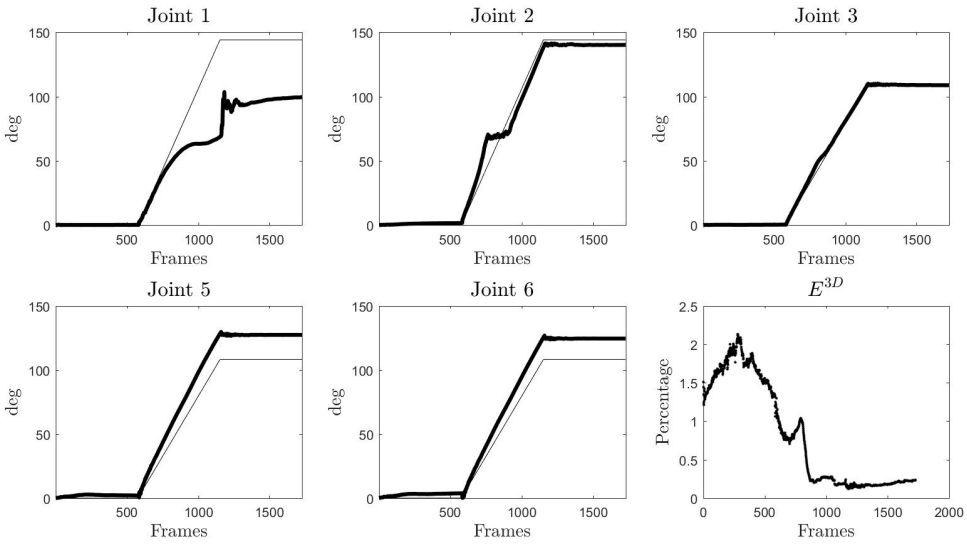

Figure 25. Comparison of actual and estimated articulation angles and the E^{3D} (lower right) for the partial NMC simulation

Due to the abrupt change in articulation rate, the filter initially diverged after the articulation started. To fix this issue, a noise inflation trigger was added. The Mahalanobis distance Ψ can be used as a metric to monitor the filter's performance.[34] It is calculated using equation (122) where $\mathbf{r}_t$ is the residual, H_t is the measurement matrix, P_t is the covariance matrix, and R is the measurement noise for the apEKF.

$$\Psi_t = \mathbf{r}_t^T (H_t P_t H_t^T + R)^{-1} \mathbf{r}_t \tag{122}$$

When the Mahalanobis distance increased above a threshold of $3,000$ (judiciously chosen) the process noise for the articulation rate was increased. This allows the filter to appropriately adjust the articulation rate rather than other parameters such as the articulation axis to account for the errors between prediction and measurement. For each frame after noise inflation is triggered, the inflated noise ($\eta_{\dot{\phi}}$) is reduced by 10% until it reaches the original value. These settings and this method worked well for this case, however more robust methods of picking these thresholds certainly should

be investigated.

The estimated articulation angles compared to truth and E_t^{3D} are shown in Figure 25. The poor results for joint 1 can be explained by looking at the number of measurements for that component. Only six feature points are measured for the component attached to joint 1 at the end of the simulation. With such few feature points, the filter likely moved the feature points in the body frame rather than adjusting the articulation angle to explain the measurements. Similarly, the inaccuracies in joints 5 and 6 likely occur because the points are allowed to move in the body frame. This means the shape of the component could change rather than the articulation angle to explain the measurements. A solution to this problem may be to fix points in the body frame after their residuals are below a threshold for a certain number of frames. While there are certainly improvements that could be made, this scenario demonstrates the filter can adapt to articulation rate changes.

4.5 Conclusion

While this work demonstrates an initial capability to track articulation, there are a few limitations and improvements that could be made:

- Stationary articulation parameters such as the axes and joints could be removed from the model. Including them in the model allows the method to improve a potentially inaccurate model, however continuing to include them could lead to confusion when the main body motion or articulation is not at a constant rate. Removing the stationary states after they have remained constant could improve long term results.

- If a joint begins to articulate that is not included in the model, the filter is likely to diverge. A method of accommodating additional joints could be developed, perhaps using multiple models to track multiple hypothesis when residuals are

high. A similar concept could be developed to collapse (merge) joints with near zero articulation rates.

- If an articulation rate changes while the component is occluded, the filter may not be able to recover when the component comes back into view. Automatic re-initialization methods could be developed to solve this issue.

This work lays out a framework for tracking an articulated object given an inspection trajectory and demonstrates initial performance under realistic levels of noise in the inspection route. The method is capable of tracking the articulation of the components throughout an NMC in which points continuously come into view and are added to the model. The effects of inaccuracies in the relative translation and rotation between inspector satellite and primary satellite are investigated yielding results which could inform system design specifications for a relative navigation system. A simulation is also presented in which immediate changes in articulation rate are correctly estimated by employing a noise inflation technique.

V. Articulation Tracking with Silhouettes

5.1 Chapter Overview

This chapter consists of method development and results for an estimation framework to track the articulated motion of a primary satellite sequentially to meet Objective 2 of this research. The work presented is also available in a submitted journal article [20]. This chapter differs from the work in Chapter IV primarily in that it does not use feature points, rather silhouette (binary) images are used. This chapter consists of an introduction (section 5.2), the method for tracking articulation with silhouettes (section 5.3), results (section 5.4), and a conclusion (section 5.5)

5.2 Introduction

As an alternative to using feature points, given a known model, the silhouette of the image can be used as a measurement to detect and track articulation. As described in section 2.3.5, the problem of tracking articulation of a known model using silhouettes has been researched heavily. Most work focuses on tracking the articulated motion of humans. The principles and methods developed for human articulation tracking can be adjusted for any situation in which a model of the articulated structure is known. With these concepts in mind, a simple recursive estimation filter has been developed which demonstrates potential for tracking articulated motion of a known model using silhouette images as measurements.

There are numerous advantages to using silhouettes instead of feature points, particularly for space applications.

1. Inaccuracies in feature point tracking: While methods of feature point tracking are generally accurate, inaccuracies can arise which would cause errors in tracking methods that use feature points. For instance, objects such as solar panels

could have may repetitive features. This could cause a tracker to track a feature point on the corner of a solar cell in one frame to the corner of a different solar cell in the next frame. Additionally, glare could produce a spot of high intensity that travels along a surface as the angle to the Sun changes. Tracking this spot would result in corresponding feature points that do not represent a single location on a rigid body.

2. Inadequate number of feature points: Some objects or components may be smooth and of uniform color which could minimize the number of feature points that can be acquired and tracked.

3. Easy background subtraction: The space environment lends itself well to silhouettes due to the fact that the background is often empty space. This allows silhouettes to be created with simple pixel intensity thresholding.

4. Low resolution or blurry images: When the resolution of an image is decreased, the number of feature points decreases while the silhouette that can be calculated remains nearly the same. To demonstrate this, imagery collected as outlined in Chapter VI was used to create Figure 26 in which an example image was resized to a lower resolution using Matlab's 'imresize' command. The 'dectectFASTFeatures' was then used with default settings to detect feature points on the satellite and the silhouette was found using the method outlined in section 6.4.1. As the resolution decreased the number of feature points detected decreased from 137 to 24 while the percentage of pixels covered by the silhouette remains nearly constant. Note that the work in this Chapter uses simulated imagery, not the real imagery shown in Figure 26.

5. No segmentation required: Feature points must be assigned to a particular component to provide valuable information regarding articulation. The use of sil-

houettes does not require points to be assigned to a particular component.

6. No illumination: In some cases a silhouette may be able to be created without the primary satellite being illuminated from the perspective of the inspector. For instance, if the primary satellite is between the inspector and an illuminated object (such as the earth or moon) the silhouette could be captured by looking at the satellites shadow assuming the optic is capable of looking at an illuminated celestial body.

5.3 Silhouette Based Tracking Method

This work uses an articulated model and an unscented Kalman filter (UKF) to track the pose and articulation angles of the satellite using simulated silhouette images. Silhouette images are binary images with a 1 in pixel locations that are occupied by the foreground (the satellite) and a 0 in pixels that are occupied by the background. Simulated silhouette images are created by projecting the corner points of the satellite model components into the image plane and setting the pixels on the interior of the convex hull defined by the corner points to 1. Simulated silhouette images are compared to the reprojection of the sigma points of a UKF to create a measurement vector which is used to update the state vector at every time step. Results are shown for linear articulation and sinusoidal articulation over a complete NMC and for a partial NMC with articulation start/stop as in section 4.4.3.

5.3.1 Simulating Imagery.

As with the feature point method outlined in Chapter IV, the silhouette method was developed using simulated data. In this case, the simulated data is a set of binary images rather than a trajectory matrix. The method of creating the binary images is similar to the method in section (simulating feature points). The main difference is

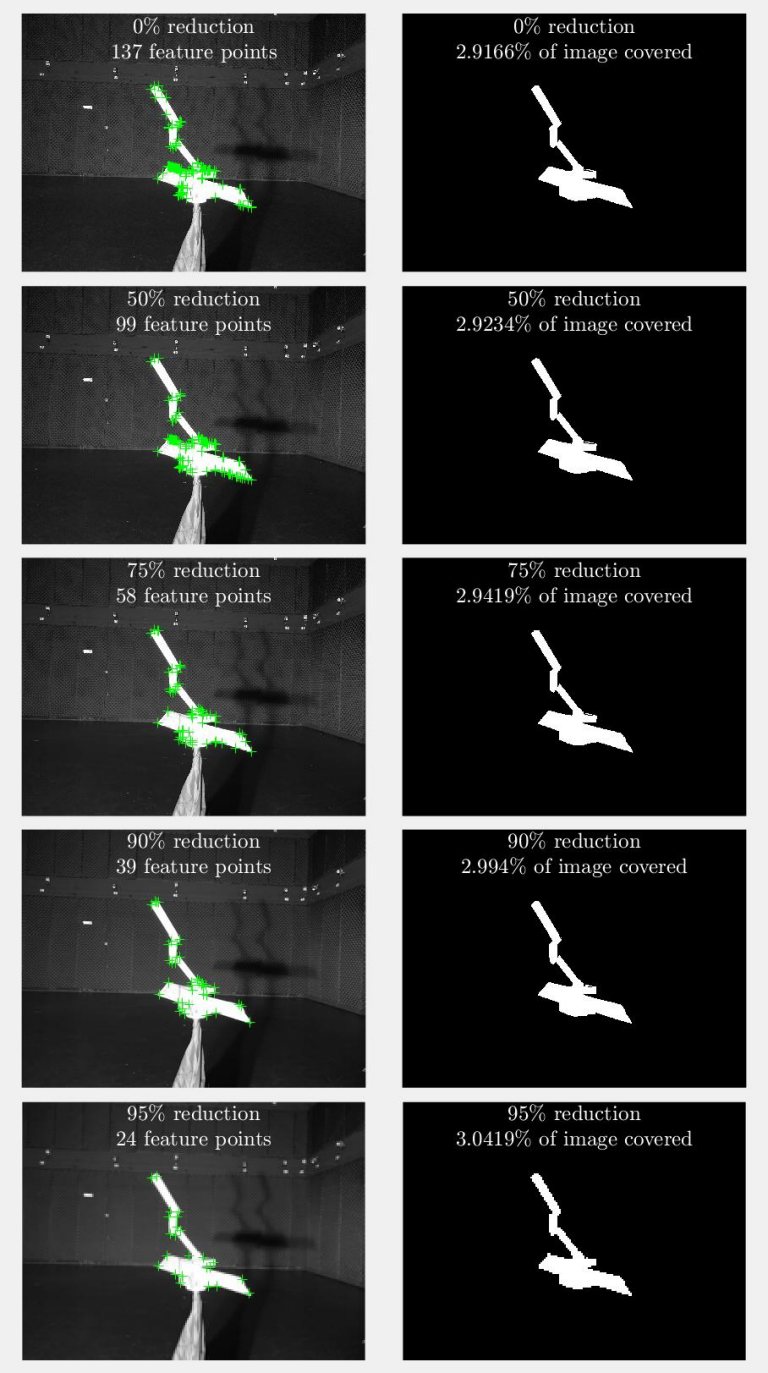

Figure 26. Comparison of feature points to silhouettes as image resolution decreases. Left side is the reduced size image with FAST feature points. Right side is the resulting silhouette from the reduced size image.

that instead of projecting points spread over the model surfaces to represent feature points, only the corner points of the illuminated and visible surfaces are projected to

the image plane.

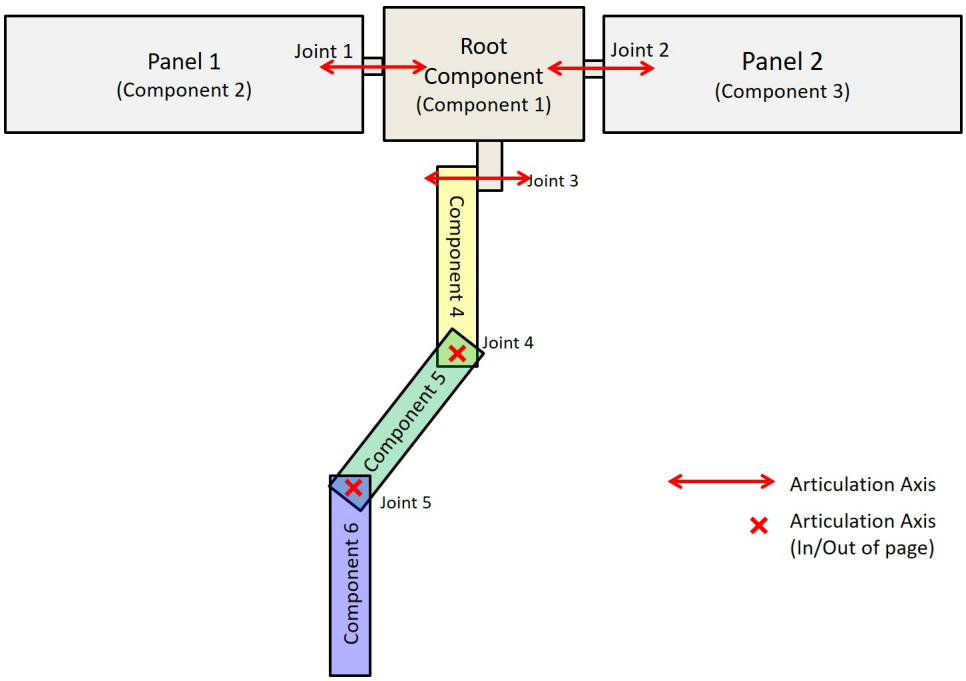

Figure 27. Satellite model diagram.

For simplicity, the satellite model was modified slightly from the model used in previous chapters. The number of joints in the appendage was decreased. The model used for this chapter is shown in Figure 27 which is a simplified version of the model used in Chapter III and IV. For each face on the model, the outward facing normal is checked against the camera vector and the Sun vector to determine if the face is illuminated and visible. If it is illuminated and visible, a binary matrix of size $np_y \times np_x$ is created where np_y and np_x are the number of pixels in the simulated image in the x and y directions respectively. Each element in the matrix represents a pixel in the image. Next, each of the corner points are projected to the image plane using a pinhole camera model and the pixels (elements) in the binary image closest to the projected points are converted to 1's. The Matlab function 'bwcovhull' is then used to convert all the pixels internal to the corner points (representing the entire surface) to value 1. After this is done for each face, the resulting matrices are

summed, and any element greater than zero is set to 1 yielding a binary silhouette image representing the entire model. Note that this method does not account for portions of faces that may be shadowed by other components. Figure 28 shows the resulting silhouette image from a particular model orientation.

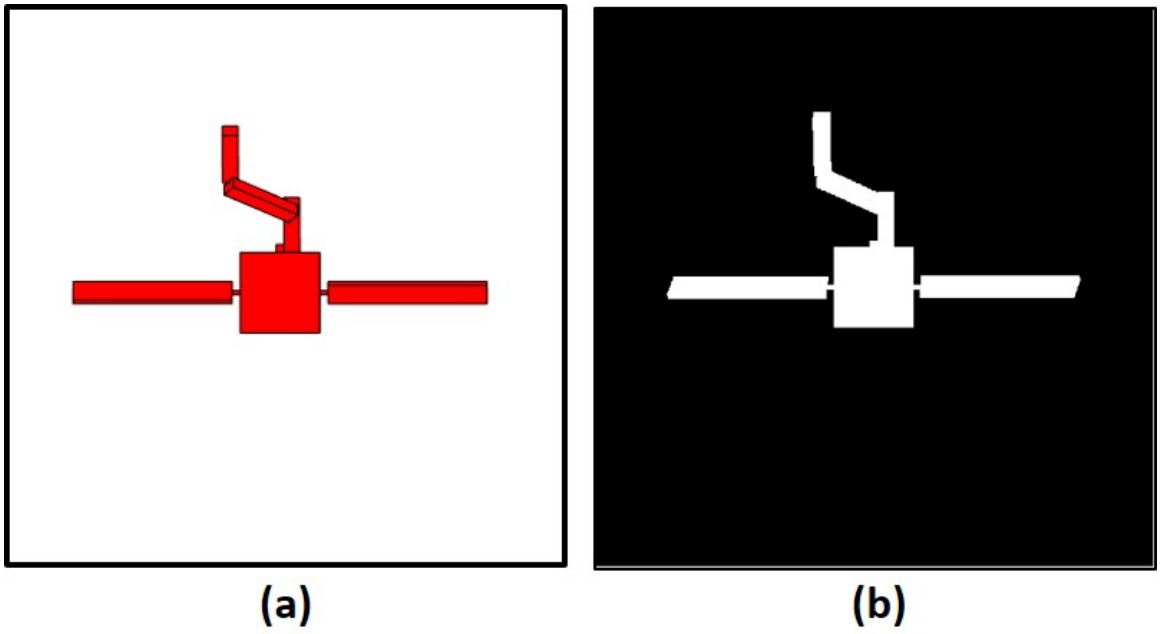

Figure 28. Example of a simulated silhouette image: a) 3D representation of the model b) resulting silhouette image

For the simulations presented here, a silhouette image size of 800×800 was used. This setting can be adjusted as needed. Further reductions can be realized if sparse matrix techniques are employed. Increasing the image size will increase the computational time, but it will also increase the fidelity of the measurements.

5.3.2 Filter Framework.

An unscented Kalman filter (UKF) was selected for this demonstration. As outlined in section 2.2.11, the UKF allows non-linear dynamics and measurement models by representing the state probability distributions using the mean and sigma points. It does not require the linearization of the measurement model as in an EKF which

would be difficult, if not impossible, given the nature of the measurements outlined in section 5.3.4. Additionally, the number of sigma points that must be evaluated is $2n + 1$ where n is the number of states. Implementing using a particle filter would likely require significantly more measurement function evaluations.

The primary satellite is modeled as a set of rectangular prisms connected by revolute joints. The states for the UKF are the pose and rates that characterize the root component's motion as well as the articulation angle and angular rates.

$$\mathbf{x} = \begin{bmatrix} (\mathbf{q}^{wr})^T & (\mathbf{T}^{wr})^T & (\omega^{wr})^T & (\dot{\mathbf{T}}^{wr})^T & \phi^1 & \dot{\phi}^1 & \cdots & \phi^M & \dot{\phi}^M \end{bmatrix} \tag{123}$$

For simplicity, the articulation parameters that are expected to remain constant are not included for estimation as they were in the apEKF outlined in section 4.3.2. The size of each rectangular prism is also held constant, however the size of the prisms as well as the stationary articulation parameters could likely be included in the estimation framework if desired.

5.3.3 Dynamics.

As in the apEKF in section 4.3.3, the articulation parameters are propagated with a constant rates. The propagation equations are repeated here for convenience. Once again, the quaternion is normalized for every use and the state vector and covariance matrix are adjusted every time step according to the method outlined in Civera [14].

$$\mathbf{q}_t^{wr} = \mathbf{q}_{t-1}^{wr} \otimes q((\omega_t + \eta_\omega \Delta t)\Delta t) \tag{124}$$

$$\omega_t^{wr} = \omega_{t-1}^{wr} + \eta_\omega \Delta t \tag{125}$$

The $\otimes$ represents the quaternion multiplication and $q(*)$ is the conversion from a rotation vector to a quaternion $q(*) = \begin{bmatrix} \frac{*}{\|*\|}\sin(\frac{\|*\|}{2}) \\ \cos(\frac{\|*\|}{2}) \end{bmatrix}$. The translation is propagated assuming a constant rate with noise $\eta_{\dot{T}}$.

$$\mathbf{T}_t^{wr} = \mathbf{T}_{t-1}^{wr} + (\dot{\mathbf{T}}_{t-1}^{wr} + \eta_{\dot{T}}\Delta t)\Delta t \quad (126)$$

$$\dot{\mathbf{T}}_t^{wr} = \dot{\mathbf{T}}_{t-1}^{wr} + \eta_{\dot{T}}\Delta t \quad (127)$$

The articulation angle ϕ^i propagates with a constant rate $\dot{\phi}^i$ with noise $\eta_{\dot{\phi}}$.

$$\phi_t^i = \phi_{t-1}^i + (\dot{\phi}_{t-1}^i + \eta_{\dot{\phi}}\Delta t)\Delta t \quad (128)$$

$$\dot{\phi}_t^i = \dot{\phi}_{t-1}^i + \eta_{\dot{\phi}}\Delta t \quad (129)$$

5.3.4 Measurement Model.

No feature points are being used, therefore the measured silhouette image must be compared to the reprojection of the shape given the estimate in a way that quantifies the quality of the estimate. Since the measurement is a binary matrix $np_x \times np_y$, it cannot be used as a measurement directly. Instead, the measurements are more akin to residuals in that they are a comparison of the measurement silhouette to the reprojected silhouette given a state vector.

To evaluate the measurement (or residual), the first step is to create a silhouette image from the estimated state. To do this, the eight corner points (cp_w^n) of the rectangular prism representing component n are projected into the camera frame (cp_i^n) using a pinhole camera model as in equations 100-102. The rotations and translations of each component to the world frame (R^{CW} and $\mathbf{T}^{wn}$) are determined hierarchically from the known model and the articulation parameters contained in the state vector as described in section 4.3.5.

Next an $np_x \times np_y$ matrix of zeros (M_r^n) is created. The corner points cp_i^n are rounded to the nearest integer and the corresponding elements of M_r^n are set to 1. The Matlab function 'bwcovhull' is used to convert all the pixels internal to the corner points to 1's. For simplicity, this method of projection does not require knowledge of the illumination source, nor does it calculate the effect of shadowing. These inaccuracies contribute to measurement noise. Combining these for all components gives the full projected silhouette binary image M_r. This information can be compared to the measurement silhouette M_m to give the measurements as follows.

The first two measurements ($\hat{z}_o$ and $\hat{z}_u$) quantify the number of pixels over-covered (1's in the reprojection but 0's in the measurement) and under-covered (1's in the measurement but 0's in the reprojection) respectively. The variable I is the indices of M that are equal to 1 and the variable m is the sum of the elements of M.

$$\hat{z}_o = \frac{1}{m_r} \sum 1 - M_m(I_r) \tag{130}$$

$$\hat{z}_u = \frac{1}{m_r} \sum 1 - M_r(I_m) \tag{131}$$

In some cases these two measurements alone may be sufficient, however including metrics specific to each component was found to improve performance.

For each component, the over-coverage z_{o_n} can be calculated similar to equation (130).

$$\hat{z}_{o_n} = \frac{1}{m_r^n} \sum 1 - M_m(I_r^n) \tag{132}$$

However, the under-covered quantity cannot be calculated for each component since it is unknown which parts of M_m correspond to which components. Instead, n_d^n points are sampled along the edges of each component. The Euclidean distance (d) from each of these edge points to the nearest point in the measurement not covered by

the full reprojection is calculated. These distances are then mapped to a zero mean normal distribution with $\sigma = 40$ and averaged to give $\hat{z}_{d_n}$.

$$\hat{z}_{d_n} = \frac{1}{n_d^n \sqrt{2\pi}\sigma} \sum e^{\frac{-d}{2\sigma^2}} \tag{133}$$

The $\hat{z}_{d_n}$ measurements quantifies the error for each component using the distance from the component to pixels under-covered by the reprojection. Under-covered pixels close to component n are likely due to incorrect placement of component n and therefore will have a large contribution to $\hat{z}_{d_n}$ while error pixels far from c^n are unlikely due to incorrect placement of component n and therefore will have a minimal contribution to $\hat{z}_{d_n}$. One limitation of this approach is that if the filter diverges to the point where the reprojected component is no longer overlapping the measurement, it will continue to diverge because moving closer to the measurement actually would increase the residual ($\hat{z}_{d_n}$) in that case. However, as long as the component maintains overlap, the measurement will serve to minimize the under-covered pixels.

The method described results in a total of $2 + 2N$ measurements where N is the number of components. Evaluating this measurement model is the most computationally expensive part of the UKF and it must be completed for every sigma point at each update of the UKF. In experimentation with $np_x = np_y = 800$ and six primary satellite components, each update takes approximately 1.7 seconds using single core processing (this does not include the image processing time to build the silhouette). In practice, the measurement model evaluation could be easily run in parallel to speed up processing time.

5.3.5 Implementation.

Given a set of initial images where the satellite is stationary, search methods could likely be implemented to find the initial pose and articulation angles. These methods

have not been investigated, but work by Cheung et al. [12] on shape from silhouettes gives insight into how they may be developed. Instead, the filter is initialized with the true pose and articulation angles, but with zero rates.

Given an initialization state vector, the dynamic model, and the measurement model previously discussed, the UKF can be implemented as described in section 2.2.11 with one slight change. In this implementation, the actual measurement is a binary $np_x \times np_y$ matrix M_m not a scalar quantity. The calculated measurement vector $\hat{\mathbf{z}}$ is in fact a set of metrics which compare the reprojected silhouette image set $M_r^{1:N}$ to the measurement M_m. This means that the measurements $\hat{\mathbf{z}}$ are really equivalent to the residuals $\mathbf{r}$ from equation (34). In the UKF framework of section 2.2.11 the residual is defined as $\mathbf{z}_{meas} - \hat{\mathbf{z}}_t$ so in this implementation the update equation (equation (35)) becomes $\hat{\mathbf{x}}_t^+ = \hat{\mathbf{x}}_t^- K(-\hat{\mathbf{z}})$.

The process noise matrix (Q from equation (28)) is calculated from the process noise values (η_x) in the same way as in section 4.3.3. The measurement noise was determined by propagating the true state vector through each time step and evaluating the measurements against the simulated silhouette image. The mean of each measurement over all frames was taken as the measurement noise value for that measurement.

The weights $\mathbf{w}_m$ and $\mathbf{w}_c$ for the UKF were determined from the parameters α, β, and κ using standard equations where n is the number of states [86].

$$\gamma = \sqrt{n + \lambda} \tag{134}$$

$$\lambda = \alpha^2(n + \kappa) - n \tag{135}$$

$$\mathbf{w}_m^1 = \frac{\lambda}{n+\lambda} \tag{136}$$

$$\mathbf{w}_c^1 = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \tag{137}$$

$$\mathbf{w}_m^i = \mathbf{w}_c^i = \frac{1}{2(n+\lambda)}, \quad \text{for } i = 2, \cdots, 2n+1 \tag{138}$$

The value of γ dictates how far the sigma points will be separated from the mean. In this implementation κ was set to $\frac{4}{\alpha^2} - n$ to make $\gamma = 2$ which spreads the sigma points 2σ from the mean. The value of α was set to 0.1 and the distribution was assumed to be Gaussian so β was set to 2.

5.3.6 Computational Requirements.

The computational requirements for this method are higher per update than the EKF method from Chapter V. In this implementation with 800×800 images each update takes approximately 1.7 seconds and does not include the time to process an image into a silhouette. The start-stop scenario outlined in section 4.4.3 would require an update every 15 seconds. The majority of that time is spent evaluating the measurement model for $2n + 1$ sigma points. Computational time could be decreased even further by increasing coding efficiency in the measurement model. Additionally, since the measurements model evaluations are independent, they could be run in parallel to enable real-time operation.

5.4 Results

Three scenarios were run to demonstrate this method. The first two consist of a full NMC with linear articulation and sinusoidal articulation respectively. The third investigates the ability to sense and accommodate articulation start and stop over a portion of an NMC consistent with existing space robotic arms articulation speeds as outlined in section 4.4.3.

5.4.1 Full NMC Results.

The first scenario evaluated is an NMC with linear articulation occurring over the length of the simulation. The simulation consists of 720 frames which translates to

one frame every 2 minutes if the primary satellite is in a geosynchronous orbit. Noise is added to the inspection route as it is in section 4.4. A diagram of the inspection route as well as some of the sampled and reprojected images are shown in Figure 29.

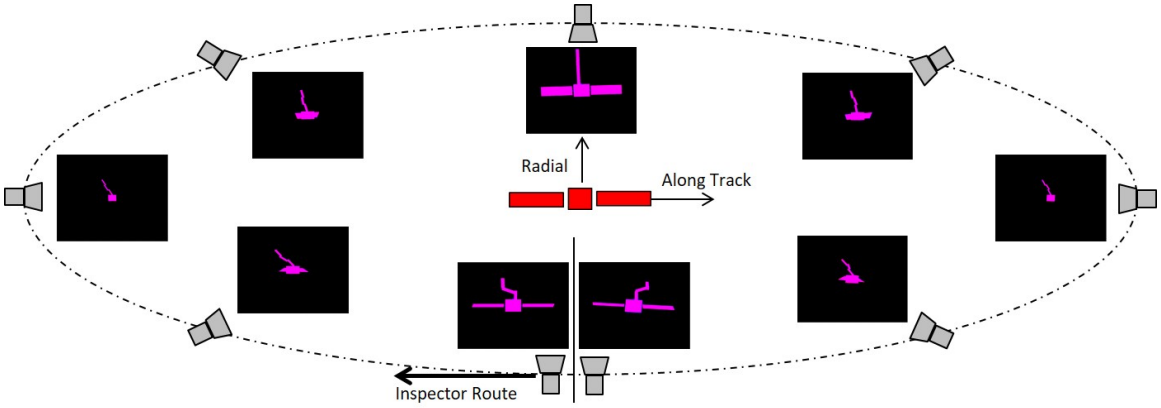

Figure 29. Diagram showing example silhouette images at frames throughout the NMC. The reprojection of the state vector shown in red is overlaid with the input silhouette image shown in blue. The purple color is where the two images overlap.

The estimated articulation angles and rates for each joint are shown in Figure 30 and demonstrate the method is capable of tracking the articulated motion. Notice errors increase for joints 1 and 2 when they are completely occluded from view. When they return to view, the filter is able to improve the estimate.

5.4.2 Sinusoidal Articulation Movement.

To demonstrate robustness to articulated motion that is non-linear, the scenario above was repeated with a single period of sinusoidal articulation instead of linear articulation. The dynamics used to propagate states in the filter were kept at constant articulation rate. Results are shown in Figure 31. The filter is able to track the motion even though it is non-linear. The extent to which the filter is capable of tracking motion that does not match the dynamics has not been investigated further. To further accommodate non-linear motion, multiple motion models could be developed and interchanged based on anticipated motion types.

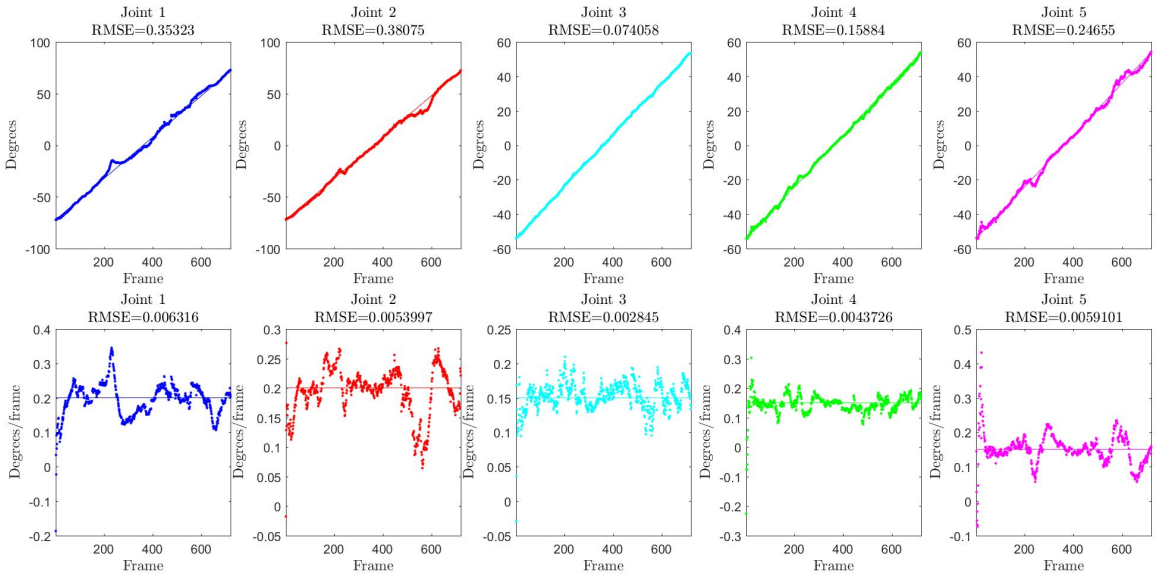

Figure 30. Results from full NMC with linear articulation. Top row plots are articulation angles compared to truth. Bottom row plots are articulation rates compared to truth.

5.4.3 Start/Stop Scenario.

The final scenario tested is the same as the scenario outlined in section 4.4.3 and shown in Figure 24. The Mahalanobis distance was once again used to sense the change and inflate the process noise as described in section 4.4.3 with a noise inflation threshold of $\Psi = 125$ and noise reduced by 50% each frame after the trigger.

5.4.4 Model Sensitivity Assessment.

A perfect articulated model was used for the previous results. It is interesting to investigate how robust this method is to inaccuracies in the truth model. To do this, the full NMC scenario with linear articulation was run with various modifications to the articulated model. One factor was modified at a time, and the simulation was run with the a consistent inspection route noise profile. To modify the axis, the azimuth and elevation were changed randomly a total of 5°. To modify a joint a random vector 0.2 units in length was added to the parent joint location. 10 trials were run for each

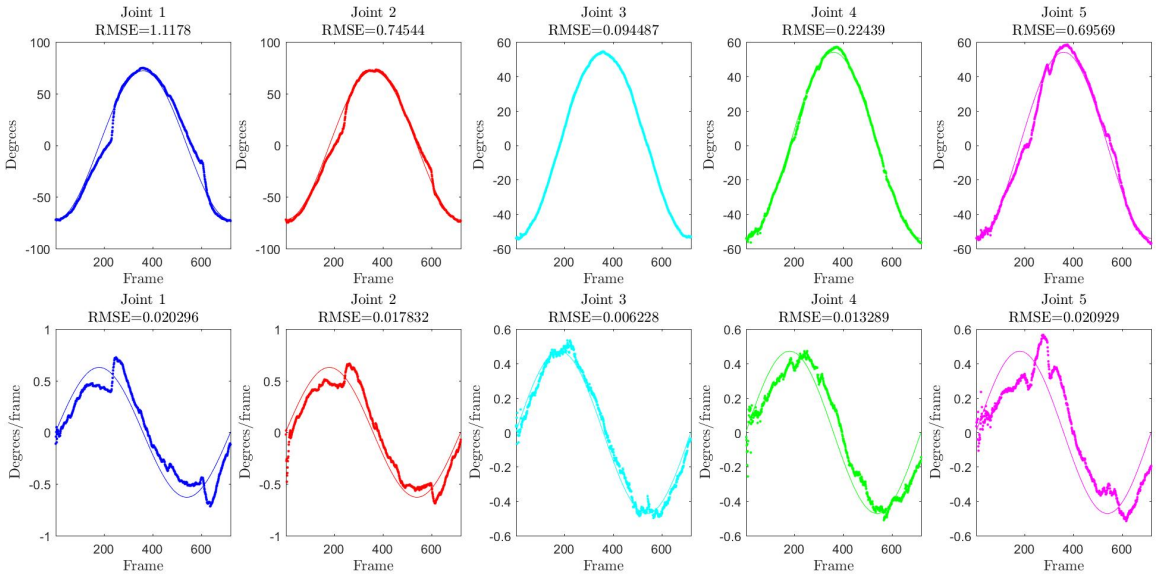

Figure 31. Results from full NMC with sinusoidal articulation. Top row plots are articulation angles compared to truth. Bottom row plots are articulation rates compared to truth.

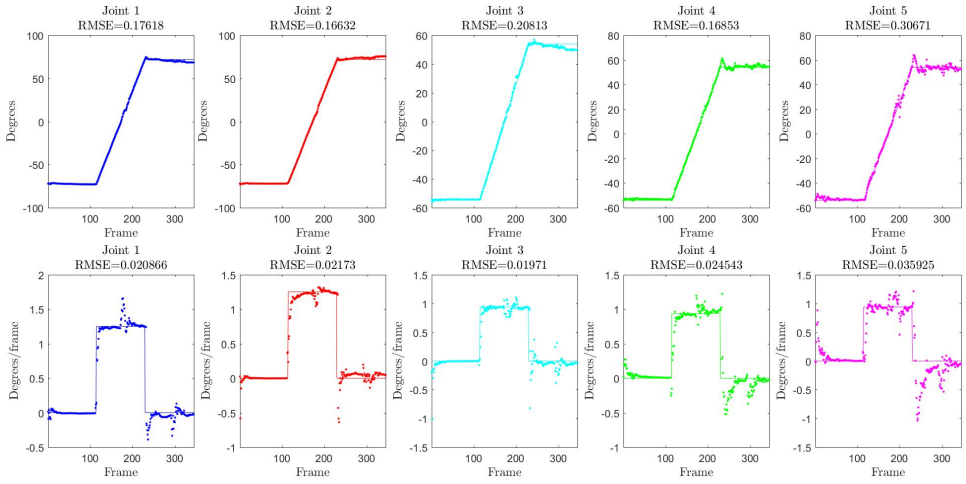

Figure 32. Results from 6% of an NMC with articulation start and stop. Top row plots are articulation angles compared to truth. Bottom row plots are articulation rates compared to truth.

factor and the results were averaged to give the values (root mean square articulation angle error) shown in Figure 33.

The filter was able to accurately track the motion of the articulated arm in most cases even with the inaccuracies in the model. Modification to any axis or joint lead to inaccuracies in tracking both panels, but particularly the first panel. The

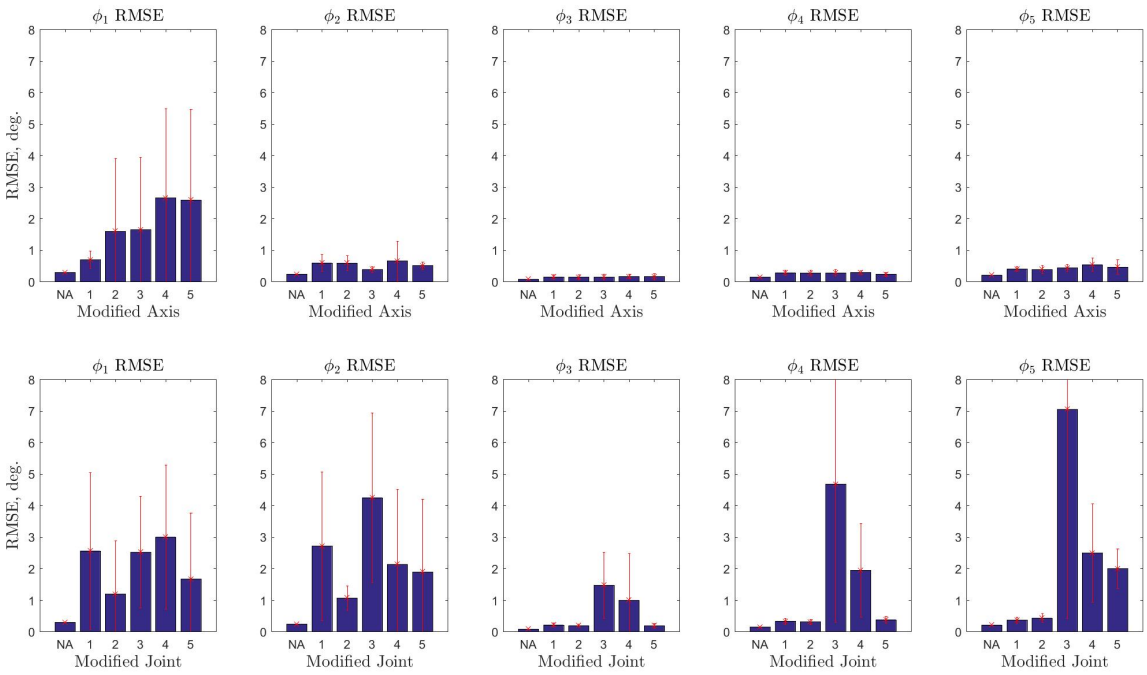

Figure 33. Model articulation axis (top row) and joint location (bottom row) modification effects on articulation tracking. Bar heights are averaged values over 10 trials; error bars are 1σ.

likely reason for this is the occurrence of an ambiguity outlined later in section 7.2.3, specifically as shown in Figure 47b. When an articulation axis and the object face are perpendicular to the optical axis, rotations in either direction will look the same. Due to the nature of the particular maneuver and noise used in these simulations, panel one is closer to the ambiguity than the other, causing the panel to diverge more often when the model is inaccurate. To remove this issue, the inspection route was modified slightly so that the panels were not as close to the ambiguity at any point during the inspection route. The results, as shown in Figure 34, demonstrate that filter can accommodate slight inaccuracies in the model, and, as expected, the angle measurements are most effected by modification to the specific angle/joint or one below it in the kinematic chain.

Using the modified inspection route, the effect of inaccuracies in the component sizes was also investigated. The size of each of the components was modified by

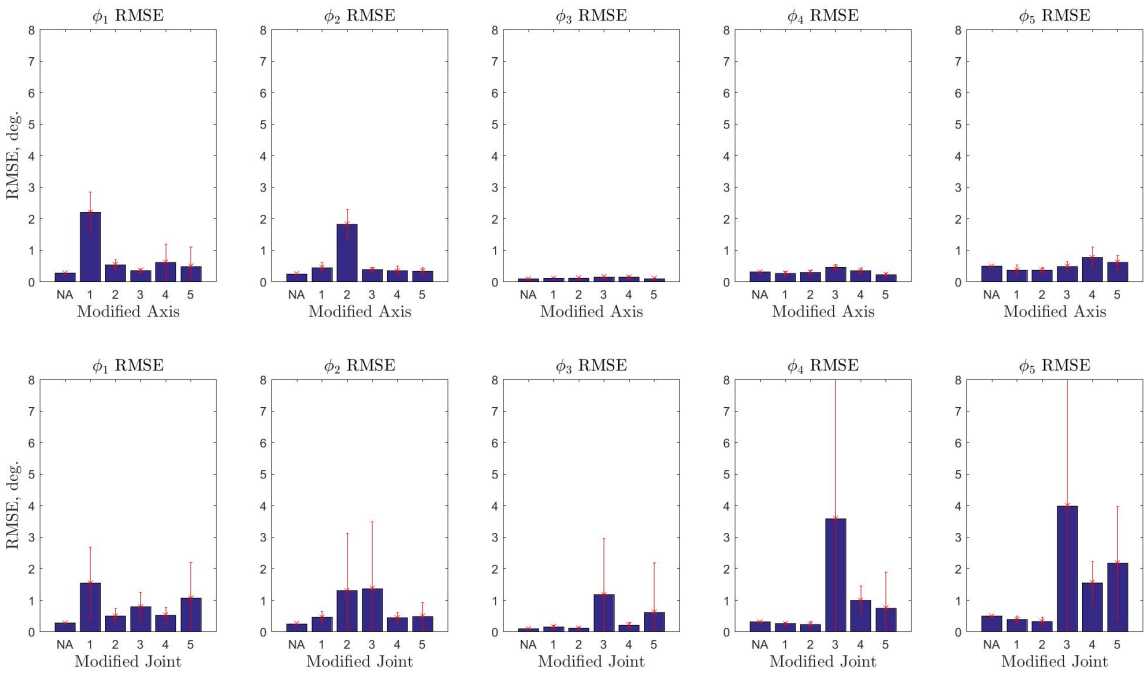

Figure 34. Results when inspection route is modified to reduce the ambiguity. Model articulation axis (top row) and joint location (bottom row) modification effects on articulation tracking. Bar heights are averaged values over 10 trials; error bars are 1σ.

increasing/decreasing the length, width, and height of the rectangular prism representing the component by an amount between ±10%. Results are shown in Figure 35. For a constant inspection route with constant inspection route noise, the results are deterministic, so a single trial was done at 2% increments between ±10%. Generally, the filter is able to accommodate the inaccuracies in the model, however when the root component size is reduced more than 5% the filter diverges, likely due to inaccuracies in the estimated root component pose. The remaining spikes in the data correspond to divergence of the filter caused by occlusion of the panels.

While the this method can accommodate some inaccuracies, issues can arise, particularly near ambiguous poses or when significant occlusion occurs. Methods could be developed to accommodate these issues such as noise inflation when an ambiguity or occlusion is sensed, allowing the filter to recover the correct articulation rather than diverging.

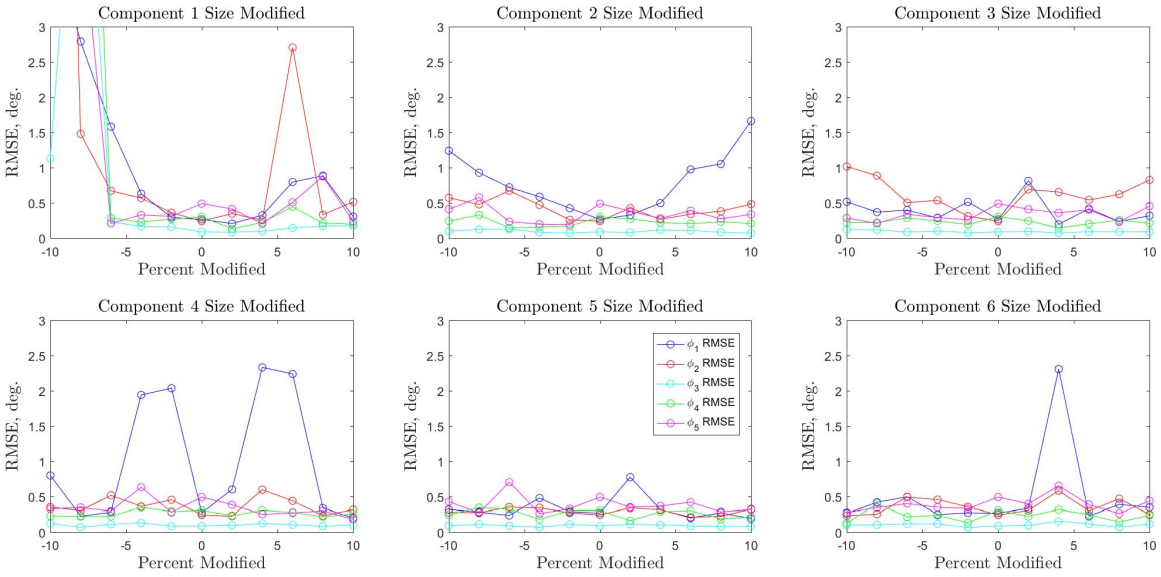

Figure 35. Model component size modification effects on articulation tracking.

5.5 Conclusions

This work demonstrates an additional method of tracking a satellites articulated motion from monocular imagery. Instead of using feature points, the method demonstrates that silhouette images can be used to track articulated motion. The silhouette method tracks the articulated angular rates with much higher accuracy, however for the silhouette method the exact model was given as prior knowledge, while the feature point method estimated some of the stationary aspects of the model. Sensitivity to inaccuracy of the satellite model was investigated with some robustness to model inaccuracies demonstrated. Additionally, if the satellite model is know to be inaccurate this method could be adjusted to include other articulation parameters in the state vector.

While the method has been demonstrated in scenarios with non-linear articulation, in situations where the motion is highly non-linear alternative methods of tracking such as those used for tracking human motion could be employed [28]. Additionally, the quality of silhouette type methods benefit from multiple cameras taking images

at different angles. While more complicated and expensive, if accuracy is desired a system using multiple inspector satellites could be developed using similar methods.

VI. Real Imagery

6.1 Chapter Overview

This chapter consists of specifications for a satellite model, the collection method of real imagery in a simulated space illumination environment, and results when applying the method outlined in Chapter V for tracking articulation to meet Objective 3 of this research. Partial results are also presented in a submitted journal article [20]. This chapter consists of an introduction (section 6.2), an overview of the imagery collection setup (section 6.3), results of tracking using the developed UKF (section 6.5), an overview/results for a modified human articulation tracking algorithm from [81] (section 6.6), and a conclusion (section 6.7).

6.2 Introduction

A satellite model similar to the one used for simulation (Figure 27) was built and used to collect imagery a simulated space illumination environment. An inspection route was simulated by rotating the satellite, moving the camera forward and backward on a track, and moving a single light source to simulate the Sun location. Images were taken at discrete locations on the inspection route with the articulation angles of the five joints on the model moved slightly between frames giving a set of images mimicking those that would be collected of an articulating satellite from a satellite on a near by inspection route. Results are presented when using these images as the input images for the silhouette method outlined in Chapter V. Additionally, an algorithm created for human tracking (the Annealed Particle Filter [81, 28]) was modified to track the satellite with results presented.

6.3 Setup

This section contains information on the physical satellite model and how images were collected to simulate the space illumination environment.

6.3.1 Satellite Model Specifications.

The satellite model was constructed out of wood. The root component (main body) is $8'' \times 8'' \times 8''$. Each of the panels are $16'' \times 6'' \times \frac{1}{8}''$ with a $\frac{3}{4}''$ dowel mounted to the back and used to connect the panel to the center of the appropriate root component side. Each of the components of the arm are $1.5'' \times 1.5'' \times 12''$ with rounded ends to accommodate articulation angle markings. They are attached to each other and the root component with $1/4 - 20$ machine screws and wing-nuts. The root component is covered in Kapton and the panels are covered in paper to mimic solar panels. The components on the arm are painted gray. Various items were mounted to the root component to simulate satellite components. Figure 36 shows images of the satellite with labeled dimensions. The joints on the arm were marked at 5° increments.

6.3.2 Image Collection.

Images were taken in a $50' \times 70'$ black room within the Air Force Research Laboratory (AFRL) $\mu AVIARI$ which supports micro air vehicle testing. The satellite was mounted to a tripod on a bearing allowing it to spin about the vertical axis. A camera was mounted to a second tripod which was placed on a platform that rolled forward and backward on a track. A set of two lights was placed behind the track. The platform was moved forward and backward, the satellite was rotated on the tripod, and the light was moved left and right to simulate an inspection route and Sun lighting. The image collection setup is shown in Figure 37.

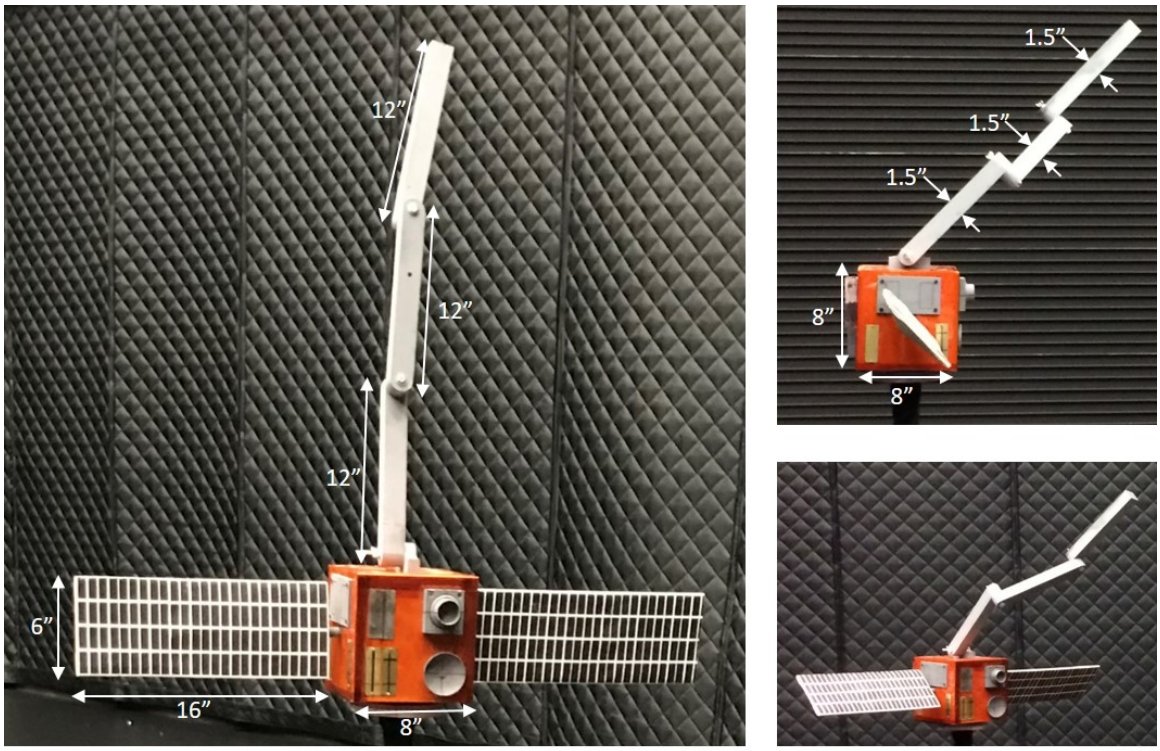

Figure 36. Satellite model dimensions

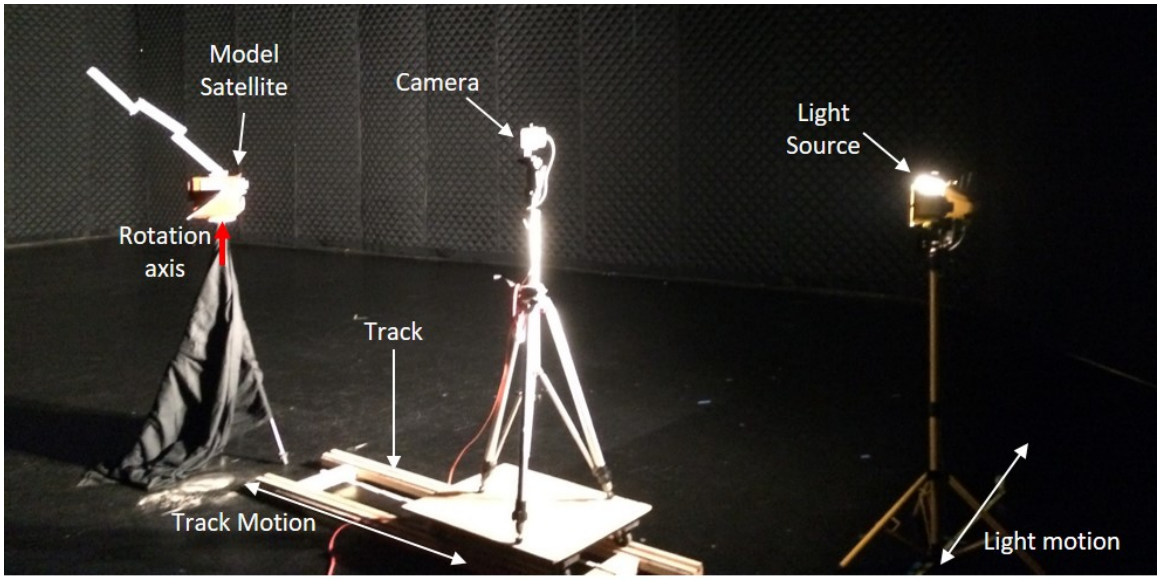

Figure 37. Imagery collection setup

An inspection route with articulated motion was simulated using stop-motion photography. The inspection route shown in Figure 38a was discretized into 90 frames.

At each frame, the rotation of the satellite with respect to the camera frame, distance from the satellite to the camera, and angle between the camera and Sun were calculated and used to determine the respective position of objects as shown in Figure 38b to simulate the inspection route. Between each frame, the joints were articulated approximately 1° each. The joints on the arm were marked at 5° intervals, and setting for the 1° increments were estimated. The panel joints were not marked, so a small articulation was applied at each interval. This imprecise motion was assumed to represent real 'noise' in an actual set of measurements on orbit.

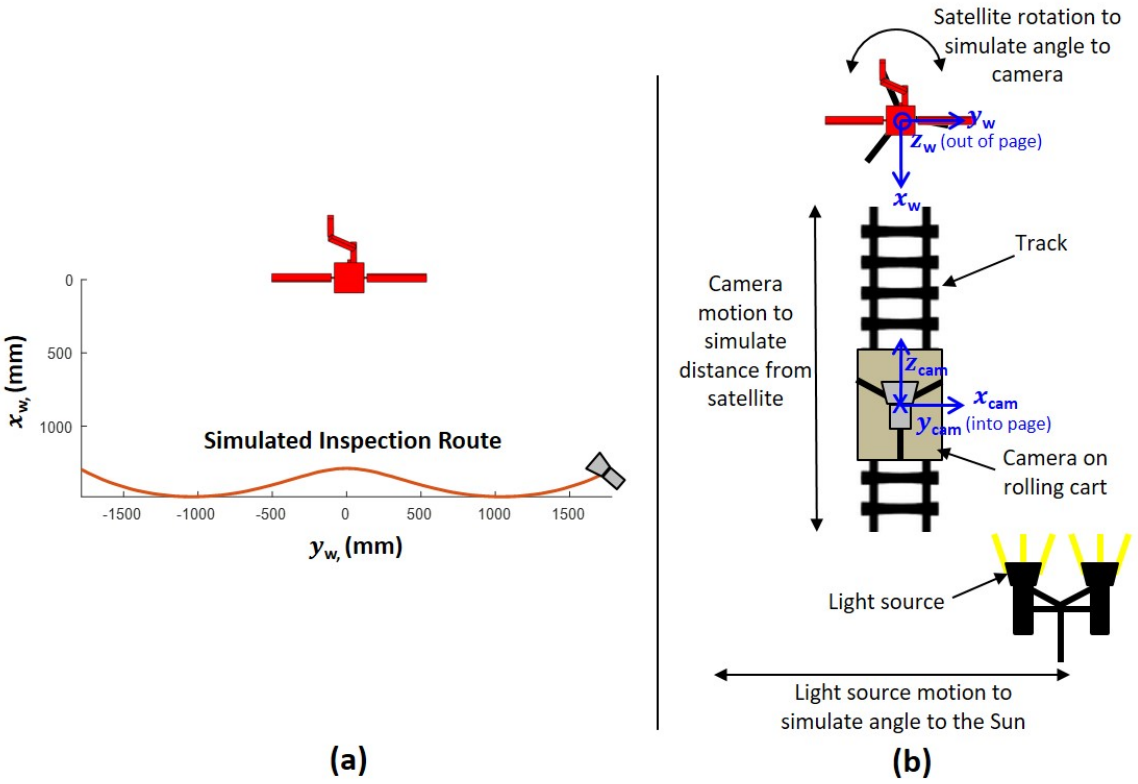

Figure 38. Inspection route simulation diagram. a) Inspection route of camera in world frame. b) Image capture setup.

6.3.3 Camera Calibration.

An iPhone 7 camera consisting of a 3024×4032 pixel array was used for data collection. Camera calibration was performed to determine the intrinsic parameters

of the camera using Matlab's Single Camera Calibration App. The resulting camera calibration matrix P_{cam} is as follows where f_x and f_y are the focal lengths, expressed in pixels, in the camera's x and y directions respectively. The principal point of the camera (x_0 and y_0) is at the location of the camera's optical center (in pixels) on the pixel array.

$$P_{cam} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3269 & 0 & 2030 \\ 0 & 3270 & 1474 \\ 0 & 0 & 1 \end{bmatrix} \tag{139}$$

Since the inspection route was being simulated as described the extrinsic portion of the camera calibration matrix was not used. The radial distortion was also not applied because it was small for this camera.

6.4 Method

The model building method of Chapter III and the tracking method of Chapter IV were not applied to the real imagery collected due to the noise in image collection. The camera/satellite motion used to simulate the inspection route were not accurate enough for these methods to be applied successfully. However, the silhouette tracking method from Chapter V was applied to the real imagery collected.

6.4.1 Image Processing.

The steps to convert an image to a silhouette image are fairly straight forward using Matlab's Image Processing toolbox. They are shown in sequence order in Figure 39. Most of the process is automated, however the tripod was removed manually from the image. Even though it was covered in a black sheet it was very bright in the

images. Since the pedestal would not be required to hold up the satellite in space, removing it manually does not detract from the realism of the experiment.

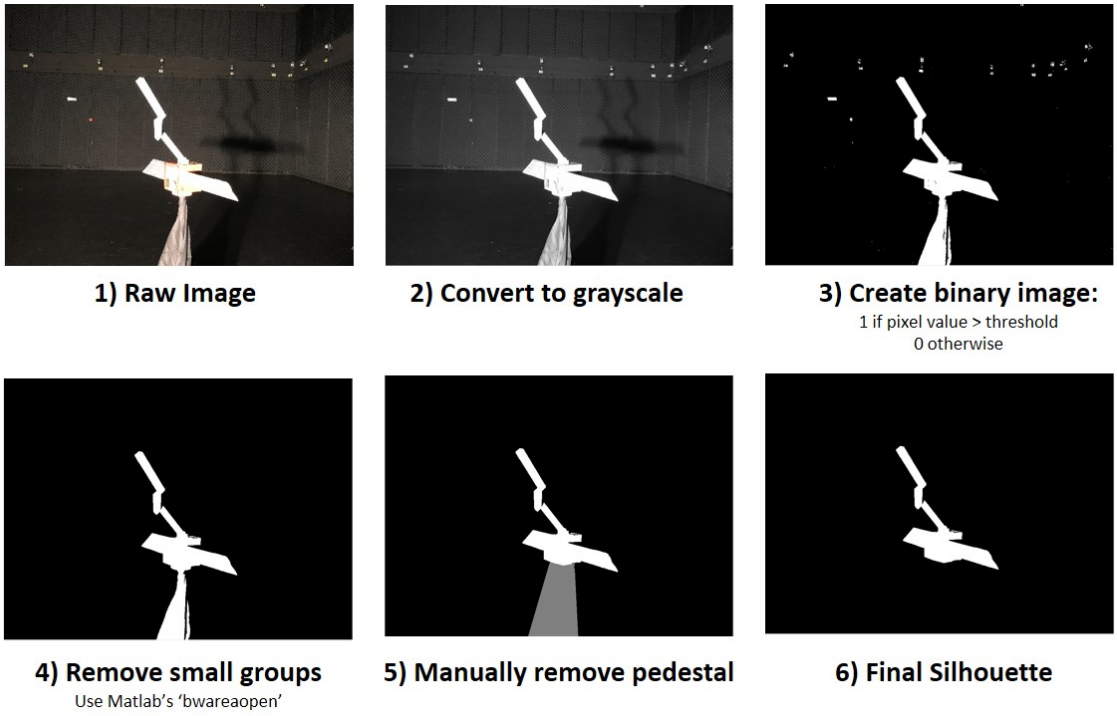

Figure 39. Steps to create a silhouette image.

6.4.2 UKF Settings.

The parameters of the UKF are set as shown in Table 7. The values for the measurement noise were determined by manually setting the state vector to align with the first image frame and calculating the measurements described in section 5.3.4. These measurements represent the error in the image when the model is 'perfectly' aligned and are therefore used as the measurement noise values.

The filter was manually initiated with the 'perfect' state based on the first image silhouette. The rate states (ω,$\dot{T}$, and $\dot{\phi}$) were initiated at zero.

Table 7. Noise settings

Initial covariance values						Process noise values		
σ_q^2	σ_T^2	σ_ω^2	$\sigma_{\dot{T}}^2$	σ_ϕ^2	$\sigma_{\dot{\phi}}^2$	σ_ω^2	$\sigma_{\dot{T}}^2$	$\sigma_{\dot{\phi}}^2$
$1e-6$	$1e-2$	$1e^{-}6$	$1e-2$	$1e-6$	$5.8e-3$	$1e-10$	$1e-8$	$1.5e-6$

Measurement noise (10^{-3})													
$\sigma_{\hat{z}_o}$	$\sigma_{\hat{z}_u}$	$\sigma_{\hat{z}_{o_1}}$	$\sigma_{\hat{z}_{d_1}}$	$\sigma_{\hat{z}_{o_2}}$	$\sigma_{\hat{z}_{d_2}}$	$\sigma_{\hat{z}_{o_3}}$	$\sigma_{\hat{z}_{d_3}}$	$\sigma_{\hat{z}_{o_4}}$	$\sigma_{\hat{z}_{d_4}}$	$\sigma_{\hat{z}_{o_5}}$	$\sigma_{\hat{z}_{d_5}}$	$\sigma_{\hat{z}_{o_6}}$	$\sigma_{\hat{z}_{d_6}}$
87.3	23.2	70.7	9.39	27.5	9.30	76.6	9.04	117	9.73	88.1	9.86	197	9.30

6.4.3 Computational Time.

Due to the larger image size, the computation time was longer than simulation with an average update requiring 70 seconds without the image processing steps to create the silhouette. This time could likely be reduced by reducing the image size or parallelizing the measurement function evaluations.

6.5 Results

Qualitative results for 12 frames are shown in Figure 40. In each image, the model reprojected from the estimated state vector ($\hat{\mathbf{x}}_t^+$) is shown in cyan by eliminating the red component of the original image at the occupied pixels in the model reprojection. As can be seen, the reprojection closely tracks the satellite at all frames.

Since precise truth data is not known from the articulation angles, quantitative comparison of accuracy is difficult. Figure 41 shows the estimated articulation angles for each frame. No attempt was made to gather truth data for the panel articulation, so the input angle is not shown, however the tracked motion does have trends appropriate with the input angle trends. Notice the large changes to the articulation rate near the middle of the simulation. These frames correspond to a time in the articulation at which there is an ambiguity in the panel measurement. At this point, the panel axis is nearly perpendicular to the camera axis and the panel outward normal is nearly parallel to the camera axis. This means a rotation in either direction will

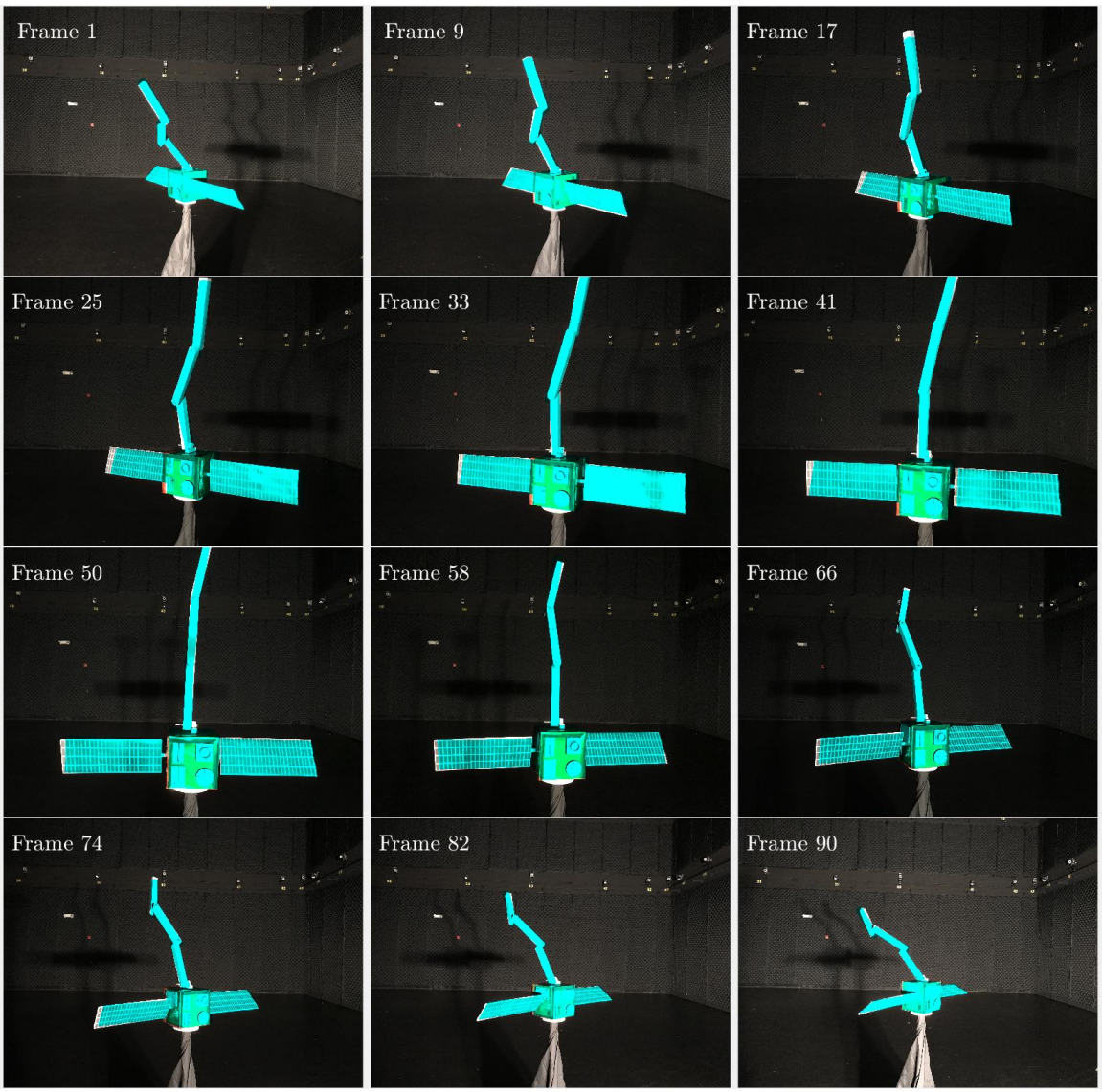

Figure 40. Qualitative results using real imagery. Each frame consists of the raw image with the reprojected silhouette overlaid in cyan.

look nearly the same as described in section 7.2.3.

An attempt was made to articulate the joints on the arm at a rate of 1° per frame from 45° to −45°, so the estimation is compared to the input articulation angle. The articulation method likely resulted in true articulation angles of ±2°. In all cases, the UKF was generally able to track the articulated motion.

The difference between the initial settings for rotation and translation of the root

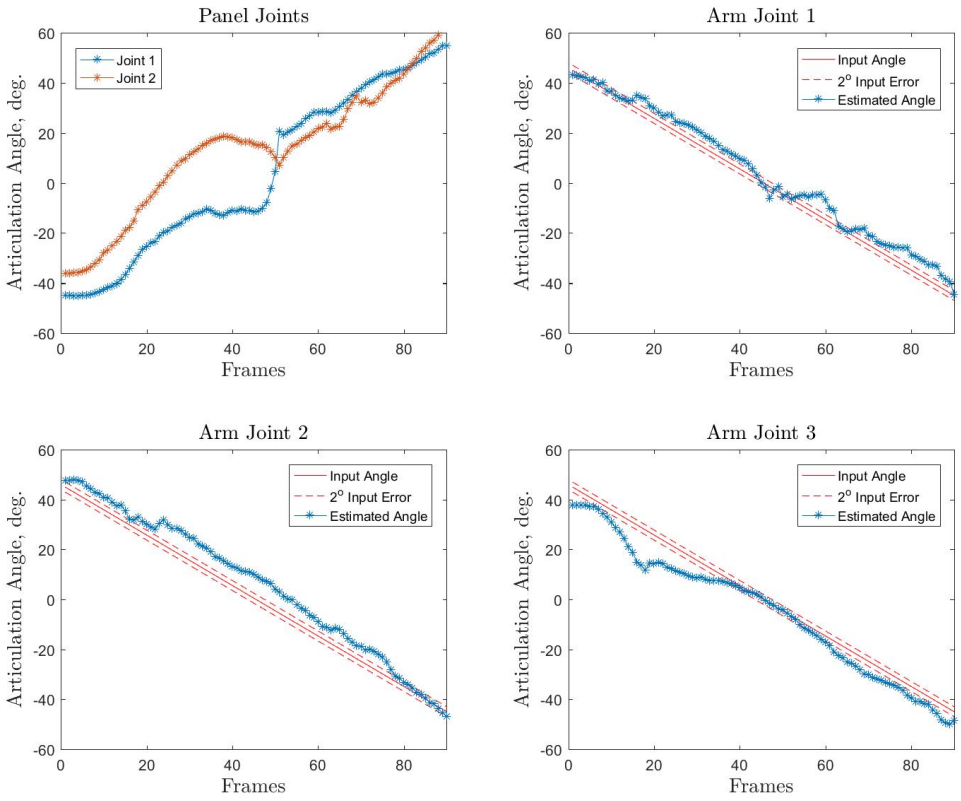

Figure 41. Estimated articulation angle results for all 5 joints. Panels input angles were not captured. Arm input angles were linear, however $\pm 2^{\circ}$ input inaccuracy is likely.

component and the estimated values are shown in Figure 42. While the satellite was rotated during image collection, the rotation was to simulate the movement of the camera along the inspection route, not a rotation of the satellite in the world frame. The satellite was intended to remain stationary throughout image collection in the world frame, therefore the values in Figure 42 are the errors in the rotation and translation estimates. The camera position and the root component body frame are both defined with respect to the world frame, however the world frame is arbitrary. Therefore, errors in the pose of the root component are compounded by errors in the pose of the camera in the world frame. Given the simple setup for image collection there are numerous sources of error in the camera pose such as camera orientation in

the tripod, satellite orientation on the tripod, accuracy in satellite rotation, accuracy in camera location on track, and track alignment. Success under these conditions suggests success in a realistic scenario with precision relative guidance, navigation and control.

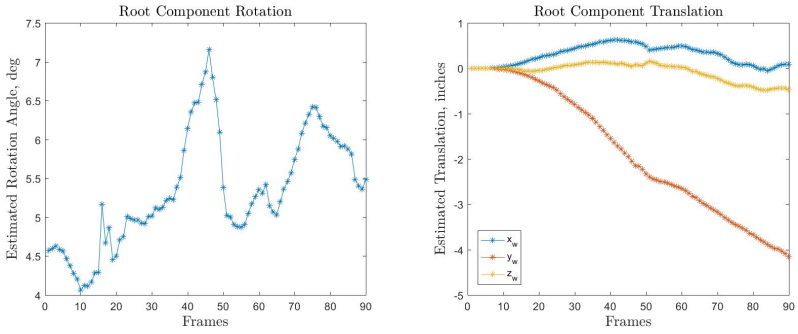

Figure 42. Estimated root component rotation and translation.

6.6 Modified HumanEva Baseline Algorithm

The problem of tracking the articulation of the satellite given a known model is nearly identical to the problem of tracking the motion of a human. As outlined in Chapter II, there is extensive work in the field of tracking human articulated motion. To demonstrate the utility of applying human tracking methods to satellite articulation tracking, the baseline algorithm available with the HumanEva dataset [81] was modified. The baseline algorithm is an implementation of the annealed particle filter (APF) introduced for human tracking by Deutscher and Reid.[28] The APF is a modification to the traditional particle filter in which multiple filtering layers are employed. The highest layer of the filter is evaluated first with particles spread over a larger search space. To transition to the next layer particles are re-sampled according to their likelihood. This allows fewer particles to be used to search high dimensional spaces. Both [81] and [28] provide in-depth explanations of the APF.

With relatively few adjustments, this baseline algorithm was capable of tracking

the motion of the satellite using the imagery collected. The main modification to the existing code consisted of modifying the articulated model from a 40 DOF human skeleton with limbs represented by cylinders to an 11 DOF satellite with components represented by rectangular prisms. Additionally, the code was modified to use only one camera. Multiple synchronized cameras are available and used in the Sigal et al. implementation, however by providing an accurate initial state, reducing the state covariance, and using the more computationally expensive bi-directional silhouette likelihood function the algorithm demonstrated capability to track the articulated motion with one camera.

6.6.1 Results.

The APF was run with 3 annealing layers each containing 200 particles. With these settings, each update took approximately 11.5 minutes running on a desktop computer without parallel processing. The likelihood function evaluates each particle against a silhouette map by comparing it with sampled pixels within each reprojected component and against an edge map by comparing it with sampled points along the outside edge of the reprojected components.

Figure 43 shows the resulting reprojected edges of the weighted mean of the particles (red) and the best particle (green) overlaid on the actual image at sampled time steps. Figures 44 and 45 show the estimated angle and root component translation/rotation.

These results are comparable to the results from the UKF method (Chapter V). Both are able to track the general trends of the articulation. The APF does not require a motion model, however the covariance of each state informs the span of the search space and could be used to expand/contract the search space based on estimated articulation rate capabilities. Reducing the covariance allows success with

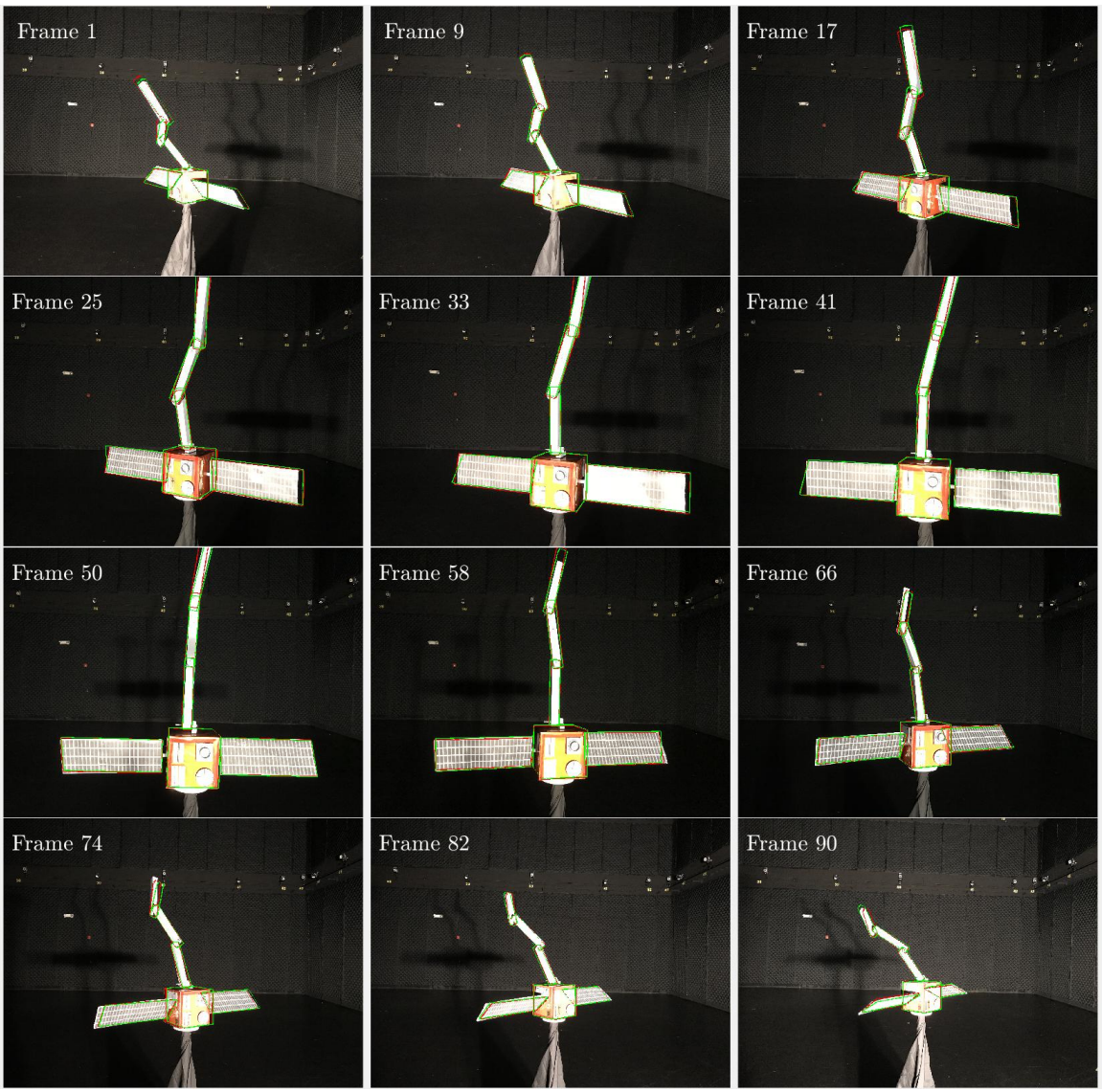

Figure 43. Qualitative results using real imagery. Each frame consistes of the raw image with an outline of the estimated state in red and the best particle in green.

fewer particles, but also deceases robustness to large changes in the state between frames. The UKF uses a motion model and therefore is less robust to articulation inconsistent with the measurement model, however this allows for fewer function evaluations and therefore a $10\times$ faster update rate.

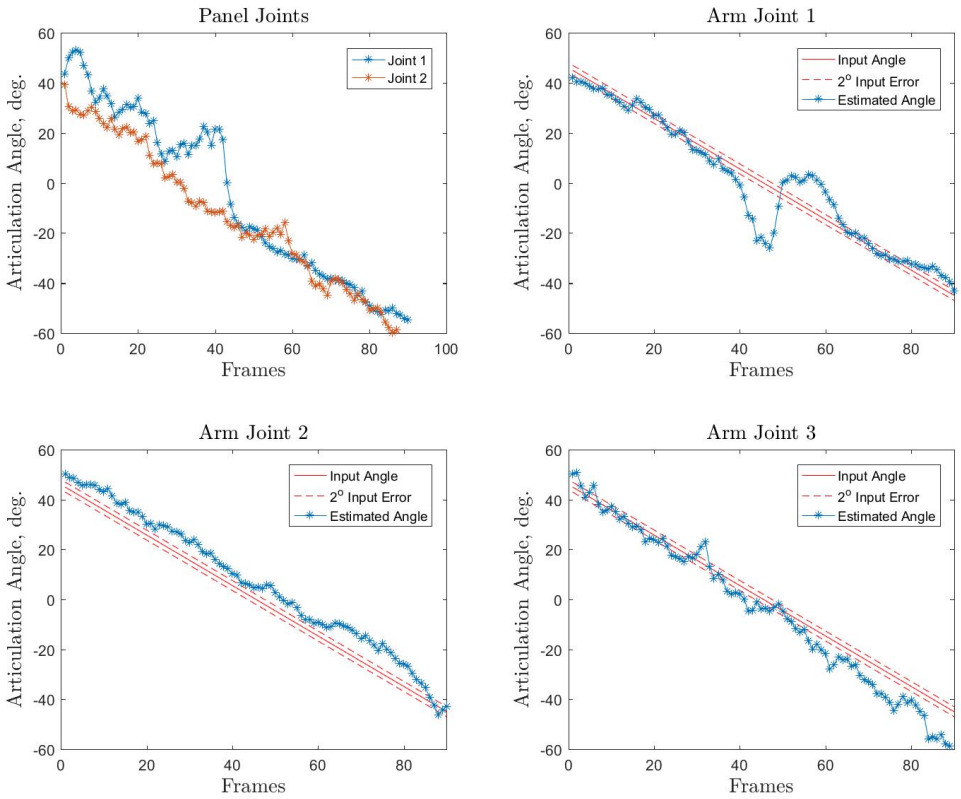

Figure 44. Estimated articulation angle results for all 5 joints. Panels input angles were not captured. Arm input angles were linear, however $\pm 2^{\circ}$ input inaccuracy is likely

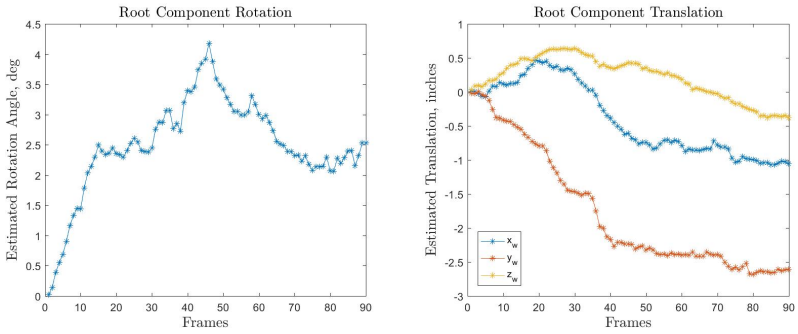

Figure 45. Estimated root component rotation and translation.

6.7 Conclusion

The images collected in a space representative illumination environment demonstrate the capabilities of the UKF silhouette based tracking method. They also were

used to demonstrate the applicability of human motion tracking algorithms such as the APF for the problem of articulation tracking. The feature point based methods of Chapters III and IV were not applied to the real imagery primarily because the image capture method did not allow for enough accuracy in the inspection route. Additionally, as evident in Figure 26, there are very few feature points that are available on the articulated arm.

VII. Conclusions and Recommendations

This chapter concludes the work by reviewing research objectives (section 7.1), providing considerations on the topic of satellite articulation sensing (section 7.2), summarizing the methods developed in this work (section 7.3), outlining the contributions of this work (section 7.4), and providing recommendations for future work (section 7.5).

7.1 Research Objectives

Three research objectives are listed below. Chapter III describes work toward Objective 1, Chapters IV and V describe work toward Objective 2, and Chapter VI describes work toward Objective 3.

7.1.1 Objective 1.

Objective 1: Develop a method of autonomously building an articulated model of a satellite through inspection with a monocular camera producing resolved imagery. This work consisted of development of algorithms and testing on simulated feature points from a nominal satellite with articulating panels and an articulating arm. The quality of the model created from inspection routes with various illumination conditions will be assessed.

Discussion: Objective 1 was met by the work presented in Chapter III. A method of building an articulated model from a trajectory matrix containing feature point information was presented and demonstrated. The effect of illumination angle was assessed, and it was demonstrated that an articulated model can be built without viewing the satellite from all sides. To accommodate uncertainty, the method was modified, and the effect of inspection route noise was assessed.

7.1.2 Objective 2.

Objective 2: Develop an estimation framework for tracking the articulated motion of the primary satellite sequentially as new imagery becomes available. This work consisted of development of an algorithm that uses an articulated model of the satellite and either simulated feature point locations or image silhouettes to track the articulated motion of the satellite.

Discussion: Objective 2 was met by the work presented in Chapters IV and V. Methods of tracking the articulation of a satellite using feature points or silhouettes from monocular imagery were developed and demonstrated. The developed silhouette tracking method from Chapter V was demonstrated to be effective with real imagery in Chapter VI. Additionally, an algorithm developed for human motion tracking was modified and demonstrated successful in tracking the articulation of a satellite in Chapter VI.

7.1.3 Objective 3.

Objective 3: Build a satellite model and collect stop motion imagery mimicking articulation in a simulated space lighting environment. Use captured imagery to validate developed tools as permitted by capture method uncertainties.

Discussion: Objective 3 was met by the work presented in Chapter VI. A satellite model consisting of two articulating panels and an articulating arm containing three joints was built. Stop-motion imagery was taken in a large dark room with a single light source to simulate the space lighting environment. The imagery was used to validate the silhouette tracking method outlined in Chapter V.

7.2 Satellite Articulation Sensing Considerations

In conducting this research many interesting aspects of this topic were revealed. This section attempts to provide an overview of these aspects for consideration of future researchers investigating the topic.

7.2.1 Building an Articulated Model.

Initial attempts to build an articulated model focused on building a complete model from an inspection route (such as an NMC) that viewed nearly all sides of the satellite. While viewing the satellite from all sides would be necessary to create point clouds that represent the entire shape of the satellite, the inaccuracies in stitching together sets of points not viewed in common frames led to inaccuracies in the overall articulated model. Figure 46 demonstrates this issue. However, if a full point cloud representing the entire shape of each component is less important, a more accurate articulated model can be created by only considering the viewing angles associated with one side of the satellite. Selecting the best inspection route will be dependent on the particular satellite being inspected, however there are a few things to consider to generally improve results:

- The inspection route should minimize component occlusion/shadowing.
- The inspection route should put the camera no more than 60° off of the Sun vector.
- The inspection route should minimize the missing data in the trajectory matrix.
- The inspection route should maximize the range of viewing angles.
- The inspection route should minimize noise in the relative pose estimate (for instance by minimizing jitter during image collection).

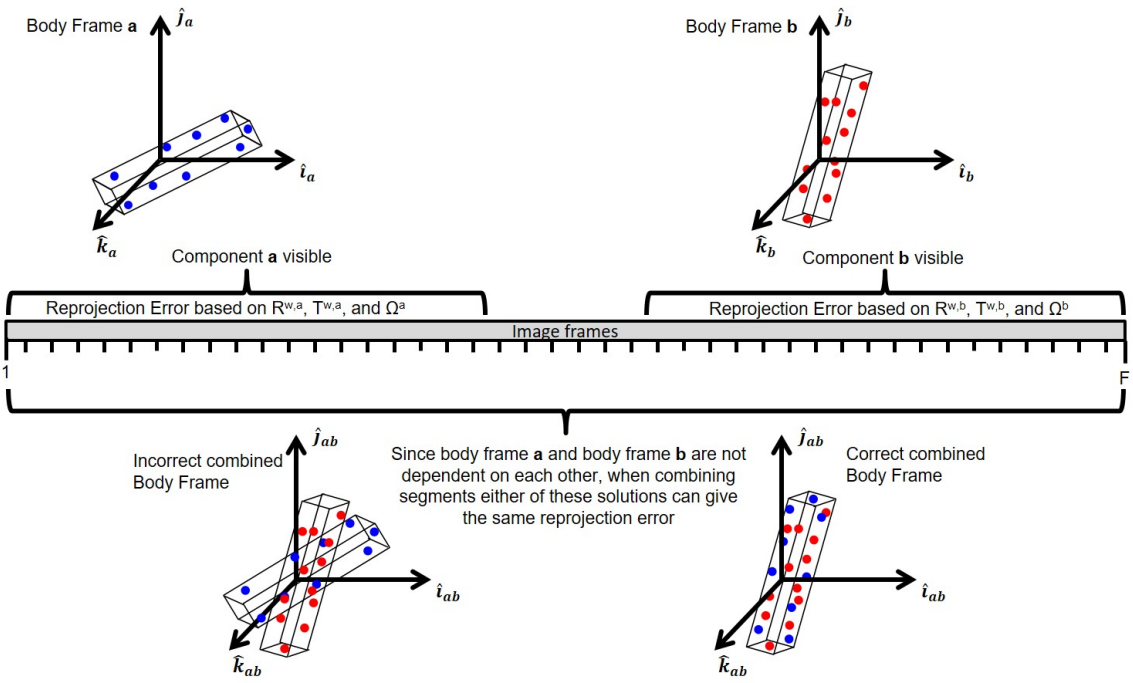

Figure 46. Illustration of ambiguity in creating a shape from points not seen together.

Consideration should also be given for alternative sensors that contain additional information. For instance, spectral sensors could provide useful data in segmenting components based on the spectral content of the return, or the 3D data available from a LIDAR sensor could remove ambiguity and complexity from any articulated model creation method.

While the methods outlined focus on autonomously building the articulated model, in certain situations it may make more sense for a method with a 'human in the loop'. Such a system could use a human operator input to determine the number of components, validate feature point segmentation, determine the kinematic chain, clear up ambiguities, or determine joint types. Additionally, given a few images of a satellite, a human operator could likely develop a kinematic chain and a set of primitive shapes that represent the shape of each component. This information could be used to build an articulated model with silhouette images.

7.2.2 Articulation Tracking.

Two articulation tracking methods were developed. Each used a different method of transforming image data to something that can be used as a sensor measurement for a recursive estimation filter. The method in Chapter IV used feature points while the method in Chapter V used silhouette. Each method has advantages and disadvantages. One of the main advantages of feature points is that they are capable of creating/maintaining a more detailed representation of the components. The shape is represented by a point cloud, so it can take on whatever shape best explains the data. The silhouette method on the other hand requires the shape to be defined by some basic shape (rectangular prism in this case). While the model could be defined by more complex shapes, additional complexity would increase computational time. The use of feature points also requires new feature points to be assigned to components as they are acquired. A method of assigning points to different components is outlined in section 4.3.7, however if a joint stopped articulating, the method would fail to assign points to the correct component. The silhouette method on the other hand does not require assignment of points. Additional comparisons between silhouettes and feature points are available in section 5.2.

Both of the developed articulation methods rely on a constant rate articulation motion model. Some robustness to non-constant rate articulation was demonstrated, however a full investigation would likely reveal some failures of a single motion model under some circumstances. For instance, if the articulation rate changes dramatically when a component is occluded, the filter will likely be unable to recover. Methods could be developed to rectify this limitation such as using multiple models, or automatically re-initializing diverged states. Alternatively, methods such as the APF (outlined in Chapter VI) that do not require a motion model could be employed.

As in building the articulated model, there may be advantages to using other

types of sensors. In addition, it would be advantageous to have multiple synchronized cameras. Multiple cameras would alleviate some of the ambiguities outlined in section 7.2.3. Work on human pose tracking shows the accuracy advantages when multiple cameras are used in tracking articulation.[46, 81]

The final consideration for articulation tracking is the realization of how closely it is related to the heavily researched field of human pose tracking. As mentioned in section 2.3.5, there is a significant body of work focused on determining how human joints are articulated from imagery. The articulated model of a human is known; it is the human skeleton. Aspects (such as limb length) may need to be different from person to person, but generally the model is the same. Therefore, the methods developed over the last few decades for human pose tracking can be applied to tracking the pose of a satellite with a known articulated model as demonstrated in section 6.6. In addition, rendering tools built specifically for simulating images taken in space could be used to produce 'training' data for methods such as [3] that learn to predict the articulation state from a single monocular image.

7.2.3 Monocular Vision Ambiguities.

Monocular imagery is a 2D representation of the 3D world. With this decrease in dimensionality there are some ambiguities that should be considered. A single 2D image of a scene contains no depth information, meaning that an identical image could be constructed with the objects in the image scaled and closer/farther from the camera. When attempting to build a model of an object that is viewed over multiple images, there are multiple solutions (as shown in Figure 47a). When considering the case of sensing articulation, this can lead to a rotation ambiguity. When an articulation axis and the object face are perpendicular to the optical axis, rotations in either direction will look the same as shown in Figure 47b.

Ambiguities also occur due to occlusions. When an object is viewed from one direction only, portions of the object can be occluded from view. This can cause problems in reconstructing the object as demonstrated in Figure 46 or could cause problems in tracking, particularly if the articulation rate changes when the component is out of view. For silhouettes, a component is also effectively occluded when it is directly between the camera and another component as shown in Figure 47c.

In the context of articulation sensing, particularly with feature points, an additional ambiguity arises when a joint is not active. If a joint is not active, the two connected components are moving as a rigid body. This means that points could be assigned to either body frame without consequence causing ambiguity in tracking methods that use feature point measurements.

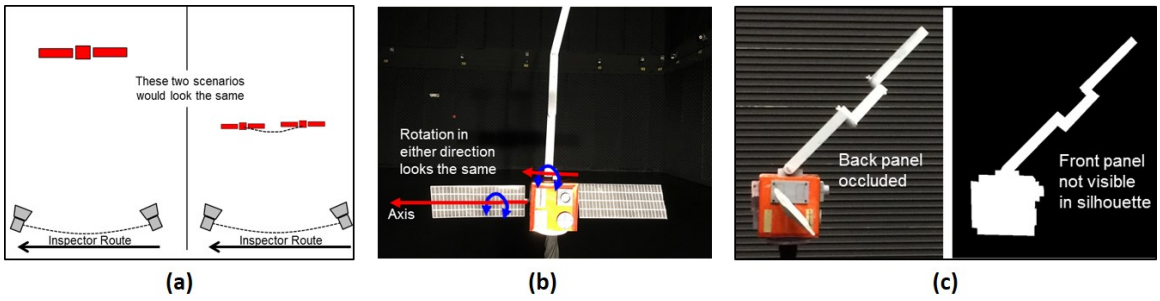

Figure 47. Examples of monocular imagery ambiguities: a) Depth ambiguity b) Articulation ambiguity c) Occlusion ambiguity

7.2.4 Articulation Rates.

While the effect of the articulation rate on the results from the methods was not assessed, there are a few considerations for the topic. Rather then considering the angular articulation rate (degrees/second) and the image capture rate (frames/second) separately, consider the articulation rate generally in terms of degrees/frame. This removes the temporal aspect and instead focuses on relative rate between angular articulation and image capture. For the methods of building the articulated model

(Chapter III) the articulation rate can be thought of in terms of the number of frames captured over a particular articulation period. For the original method (section 3.3) reducing the number of frames, which increases the articulation rate, may exacerbate the issues involved with combining segments since there will be less frame overlap between points. Alternatively, increasing the number of frames (decreasing the articulation rate) may improve results, however it would also increase computational time. For the modified method (section 3.5.1) reducing the number of frames is likely to have an effect only if it reduces the accuracy of the segmentation or kinematic chain.

For the tracking methods (Chapters IV and V), the articulation rate can be thought of as degrees/update. For the feature point method (Chapter IV) articulation rate must be low enough that the linearized measurement model remains accurate. For the silhouette method the expected articulation rate should be considered in setting the initial covariance values and weights; if the rate is too high as compared to the initial covariance, the sigma points may not be spaced far enough from the mean to enable the filter to converge to the correct rates. In both cases a lower articulation rate (faster updates) would allow the filter to better accommodate changes to the articulation rates. For the APF method outlined in section 6.6 the effect of the articulation rate could be minimized by increasing the number of particles (increasing the computational requirement) however applying the method with a single camera may still yield poor results due to ambiguities outlined in section 7.2.3.

7.3 Summary of Methods Presented

An overview of each of the methods developed is included in Table 8. Figure 1 gives a graphical overview of how the different methods fit into the overall problem and how they relate to the stated research objectives of sections 1.2 and 7.1.

Table 8. Overview of methods. TM=Trajectory Matrix, IR=Inspection Route, FP=Feature Point

	Build Articulated Model		Articulation Tracking	
	Original Method (sec. 3.3)	**Modified Method (sec. 3.5.1)**	**Feature points (Chapter IV)**	**Silhouette (Chapter V)**
Prior Knowledge and Assumptions	-Number of components unknown. -Main body not maneuvering -No IR or TM noise (if main body can be segmented, can accommodate IR and TM noise) -Accommodates significant missing data; better results with less missing data	-Number of components known -Main body can be maneuvering -TM and IR noise acceptable -Viewing angle such that kinematic chain can be created directly from TM. -Only solves for FPs seen in all frames	-FPs used for measurement -Stationary articulation parameters included in state -Initiated with model and first image frame -Demonstrated with IR and FP noise	-Silhouettes used for measurements -Stationary articulation parameters excluded from state -Initiated with true angles/pose and zero rates -Demonstrated with IR noise -Components modeled with primitive shapes
	-Depth to satellite $\gg$ depth within satellite (scaled orthographic camera model used for initializations)		-Articulated model available	
	-FPs can be reliably tracked with correct correspondence -FPs adequately cover components			-Silhouettes can be reliably created
	-IR known			
Type of Articulation	-Constant rate articulation assumption used for initialization only. -Non-constant rate articulation not tested.		-Constant rate motion model	
			-Bilinear demonstrated	-Sinusoidal and bilinear demonstrated
	-All joints are revolute (hinge)			
Articulation Rate	-Tested with rates of $3.3e-4$ to $4.4e-4$ degrees per frame		-Tested with rates of $1.1e-5$ to $7.6e-5$ degrees per update	-Tested with rates of $6.1e-5$ to $3.8e-4$ degrees per update
Parameter to be Tuned	-See Table 3	-Parameters γ_{sc+}, γ_{sc-}, γ_{BB}, and η from Table 3 -Range of expected values for initializations	-Process noise -Measurement noise -Criteria for adding points	-Process noise -Measurement noise -UKF weights -Value for σ from equation (133)
Approximate Computational Time	-5.5 hrs. (Table 2)	-7 hrs. (100 frames; 276 points) -This could be minimized by reducing the number of seeds evaluated in line 9 of Algorithm 2	-0.3 sec. per update	-Time per update is image size dependent -1.7 sec. for 800x800 image -70 sec. for 3024x4032 image

7.4 Contributions

The main contribution of this work is the investigation into methods of characterizing and tracking satellite articulation in space using monocular computer vision. While previous work exists on characterizing articulation within the computer vision community, no previous work was found in applying computer vision to characterize articulation in space. Aspects of existing computer vision algorithms have been used to develop methods of building an articulated model and tracking satellite articulation in real-time. Specifically, the contributions from this research are:

1. Introduction of computer vision for the purpose of building an articulated model and tracking satellite articulation in space using monocular imagery.

2. Development and demonstration of a method for building an articulated model of a satellite from monocular imagery taken from a known inspection route.

3. Development and demonstration of methods for sequentially estimating the pose and articulation angles of a satellite using both feature points and silhouette images.

4. Demonstration of tracking articulation angles using real images of a satellite model taken in a simulated space lighting environment.

5. Demonstration that algorithms developed for human articulation tracking can be modified to track the articulated motion of a satellite.

Additionally, four conference papers have been published in conjunction with this work [19, 18, 21, 17].

7.5 Recommendations for Future Work

There are numerous areas for continuation of the current work:

- Human tracking method applications: The use of human tracking algorithms for tracking satellite articulation could be investigated in more depth. The potential of using these type of algorithms has been demonstrated, but a thorough investigation of the existing work and its applicability had not been investigated.

- Optimize inspection route: The best inspection route depends on the satellite being viewed and the purpose of the inspection, but with some assumptions a cost function could be developed based on the information available from images taken at different relative poses. Images of some nominal satellites could be rendered and measures of the information availability could be developed that compare detected feature points, silhouettes, or edge maps to a baseline images. Completeness of the inspection could be measured by evaluating the amount of the satellite that was viewed by the camera over a given route. These measures could be used to build a simpler surrogate metric that could be used in the cost function of an optimal control problem.

- Multiple cameras: There are added benefits for inspection in having multiple cameras collecting imagery at the same time from different angles. Many of the human articulation tracking algorithms employ multiple cameras. A cost/benefit analysis could be performed to identify if and when it would be worth the added cost to use multiple cameras.

- Building an articulated model from silhouettes: Methods exist for building shape from silhouettes [12, 13]. These algorithms could be investigated for use in the space environment.

- Machine learning for pose estimation: Training imagery could be rendered of a known satellite in various articulation configurations from various angles. This could be used in machine learning applications to 'learn' how to predict artic-

ulation from imagery.

- Alternative sensors: The pros/cons of using alternative sensors such as LIDAR or stereo vision for inspection could be investigated.

The work presented represents significant progress in the use of computer vision techniques for articulation sensing in space. It outlines methods for building an articulated model and tracking articulation to answer the objectives outlined in Chapter I. Furthermore, it demonstrates the potential of applying human articulation tracking methods to the problem of spacecraft articulation tracking.

Appendices

Appendix A. Gradients for Chapter III

A.1 Rigid Body Optimization Gradients

Derivation for the derivatives used in rigid body optimization (section 3.3.4) are as follows where $x = [\mathbf{a}, \phi, \mathbf{T}, \mathbf{\Omega}]^T$.

$$f(x) = \sum_{i=1}^{F} \|W_i - P_i(R_i\mathbf{\Omega} + \tilde{T}_i)\|^2 + \lambda_{rb} \sum_{i=2}^{F} \left[\arccos(.5(trace(R_i R_{i-1}^T) - 1)) + \|T_i - T_{i-1}\|^2\right] \quad \text{(A.1)}$$

The optimization variables are contained in the vector x which consist of a $3F \times 1$ vector of Euler axes (not normalized) $\mathbf{a} = [\vec{a}_1^T, \vec{a}_2^T, \cdots, \vec{a}_F^T]$, a $F \times 1$ vector of Euler angles $\phi = [\phi_1, \phi_2, \cdots, \phi_F]$, a $3F \times 1$ vector of translations $\mathbf{T} = [\vec{T}_1^T, \vec{T}_2^T, \cdots, \vec{T}_F^T]$, and a $3P \times 1$ vector of shape coordinates $\mathbf{\Omega} = [\vec{s}_1^T, \vec{s}_2^T, \cdots, \vec{s}_P^T]$. These optimization variables translate to the variables in the cost function as follows.

$$\begin{aligned} R_i &= \cos(\phi_i)\mathbf{I} + (1 - \cos(\phi_i))\hat{a}_i\hat{a}_i^T - \sin(\phi_i)\hat{a}_i^x & \text{(A.2)} \\ \hat{a}_i &= \frac{\vec{a}_i}{\|\vec{a}_i\|} & \text{(A.3)} \\ \tilde{T}_i &= T_i - R_i\bar{\Omega} & \text{(A.4)} \\ \bar{\Omega} &= \frac{1}{P}\sum_{j=1}^{P} \vec{s}_j & \text{(A.5)} \end{aligned}$$

The gradient of the function consists of the partial derivatives of each of the optimization variables $\frac{\partial f}{\partial x} = \left[\frac{\partial f}{\partial \mathbf{a}}, \frac{\partial f}{\partial \phi}, \frac{\partial f}{\partial \mathbf{T}}, \frac{\partial f}{\partial \mathbf{\Omega}}\right]^T$. The chain rule is used to relate each of these to the components of the cost function. Start with the partial between the rotation matrix R_i and the rotation parameters ($\vec{a}_i$ and ϕ_i).

$$\frac{\partial f}{\partial \vec{a}_i} = \frac{\partial f}{\partial R_i}\frac{\partial R_i}{\partial \hat{a}_i}\frac{\partial \hat{a}_i}{\partial \vec{a}_i} \tag{A.6}$$

$$\hat{a}_i = [\hat{a}_i^{(1)}, \hat{a}_i^{(2)}, \hat{a}_i^{(3)}]^T \tag{A.7}$$

$$R_i = \begin{bmatrix} R_i^{(1)} & R_i^{(2)} & R_i^{(3)} \\ R_i^{(4)} & R_i^{(5)} & R_i^{(6)} \\ R_i^{(7)} & R_i^{(8)} & R_i^{(9)} \end{bmatrix} \tag{A.8}$$

$$\frac{\partial R_i}{\partial \hat{a}_i} = \begin{bmatrix} \frac{\partial R_i^{(1)}}{\partial \hat{a}_i^{(1)}} & \frac{\partial R_i^{(1)}}{\partial \hat{a}_i^{(2)}} & \frac{\partial R_i^{(1)}}{\partial \hat{a}_i^{(3)}} \\ \frac{\partial R_i^{(2)}}{\partial \hat{a}_i^{(1)}} & \frac{\partial R_i^{(2)}}{\partial \hat{a}_i^{(2)}} & \frac{\partial R_i^{(2)}}{\partial \hat{a}_i^{(3)}} \\ \vdots & \vdots & \vdots \\ \frac{\partial R_i^{(9)}}{\partial \hat{a}_i^{(1)}} & \frac{\partial R_i^{(9)}}{\partial \hat{a}_i^{(2)}} & \frac{\partial R_i^{(9)}}{\partial \hat{a}_i^{(3)}} \end{bmatrix} \tag{A.9}$$

The derivatives of R_i with respect to $\hat{a}_i$ are as follows. They are written as 3×3 matrices, however they should be reshaped to 9×3 to be used in the chain rule as written.

$$\frac{\partial R_i}{\partial \hat{a}_i^{(1)}} = \begin{bmatrix} 2(1-\cos(\phi_i))\hat{a}_i^{(1)} & (1-\cos(\phi_i))\hat{a}_i^{(2)} & (1-\cos(\phi_i))\hat{a}_i^{(3)} \\ (1-\cos(\phi_i))\hat{a}_i^{(2)} & 0 & \sin(\phi_i) \\ (1-\cos(\phi_i))\hat{a}_i^{(3)} & -\sin(\phi_i) & 0 \end{bmatrix} \tag{A.10}$$

$$\frac{\partial R_i}{\partial \hat{a}_i^{(2)}} = \begin{bmatrix} 0 & (1-\cos(\phi_i))\hat{a}_i^{(1)} & -\sin(\phi_i) \\ (1-\cos(\phi_i))\hat{a}_i^{(2)} & 2(1-\cos(\phi_i))\hat{a}_i^{(2)} & (1-\cos(\phi_i))\hat{a}_i^{(2)} \\ \sin(\phi_i) & (1-\cos(\phi_i))\hat{a}_i^{(3)} & 0 \end{bmatrix} \tag{A.11}$$

$$\frac{\partial R_i}{\partial \hat{a}_i^{(3)}} = \begin{bmatrix} 0 & \sin(\phi_i) & (1-\cos(\phi_i))\hat{a}_i^{(1)} \\ -\sin(\phi_i) & 0 & (1-\cos(\phi_i))\hat{a}_i^{(2)} \\ (1-\cos(\phi_i))\hat{a}_i^{(1)} & (1-\cos(\phi_i))\hat{a}_i^{(2)} & 2(1-\cos(\phi_i))\hat{a}_i^{(3)} \end{bmatrix} \tag{A.12}$$

The Euler axis should be constrained to be unit length. Instead of enforcing unit

length as a hard constraint in the optimization, the vector is normalized before use in the cost function. The following derivatives relate the Euler axis $\hat{a}_i$ to the optimization variable $\vec{a}_i$. I represents the 3×3 identity matrix.

$$\frac{\partial \hat{a}_i}{\partial \vec{a}_i} = \begin{bmatrix} \frac{\partial \hat{a}_i^{(1)}}{\partial a_i^{(1)}} & \frac{\partial \hat{a}_i^{(1)}}{\partial a_i^{(2)}} & \frac{\partial \hat{a}_i^{(1)}}{\partial a_i^{(3)}} \\ \frac{\partial \hat{a}_i^{(2)}}{\partial a_i^{(1)}} & \frac{\partial \hat{a}_i^{(2)}}{\partial a_i^{2}} & \frac{\partial \hat{a}_i^{(2)}}{\partial a_i^{(3)}} \\ \frac{\partial \hat{a}_i^{(3)}}{\partial a_i^{(3)}} & \frac{\partial \hat{a}_i^{(3)}}{\partial a_i^{(2)}} & \frac{\partial \hat{a}_i^{(3)}}{\partial a_i^{(3)}} \end{bmatrix} \tag{A.13}$$

$$\frac{\partial \hat{a}_i}{\partial \vec{a}_i} = \frac{1}{\|\hat{a}_i\|} I - \frac{1}{\|\hat{a}_i\|^3} (\vec{a}_i \vec{a}_i^T) \tag{A.14}$$

The derivative of R_i with respect to the Euler angle ϕ_i is as follows.

$$\frac{\partial R_i}{\partial \phi_i} = \begin{bmatrix} -\sin(\phi_i) & \cos(\phi_i)\hat{a}_i^{(3)} & -\cos(\phi_i)\hat{a}_i^{(2)} \\ -\cos(\phi_i)\hat{a}_i^{(3)} & -\sin(\phi_i) & -\cos(\phi_i)\hat{a}_i^{(1)} \\ \cos(\phi_i)\hat{a}_i^{(2)} & -\cos(\phi_i)\hat{a}_i^{(1)} & -\sin(\phi_i) \end{bmatrix} + \sin(\phi_i)(\hat{a}_i \hat{a}_i^T) \tag{A.15}$$

The partial derivative of the centered translation $\tilde{T}_i$ with respect to the rotation matrix R_i, the actual translation T_i, and the shape $\mathbf{\Omega}$ must also be calculated. The symbol $\otimes$ denotes the Kronecker product.

$$\frac{\partial \tilde{T}_i}{\partial T_i} = I \tag{A.16}$$

$$\frac{\partial \tilde{T}_i}{\partial R_i} = -I \otimes (\bar{\Omega})^T \tag{A.17}$$

$$\frac{\partial \tilde{T}_i}{\partial \Omega_j} = -\frac{1}{P} R_i \tag{A.18}$$

Now that the rotation matrix has been related to the optimization variables, it must now be related to the cost function. The cost function can be broken into two parts: D is the reprojection error, and L is the smoothness constraint. Note that $\|\cdot\|$

is the Frobenius norm, or the summation of each element squared.

$$D = \sum_{i=1}^{F} \|W_i - P_i(R_i\mathbf{\Omega} + \tilde{T}_i)\|^2 \tag{A.19}$$

$$L = \lambda_{rb} \sum_{i=2}^{F} \arccos(.5(trace(R_i R_{i-1}^T) - 1)) + \|T_i - T_{i-1}\|^2 \tag{A.20}$$

$$\frac{\partial f}{\partial R} = \frac{\partial D}{\partial R} + \frac{\partial L}{\partial R} \tag{A.21}$$

$$\frac{\partial f}{\partial T} = \frac{\partial D}{\partial T} + \frac{\partial L}{\partial T} \tag{A.22}$$

$$\frac{\partial f}{\partial \Omega} = \frac{\partial D}{\partial \Omega} \tag{A.23}$$

The reprojection error for frame i is D_i which is the summation of the reprojection error for the P points available in frame i. The terms u and v are the image coordinates in the horizontal and vertical dimensions respectively.

$$D_i = \sum_{u,v} \sum_{j=1}^{P} G_{i,j}^2 \tag{A.24}$$

$$G_{i,j} = \begin{bmatrix} u_j \\ v_j \end{bmatrix} - P_i(R_i\Omega_j + \tilde{T}_i) \tag{A.25}$$

$$\frac{\partial D_i}{\partial R_i} = \sum_{u,v} \sum_{j=1}^{P} 2G_{i,j} \left[\frac{\partial G_{i,j}}{\partial R_i} + \frac{\partial G_{i,j}}{\partial \tilde{T}_i} \frac{\partial \tilde{T}_i}{\partial R_i} \right] \tag{A.26}$$

Define $\tilde{G}$ as the second term of G written in homogeneous coordinates. f is the focal

length of the camera using a pinhole camera model.

$$\tilde{G}_{i,j} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[R_{CW} | \mathbf{t} \right] \begin{bmatrix} R_i & T_i \\ 0,\ 0,\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_j \\ 1 \end{bmatrix} \tag{A.27}$$

$$\tilde{G}_{i,j} = \begin{bmatrix} \tilde{G1}_{i,j} \\ \tilde{G2}_{i,j} \\ \omega \end{bmatrix} \tag{A.28}$$

$$G_{i,j} = \begin{bmatrix} u_j \\ v_j \end{bmatrix} - \begin{bmatrix} \frac{\tilde{G1}_{i,j}}{\omega} \\ \frac{\tilde{G2}_{i,j}}{\omega} \end{bmatrix} \tag{A.29}$$

The rotation, translation and shape derivatives are then calculated as follows using Matlab matrix notation where appropriate. The symbol $\odot$ denotes element-wise multiplication.

Rotation derivatives:

$$\frac{\partial G_{i,j}}{\partial R_i} = - \begin{bmatrix} \frac{\omega \frac{\partial \tilde{G1}_{i,j}}{\partial R_i} - \tilde{G1}_{i,j} \frac{\partial \omega}{\partial R_i}}{\omega^2} \\ \frac{\omega \frac{\partial \tilde{G2}_{i,j}}{\partial R_i} - \tilde{G2}_{i,j} \frac{\partial \omega}{\partial R_i}}{\omega^2} \end{bmatrix} \tag{A.30}$$

$$\frac{\partial \tilde{G1}_{i,j}}{\partial R_i} = - \begin{bmatrix} \frac{\partial \tilde{G1}_{i,j}}{\partial R_i^1}, & \frac{\partial \tilde{G1}_{i,j}}{\partial R_i^2}, & \dots & \frac{\partial \tilde{G1}_{i,j}}{\partial R_i^9} \end{bmatrix} \tag{A.31}$$

$$\frac{\partial \tilde{G1}_{i,j}}{\partial R_i} = f \left[(R_{CW}^{(1,:)} \otimes [1,1,1]) \odot ([1,1,1] \otimes s_j^T) \right] \tag{A.32}$$

$$\frac{\partial \tilde{G2}_{i,j}}{\partial R_i} = f \left[(R_{CW}^{(2,:)} \otimes [1,1,1]) \odot ([1,1,1] \otimes s_j^T) \right] \tag{A.33}$$

$$\frac{\partial \tilde{\omega}_{i,j}}{\partial R_i} = \left[(R_{CW}^{(3,:)} \otimes [1,1,1]) \odot ([1,1,1] \otimes s_j^T) \right] \tag{A.34}$$

Translation derivatives:

$$\frac{\partial D_i}{\partial T_i} = \sum_{u,v}\sum_{j=1}^{P} 2G_{i,j}\frac{\partial G_{i,j}}{\partial \tilde{T}_i}\frac{\partial \tilde{T}_i}{\partial T_i} \tag{A.35}$$

$$\frac{\partial G_{i,j}}{\partial \tilde{T}_i} = -\begin{bmatrix} \frac{\omega\frac{\partial \tilde{G1}_{i,j}}{\partial \tilde{T}_i} - \tilde{G1}_{i,j}\frac{\partial \omega}{\partial \tilde{T}_i}}{\omega^2} \\ \frac{\omega\frac{\partial \tilde{G2}_{i,j}}{\partial \tilde{T}_i} - \tilde{G2}_{i,j}\frac{\partial \omega}{\partial \tilde{T}_i}}{\omega^2} \end{bmatrix} \tag{A.36}$$

$$\frac{\partial \tilde{G1}_{i,j}}{\partial \tilde{T}_i} = fR_{CW}^{(1,:)} \tag{A.37}$$

$$\frac{\partial \tilde{G2}_{i,j}}{\partial \tilde{T}_i} = fR_{CW}^{(2,:)} \tag{A.38}$$

$$\frac{\partial \omega}{\partial \tilde{T}_i} = R_{CW}^{(3,:)} \tag{A.39}$$

Shape derivatives:

$$\frac{\partial D}{\partial \Omega_j} = \sum_{u,v}\sum_{i=1}^{F} 2G_{i,j}\left[\frac{\partial G_{i,j}}{\partial \Omega_j} + \frac{\partial G_{i,j}}{\partial \tilde{T}_i}\frac{\partial \tilde{T}_i}{\partial \Omega_j}\right] \tag{A.40}$$

$$\frac{\partial G_{i,j}}{\partial \Omega_j} = -\begin{bmatrix} \frac{\omega\frac{\partial \tilde{G1}_{i,j}}{\partial \Omega_j} - \tilde{G1}_{i,j}\frac{\partial \omega}{\partial \Omega_j}}{\omega^2} \\ \frac{\omega\frac{\partial \tilde{G2}_{i,j}}{\partial \Omega_j} - \tilde{G2}_{i,j}\frac{\partial \omega}{\partial \Omega_j}}{\omega^2} \end{bmatrix} \tag{A.41}$$

$$\frac{\partial \tilde{G1}_{i,j}}{\partial \Omega_j} = fR_{CW}^{(1,:)}R_i \tag{A.42}$$

$$\frac{\partial \tilde{G2}_{i,j}}{\partial \Omega_j} = fR_{CW}^{(2,:)}R_i \tag{A.43}$$

$$\frac{\partial \omega}{\partial \Omega_j} = R_{CW}^{(3,:)}R_i \tag{A.44}$$

The smoothness constraint is only applied to adjacent frames, so the derivatives are different if frame i is the first frame in a sequence, the middle frame in a sequence

or the last frame in a sequence. For the first frame in a sequence,

$$
\begin{aligned}
L_i &= \lambda_{rb}\left[\arccos(.5(trace(R_{i+1}R_i^T)-1)) + \|T_{i+1}-T_i\|\right] && \text{(A.45)}\\
U_i &= 0.5(trace(R_iR_{i-1}^T)-1) && \text{(A.46)}\\
\frac{\partial L_i}{\partial R_i} &= -\frac{\lambda_{rb}}{\sqrt{1-U_{i+1}^2}}(.5R_{i+1}) && \text{(A.47)}\\
V_i &= \|T_i - T_{i-1}\| && \text{(A.48)}\\
&= \sqrt{(T_i^1-T_{i-1}^1)^2+(T_i^2-T_{i-1}^2)^2+(T_i^3-T_{i-1}^3)^2} && \text{(A.49)}\\
\frac{\partial L_i}{\partial T_i} &= -\frac{\lambda_{rb}}{2\sqrt{V_{i+1}^2}}\left[-2(T_{i+1})^T+2(T_i)^T\right] && \text{(A.50)}
\end{aligned}
$$

For middle frames in a sequence,

$$
\begin{aligned}
L_i = \lambda_{rb}[&\arccos(.5(trc(R_{i+1}R_i^T)-1)) + \|T_{i+1}-T_i\| + \\
&\arccos(.5(trc(R_iR_{i-1}^T)-1)) + \|T_i - T_{i-1}\|] \quad \text{(A.51)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_i}{\partial R_i} &= -\frac{\lambda_{rb}}{\sqrt{1-U_i^2}}(.5R_{i-1}) - \frac{\lambda_{rb}}{\sqrt{1-U_{i+1}^2}}(.5R_{i+1}) && \text{(A.52)}\\
\frac{\partial L_i}{\partial T_i} &= -\frac{\lambda_{rb}}{2\sqrt{V_i^2}}\left[-2(T_i)^T+2(T_{i-1})^T\right] - \frac{\lambda_{rb}}{2\sqrt{V_{i+1}^2}}\left[-2(T_{i+1})^T+2(T_i)^T\right] && \text{(A.53)}
\end{aligned}
$$

For last frames in a sequence,

$$
\begin{aligned}
L_i &= \lambda_{rb}\left[\arccos(.5(trc(R_iR_{i-1}^T)-1)) + \|T_i - T_{i-1}\|\right] && \text{(A.54)}\\
\frac{\partial L_i}{\partial R_i} &= -\frac{\lambda}{\sqrt{1-U_i^2}}(.5R_{i-1}) && \text{(A.55)}\\
\frac{\partial L_i}{\partial T_i} &= -\frac{\lambda}{2\sqrt{V_i^2}}\left[-2(T_i)^T+2(T_{i-1})^T\right] && \text{(A.56)}
\end{aligned}
$$

A.2 Articulation Parameter Gradients

Derivation for the derivatives used in articulation parameter optimization (section 42) are as follows:

$$f(x) = \sum_{j=1}^{N-1} \Bigg[\underbrace{\sum_{i=1}^{CF} \left[\| W_i - P_i(R_i^{w,c_j} \Omega^{c_j} + T_i^{w,c_j}) \|^2 \right]}_{\text{Reprojection Error, A}} + \underbrace{\lambda_{jp} \sum_{k=1}^{12} ln(1 + e^{d_k \eta})}_{\text{Joint Penalty, B}} + \underbrace{\lambda_{sc} \sum_{i=2}^{CF} (1 - cos(\phi_i - \phi_{i-1}))^2}_{\text{Smoothness Constraint, SC}} \Bigg] \quad \text{(A.57)}$$

$$x = [\Omega^{c_1}, \vec{a}^{p_1}, \vec{a}^{c_1}, \phi^1, J^{p_1}, J^{c_1}, \cdots, \Omega^{c_{N-1}}, \vec{a}^{p_{N-1}}, \vec{a}^{c_{N-1}}, \phi^{N-1}, J^{p_{N-1}}, J^{c_{N-1}}]^T \quad \text{(A.58)}$$

Many of the derivatives for the articulation parameter cost function are the same as for rigid body optimization. The primary difference is finding the partial derivatives of the rotation matrix used for each component with respect to the optimization variables lower in the kinematic chain. For each parameter that defines joint j between components x_k and x_{k-1}, the derivatives must be calculated for each component that contains x_{k-1} in its hierarchy. Define X^k as the set of components influenced by component x_k, or in other words X^k is the set of components below component x_k in the kinematic chain.

$$\frac{\partial f}{\partial R_i^{w,x_k}} = \sum_{q=1}^{X^k} \frac{\partial f}{\partial R_i^{w,x_q}} \tag{A.59}$$

$$\frac{\partial f}{\partial T_i^{w,x_k}} = \sum_{q=1}^{X^k} \frac{\partial f}{\partial T_i^{w,x_q}} \tag{A.60}$$

To calculate the derivatives of the rotation matrices R_i^{w,x_k} and the translation vectors T_i^{w,x_k} a hierarchical approach is required. Define $X = [x_1, x_2, \cdots, x_l]$ as a single chain starting at the root and ending with a component that has no children where each element of X identifies a component, x_k is the parent and x_{k-1} is the child for any particular joint.

$$\frac{\partial R_i^{w,x_k}}{\partial R_i^{x_{l-1},x_l}} = \begin{cases} \frac{\partial R_i^{w,x_k}}{\partial R_i^{x_{l-1},x_l}} & k = l \\ \frac{\partial R_i^{w,x_k}}{\partial R_i^{w,x_{k-1}}} \frac{\partial R_i^{w,x_{k-1}}}{\partial R_i^{x_{l-1},x_l}} & k > l \end{cases} \tag{A.61}$$

$$\frac{\partial T_i^{w,x_k}}{\partial R_i^{x_{l-1},x_l}} = \begin{cases} \frac{\partial T_i^{w,x_k}}{\partial R_i^{w,x_k}} & k = l \\ \frac{\partial T_i^{w,x_k}}{\partial R_i^{w,x_{k-1}}} \frac{\partial R_i^{w,x_{k-1}}}{\partial R_i^{x_{l-1},x_l}} + \frac{\partial T_i^{w,x_k}}{\partial T_i^{w,x_{k-1}}} \frac{\partial T_i^{w,x_{k-1}}}{\partial R_i^{x_{l-1},x_l}} + \frac{\partial T_i^{w,x_k}}{\partial R_i^{w,x_k}} & k > l \end{cases} \tag{A.62}$$

To calculate these partial derivatives, the derivative relating the actual optimization parameters to the rotation matrices must be calculated. The symbol $\otimes$ denotes the Kronecker product.

$$\frac{\partial R_i^{w,x_k}}{\partial R_i^{x_{k-1},x_k}} = R_i^{w,x_{k-1}} \otimes I \tag{A.63}$$

$$\frac{\partial R_i^{w,x_k}}{\partial R_i^{w,x_{k-1}}} = I \otimes (R_i^{x_{k-1},x_k})^T \tag{A.64}$$

From equation set (74) the derivatives of $R_i^{x_{k-1},x_k}$ with respect to rotational optimization variables for the joint between x_{k-1} and x_k ($\hat{a}_p^{x_k,x_{k-1}}$, $\hat{a}_c^{x_k,x_{k-1}}$, $\phi_i^{x_k,x_{k-1}}$) can be found.

$$\frac{\partial R_i^{x_{k-1},x_k}}{\partial \hat{a}_p^{x_k,x_{k-1}}} = (R_i^{x_{k-1},aa} \otimes I)\frac{\partial R_i^{aa,x_k}}{\partial \hat{a}_p^{x_k,x_{k-1}}} + (I \otimes (R^{aa,x_k})^T)\frac{\partial R_i^{x_{k-1},aa}}{\partial \hat{a}_p^{x_k,x_{k-1}}} \tag{A.65}$$

$$\frac{\partial R_i^{aa,x_k}}{\partial \hat{a}_p^{x_k,x_{k-1}}} = \frac{\partial R_i^{aa,x_k}}{\partial \hat{a}_a^{x_k,x_{k-1}}}\frac{\partial \hat{a}_a^{x_k,x_{k-1}}}{\partial \vec{a}_a^{x_k,x_{k-1}}}\frac{\partial \vec{a}_a^{x_k,x_{k-1}}}{\partial \hat{a}_p^{x_k,x_{k-1}}} + \frac{\partial R_i^{aa,x_{k-1}}}{\partial \psi_a^{x_k,x_{k-1}}}\frac{\partial \psi_a^{x_k,x_{k-1}}}{\partial \hat{a}_p^{x_k,x_{k-1}}} \tag{A.66}$$

The derivatives $\frac{\partial R_i^{x_{k-1},aa}}{\partial \hat{a}_p^{x_k,x_{k-1}}}$, $\frac{\partial R_i^{aa,x_k}}{\partial \hat{a}_a^{x_k,x_{k-1}}}$, $\frac{\partial \hat{a}_a^{x_k,x_{k-1}}}{\partial \vec{a}_a^{x_k,x_{k-1}}}$, and $\frac{\partial R_i^{aa,x_{k-1}}}{\partial \psi_a^{x_k,x_{k-1}}}$ are identical in structure to the derivatives of a rotation matrix with respect to an Euler axis and angle shown in section A.1.

$$\vec{a}_a = \hat{a}_p \times \hat{a}_c = \begin{bmatrix} \hat{a}_p^{(2)}\hat{a}_c^{(3)} - \hat{a}_p^{(3)}\hat{a}_c^{(2)} \\ \hat{a}_p^{(3)}\hat{a}_c^{(1)} - \hat{a}_p^{(1)}\hat{a}_c^{(3)} \\ \hat{a}_p^{(1)}\hat{a}_c^{(2)} - \hat{a}_p^{(2)}\hat{a}_c^{(1)} \end{bmatrix} \tag{A.67}$$

$$\frac{\partial \vec{a}_a}{\hat{a}_p} = \begin{bmatrix} 0 & \hat{a}_c^{(3)} & -\hat{a}_c^{(2)} \\ -\hat{a}_c^{(3)} & 0 & \hat{a}_c^{(1)} \\ \hat{a}_c^{(2)} & -\hat{a}_c^{(1)} & 0 \end{bmatrix} \tag{A.68}$$

$$\psi_a = \arccos(\hat{a}_p \cdot \hat{a}_c) \tag{A.69}$$

$$\frac{\partial \psi_a}{\hat{a}_p} = -\frac{1}{\sqrt{1-(\hat{a}_p \cdot \hat{a}_c)^2}}(\hat{a}_c)^T \tag{A.70}$$

Derivatives with respect to $\hat{a}_c^{x_k,x_{k-1}}$ are similar except that $R_i^{x_{k-1},aa}$ is not depen-

dent on $\hat{a}_c^{x_k,x_{k-1}}$ so the product rule is not required.

$$\frac{\partial R_i^{x_{k-1},x_k}}{\partial \hat{a}_c^{x_k,x_{k-1}}} = (R_i^{x_{k-1},aa} \otimes I) \frac{\partial R_i^{aa,x_k}}{\partial \hat{a}_c^{x_k,x_{k-1}}} \tag{A.71}$$

$$\frac{\partial R_i^{aa,x_k}}{\partial \hat{a}_c^{x_k,x_{k-1}}} = \frac{\partial R_i^{aa,x_k}}{\partial \hat{a}_a^{x_k,x_{k-1}}} \frac{\partial \hat{a}_a^{x_k,x_{k-1}}}{\partial \vec{a}_a^{x_k,x_{k-1}}} \frac{\partial \vec{a}_a^{x_k,x_{k-1}}}{\partial \hat{a}_c^{x_k,x_{k-1}}} + \frac{\partial R_i^{aa,x_{k-1}}}{\partial \psi_a^{x_k,x_{k-1}}} \frac{\partial \psi_a^{x_k,x_{k-1}}}{\partial \hat{a}_c^{x_k,x_{k-1}}} \tag{A.72}$$

The derivative $\frac{\partial R_i^{x_{k-1},aa}}{\partial \hat{a}_c^{x_k,x_{k-1}}}$ is identical in structure to the derivatives of a rotation matrix with respect to an Euler axis and angle shown in section A.1.

$$\frac{\partial \vec{a}_a}{\hat{a}_c} = \begin{bmatrix} 0 & -\hat{a}_p^{(3)} & \hat{a}_p^{(2)} \\ \hat{a}_p^{(3)} & 0 & -\hat{a}_p^{(1)} \\ -\hat{a}_p^{(2)} & \hat{a}_p^{(1)} & 0 \end{bmatrix} \tag{A.73}$$

$$\frac{\partial \psi_a}{\hat{a}_c} = -\frac{1}{\sqrt{1-(\hat{a}_p \cdot \hat{a}_c)^2}} (\hat{a}_p)^T \tag{A.74}$$

The derivatives with respect to the articulation angle (ϕ_i) are not dependent on $R_i^{x_{k-1},aa}$ and can be calculated using the 3×3 representation of $\frac{\partial R}{\partial \phi}$ from section A.1. After multiplication, $\frac{\partial R_i^{x_{k-1},x_k}}{\partial \phi_i}$ must be reshaped to a 9×1 matrix.

$$\frac{\partial R_i^{x_{k-1},x_k}}{\partial \phi_i} = \frac{\partial R_i^{x_{k-1},aa}}{\partial \phi_i} R^{aa,x_k} \tag{A.75}$$

The rotation also effects the translation. The remaining derivatives required for $\frac{\partial T_i^{w,x_k}}{\partial R_i^{x_{l-1},x_l}}$ are as follows.

$$\frac{\partial T_i^{w,x_k}}{\partial R_i^{w,x_k}} = I \otimes (-J_c^{x_k,x_{k-1}})^T \tag{A.76}$$

$$\frac{\partial T_i^{w,x_k}}{\partial R_i^{w,x_{k-1}}} = I \otimes (J_p^{x_k,x_{k-1}})^T \tag{A.77}$$

$$\frac{\partial T_i^{w,x_k}}{\partial T_i^{w,x_{k-1}}} = I \tag{A.78}$$

Next, the derivatives with respect to the joint locations (J_p and J_c) must be calculated. Since the joint locations do not have an effect on the rotation matrices, only

the derivatives with respect to the translations and the joint penalty (B) are required. BB_p and BB_c are the bounding boxes for the parent and the child respectively.

$$\frac{\partial T_i^{w,x_k}}{\partial J_p^{x_{l-1},x_l}} = \begin{cases} R_i^{w,x_{k-1}} & k = l \\ \frac{\partial T_i^{w,x_k}}{\partial T_i^{w,x_{k-1}}} \frac{\partial T_i^{w,x_{k-1}}}{\partial J_p^{x_{l-1},x_l}} & k > l \end{cases} \tag{A.79}$$

$$\frac{\partial T_i^{w,x_k}}{\partial J_c^{x_{l-1},x_l}} = \begin{cases} -R_i^{w,x_k} & k = l \\ \frac{\partial T_i^{w,x_k}}{\partial T_i^{w,x_{k-1}}} \frac{\partial T_i^{w,x_{k-1}}}{\partial J_c^{x_{l-1},x_l}} & k > l \end{cases} \tag{A.80}$$

$$d^j = [(J_p^j - BB_p^{max})^T, (BB_p^{min} - J_p^j)^T, (J_c^j - BB_c^{max})^T, (BB_c^{min} - J_c^j)^T] \tag{A.81}$$

$$\frac{\partial d_k^j}{\partial J_p^j} = [1, 1, 1, -1, -1, -1, 0, 0, 0, 0, 0, 0] \tag{A.82}$$

$$\frac{\partial d_k^j}{\partial J_c^j} = [0, 0, 0, 0, 0, 0, 1, 1, 1, -1, -1, -1] \tag{A.83}$$

$$\frac{\partial B_j}{\partial J_p^j} = \lambda_{jp} \sum_{k=1}^{12} \frac{1}{1+e^{d_k \eta}} (\eta e^{d_k \eta}) \frac{\partial d_k^j}{\partial J_p^j} \tag{A.84}$$

$$\frac{\partial B_j}{\partial J_c^j} = \lambda_{jp} \sum_{k=1}^{12} \frac{1}{1+e^{d_k \eta}} (\eta e^{d_k \eta}) \frac{\partial d_k^j}{\partial J_c^j} \tag{A.85}$$

Finally, the derivatives with respect to the smoothness constraint (SC) must be calculated.

$$\frac{\partial f}{\partial \phi_i} = \frac{\partial A}{\partial \phi_i} + \frac{\partial SC}{\partial \phi_i} \tag{A.86}$$

$$\Delta_i = \phi_i - \phi_{i-1} \tag{A.87}$$

$$\frac{\partial SC}{\partial \phi_i} = \begin{cases} -2\lambda sc(1 - \cos(\Delta_{i+1})) \sin(\Delta_{i+1}) & \text{first frame} \\ -2\lambda sc\left[(1 - \cos(\Delta_{i+1})) \sin(\Delta_{i+1}) - (1 - \cos(\Delta_i)) \sin(\Delta_i)\right] & \text{middle frames} \\ 2\lambda_s c(1 - \cos(\Delta_i)) \sin(\Delta_i) & \text{last frames} \end{cases} \tag{A.88}$$

The derivative of the reprojection error (A) with respect to Ω^{c_j}, R_i^{w,c_j}, and T_i^{w,c_j} are calculated as defined in section A.1.

Appendix B. Jacobians for Chapter IV

B.1 Articulation Parameter Propagation Jacobian

The propagation Jacobian is defines as $F = \frac{\partial f(x_{t-1})}{\partial x_{t-1}}$ where $x_t = f(x_{t-1})$ propagates the state from x_{t-1} to x_t. Since many of the states (θ^i, ψ^i, $\mathbf{J_p}^i$, $\mathbf{J_c}^i$) are constant with time, their propagation equations are $x_t = x_{t-1} + \eta\Delta t$ where η is the appropriate process noise and the corresponding portion of the Jacobian is the identity matrix. The only remain states are $\mathbf{q}^{wr}$, $\mathbf{T}^{wr}$, ω^{wr}, $\dot{\mathbf{T}}^{wr}$, ϕ^i, and $\dot{\phi}^i$. Those states associated with the root component are addressed first. These derivatives are similar to those worked out in Civera[14] for EKF SLAM.

$$x_t = f(x_{t-1}) = \begin{bmatrix} \mathbf{q}_t^{wr} \\ \mathbf{T}_t^{wr} \\ \omega_t^{wr} \\ \dot{\mathbf{T}}_t^{wr} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{t-1}^{wr} \otimes q((\omega_{t-1} + \eta_\omega \Delta t)\Delta t) \\ \mathbf{T}_{t-1}^{wr} + (\dot{\mathbf{T}}_{t-1}^{wr} + \eta_T \Delta t)\Delta t \\ \omega_{t-1}^{wr} + \eta_\omega \Delta t \\ \dot{\mathbf{T}}_{t-1}^{wr} + \eta_T \Delta t \end{bmatrix} \tag{B.1}$$

$$F = \frac{\partial f(x_{t-1})}{\partial x_{t-1}} = \begin{bmatrix} \frac{\partial \mathbf{q}_t^{wr}}{\partial \mathbf{q}_{t-1}^{wr}} & \mathbf{0} & \frac{\partial \mathbf{q}_t^{wr}}{\partial \omega_{t-1}^{wr}} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{T}_t^{wr}}{\partial \mathbf{T}_{t-1}^{wr}} & \mathbf{0} & \frac{\partial \mathbf{T}_t^{wr}}{\partial \dot{\mathbf{T}}_{t-1}^{wr}} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \omega_t^{wr}}{\partial \omega_{t-1}^{wr}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \dot{\mathbf{T}}_t^{wr}}{\partial \dot{\mathbf{T}}_{t-1}^{wr}} \end{bmatrix} \tag{B.2}$$

Define $\omega' = (\omega_{t-1} + \eta_\omega \Delta t)\Delta t$ and $\mathbf{q}' = q(\omega')$. Quaternion multiplication is defined as in Markley[56] with $\mathbf{q} = [(\mathbf{q}_v)^T, q_4]^T$.

$$\mathbf{q} \otimes \mathbf{q}' = \begin{bmatrix} q_4 \mathbf{q}'_v + {q'}_4 \mathbf{q}_v - \mathbf{q}_v \times \mathbf{q}'_v \\ q_4 {q'}_4 - \mathbf{q}_v \cdot \mathbf{q}'_v \end{bmatrix} \tag{B.3}$$

With these definitions, the derivatives for the quaternion are as follows. For convenience of notation, $\mathbf{q}_t^{wr}$ is represented by $\mathbf{q}$

$$\frac{\partial \mathbf{q}_t^{wr}}{\partial \mathbf{q}_{t-1}^{wr}} = \begin{bmatrix} q'_4 & -q'_3 & q'_2 & q'_1 \\ q'_3 & q'_4 & -q'_1 & q'_2 \\ -q'_2 & q'_1 & q'_4 & q'_3 \\ -q'_1 & -q'_2 & -q'_3 & q'_4 \end{bmatrix} \tag{B.4}$$

$$\frac{\partial \mathbf{q}_t^{wr}}{\partial \omega_{t-1}^{wr}} = \frac{\partial \mathbf{q}_t^{wr}}{\partial \mathbf{q}'} \frac{\partial \mathbf{q}'}{\partial \omega'} \frac{\partial \omega'}{\partial \omega_{t-1}^{wr}} \tag{B.5}$$

$$\frac{\partial \mathbf{q}_t^{wr}}{\partial \mathbf{q}'} = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \tag{B.6}$$

$$\frac{\partial \mathbf{q}'}{\partial \omega'} = \begin{bmatrix} (\frac{1}{\|\omega'\|} I - \frac{\omega'}{\|\omega'\|} \frac{\partial \|\omega'\|}{\partial \omega'}) \sin(\frac{\|\omega'\|}{2} + \frac{\omega'}{\|\omega'\|}(0.5 \cos(\frac{\|\omega'\|}{2})) \frac{\partial \|\omega'\|}{\partial \omega'} \\ -0.5 \sin(\frac{\|\omega'\|}{2}) \frac{\partial \|\omega'\|}{\partial \omega'} \end{bmatrix} \tag{B.7}$$

$$\frac{\partial \|\omega'\|}{\partial \omega'} = \frac{\omega'^T}{\|\omega'\|} \tag{B.8}$$

$$\frac{\partial \omega'}{\partial \omega_{t-1}^{wr}} = I \Delta t \tag{B.9}$$

$$\frac{\partial \mathbf{q}_t^{wr}}{\partial \omega_{t-1}^{wr}} = \frac{\partial \mathbf{q}_t^{wr}}{\partial \mathbf{q}'} \frac{\partial \mathbf{q}'}{\partial \omega'} \frac{\partial \omega'}{\partial \omega_{t-1}^{wr}} \tag{B.10}$$

The derivative for $\mathbf{T}_t^{wr}$, ω_t^{wr}, and $\dot{\mathbf{T}}_t^{wr}$ are straight forward: $\frac{\partial \mathbf{T}_t^{wr}}{\partial \mathbf{T}_{t-1}^{wr}} = I$, $\frac{\partial \mathbf{T}_t^{wr}}{\partial \dot{\mathbf{T}}_{t-1}^{wr}} = I\Delta t$, $\frac{\partial \omega_t^{wr}}{\partial \omega_{t-1}^{wr}} = I$, and $\frac{\partial \dot{\mathbf{T}}_t^{wr}}{\partial \dot{\mathbf{T}}_{t-1}^{wr}} = I$. The derivative for ϕ^i and $\dot{\phi}^i$ are also straight forward: $\frac{\partial \phi_t^i}{\partial \phi^i} = 1$, $\frac{\partial \phi_t^i}{\partial \dot{\phi}^i} = \Delta t$, and $\frac{\partial \dot{\phi}_t^i}{\partial \dot{\phi}^i} = 1$.

Similarly, the G matrix is found by taking the derivative of the propagation equations with respect to the process noise. A process noise channel η is assigned for each type of state as follows: $\eta = [\eta_\omega, \eta_{\dot{T}}, \eta_\Omega, \eta_{\theta,\psi}, \eta_J, \eta_{\dot{\phi}}]$. The portion of the G matrix

corresponding the the root component motion is shown in the following equations.

$$G = \frac{\partial f(x_{t-1})}{\partial \eta} = \begin{bmatrix} \frac{\partial \mathbf{q}_t^{wr}}{\partial \eta_\omega} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{T}_t^{wr}}{\partial \eta_{\dot{T}}} \\ \frac{\partial \omega_t^{wr}}{\partial \eta_\omega} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \dot{\mathbf{T}}_t^{wr}}{\partial \eta_{\dot{T}}} \end{bmatrix} \tag{B.11}$$

$$\frac{\partial \mathbf{q}_t^{wr}}{\partial \eta_\omega} = \frac{\partial \mathbf{q}_t^{wr}}{\partial \mathbf{q}'} \frac{\partial \mathbf{q}'}{\partial \omega'} \frac{\partial \omega'}{\partial \eta_\omega} \tag{B.12}$$

$$\frac{\partial \omega'}{\partial \eta_\omega} = \frac{\partial \mathbf{T}_t^{wr}}{\partial \eta_{\dot{T}}} = (\Delta t)^2 \tag{B.13}$$

$$\frac{\partial \omega_t^{wr}}{\partial \eta_\omega} = \frac{\partial \dot{\mathbf{T}}_t^{wr}}{\partial \eta_{\dot{T}}} = \Delta t \tag{B.14}$$

The derivative for $\frac{\phi^i}{\eta_\phi} = (\Delta t)^2$ and the remaining derivative are zero or Δt depending on if a particular noise channel contributes to a particular state.

B.2 Measurement Model Jacobian

The measurement model Jacobian is defines as $H = \frac{\partial \mathbf{h}(\mathbf{x}_t^-)}{\partial \mathbf{x}_t^-}$ where $\hat{\mathbf{z}} = \mathbf{h}(\mathbf{x}_{t-})$ transforms the state into measurement space. All states relate to the measurement through equations (100) and (102), so begin with these derivatives letting a represent any of the variables in equation (100) that are functions of states (R^{wn} and $\mathbf{T}^{wn}$). The symbol $\odot$ represents element-wise multiplication, the symbol $\otimes$ represents the Kronecker product, and Matlab notation is used to refer to particular rows or columns

of matrices.

$$\frac{\partial \hat{\mathbf{z}}_j}{\partial a} = \frac{1}{\alpha_j^2}\begin{bmatrix} \alpha_j \frac{\partial u'_j}{\partial a} - u'_j \frac{\partial \alpha_j}{\partial a} \\ \alpha_j \frac{\partial v'_j}{\partial a} - v'_j \frac{\partial \alpha_j}{\partial a} \end{bmatrix} \tag{B.15}$$

$$\frac{\partial u'_j}{\partial R^{wn}} = f\left[(R^{CW}_{(1,:)} \otimes [1,1,1]) \odot ([1,1,1] \otimes s_j^T)\right] \tag{B.16}$$

$$\frac{\partial v'_j}{\partial R^{wn}} = f\left[(R^{CW}_{(2,:)} \otimes [1,1,1]) \odot [1,1,1] \otimes s_j^T)\right] \tag{B.17}$$

$$\frac{\partial \alpha_j}{\partial R^{wn}} = (R^{CW}_{(3,:)} \otimes [1,1,1]) \odot ([1,1,1] \otimes s_j^T) \tag{B.18}$$

$$\frac{\partial u'_j}{\partial \mathbf{T}^{wn}} = f R^{CW}_{(1,:)} \tag{B.19}$$

$$\frac{\partial v'_j}{\partial \mathbf{T}^{wn}} = f R^{CW}_{(2,:)} \tag{B.20}$$

$$\frac{\partial \alpha_j}{\partial \mathbf{T}^{wn}} = R^{CW}_{(3,:)} \tag{B.21}$$

The derivatives for each point $\mathbf{s}_j$ to be used in the ppEKF are $\frac{\partial u'_j}{\partial \mathbf{s}_j} = f R^{CW}_{(1,:)} R^{wn}$, $\frac{\partial v'_j}{\partial \mathbf{s}_j} = f R^{CW}_{(2,:)} R^{wn}$, and $\frac{\partial \alpha_j}{\partial \mathbf{s}_j} = R^{CW}_{(3,:)} R^{wn}$ where R^{wn} is the rotation matrix associated with the component containing point $\mathbf{s}_j$.

The derivatives with respect to the root component quaternion are as follows.

Note that $\hat{\mathbf{q}}^{wr}$ is the normalized quaternion $\hat{\mathbf{q}}^{wr} = \frac{\mathbf{q}^{wr}}{\|\mathbf{q}^{wr}\|}$.

$$\frac{\partial R^{wr}}{\partial \hat{\mathbf{q}}^{wr}} = 2 \begin{bmatrix} \hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 & \hat{q}_4 \\ \hat{q}_2 & \hat{q}_1 & \hat{q}_4 & \hat{q}_3 \\ \hat{q}_3 & -\hat{q}_4 & \hat{q}_1 & -\hat{q}_2 \\ \hat{q}_2 & \hat{q}_1 & -\hat{q}_4 & -\hat{q}_3 \\ -\hat{q}_1 & \hat{q}_2 & -\hat{q}_3 & \hat{q}_4 \\ \hat{q}_4 & \hat{q}_3 & \hat{q}_2 & \hat{q}_1 \\ \hat{q}_3 & \hat{q}_4 & \hat{q}_1 & \hat{q}_2 \\ -\hat{q}_4 & \hat{q}_3 & \hat{q}_2 & -\hat{q}_1 \\ -\hat{q}_1 & -\hat{q}_2 & \hat{q}_3 & \hat{q}_4 \end{bmatrix} \tag{B.22}$$

$$\frac{\partial \hat{\mathbf{q}}^{wr}}{\partial \mathbf{q}^{wr}} = \frac{1}{\|\mathbf{q}^{wr}\|} I - \frac{1}{\|\mathbf{q}^{wr}\|^3} (\mathbf{q}^{wr} (\mathbf{q}^{wr})^T) \tag{B.23}$$

To calculate the derivatives of the rotation matrices R^{w,x_k} and the translation vectors $\mathbf{T}^{w,x_k}$ a hierarchical approach is required. Define $X^k = [x_1, x_2, \cdots, x_l]$ as a single chain starting at the root and ending with a component that has no children where each element of X identifies a component, x_k is the parent and x_{k-1} is the child for any particular joint.

$$\frac{\partial R^{w,x_k}}{\partial R^{x_{l-1},x_l}} = \begin{cases} \frac{\partial R^{w,x_k}}{\partial R^{x_{l-1},x_l}} & k = l \\ \frac{\partial R^{w,x_k}}{\partial R^{w,x_{k-1}}} \frac{\partial R^{w,x_{k-1}}}{\partial R^{x_{l-1},x_l}} & k > l \end{cases} \tag{B.24}$$

$$\frac{\partial \mathbf{T}^{w,x_k}}{\partial R^{x_{l-1},x_l}} = \begin{cases} \frac{\partial \mathbf{T}^{w,x_k}}{\partial R^{w,x_k}} & k = l \\ \frac{\partial \mathbf{T}^{w,x_k}}{\partial R^{w,x_{k-1}}} \frac{\partial R^{w,x_{k-1}}}{\partial R^{x_{l-1},x_l}} + \frac{\partial \mathbf{T}^{w,x_k}}{\partial \mathbf{T}^{w,x_{k-1}}} \frac{\partial \mathbf{T}^{w,x_{k-1}}}{\partial R^{x_{l-1},x_l}} + \frac{\partial \mathbf{T}^{w,x_k}}{\partial R^{w,x_k}} & k > l \end{cases} \tag{B.25}$$

$$\frac{\partial R^{w,x_k}}{\partial R^{x_{k-1},x_k}} = R^{w,x_{k-1}} \otimes I \tag{B.26}$$

$$\frac{\partial R^{w,x_k}}{\partial R^{w,x_{k-1}}} = I \otimes (R^{x_{k-1},x_k})^T \tag{B.27}$$

Next, the derivatives with respect to the joint parameters must be calculated. The translations are dependent on the joints and the rotation matrices.

$$\frac{\partial \mathbf{T}^{w,x_k}}{\partial R^{w,x_k}} = I \otimes (-J_c^{x_k,x_{k-1}})^T \tag{B.28}$$

$$\frac{\partial \mathbf{T}^{w,x_k}}{\partial R^{w,x_{k-1}}} = I \otimes (J_p^{x_k,x_{k-1}})^T \tag{B.29}$$

$$\frac{\partial \mathbf{T}^{w,x_k}}{\partial \mathbf{T}^{w,x_{k-1}}} = I \tag{B.30}$$

$$\frac{\partial \mathbf{T}^{w,x_k}}{\partial J_p^{x_{l-1},x_l}} = \begin{cases} R^{w,x_{k-1}} & k = l \\ \frac{\partial \mathbf{T}^{w,x_k}}{\partial \mathbf{T}^{w,x_{k-1}}} \frac{\partial \mathbf{T}^{w,x_{k-1}}}{\partial J_p^{x_{l-1},x_l}} & k > l \end{cases} \tag{B.31}$$

$$\frac{\partial \mathbf{T}^{w,x_k}}{\partial J_c^{x_{l-1},x_l}} = \begin{cases} -R^{w,x_k} & k = l \\ \frac{\partial \mathbf{T}^{w,x_k}}{\partial \mathbf{T}^{w,x_{k-1}}} \frac{\partial \mathbf{T}^{w,x_{k-1}}}{\partial J_c^{x_{l-1},x_l}} & k > l \end{cases} \tag{B.32}$$

Finally, the derivatives of the rotation matrices R^{x_{k-1},x_k} between two linked components with respect to the angle θ, ψ, and ϕ can be calculated using the articulation axis $\hat{\mathbf{a}}$ as an intermediate variable where $\hat{\mathbf{a}} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\theta)\cos(\psi) & \sin(\psi) \end{bmatrix}^T$.

$$\frac{\partial R^{x_{k-1},x_k}}{\partial \theta} = \frac{\partial R^{x_{k-1},x_k}}{\partial \hat{\mathbf{a}}} \frac{\partial \hat{\mathbf{a}}}{\partial \theta} \tag{B.33}$$

$$\frac{\partial R^{x_{k-1},x_k}}{\partial \psi} = \frac{\partial R^{x_{k-1},x_k}}{\partial \hat{\mathbf{a}}} \frac{\partial \hat{\mathbf{a}}}{\partial \psi} \tag{B.34}$$

$$\frac{\partial R^{x_{k-1},x_k}}{\partial \hat{\mathbf{a}}} = \begin{bmatrix} 2(1-\cos(\phi))\hat{a}^1 & 0 & 0 \\ (1-\cos(\phi))\hat{a}^2 & (1-\cos(\phi))\hat{a}^1 & \sin(\phi) \\ (1-\cos(\phi))\hat{a}^3 & -\sin(\phi) & (1-\cos(\phi))\hat{a}^1 \\ (1-\cos(\phi))\hat{a}^2 & (1-\cos(\phi))\hat{a}^2 & 0 \\ 0 & 2(1-\cos(\phi))\hat{a}^3 & -\sin(\phi) \\ \sin(\phi) & (1-\cos(\phi))\hat{a}^2 & (1-\cos(\phi))\hat{a}^2 \\ (1-\cos(\phi))\hat{a}^3 & \sin(\phi) & (1-\cos(\phi))\hat{a}^1 \\ -\sin(\phi) & (1-\cos(\phi))\hat{a}^3 & (1-\cos(\phi))\hat{a}^2 \\ 0 & 0 & 2(1-\cos(\phi))\hat{a}^3 \end{bmatrix} \tag{B.35}$$

$$\frac{\partial \hat{\mathbf{a}}}{\partial \theta} = \begin{bmatrix} -\sin(\theta)\cos(\psi) & \cos(\theta)\cos(\psi) & 0 \end{bmatrix}^T \tag{B.36}$$

$$\frac{\partial \hat{\mathbf{a}}}{\partial \theta} = \begin{bmatrix} -\cos(\theta)\sin(\psi) & -\sin(\theta)\sin(\psi) & \cos(\psi) \end{bmatrix}^T \tag{B.37}$$

$$\frac{\partial R^{x_{k-1},x_k}}{\partial \phi} = \begin{bmatrix} -\sin(\phi) & \cos(\phi)\hat{a}^3 & -\cos(\phi)\hat{a}^2 \\ -\cos(\phi)\hat{a}^3 & -\sin(\phi) & -\cos(\phi)\hat{a}^1 \\ \cos(\phi)\hat{a}^2 & -\cos(\phi)\hat{a}^1 & -\sin(\phi) \end{bmatrix} + \sin(\phi)(\hat{\mathbf{a}}\hat{\mathbf{a}}^T) \tag{B.38}$$

The remaining derivatives are zero.

Appendix C. AIAA SciTech 2017 Paper

A conference paper presented at AIAA SciTech in January 2017 is included in this Appendix. It outlines a method of sensing articulation for a single articulating panel using structure from motion techniques. The methods presented are not discussed elsewhere in the document so the paper is included here in its entirety.

AIAA SciTech Forum
9 - 13 January 2017, Grapevine, Texas
55th AIAA Aerospace Sciences Meeting

10.2514/6.2017-1329

Satellite Articulation Sensing using Computer Vision

David H. Curtis* and Richard G. Cobb†
Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433

Autonomous on-orbit satellite servicing benefits from an inspector satellite that can gain as much information as possible about the primary satellite. This includes performance of articulated objects such as solar arrays, antennas, and sensors. This paper presents a method of sensing and characterizing single-axis articulation of a solar panel on a target satellite from an inspector satellite in a natural circumnavigation trajectory around the target. The method presented uses trajectories of feature points on the target satellite to sense articulated motion. Motion segmentation is then used to separate feature points into groups consisting only of points that undergo the same motion. Structure from motion methods are used to develop point clouds representing each of the distinct objects. Finally, the axis and angle of the articulation are identified. The method is demonstrated using simulated data where point cloud radial error on the order of 2%, articulation axis error of approximately 0.04 radians, and relative articulation angle mean square error of approximately 0.002 radians were obtained.

Nomenclature

W	Trajectory matrix
u, v	Horizontal and vertical image coordinates
F	Number of frames in sequence or increment
P	Number of feature points
$\tilde{W}$	Normalized trajectory matrix
$\hat{\mathbf{i}},\hat{\mathbf{j}}$	Camera unit vectors expressed in body frame
x,y,z	Point cloud 3D coordinates
R	Motion matrix
S	Shape matrix
r	Trajectory matrix rank
k	Noise adjustment parameter
A	Affinity matrix
E	Entropy
Q	Metric constraint
$R_{a,b}$	Rotation matrix from frame b to frame a
T	Articulation axis motion subspace
m	Articulation axis point trajectory
$\mathbf{b}$	Articulation axis
$\theta(t)$	Articulation angle
σ	Trajectory matrix singular value
t	Time or frame number

I. Introduction

Autonomous satellite inspection and monitoring can improve the effectiveness of space operations by giving space operators information about the current state of health of a target satellite. Spacecraft could

*Graduate Student, Department of Aeronautics and Astronautics, 2950 Hobson Way, AIAA Member.
†Professor, Department of Aeronautics and Astronautics, 2950 Hobson Way, AIAA Associate Fellow.

be designed with a small inspector satellite capable of performing monitoring services for the main satellite, such as verifying performance of sensors, solar arrays, robotic arms, or communication antennas. To perform these monitoring services, it will be necessary for the inspector satellite to determine as much as possible about the satellite, and to draw conclusions autonomously. Many methods exist for determining satellite pose using computer vision; however, most make the assumption that the satellite is a rigid body, with feature points that do not move with respect to one another. This assumption is inaccurate in situations in which the satellite has components that articulate. To determine if the satellite is articulating, or articulating correctly, a computer vision algorithm is developed that autonomously senses articulation of a satellite component.

A. Background

Many algorithms have been developed using computer vision to determine the relative pose between a inspector satellite and a primary satellite in space. Some of these algorithms rely upon markers on the primary satellite that assist the computer vision algorithm in determining pose.[1,2] Others rely on prior knowledge of the primary satellites configuration.[3,4] Some methods rely on stereo vision systems,[5,6] some on monocular vision systems,[7,8,4] some use laser illumination of reflective markers.[9] They all use a computer vision method that identifies features in images and matches those features from frame to frame (or in corresponding frames in the case of stereo systems). The relative position of the features is then estimated using some type of estimation filter such as an extended Kalman filter or a particle filter. The attitude of the primary satellite is then estimated by using feature points to define a primary reference frame,[7,10] or by using a known model of the primary satellite.[3] Feature points can also be used to create a 3D model of the satellite.[11]

Previous work by Tweddle using stereo cameras has demonstrated that mass moments of inertia can be estimated for a spinning satellite using computer vision algorithms.[5] Similar work by Yu estimated mass center and mass moments of inertia for tumbling satellites.[10] If the mass moment of inertia were estimated, changes in the mass moments of inertia estimates could be sensed,[12] which would indicate some change such as articulation. However, this is a limiting case since the estimation of mass moments of inertia from computer vision is dependent on the angular velocity of the primary satellite and is ambiguous as to the real physical geometry when articulation is possible. A well controlled operational satellite is unlikely to be spinning or tumbling, therefore this method is unlikely to be applicable to sensing articulation in most cases.

While both the inspector and primary satellite are moving, the problem of finding the relative position between the two can be looked at using the assumption that either the primary is moving and the inspector is stationary, or the inspector is moving and the primary is stationary. Ghadiok et al. made the assumption that the inspector was stationary and the primary was moving so that angular velocity measurements from the inspector could be used to improve the accuracy of the relative position of feature points. They used a nonlinear complementary filter to fuse the feature position measurements from the computer vision algorithm with inspector gyro measurements.[11] Similarly, Philip et al. assumed the primary was stationary and used the inspector gyro measurements along with a Kalman filter and extended Kalman filter to estimate primary position and attitude respectively.[7]

Within the computer vision field, the task of separating different motions from an image stream is called motion segmentation. The term motion segmentation refers to any method that attempts to identify and separate different motions in a video sequence. Zappella et al. summarize a number of motion segmentation algorithms and classifies them based on the method used for segmentation.[13] While there are a number of strategies for motion segmentation outlined, the manifold clustering strategy seems the most applicable to the task of sensing and characterizing articulation.

Manifold clustering is a feature-based strategy based on analysis of a trajectory matrix. A trajectory matrix is a matrix of the 2D image coordinates of each feature point tracked through each frame. The trajectory matrix, W, is 2F x P in size where F is the number of frames in the sequence and P is the number of feature points. The make up of the trajectory matrix is shown in equation (1) where $u_{i,j}$ and $v_{i,j}$ are the

horizontal and vertical pixel-wise positions of the j-th feature point in the i-th frame in the sequence.

$$W = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,P} \\ v_{1,1} & v_{1,2} & \cdots & v_{1,P} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,P} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ u_{F,1} & u_{F,2} & \cdots & u_{F,P} \\ v_{F,1} & v_{F,2} & \cdots & v_{F,P} \end{bmatrix} \tag{1}$$

Assuming an affine camera model, the trajectory matrix of feature points on a rigid body will have a rank of at most four. Using that fact, motion segmentation becomes a problem of clustering the columns of the trajectory matrix into independent subspaces.

There are many methods of segmenting the trajectory matrix. Rao et al.[14] developed a method based on Agglomerative Lossy Compression which minimizes a cost function based on the 'coding length', or the number of bits required to describe the segmentation. The method begins by treating each point as a separate subspace. Then it merges subspaces and calculates the effect of the merge on the coding length. Merges that decrease the coding length are kept and the process is repeated until more improvement is not possible. Vidal et al.[15] proposed a different solution where they first project the trajectory matrix into a 5-dimensional subspace. They prove that different motions, even if the motions are partially dependent (such as articulated motions), will remain different along at least one dimension when projected onto a 5-dimensional subspace. Once projected, an n-degree polynomial (where n is the expected number of independent motions) of five variables is found. The derivative of that polynomial is evaluated for each point yielding a 5 element vector. If the two points are in the same subspace, the angle between these vectors will be zero (or π). The points can then be segmented according to the angle between the vectors.

Yan and Pollefeys[16] develop a method that has been termed Local Subspace Affinity (LSA). To segment the trajectory matrix according to different motions, LSA uses the concept that trajectories of feature points on the same object "lie in a low dimensional linear manifold" and trajectories of feature points on objects with different motions "result in different linear manifolds". LSA also uses the concept of locality which means that the basis of a trajectory and its nearest neighbors will lie in the same linear manifold as other trajectories of the same motion. Zappella et al.[17] enhanced the LSA algorithm by adding a method of automatically determining the rank of the trajectory matrix and the number of motions. To do this, they used the entropy of the affinity matrix as a measure of the quality of the guessed rank. The ELSA (Enhanced Local Subspace Affinity) algorithm, including code made available by Zappella, was used in the current work. There are many more methods of segmenting feature points. Zappella et al.[13] outlines a number of them and also provides a chart showing the attributes of each of the methods.

After motion segmentation, the trajectory matrix can be split into an individual trajectory matrix for each rigid body. For a trajectory matrix consisting of feature points of a rigid body under orthographic projection, Tomasi and Kanade[18] present an elegant approach to solving for the camera motion (motion matrix, R) and the shape of the object (shape matrix, S). Defining $\tilde{W}$ as the trajectory matrix minus the average of each row, they prove that

$$\tilde{W} = \begin{bmatrix} \hat{\mathbf{i}}_1^T \\ \vdots \\ \hat{\mathbf{i}}_F^T \\ \hat{\mathbf{j}}_1^T \\ \vdots \\ \hat{\mathbf{j}}_F^T \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_P \\ y_1 & \cdots & y_P \\ z_1 & \cdots & z_P \end{bmatrix} \tag{2}$$

$$\tilde{W} = RS \tag{3}$$

where R is the motion matrix which represents the orientation of the camera with respect to the object ($\mathbf{i}$ and $\mathbf{j}$), and S is the shape matrix which represents the 3D location of each feature point with respect to the object centroid. The $\tilde{W}$ matrix is at most rank 3 (since the translation has been subtracted out). The

motion matrix and the shape matrix can be found by taking the singular value decomposition of $\tilde{W}$, and extracting the first 3 rows and columns as follows.

$$\tilde{W} = U\Sigma V^T \quad (4)$$

$$\hat{R} = U_{(1:3,:)}[\Sigma_{(1:3,1:3)}]^{\frac{1}{2}} \quad (5)$$

$$\hat{S} = [\Sigma_{(1:3,1:3)}]^{\frac{1}{2}} V_{(1:3,:)} \quad (6)$$

$$\tilde{W} = \hat{R}\hat{S} \quad (7)$$

To convert $\hat{R}$ and $\hat{S}$, to R and S they must be multiplied by Q and Q^{-1} respectively where Q is chosen such that $\mathbf{i}_{1-F}$ and $\mathbf{j}_{1-F}$ are unit length and orthogonal to each other. This method of gaining structure from motion is heavily cited in the computer vision literature. The original method is derived for orthographic projection,[18] however similar methods have been developed using a weak perspective camera model[19] and a paraperspective camera model.[20]

Manifold clustering type motion segmentation algorithms rely on the fact that a trajectory matrix consisting of feature point trajectories from a rigid body will be at most rank four. However, not all of the algorithms can handle the case of articulation where multiple rigid bodies are linked. In the case of linked rigid bodies, the subspaces of the motions are not entirely independent. In fact, a universal joint results in a one dimensional intersection of subspaces and hinge joint results in a two dimensional intersection of subspaces.[21,22] Tresadern and Reid[22] propose a method of solving for articulated structure and motion by manipulating the motion matrix and shape matrix so the shape matrix is of block diagonal form. Yan and Pollefeys[21] solve for a trajectory of a feature point on the joint by finding the intersection of the two subspaces and solving for the linear combination of the columns of the intersection that maintains appropriate rank when augmented to the separated trajectory matrix. Additional work[23,24] discusses recovery of articulated motion, but in the context of the motion of human or animal appendages.

II. Approach

The method developed herein attempts to detect and characterize the articulation and shape of a simulated satellite with an articulated solar panel from a simulated orthographic camera in an elliptical circumnavigation route around the primary satellite. Observation of the primary satellite begins when no articulation is present, however an adjustment to this method is mentioned for cases when articulation is occurring throughout the observation. The method begins with a trajectory matrix of feature points. Image processing and feature point tracking is not included, but will be included in future work. The primary concepts employed by this method are motion segmentation,[17] structure from motion,[18] and articulation axis recovery.[21] The contribution for this paper is the combination of these methods to characterize the primary satellite with two distinct point clouds and the articulation of the solar array with an articulation axis and angle.

A. Simulation set-up

The current method was developed and tested using a simulated trajectory matrix that was generated for a point cloud representing a simple satellite. Figure 1a shows the model satellite which is approximately 14 units wide (x-direction), 3 units deep (y-direction), and 5 units tall (z-direction). Points were randomly spread over each face to build the point cloud shown in Figure 1b. Note the articulated solar panel was modeled with curvature. Aspects of this method rely on the normalized trajectory matrix ($\tilde{W}$) to be rank 3. If all of the points on the objects are on the same plane, $\tilde{W}$ will only be of rank 2. This is a limitation that will be investigated in future work.

The trajectory matrix was created by converting the simulated point cloud into the imagery coordinates that would be seen from a camera viewing the points as feature points while the inspector satellite is in a 2x1 elliptical circumnavigation route around the primary. Figure 2a shows the route the inspector takes around the primary satellite. Figure 2b shows one frame from a sequence of frames of feature point locations, simulating the results of a feature point tracking algorithm. Note that current work does not include feature point detection and matching. The 3D point cloud is translated directly to a 2D location via orthographic projection and is added to the trajectory matrix. A shadowing algorithm is included so only the points that are within view of the camera are projected into image coordinates at each frame.

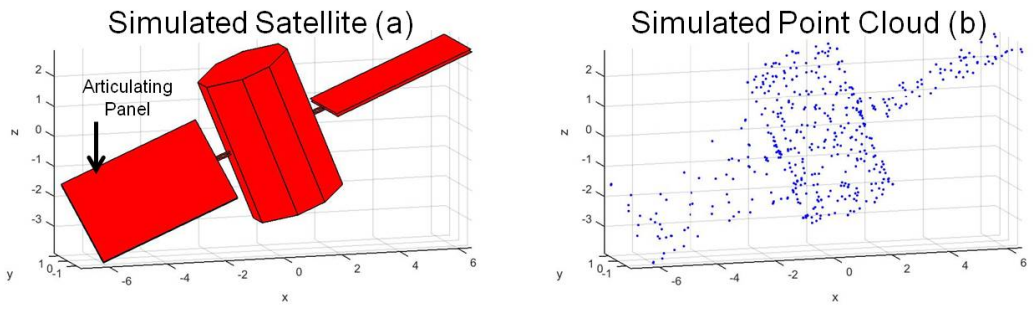

Figure 1. Simulated satellite used in testing: (a) Simulated Satellite (b) Simulated Point Cloud

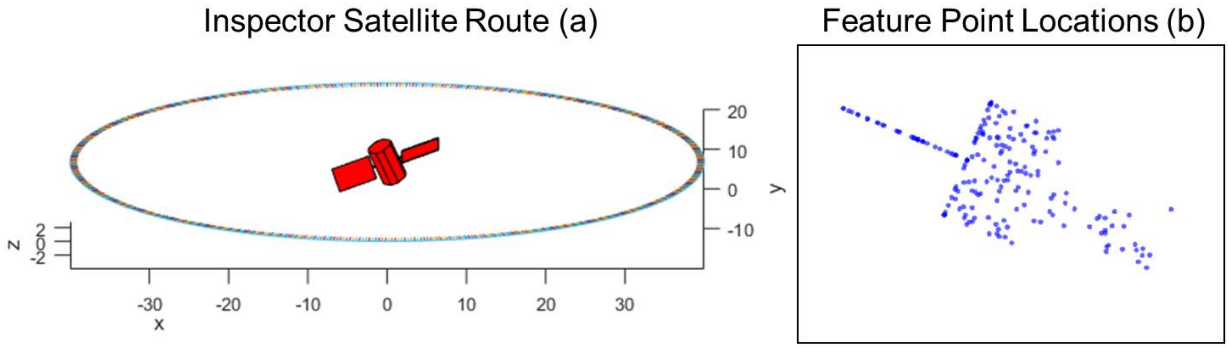

Figure 2. Inspector satellite route and feature point locations: (a) Inspector satellite route (b) Feature point locations

This elliptical circumnavigation route may not be the optimal trajectory for an inspection mission, however it provides a realistic yet simple test case for algorithm development. Additional trajectories will be investigated in future work.

When articulating, the articulation angle of the solar array follows a sine wave fluctuating $\pm 30°$ from its start position at a rate of four cycles per circumnavigation orbit. For the remainder of the paper 'main body' refers to the main body of the primary satellite and 'object 2' refers to the articulating solar array.

B. Method

1. Sense Articulation Start

Figure 3 provides an overview of the current method that is the main contribution of this paper. The steps of the process are further explained below.

The first step is to determine when articulation starts. The most straight forward way to sense if the image stream consists of motion from a single rigid body or multiple rigid bodies is to check the rank of the trajectory matrix (W), equation (1). When articulation starts, the rank of the trajectory matrix increases. A simple way to detect this is to look at the ratio of the fourth and fifth largest singular values (σ_5/σ_4) of W. To determine when articulation starts, a window consisting of ten frames of data is progressed through the trajectory matrix. The frame of articulation start is taken as the first time when σ_5/σ_4 raises above a threshold. Care must be taken or this method will produce poor results if the threshold is not tuned in accordance with the noise in the trajectory matrix. It is also computationally expensive. Improvements to this area are certainly possible.

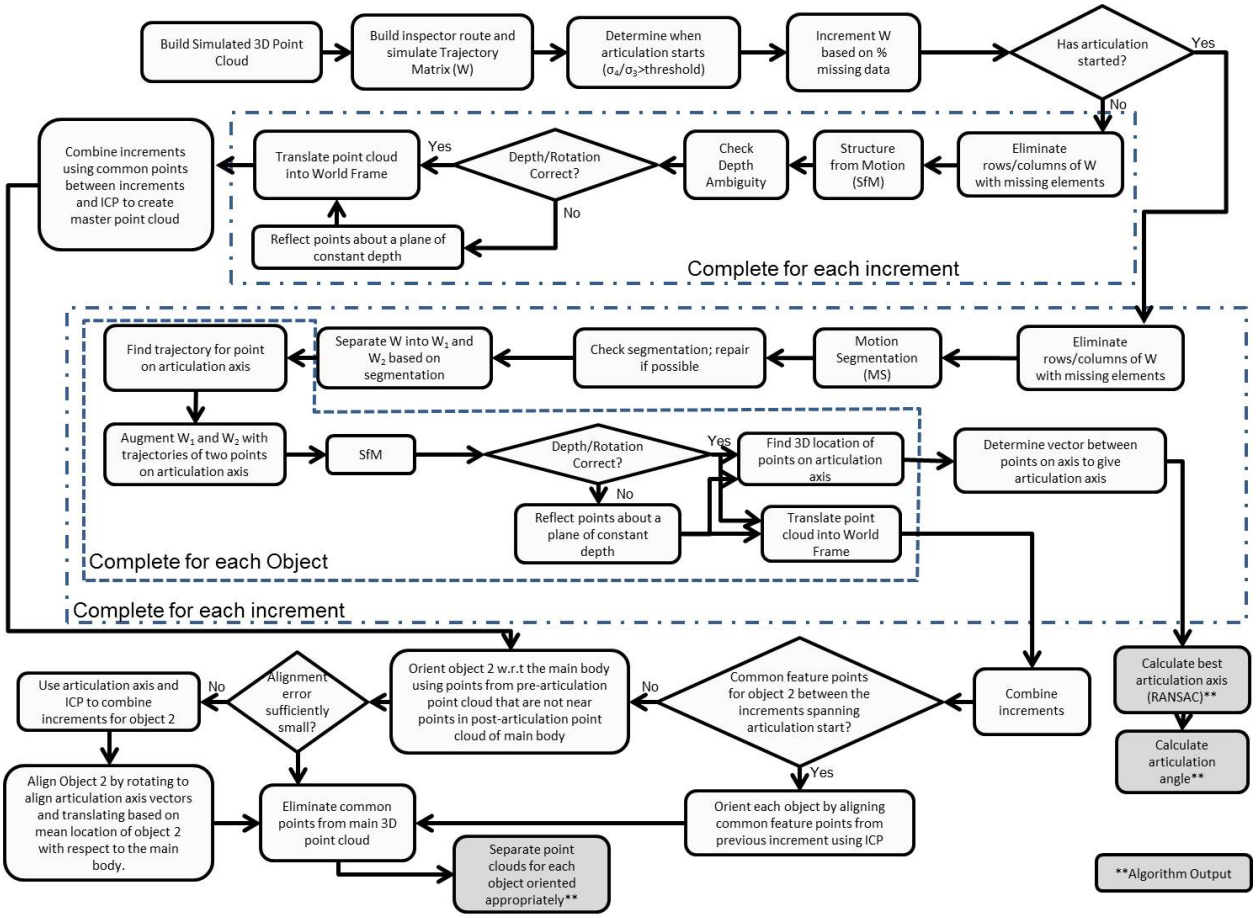

Figure 3. Overview of the proposed method

2. *Increment Trajectory Matrix*

The full trajectory matrix is 2FxP, where F is the number of frames and P is the number of feature points. Since points drop in and out of view, the matrix has undefined elements. While some motion segmentation algorithms are capable of handling missing data,[14,25,15] the extent of the missing data in a trajectory matrix from a circumnavigation inspection trajectory is higher than these algorithms can handle. Therefore, the trajectory matrix is first incremented into sections containing a manageable amount of missing elements. The trajectory matrix is incremented by progressing through frames (rows) and eliminating points (columns) that contain less more than 50% missing data until the overall increment has more than 10% missing data. The trajectory matrix is incremented in two groups, one before the articulation start

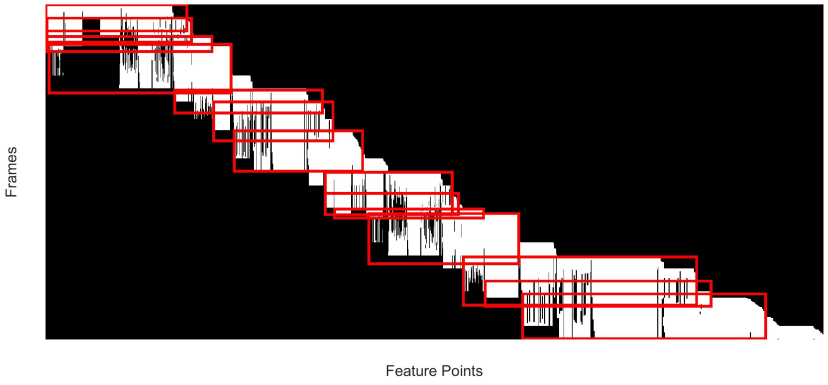

Figure 4. Example of a full trajectory matrix with increments. Empty elements are black, full elements are white, and red boxes represent the increments

frame and one after articulation has started. An example trajectory matrix with incrementation is shown in Figure 4.

3. Motion Segmentation

The increments after the articulation starts must be segmented to determine which points are part of which object. To do this, the Enhanced Local Subspace Affinities (ELSA) algorithm developed by Zappella et al.[17] is used with a slight modification to force segmentation into two groups. The ELSA algorithm does not deal with missing data, so first the missing data points must be removed. Methods of filling missing data in trajectory matrices such as (Rao and others) exist and will be applied in future work, however, for this work the rows/columns of the trajectory matrix are eliminated in a way which attempts to delete as little data as possible.

The trajectory matrix is then segmented using the ELSA algorithm with a slight modification. The ELSA algorithm is capable of determining the number of separate motions in a given trajectory matrix. Most of the time the number of segments is appropriately calculated, however sometimes ELSA determines there are more than two motions. Future work will remedy this problem in a more robust manner, but currently the solution is to limit the number of objects to two. A full description of the ELSA algorithm[17] and the underlying LSA algorithms[16] are available in the respective papers; an overview of the method is provided below.

The goal of manifold clustering algorithms such as ELSA is to separate feature point trajectories into groups that represent different motions. Figure 5 graphically shows an example of feature point trajectories from the simulated satellite used in this work both before segmentation (Figure 5a) and after (Figure 5b-c) the trajectory matrix has been segmented into two different objects. Each line in Figure 5 represents the image coordinates (u and v) of a feature point over time.

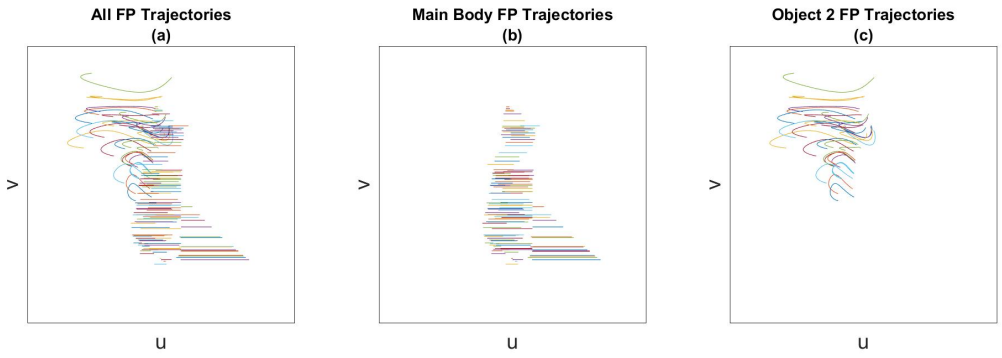

Figure 5. Motion Segmentation Illustration: (a) All Feature Point (FP) Trajectories, (b) Main body FP Trajectories, (c) Object 2 FP Trajectories

The ELSA algorithm begins by decomposing the trajectory matrix (W) with singular value decomposition to create $U_{2F\,x\,r}$, $\Sigma_{r\,x\,r}$, and $V_{r\,x\,P}$. Equation (8) can be used to find the rank (r) of W. In equation (8) σ_i are the ordered singular values of the trajectory matrix (W). The columns of the V matrix can now be thought of as the direction of the feature points motion in r-dimensional space, or their location on an r-dimensional sphere. Columns corresponding to feature points with the same motion will lie on lower dimensional cuts of this r-dimensional sphere.

$$r = \underset{r}{\operatorname{argmin}} \left(\frac{\sigma_{r+1}^2}{\sum_{i=1}^{r} \sigma_i^2} + kr \right) \tag{8}$$

Features will likely have the same motion as features closest to themselves in the transformed space (the r-dimensional sphere). 'Closeness' can be measured by Euclidean distance or by the principal angle between them. For each feature point, the local subspace is found by first finding its n nearest neighbors (using Euclidean distance or principal angle) and then finding a basis for the combination of those points using the SVD. The size of the basis is determined by finding the rank of the local subspace using equation (8).

Next, the principal angles between these subspaces must be determined. To do this, first define two subspaces as $C(\alpha)$ and $C(\beta)$, each with normalized columns. The principal angles (θ_{ss}) between these two

subspaces are the angles between each column of $C(\alpha)$ and the closest (largest dot product) column of $C(\beta)$. These principal angles can then be used in equation (9) to calculate an affinity (A) between each subspace.

$$A_{i,j} = e^{-\sum_{k=1}^{M} sin^2(\theta_{ss_k})} \tag{9}$$

The affinity matrix is a measure of the similarity between subspaces created from each trajectory and its nearest neighbors. When the estimation of rank used to decompose W into $U_{2F\,x\,r}$, $\Sigma_{r\,x\,r}$, and $V_{r\,x\,P}$ (and therefore to create the affinity matrix) is nearly correct, the principal angles between subspaces generated by trajectories under the same motion are smaller while the subspaces generated by trajectories under different motions are larger. Visually, this causes the affinity matrices with the correct rank (generated from the correct selection of k) to look like a block diagonal with the number of blocks corresponding to the number of motions.

The primary addition of the ELSA algorithm[17] to the LSA algorithm[16] is that, affinities are calculated using different values of k in equation (8), which is directly related to the rank estimate. The quality of the rank estimate can be quantified by the entropy, defined in equation (10), of the affinity matrix ($E(A(r))$). In equation (10), $h_{A(r)}$ is the histogram count when sorting the values in the $A(r)$ affinity matrix into 256 bins.

$$E(A(r)) = -\sum_{i=0}^{256} h_{A(r)}(i) log_2(h_{A(r)}(i)) \tag{10}$$

With entropy as a measure of the quality of the affinity matrix, the choice of k that produces the highest entropy can be determine. Once k is found, the rank of the trajectory matrix is found from equation (8) and the corresponding affinity matrix can be used for the next steps. The ELSA algorithm next estimates the number of motions by looking at the eigenvalues of the symmetric normalized Laplacian matrix. As mentioned earlier, for the current work, the number of motions is given as two, so this step is ignored.

Once the affinity matrix is calculated, a spectral clustering algorithm is used to determine which columns of the affinity matrix match up to which motion. The normalized cuts method outlined by Shi and Malik[26] is proposed by Yan and Pollefeys.[16,27] The normalized cuts method recursively segments the affinity matrix until the number of segments is met. Numerous cuts (or segmentations) of the affinity matrix are evaluated to determine which one minimizes the 'Ncuts', or normalized cuts, criteria shown in equation (11). Shi and Malik[26] prove the normalized cuts criteria measures both the total dissimilarity between the different groups (G_1 and G_2) and the similarity within the groups.

$$Ncut(G_1, G_2) = \frac{cut(G_1, G_2)}{assoc(G_1, A)} + \frac{cut(G_1, G_2)}{assoc(G_2, A)} \tag{11}$$

In equation (11) the 'cut(G_1,G_2)' operator is the sum of all positions in the affinity matrix where the column index is in group G_1 and the row index is in group G_2. The 'assoc(G_1,A)' operator is sum of each of the rows of A corresponding to group G_1.

This process of motion segmentation is repeated for all increments after the start of articulation. The result at each increment is a vector of length P containing the group identifiers for each point.

4. Structure from Motion

For the increments before articulation starts, the structure from motion process can be accomplished without segmentation since all points are on the same rigid body. The motion matrix (R) and the shape matrix (S) are calculated for each increment using the singular value decomposition method presented by Tomasi and Kanade.[18] After singular value decomposition, the metric constraints, outlined in equation set (12), are used to find Q. These constraints are linear in the elements of Ω where $\Omega = QQ^T$. Linear least squares can be used to solve for the elements of Ω, and provided Ω is positive definite, Q can be found through Cholesky factorization of Ω. If Ω is not positive definite, a non-linear least squares solver is used to find the elements of Q directly. The metric constraint is applied as follows: $S = Q^{-1}\hat{S}$ and $R = \hat{R}Q$.

$$\begin{aligned} \hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{i}}_f &= 1 \\ \hat{\mathbf{j}}_f^T Q Q^T \hat{\mathbf{j}}_f &= 1 \\ \hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{j}}_f &= 0 \end{aligned} \tag{12}$$

One issue with the Tomasi and Kanade's SVD approach to finding R and S is that there can be a depth ambiguity.[28] Since there is no depth information contained in the orthographic projection, it is possible to have two sets of motion and shape matrices that adhere to the metric constraints. One combination will be the correct one, with the correct pairing of rotation and shape, while the other one will be a reflection of the correct shape across some plane of constant depth and motion matrix consisting of motion in the opposite direction. Figure 6 demonstrates this depth ambiguity by showing the calculated shape matrix and the truth point cloud projected into the camera frame. The fact that the shape is not correct is evident when looking perpendicular to the camera axis (the z axis) (Figure 6a), while Figure 6b shows that the points match up exactly when looking down the camera axis.

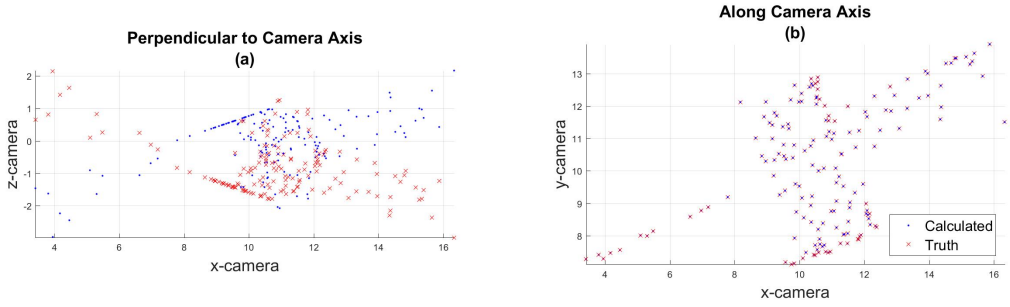

Figure 6. Demonstrated depth ambiguity: Calculated point cloud compared to truth in the camera frame: (a) Viewed perpendicular to camera axis, (b) Viewed along camera axis

Multiple methods of solving this ambiguity problem have investigated. Szeliski and Kang[29] suggest that the depth reversal can be sensed by reflecting the shape about a plane at a constant depth, recalculating the trajectory matrix and determining if the error is reduced. This method was attempted, however the difference in error after reflection was so small it did not give good results. Instead for our application, the known direction of the circumnavigation route was compared to the calculated motion matrix. As is evident from Figure 7, given the known direction of circumnavigation, if the motion matrix is correct the cross product of consecutive $\hat{\mathbf{i}}$ will be in the same direction as $\hat{\mathbf{j}}$. To correct for this ambiguity, for each increment (and each object if applicable) equation set (13) is used to calculate the angle between the cross product of consecutive $\hat{\mathbf{i}}$ and the $\hat{\mathbf{j}}$ vector. If the average of these angles is over $\pi/2$ the shape matrix is reflected through a plane of constant depth.

$$\phi = cos^{-1}\left(\frac{\hat{\mathbf{i}}(t)\times\hat{\mathbf{i}}(t-1)}{|||\hat{\mathbf{i}}(t)\times\hat{\mathbf{i}}(t-1)|||}\cdot\hat{\mathbf{j}}(t)\right) \tag{13}$$

For each increment, the motion and shape matrices are calculated using the SVD method. Next, the algorithm steps through each frame and calculates a rotation matrix that will convert the shape matrix into the camera frame. Equation (14) and (15) are used to calculate this rotation matrix; $\hat{\mathbf{i}}(t)$ and $\hat{\mathbf{j}}(t)$ are the rows of the motion matrix corresponding to camera frame t. In most cases, M and $R_{cam,b}$ are equal, however noise can cause the metric constraints on R not to be met perfectly, in which case the method in

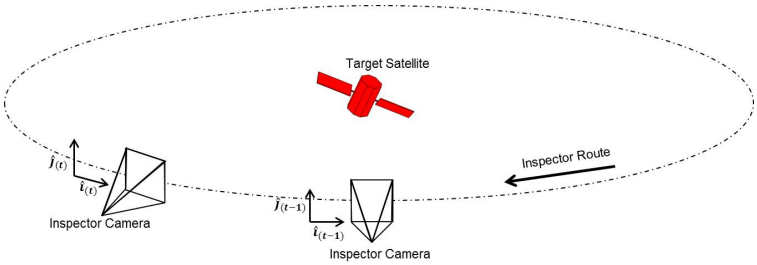

Figure 7. Diagram of inspector camera orientation

equation (15) and outlined by Horn et al.[30] produces a best-fit orthonormal rotation matrix.

$$M = \begin{bmatrix} \hat{\mathbf{i}}(t)^T \\ \hat{\mathbf{j}}(t)^T \\ (\hat{\mathbf{i}}(t) \times \hat{\mathbf{j}}(t))^T \end{bmatrix} \tag{14}$$

$$R_{cam,b}(t) = M(M^T M)^{-1/2} \tag{15}$$

$$S_w(t) = R_{w,cam} R_{cam,b} S \tag{16}$$

The $S_w(t)$ values are then averaged to give the point cloud in the world frame, although all of these values should be identical assuming the depth ambiguity is solved appropriately.

For increments after the articulation starts, the first step is to separate the trajectory matrix according to the segmentation. The structure of each of the independent objects is calculated in the same way as above with some additional steps, including a method of determining the articulation axis, a check on the quality of the segmentation, and a slightly different ambiguity test for the articulated object. The method of determining the articulation axis will be discussed in the next section.

To check the quality of the segmentation, the ratio of the third and fourth largest singular values (σ_4/σ_3) is calculated for each of the trajectory matrices from each group ($\tilde{W}_1$ and $\tilde{W}_2$). If the ratio is high enough to suggest the trajectory matrix contains more than one motion, attempts are made to find the points that are poorly segmented. It is assumed that the most likely points to be poorly segmented are the ones that are furthest from $\bar{W}$ (the mean of the rows of the trajectory matrix). The furthest points from $\bar{W}$ in each frame are eliminated from W one at a time. If removal of the point caused σ_4 to decrease significantly that point is switched to the other group. If this method fails to repair a segmentation sufficiently, the data from the object in that increment is not used in the next step. This method is effective and simple for a small number of mis-segmented points far from the center of the group, however other methods of repair such as Zappella et al.[31] may be more effective.

5. Combining Point Clouds from each Increment

The structure from motion process yields point clouds for each object (after articulation start) in each increment. These point clouds must be combined to create a point could that represents the entire primary satellite. Combining the point clouds from the increments before the articulation start is fairly straight forward. An iterative closest point (ICP) algorithm (developed by Kjer and Wilm;[32] available at MathWorks.com[33]) is used to match up the points that are common between increments. The ICP algorithm[34,35] finds the rotation and translation that minimizes the distance between two point clouds. The point cloud from each successive increment is added to the previous point cloud by applying the translation/rotation that aligns the common points between increments.

The same process continues after the articulation starts, however two point clouds, one for each object, must be maintained. The point cloud of the entire satellite without articulation serves as a master for how the articulated object is oriented with respect to the main body. If there are a sufficient number of points for both objects that are common between the last increment before articulation starts and the first increment that includes articulation, the process for orienting the point clouds is as previously described with a slight modification. For the ICP algorithm to converge to the appropriate rotation/translation, the two point clouds must be close to the same orientation. Before articulation, all the points are in the world frame, so their orientation was nearly correct. Since the articulation changes the orientation of object 2 (the articulated solar array), the orientation of the point cloud is not always close enough for ICP to converge to the correct rotation/translation. To solve this problem, a method from Horn et al.[30] is used to provide a rotation matrix that will match up the points.

$$M = \sum_{i=1}^{n} \mathbf{r_{l,i}} (\mathbf{r_{r,i}})^T \tag{17}$$

$$R_{l,r} = M(M^T M)^{-1/2} \tag{18}$$

In equation (18), $\mathbf{r_l}$ are the vectors from point cloud center to each point in the existing point cloud and $\mathbf{r_r}$ are the vectors from the point cloud center to each point in the point cloud to be added. $R_{l,r}$ can then be used to bring the new points close enough in orientation so that ICP converges to the correct solution.

If there are not enough common points between the last increment before articulation starts and the first increment with articulation to orient object 2 with respect to the main body, an alternative method is used. The points found in the master point cloud (developed before articulation starts) that are not near any points attributed to the main body after the articulation starts are likely part of object 2, therefore, these points are used to orient object 2 with respect to the main body. Performing ICP using these points and the point cloud from object 2 determines the rotation and translation necessary for alignment. If the error from ICP is over a threshold, the translation is determine by aligning the articulation axis points for object 2 to calculated articulation axis (Section 6). To find the appropriate translation, an additional step is taken in the structure from motion phase. The trajectory matrix for the points in the main body is augmented with $\bar{W}_2$, the row average of the trajectory matrix from object 2. Next, the shape matrix is found for this augmented matrix. The point corresponding to $\bar{W}_2$ represents the approximate location of the centroid of object 2 with respect to the main body. This value is stored for each increment, averaged, and applied to object 2 to put it in approximately the correct location with respect to the main body. This method can also be applied to orient object 2 with respect to the main body in the case where articulation is occurring throughout the observation time.

During the increments before articulation started, all of the points are considered part of the main body. Once articulation is sensed, and the object 2 point cloud is built, the points in the main body that are actually part of object 2 must be removed from the point cloud. To do this, the radial distances between the points in object 2 and the points in the main body are calculated. Any points in the main body that are near points in object 2 are eliminated from the main body point cloud.

The result of this process is a set of two point clouds that represent the main body and the solar array in the world frame. The solar array will be represented in the same orientation as before articulation started. A current limitation of this method is the way the groups are matched up from increment to increment. Matching is currently done by assigning the label 'group 1' to the group with more points and 'group 2' to the group with two points. While this method yields good results for the scenario considered herein, this method is not robust to a variety of different scenarios and will be improved in future work.

6. Find Articulation Axis and Angle

To characterize the articulation it is necessary to determine how the solar array moves with respect to the main satellite body. Yan and Pollefeys'[21] work shows that for articulated motion, the intersection of the motion subspaces of the two objects is the motion subspace of the link. If the link is a hinge joint (such as when a solar panel rotates on an axis) this is a 2-dimensional subspace. To find that subspace, take the SVD of the trajectory matrices of the two objects: $W_1 = U_1\Sigma_1 V_1^T$, $W_2 = U_2\Sigma_2 V_2^T$. The motion subspaces ($\tilde{U}_1$ and $\tilde{U}_2$) are formed by the first four columns of U_1 and U_2. Since the intersection is a 2-dimensional subspace, the matrix $[\tilde{U}_1 \| \tilde{U}_2]$ will have two singular values equal to zero. Taking the SVD of $[\tilde{U}_1 \| \tilde{U}_2]$ and setting N equal to the columns of V corresponding to the zero singular values yields,

$$\left[\tilde{U}_1 \| \tilde{U}_2\right] N = 0 \tag{19}$$

$$\left[\tilde{U}_1 \| \tilde{U}_2\right] \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} = 0 \tag{20}$$

$$T = \tilde{U}_1 N_1 = -\tilde{U}_2 N_2 \tag{21}$$

where T is the subspace representing the intersection of W_1 and W_2.

Once the subspace of the intersection is found, specific trajectories that lie on the axis can be found. The requirements for a trajectory (m) to lie on the axis are that it lies in the subspace T and it does not increase the rank of W_1 (or W_2) when it is appended as an additional column. To meet the first constraint m must be a linear combination of the columns of T, or

$$m(\alpha, \beta) = T \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{22}$$

where α and β are scalars. The second criteria is met by ensuring the augmented matrix, $[\tilde{W}_1 | (m(\alpha, \beta) - \bar{W}_1)]$ has a rank of 3, where $\bar{W}_1$ is the row average of W_1 and $\tilde{W}_1$ is $W_1 - \bar{W}_1$. Yan and Pollefeys manipulate this constraint to yield an equation in which the summation of five determinants is equal to zero.[21] Solving

these constraints yields a linear equation in α and β. Values of α and β can be selected to give a point on the trajectory at a particular u or v coordinate using equation (22).

For each object in each increment, two trajectories representing two points on the articulation axis are augmented to the trajectory matrix. The point cloud calculated from this augmented trajectory matrix contains the 3D points on the articulation axis. These points are translated to the world frame as described previously. The difference between these two points represents the direction of the articulation axis in the world frame. Due to mis-segmented points and noise, the articulation axis is not the same in each increment. Therefore, a RANSAC routine was developed to reject outliers and select the best articulation axis ($\mathbf{b}$).

If required, the articulation axis can be used to combine point clouds for object 2 from different increments. First, the structure from motion is applied for object 2 in each increment. For each increment, the vector between the axis points for that increment ($\mathbf{v}_i$) is found. This is aligned with the best articulation axis ($\mathbf{b}$) by converting them to unit vectors and applying the rotation matrix (R_{align}) found using the equation set (23).

$$\begin{gathered}\Phi = cos^{-1}(\mathbf{b}\cdot\mathbf{v}_i)\\ \hat{a} = \mathbf{b}\times\mathbf{v}_i\\ R_{align} = cos(\Phi)[I] + (1-cos(\Phi))\hat{a}\hat{a}^T - sin(\Phi)\begin{bmatrix}0 & -\hat{a}_3 & \hat{a}_2\\ \hat{a}_3 & 0 & -\hat{a}_1\\ -\hat{a}_2 & \hat{a}_1 & 0\end{bmatrix}\end{gathered} \tag{23}$$

The articulation angle ($\theta(t)$) within each increment is determined by looking at how much object 2 has rotated about the articulation axis from the start of the increment. First, the body frame is rotated such that $\mathbf{i}_w^b(t=1)$ (the x-axis of the body frame expressed in the world frame) at frame one is aligned with the rotation axis as shown in Figure 8a. This produces a reference frame of unit vectors in the body-aligned frame expressed in the world frame, $R_{w,ba}(t=1)$. Next, $R_{w,ba}(t)$ is calculated for each frame in the same way. The articulation angle is the angle between the y-axis of $R_{w,ba}(t)$ and the y-axis of $R_{w,ba}(t=1)$ (Figure 8b).

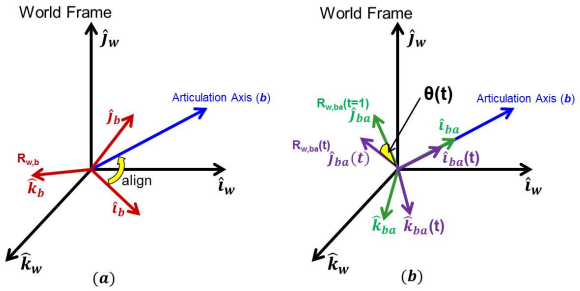

Figure 8. Articulation angle calculation method: (a) Orient object 2 frame with Articulation Axis (b) Compare $R_{w,ba}(t=1)$ to $R_{w,ba}(t)$ to find $\theta(t)$

III. Results and Limitations

A. Results

Table 1. Results.

Metric	Single Trial (Figures 9 & 10)	Mean of 100 Trials
Mean point-wise radial error (units)	0.092 units	0.075 units
Percent of points with under 0.2 unit radial error	91%	96%
Articulation axis error (radians)	0.0067 rad	0.035 rad
Articulation angle mean square error (radians)	0.000022 rad	0.0016 rad

For all results presented, the inspector satellite circumnavigated the primary satellite twice. The solar array articulation began after the first circumnavigation. The trajectory matrix contained a total of 600 simulated frames of feature point information. The truth point cloud contained approximately 500 points total, however, the simulated trajectory matrix contained approximately 1400 points. This is because points

that are re-seen after dropping out of view for a significant amount of time were considered new points. This is to more closely aligns to a situation in which feature point trajectories are generated using imagery and a feature point tracking algorithm. Figure 9 shows the calculated point clouds and the truth point cloud with the solar array oriented as it was before articulation starts. Note that not all 500 points are shown. Points were removed during the process to ensure the trajectory matrix was full for motion segmentation and structure from motion calculations.

To measure the quality of the results, the radial distance between the truth points and the calculated points is calculated. The average distance and the percentage of distances under 0.2 units is shown in Table 1. To measure the quality of the articulation axis, the angle between the true articulation axis and the calculated articulation axis is determined. Due to the random point cloud generation and other aspects this algorithm is not deterministic. Therefore the results presented in Table 1 are for the simulation that was used to generate Figure 9 as well as the averaged results for 100 simulations (trials).

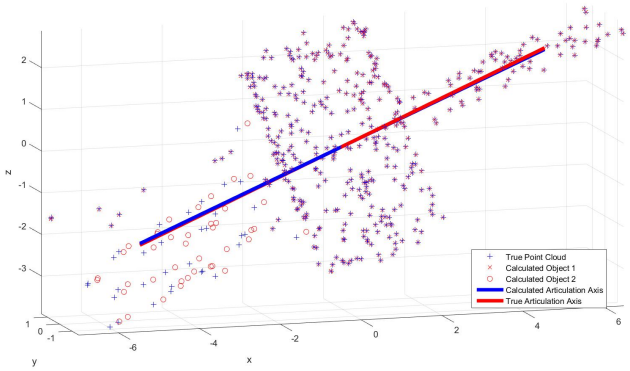

Figure 9. Point cloud and articulation axis results for a single trial.

The articulation angle is calculated for each increment in which good data was available. Data was considered 'bad' if the ratio σ_4/σ_3 was over a threshold. Figure 10 shows the true and calculated articulation angle as measured from the start of the increment with the mean square error (MSE) for that increment.

Results demonstrate that this method is capable of determining distinct point clouds for a satellite with an articulated solar array as well as being able to accurately describe the articulation.

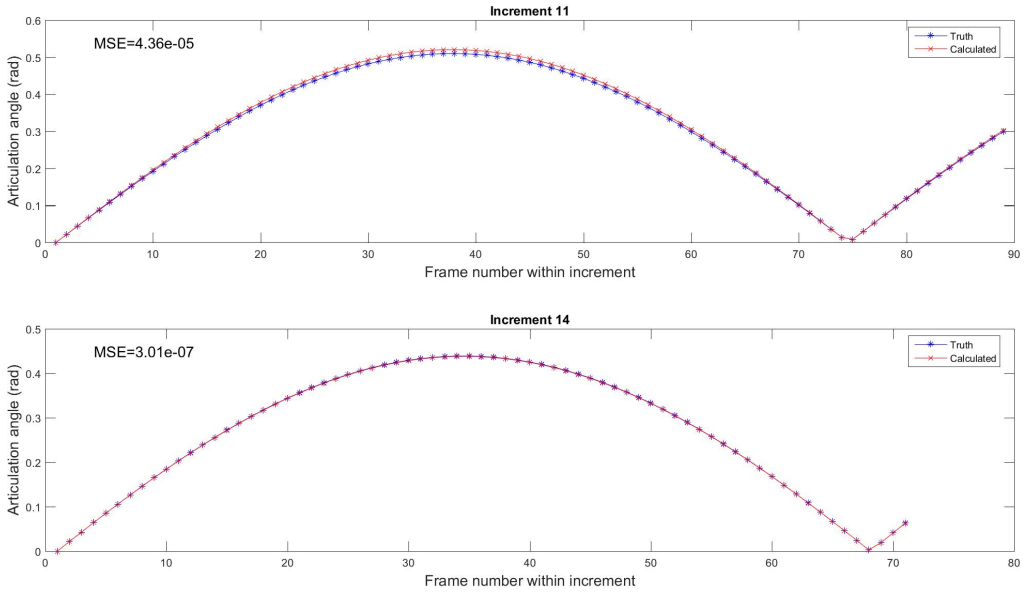

Figure 10. Articulation angle results for a single trial.

B. Limitations

This method produces good results for the simulated scenario described, however limitations exist.

- The feature points on the articulated object can not be co-planar. This is a significant limitation since solar arrays are often planar. Much of the underlying assumptions for the techniques used in this method, such as structure from motion using factorization, rely on $\tilde{W}$ to be rank 3. If all the points in the trajectory matrix are co-planar, $\tilde{W}$ will only be rank 2. Methods of fixing this limitation likely exist.
- The current implementation is not robust to noise in the feature point trajectories. If the trajectory matrix is created from actual (or rendered) imagery, there will be noise in the measurements. Aspects of this method, for instance sensing when articulation starts, are likely to fail unless adequately tuned for noise.
- The method of matching objects from increment to increment is not robust. The number of feature points in each object is currently used. This works well for this simulation since there are far more feature points on the main body, however it would not work well if the number of feature points on each object fluctuates significantly with view angle.
- There is currently no metric on quality of the results. The algorithm can not currently determine the expected accuracy of the calculated point clouds, orientation, articulation axis, or articulation angle.

IV. Conclusion and Future Work

Existing computer vision methods for motion segmentation and structure from motion are used to develop a method for sensing articulation of a solar panel on a simulated satellite. The method is demonstrated to be capable of detecting articulation, creating 3D point clouds for the main body and the articulated object, calculating the articulation angle, and calculating the articulation axis. This work contributes toward the goal of autonomous on-orbit satellite inspection and servicing. Future work in the area of sensing and characterizing articulated motion in satellites includes:

- Implement a Kalman filter approach for developing the point clouds. Kalman filters are often used in computer vision to determine the 3D location of feature points, however one of the primary assumptions in these implementations is that the feature points do not move with respect to each other. However, the motion segmentation methods described could be used to separate the points into groups when articulation starts, then separate Kalman filters could be used for the separate objects.
- Add capability to detect multiple objects and multiple types of articulation.
- Investigate different inspector routes to determine which routes provide the best characterization.
- Add the capability to track feature points in imagery.

Acknowledgments

The authors would like to acknowledge Dr Frank Chavez and others at the Air Force Research Laboratory Space Vehicles directorate for their guidance in this research.

References

[1]Tweddle, B. E. and Saenz-Otero, A., "Relative Computer Vision-Based Navigation for Small Inspection Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 5, 2015, pp. 969–978.

[2]Ho, C.-C. J. and McClamroch, N. H., "Automatic spacecraft docking using computer vision-based guidance and control techniques," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 2, 1993, pp. 281–288.

[3]Naasz, B. J., Van Eepoel, J., Queen, S. Z., Southward, C. M., and Hannah, J., "Flight results from the HST SM4 Relative Navigation Sensor system," *Advances in the Astronautical Sciences*, Vol. 137, 2010, pp. 723–744.

[4]Hannah, S. J., "A relative navigation application of ULTOR technology for automated rendezvous and docking," *Proceedings of SPIE, Spaceborne Sensors III*, edited by R. T. Howard and R. D. Richards, Vol. 6220, 2006, pp. 62200E–1–12.

[5]Tweddle, B. E. and Miller, D. W., *Computer Vision-Based Localization and Mapping of an Unkown, Uncooperative and Spinning Target for Spacecraft Proximity Operations*, Dissertation, Massachusetts Institute of Technology, 2013.

[6]Fasano, G., Grassi, M., and Accardo, D., "A stereo-vision Based System for Autonomous Navigation of an In-Orbit Servicing Platform," *AIAA Infotech@ Aerospace*, No. April, Seattle, Washington, 2009, pp. 1–10.

[7]Philip, N. K. and Ananthasayanam, M. R., "Relative position and attitude estimation and control schemes for the ÿnal phase of an autonomous docking mission of spacecraft," *Acta Astronautica*, Vol. 52, 2003, pp. 511–522.

[8]Kelly, S. J., *A Monocular SLAM method to Estimate Relative Pose During Satellite Proximity Operations*, Thesis, Air Force Institute of Technology, 2015.

[9]Howard, R. T., Heaton, A. F., Pinson, R. M., and Carrington, C. K., "Orbital Express Advanced Video Guidance Sensor," *IEEE Aerospace Conference Proceedings*, 2008, pp. 1–10.

[10]Yu, F., He, Z., Qiao, B., and Yu, X., "Stereo-vision-based relative pose estimation for the rendezvous and docking of noncooperative satellites," *Mathematical Problems in Engineering*, Vol. 2014, 2014, pp. 12.

[11]Ghadiok, V., Goldin, J., and Geller, D., "Gyro-Aided Vision-Based Relative Pose Estimation for Autonomous Rendezvous and Docking," *Advances in the Astronautical Sciences*, Vol. 149, 2013, pp. 713–728.

[12]Hess, J. A., Swenson, E. D., Leve, F. A., Black, J., and Goff, G. M., "Adaptive Estimation of Nonlinear Spacecraft Attitude Dynamics with Time-Varying Moments of Inertia Using On-Board Sensors," *AIAA Guidance, Navigation, and Control Conference*, No. January, 2016, pp. 1–18.

[13]Zappella, L., Salvi, J., and Llado, X., "New trends in motion segmentation," *Pattern Recognition*, chap. 3, INTECH Open Access Publisher, Girona, Spain, 2009, pp. 31–46.

[14]Rao, S., Tron, R., Vidal, R., and Ma, Y., "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 10, 2010, pp. 1832–1845.

[15]Vidal, R., Tron, R., and Hartley, R., "Multiframe motion segmentation with missing data using PowerFactorization and GPCA," *International Journal of Computer Vision*, Vol. 79, No. 1, 2008, pp. 85–105.

[16]Yan, J. and Pollefeys, M., "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," *European conference on computer vision*, Springer Berlin Heidelberg, 2006, pp. 94–106.

[17]Zappella, L., Llad, X., Provenzi, E., and Salvi, J., "Enhanced Local Subspace Affinity for feature-based motion segmentation," *Pattern Recognition*, Vol. 44, No. 2, 2011, pp. 454–470.

[18]Tomasi, C. and Kanade, T., "Shape and motion from image streams under orthography: a factorization method." *International Journal of Computer Vision*, Vol. 9, No. 2, 1992, pp. 137–154.

[19]Weinshall, D. and Tomasi, C., "Linear and Incremental Acquisition of Invariant Shape Models from Image Sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, 1995, pp. 512–517.

[20]Poelman, C. J. and Kanade, T., "A Paraperspective Factorization Method for Shape and Motion Recovery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 3, 1997, pp. 206–218.

[21]Yan, J. and Pollefeys, M., "A factorization-based approach to articulated motion recovery," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, 2005, pp. 815–821.

[22]Tresadern, P. and Reid, I., "Articulated structure from motion by factorization," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, 2005, pp. 1110–1115.

[23]Ross, D. A., Tarlow, D., and Zemel, R. S., "Learning articulated structure and motion," *International Journal of Computer Vision*, Vol. 88, No. 2, 2010, pp. 214–237.

[24]Daubney, B., Gibson, D., and Campbell, N., "Real time pose estimation of articulated objects using low-level motion," *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–8.

[25]Gruber, A. and Weiss, Y., "Multibody factorization with uncertainty and missing data using the EM algorithm," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1, IEEE, 2004, pp. 707–714.

[26]Shi, J. and Malik, J., "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, 2000, pp. 888–905.

[27]Yan, J. and Pollefeys, M., "A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 5, 2008, pp. 865–877.

[28]Szeliski, R., *Computer Vision: Algorithms and Applications*, Springer, London, 2011.

[29]Szeliski, R. and Kang, S. B., "Recovering 3D Shape and Motion from Image Streams using Non-Linear Least Squares," *Computer Vision and Pattern Recognition*, IEEE, 1993, pp. 752–753.

[30]Horn, B. K. P., Hilden, H. M., and Negahdaripour, S., "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, Vol. 5, No. 7, 1988, pp. 1127–1135.

[31]Zappella, L., Del Bue, A., Lladó, X., and Salvi, J., "Joint estimation of segmentation and structure from motion," *Computer Vision and Image Understanding*, Vol. 117, No. 2, 2013, pp. 113–129.

[32]Kjer, H. M. and Wilm, J., *Evaluation of surface registration algorithms for PET motion correction Bachelor thesis*, Ph.D. thesis, Technical University of Denmark, 2010.

[33]Wilm, J. and Kjer, H. M., "https://www.mathworks.com/matlabcentral/fileexchange/27804-iterative-closest-point," 2013.

[34]Besl, P. J. and McKay, N. D., "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, 1992, pp. 239–256.

[35]Chen, Y. and Medioni, G., "Object Modeling by Registration of Multiple Range Images," *International Conference on Robotics and Automation*, 1991, pp. 2724–2729.

Bibliography

1. XSS-11 Points To Fixing Satellites On Orbit. http://www.spacetoday.org/Satellites/XSS_11/XSS_11.html, 2006.

2. 88th ABW Public Affairs. Successful launch for AFRL Eagle spacecraft experiment on AFSPC-11 mission. http://www.schriever.af.mil/News/Article-Display/Article/1496633/successful-launch-for-afrl-eagle-spacecraft-experiment-on-afspc-11-mission/, 2018.

3. Ankur Agarwal and Bill Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):1–15, 2006.

4. Priyanshu Agarwal, Suren Kumar, Julian Ryde, Jason J. Corso, and Venkat N. Krovi. Estimating dynamics on-the-fly using monocular video for vision-based robotics. *IEEE/ASME Transactions on Mechatronics*, 19(4):1412–1423, 2014.

5. Air Force Research Laboratory (AFRL). Fact Sheet Automated Navigation and Guidance Experiment for Local Space (ANGELS). https://smallsat.org/conference/angels-fact-sheet.pdf, 2014.

6. Bruce Guenther Baumgart. Geometric Modeling for Computer Vision. Technical Report No. STAN-CS-74-463, Standford University Department of Computer Science, 1974.

7. A. M. Buchanan and A. W. Fitzgibbon. Damped Newton Algorithms for Matrix Factorization with Missing Data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 316–322, 2005.

8. Ingo Buerk. Matlab code: SpectralClustering. https://www.mathworks.com/matlabcentral/fileexchange/34412-fast-and-efficient-spectral-clustering, 2012.

9. I-Cheng Chang and Shih-Yao Lin. 3D human motion tracking based on a progressive particle filter. *Pattern Recognition*, 43(10):3621–3635, 2010.

10. Pei Chen. Optimization algorithms on subspaces: Revisiting missing data problem in low-rank matrix. *International Journal of Computer Vision*, 80(1):125–142, 2008.

11. German K.M. Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, pages 77–84, 2003.

12. Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time part I: Theory and algorithms. *International Journal of Computer Vision*, 62(3):221–247, 2005.

13. Kong-Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time part II: Applications to human modeling and markerless motion tracking. *International Journal of Computer Vision*, 63(3):225–245, 2005.

14. Javier Civera Sancho. *Real-Time EKF-Based Structure from Motion*. Dissertation, University of Zaragoza, 2009.

15. Joao Costeira and Takeo Kanade. A Multi-Body Factorization Method for Motion Analysis. In *Proceedings Fifth International Conference on Computer Vision*, pages 1071–1076, 1995.

16. John L. Crassidis, F. Landis Markley, and Yang Cheng. Survey of Nonlinear Attitude Estimation Methods. *Journal of Guidance, Control, and Dynamics*, 30(1):12–28, 2007.

17. David Curtis and Richard Cobb. Illumination effects on satellite articulation characterization from a trajectory matrix using optimization. In *IEEE Aerospace Conference*, Big Sky MT, 2018. IEEE.

18. David H. Curtis and Richard G. Cobb. Satellite articulation characterization from an image trajectory matrix using optimization. In *Advanced Maui Optical Space Surveillance Technologies Conference*, Maui HI, 2017.

19. David H. Curtis and Richard G. Cobb. Satellite Articulation Sensing using Computer Vision. In *Proceedings of AIAA SciTech 2017*, Orlando FL, 2017.

20. David H. Curtis and Richard G. Cobb. Satellite Articulation Sensing using Computer Vision. *Submitted to: AIAA Journal of Spacecraft and Rockets*, 2018.

21. David H. Curtis and Richard G. Cobb. Satellite Articulation Tracking Using Monocular Computer Vision. In *AAS 41st Annual Guidance and Control Conference*, pages 1–13, Breckenridge, CO, 2018.

22. DARPA. Orbital Express. http://archive.darpa.mil/orbitalexpress, 2007.

23. Ben Daubney, David Gibson, and Neill Campbell. Real time pose estimation of articulated objects using low-level motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

24. Andrew J. Davison. Real-time Simultaneous Localisation and Mapping with a Single Camera. In *IEEE International Conference on Computer Vision*, volume 2, pages 1403–1410, Washington DC, 2003.

25. Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

26. Alessio Del Bue, João Xavier, Lourdes Agapito, and Marco Paladini. Bilinear modeling via augmented lagrange multipliers (BALM). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1496–1508, 2012.

27. John D'Errico. Matlab code: Inhull. https://www.mathworks.com/matlabcentral-/fileexchange/10226-inhull, 2006.

28. Jonathan Deutscher and Ian Reid. Articulated Body Motion Capture by Stochastic Search. *International Journal of Computer Vision*, 61(2):185–205, 2005.

29. Giancarmine Fasano, Michele Grassi, and Domenico Accardo. A stereo-vision Based System for Autonomous Navigation of an In-Orbit Servicing Platform. In *AIAA Infotech@ Aerospace Conference*, pages 1–10, Seattle, Washington, 2009.

30. João Fayad, Chris Russell, and Lourdes Agapito. Automated articulated structure and 3D shape recovery from point correspondences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 431–438, Barcelona, Spain, 2011.

31. Kenneth A. Fisher and Peter S. Maybeck. Multiple model adaptive estimation with filter spawning. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):755–768, 2002.

32. Angel Flores-Abad, Ou Ma, Khanh Pham, and Steve Ulrich. A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1–26, 2014.

33. Vaibhav Ghadiok, Jeremy Goldin, and David Geller. Gyro-Aided Vision-Based Relative Pose Estimation for Autonomous Rendezvous and Docking. *Advances in the Astronautical Sciences*, 149:713–728, 2013.

34. Gary M. Goff, Johnathan T. Black, and Joseph A. Beck. Orbit Estimation of a Continuously Thrusting Spacecraft Using Variable Dimension Filters. *Journal of Guidance, Control, and Dynamics*, 38(12):2407–2420, 2015.

35. S. Burak Gokturk, Hakan Yalcin, and Cyrus Bamji. A time-of-flight depth sensor - System description, issues and solutions. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2004.

36. Paulo F. U. Gotardo and Aleix M. Martinez. Computing Smooth Time-Trajectories for Camera and Deformable Shape in Structure from Motion with Occlusion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 33, pages 2051–2065, 2011.

37. Haichao Gui and Anton H. J. de Ruiter. Quaternion Invariant Extended Kalman Filtering for Spacecraft Attitude Estimation. *Journal of Guidance, Control, and Dynamics*, 41(4):863–878, 2018.

38. Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with Microsoft Kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, 2013.

39. S. Joel Hannah. A relative navigation application of ULTOR technology for automated rendezvous and docking. In *Proceedings of SPIE, Spaceborne Sensors III*, volume 6220, pages 1–12, 2006.

40. Richard Hartley and Frederik Schaffalitzky. PowerFactorization : 3D reconstruction with missing or uncertain data. In *Proceedings of the Australia-Japan Advanced Workshop on Computer Vision*, volume 74, pages 1–9, 2003.

41. Karol Hausman, Scott Niekum, Sarah Osentoski, and Gaurav S. Sukhatme. Active articulation model estimation through interactive perception. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3305–3312, 2015.

42. Joshuah A. Hess, Eric D. Swenson, Frederick A. Leve, Jonathan Black, and Gary M. Goff. Adaptive Estimation of Nonlinear Spacecraft Attitude Dynamics with Time-Varying Moments of Inertia Using On-Board Sensors. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–18, San Diego CA, 2016.

43. Chi-Chang J. Ho and N. Harris McClamroch. Automatic spacecraft docking using computer vision-based guidance and control techniques. *Journal of Guidance, Control, and Dynamics*, 16(2):281–288, 1993.

44. Richard T. Howard, Andrew F. Heaton, Robin M. Pinson, and Connie K. Carrington. Orbital Express Advanced Video Guidance Sensor. In *IEEE Aerospace Conference Proceedings*, pages 1–10, Big Sky MT, 2008.

45. Laurent Itti, Christof Koch, and Ernst Niebur. A Model of Saliency-based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

46. Vijay John, Emanuele Trucco, and Spela Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11):1530–1547, 2010.

47. Simon J. Julier and Jeffrey K. Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194, Orlando, FL, 1997. International Society for Optics and Photonics.

48. Scott J. Kelly. *A Monocular SLAM method to Estimate Relative Pose During Satellite Proximity Operations*. Thesis, Air Force Institute of Technology, 2015.

49. Suren Kumar, Vikas Dhiman, Madan Ravi Ganesh, and Jason J. Corso. Spatiotemporal Articulated Models for Dynamic SLAM. *CoRR*, 2010.

50. Abhijit Kundu. *Monocular Multibody Visual SLAM*. Thesis, International Institute of Information Technology, Hyderabad India, 2011.

51. Abhijit Kundu, K. Madhava Krishna, and C. V. Jawahar. Realtime multibody visual SLAM with a smoothly moving monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2080–2087, 2011.

52. Carl Christian Liebe, Alex Abramovici, Randy K. Bartman, Robert L. Bunker, Jacob Chapsky, Cheng Chih Chu, Daniel Clouse, James W. Dillon, Bob Hausmann, Hamid Hemmati, Richard P. Kornfeld, Clint Kwa, Sohrab Mobasser, Michael Newell, Curtis Padgett, W. Thomas Roberts, Gary Spiers, Zachary Warfield, and Malcolm Wright. Laser radar for spacecraft guidance applications. *IEEE Aerospace Conference Proceedings*, 6:2647–2662, 2003.

53. Guangcan Liu, Zhouchen Lin, and Yong Yu. Robust Subspace Segmentation by Low-Rank Representation. In *27th International Conference on Machine Learning (ICML-10)*, pages 663–670, Haifa, Israel, 2010.

54. Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 674–679, 1981.

55. Ulrike Von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4):395–416, 2006.

56. F. Landis Markley. Attitude Error Representations for Kalman Filtering. *Journal of Guidance, Control, and Dynamics*, 26(2):311–317, 2003.

57. Manuel Marques and João Costeira. Optimal shape from motion estimation with missing and degenerate data. In *2008 IEEE Workshop on Motion and Video Computing, WMVC*, pages 2–7, 2008.

58. Roberto Martin Martin and Oliver Brock. Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2494–2501, Chicago IL, 2014.

59. Peter S. Maybeck. *Stochastic Models, Estimation and Control: Volume 1*. Academic Press, New York, 1979.

60. Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

61. Michael Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Disertation, Carnegie Mellon University, 2003.

62. Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, volume 68, pages 593–598, 2002.

63. Joseph L. Mundy and Andrew Zisserman. *Geometric Invariance in Computer Vision*. The MIT Press, 1992.

64. Bo Naasz, Doug Zimpfer, Ray Barrington, and Tom Mulder. Flight Dynamics and GN&C for Spacecraft Servicing Missions. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–16, Toronto, Ontario Canada, 2010.

65. Bo J. Naasz, John Van Eepoel, Steven Z. Queen, C. Michael Southward, and Joel Hannah. Flight results from the HST SM4 Relative Navigation Sensor system. In *33rd Annual AAS Guidance and Control Conference*, Breckenridge, Colorado, 2010.

66. NASA. Extending the Reach of the Space Station. Technical report, NASA, 2001.

67. Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an Algorithm. *Advances in Neural Information Processing Systems*, pages 849–856, 2002.

68. Kemal Egemen Ozden, Konrad Schindler, and Luc Van Gool. Multibody Structure-from-Motion in Practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1134–1141, 2010.

69. Marco Paladini, Alessio Del Bue, João Xavier, Lourdes Agapito, Marko Stošić, and Marija Dodig. Optimal metric projections for deformable and articulated structure-from-motion. *International Journal of Computer Vision*, 96(2):252–276, 2012.

70. Xavier Perez-Sala, Sergio Escalera, Cecilio Angulo, and Jordi Gonzàlez. A survey on model based approaches for 2D and 3D visual human pose recovery. *Sensors*, 14(3):4189–4210, 2014.

71. N. K. Philip and M. R. Ananthasayanam. Relative position and attitude estimation and control schemes for the final phase of an autonomous docking mission of spacecraft. *Acta Astronautica*, 52:511–522, 2003.

72. Sudeep Pillai, Matthew R. Walter, and Seth Teller. Learning Articulated Motions From Visual Demonstration. In *Robotics: Science and Systems*, 2014.

73. Bryan Poling. A Tutorial On Camera Models. Technical report, University of Minnesota, 2012.

74. Shankar Rao, Roberto Tron, Rene Vidal, and Yi Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2010.

75. Shankar R. Rao, Allen Y. Yang, S. Shankar Sastry, and Yi Ma. Robust algebraic segmentation of mixed rigid-body and planar motions from two views. *International Journal of Computer Vision*, 88(3):425–446, 2010.

76. David A. Ross, Daniel Tarlow, and Richard S. Zemel. Learning articulated structure and motion. *International Journal of Computer Vision*, 88(2):214–237, 2010.

77. Chris Russell, Rui Yu, and Lourdes Agapito. Video Pop-up: Monocular 3D Reconstruction of Dynamic Scenes. In *European Conference on Computer Vision*, pages 583–598. Springer, 2014.

78. Chris Sabol, Rich Burns, and Craig A. McLaughlin. Satellite formation flying design and evolution. *Journal of Spacecraft and Rockets*, 38(2):270–278, 2001.

79. Sanjay Saini, Dayang Rohaya Bt Awang Rambli, Suziah Bt Sulaiman, M Nordin B Zakaria, and Azfar B Tomi. Particle swarm optimization based articulated human pose tracking using enhanced silhouette extraction. In *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*, volume 9443, page 944306. International Society for Optics and Photonics, 2015.

80. Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

81. Leonid Sigal, Alexandru O. Balan, and Michael J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1-2):4–27, 2010.

82. Cristian Sminchisescu and Bill Triggs. Kinematic Jump Processes For Monocular 3D Human Tracking. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages 69–76, 2003.

83. Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, London, 2011.

84. Richard Szeliski and Sing Bing Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.

85. Bugra Tekin, Isinsu Katircioglu, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Structured Prediction of 3D Human Pose with Deep Neural Networks. In *British Machine Vision Conference*, pages 1–11, York, UK, 2016.

86. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, 2006.

87. Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Technical Report April, Carnegie Mellon University, 1991.

88. Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method. Technical report, Carnegie Mellon University, 1991.

89. Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

90. P. Tresadern and I. Reid. Articulated structure from motion by factorization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1110–1115, 2005.

91. Brent E. Tweddle and Alvar Saenz-Otero. Relative Computer Vision-Based Navigation for Small Inspection Spacecraft. *Journal of Guidance, Control, and Dynamics*, 38(5):969–978, 2015.

92. Brent Edward Tweddle. *Computer Vision-Based Localization and Mapping of an Unkown, Uncooperative and Spinning Target for Spacecraft Proximity Operations*. Dissertation, Massachusetts Institute of Technology, 2013.

93. Rene Vidal. Subspace Clustering. *IEEE Signal Processing Magazine*, (March):52–68, 2011.

94. René Vidal, Roberto Tron, and Richard Hartley. Multiframe motion segmentation with missing data using PowerFactorization and GPCA. *International Journal of Computer Vision*, 79(1):85–105, 2008.

95. Chieh-chih Wang, Tzu-chien Lo, and Shao-wen Yang. Interacting Object Tracking in Crowded Urban Areas. In *IEEE International Conference on Robotics and Automation*, pages 10–14, Rome, 2007. IEEE.

96. Chieh-chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-whyte. Simultaneous Localization , Mapping and Moving Object Tracking. *The International Journal of Robotics Research*, 26:889–916, 2007.

97. Somkiat Wangsiripitak and David W Murray. Avoiding moving outliers in visual slam by tracking moving objects. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 375–380, Kobe, Japan, 2009.

98. Jingyu Yan and Marc Pollefeys. A factorization-based approach to articulated motion recovery. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 815–821, 2005.

99. Jingyu Yan and Marc Pollefeys. Articulated Motion Segmentation Using RANSAC With Priors. In *ICCV Workshop on Dynamical Vision*, pages 75–85, 2005.

100. Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision*, pages 94–106. Springer, Berlin, Heidelberg, 2006.

101. Jingyu Yan and Marc Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):865–877, 2008.

102. Feng Yu, Zhen He, Bing Qiao, and Xiaoting Yu. Stereo-vision-based relative pose estimation for the rendezvous and docking of noncooperative satellites. *Mathematical Problems in Engineering*, 2014, 2014.

103. Kaan Yucer, Oliver Wang, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. Reconstruction of Articulated Objects from a Moving Camera. In *IEEE International Conference on Computer Vision Workshops*, pages 28–36, 2015.

104. L. Zappella, X. Llad, E. Provenzi, and J. Salvi. Enhanced Local Subspace Affinity for feature-based motion segmentation. *Pattern Recognition*, 44(2):454–470, 2011.

105. L Zappella, J Salvi, and X Llado. New trends in motion segmentation. In *Pattern Recognition*, chapter 3, pages 31–46. INTECH Open Access Publisher, Girona, Spain, 2009.

106. Peter Boyi Zhang and Yeung Sam Hung. Articulated Structure from Motion through Ellipsoid Fitting. In *International Conference of Image Processing, Computer Vision and Pattern Recognition*, pages 179–186, 2015.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
07–06–2018	Doctoral Dissertation	Sept 2015 — Aug 2018

4. TITLE AND SUBTITLE

Satellite Articulation Sensing using Computer Vision

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Curtis, David H., Maj.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENY-DS-18-S-060

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

N/A

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Autonomous on-orbit satellite servicing benefits from an inspector satellite that can gain as much information as possible about the primary satellite. This includes performance of articulated objects such as solar arrays, antennas, and sensors. A method for building an articulated model from monocular imagery using tracked feature points and the known relative inspection route is developed. Two methods are also developed for tracking the articulation of a satellite in real-time given an articulated model using both tracked feature points and image silhouettes. Performance is evaluated for multiple inspection routes and the effect of inspection route noise is assessed. Additionally, a satellite model is built and used to collect stop-motion images simulating articulated motion over an inspection route under simulated space illumination. The images are used in the silhouette articulation tracking method and successful tracking is demonstrated qualitatively. Finally, a human pose tracking algorithm is modified for tracking the satellite articulation demonstrating the applicability of human tracking methods to satellite articulation tracking methods when an articulated model is available.

15. SUBJECT TERMS

Computer vision, rendezvous and proximity operations (RPO), space situational awareness, satellite inspection, articulation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Richard G. Cobb, AFIT/ENY
U	U	U	U	222	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4559; richard.cobb@afit.edu