MULTI-CAMERA UNCALIBRATED VISUAL SERVOING

A Thesis
Presented to
The Academic Faculty

by

Matthew Marshall

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mechanical Engineering

Georgia Institute of Technology
August 2013

MULTI-CAMERA UNCALIBRATED VISUAL SERVOING

Dedicated to the memory of Floreine Langston, who valued education.

ACKNOWLEDGEMENTS

Thanks to the members of my reading committee for their assistance, especially Dr. Sadegh for his early contribution. Thanks to Dr. Lipkin for constantly being willing to talk about all things engineering and for his patience. If this document is at all concise then the credit goes to him. Also, his refusal to accept partial understanding is an inspiration. Thanks to my wife, Laura, for all her help; we learned a lot about ourselves. Financial and technical support came from Georgia Tech Research Institute, Food Processing Technology Division. Thanks to G. V. McMurray, FPTD division chief, for his steadfastness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

ALS	autocovariance least-squares; an adaptive filtering technique.
DAKF	decentralized adaptive Kalman filter.
DKF	decentralized Kalman filter; a sensor-fusion architecture.
EKF	extended Kalman filter; used for non-linear systems.
EWRLS	exponentially weighted recursive least squares; means of implementing a sliding window for recursive least squares.
GN	Gauss-Newton.
IBVS	image-based visual servoing.
KF	Kalman filter.
PBVS	position-based visual servoing.
RLS	recursive least squares.
TCP	tool center point; the origin of the robot tool coordinate system.
UVS	uncalibrated visual servoing.
VS	visual servoing.
α	camera inclusion threshold for centralized sensor-fusion architecture with Gauss-Newton.
β	scalar multiple for the process noise covariance estimate; $Q = \beta I$.
$\boldsymbol{\theta}$	robot joint coordinates.
$\boldsymbol{f}$	image-plane error; $\boldsymbol{f} \equiv \boldsymbol{y} - \boldsymbol{y}^*$.
$\boldsymbol{u}$	system input vector.
$\boldsymbol{v}$	measurement equation error.
$\boldsymbol{y}$	image-plane coordinates of the robot end-effector.
$\boldsymbol{z}$	measurement vector.
$\boldsymbol{\nu}_k$	innovation at time k; $\boldsymbol{\nu}_k \equiv \boldsymbol{z}_k - H_k\hat{\boldsymbol{x}}_{k\|k-1}$.
$\boldsymbol{\phi}$	joint-space vector between the current robot position and the one at which the image-plane error $\boldsymbol{f}$ is minimized; $\boldsymbol{\phi} \equiv \boldsymbol{\theta} - \boldsymbol{\theta}^*$.
$\boldsymbol{\theta}^*$	the robot joint coordinates at which the image-plane error $\boldsymbol{f}$ is minimized.
$\boldsymbol{h}_\omega$	change in robot joint velocity from earlier time to current.

$\boldsymbol{h}_\theta$	change in robot position from earlier time to current.
$\boldsymbol{w}$	process equation error.
$\boldsymbol{y}^*$	desired image-plane coordinates of the robot end-effector.
$\hat{\boldsymbol{x}}$	state estimate.
κ	scalar multiple for the initial measurement noise covariance estimate; $R_0 = \kappa I$.
λ	forgetting factor.
μ	maximum allowable norm for a joint offset command to the robot.
Φ	transition matrix.
$\widehat{C}_{\nu,k}$	the estimated innovation covariance at time k.
$\widehat{R}$	adaptive measurement noise covariance matrix.
B	process equation input matrix.
C	total number of cameras in a system.
C_ν	expected innovation covariance.
$E^{(i)}$	local variance error information for the i-th filter.
$e^{(i)}$	local state error information for the i-th filter.
F	process model.
H	measurement model.
I	the identity matrix.
J	a linearization of the relationship between joint displacements and image-plane displacements; $J \equiv \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}$.
K	the Kalman gain.
k	time increment.
N	sample size for adaptive filtering.
P	error covariance matrix.
Q	process noise covariance.
R	measurement noise covariance matrix.
W	weighting matrix for objective function.

SUMMARY

This thesis develops and analyzes a Kalman filter based decentralized adaptive controller for uncalibrated visual servoing that exploits the benefits of multiple cameras. Visual servoing uses image data for closed-loop robotic control. This can be performed in a calibrated paradigm where camera parameters are used to estimate the relationship between robot joint motions and changes in image-plane coordinates or this relationship can be estimated numerically without camera parameters in what is known as *uncalibrated visual servoing.* With good calibration, the former can yield higher precision. However, it is known that multiple cameras increase accuracy in uncalibrated visual servoing [41, 43, 68]. Benefits of multi-sensor systems can include system survivability, improved state estimates, and the ability to substitute inexpensive sensors for an expensive one. Multi-camera uncalibrated visual servoing that can survive camera occlusion or failure is reported in the literature [37] but it does not provide a methodical camera weighting ability. The adaptive technique employed in this thesis is shown to yield lower average error and smaller outliers than the existing method.

The Kalman filter estimates the direction in joint space for the robot to move to minimize image-plane error in what is shown to be a completely observable system. An adaptive filtering technique is chosen, modified to ensure positive definite covariance estimates, and applied to the visual servoing controller. It automatically adjusts the covariance matrix associated with each camera to provide data weighting, which reduces system sensitivity to noise and poor camera placement. The controller is placed in a decentralized architecture that parallelizes computation and adds robustness to camera occlusion.

Visual servoing simulations and experiments are performed for a six-axis robot manipulator with both moving and static targets. A non-adaptive version of the controller, if tuned properly, yields equivalent performance to Gauss-Newton for low-noise scenarios and improves performance in the presence of noise. The adaptive version makes tuning simpler and further reduces error. Simulation and experimental results include the following points:

- For moving targets the tracking error, averaged over many trials, is reduced by 15% using the decentralized adaptive Kalman filter (DAKF) controller compared to Gauss-Newton (GN). DAKF also reduces outliers. For example, the worst DAKF performance has only 47% of the tracking error for the worst GN trial.
- More pronounced improvement is noted for static targets. Over 100 trials, DAKF reduces the GN average convergence time by 43%. Also, the worst DAKF convergence time is less than one third of the worst GN convergence time.
- Discrepancies between experimental and simulation results for different scenarios range between 2% and 19%, which is acceptably low considering the variations in camera placement between them.

The thesis makes the following contributions:

- The first experimental data for uncalibrated visual servoing simultaneously using more than two cameras
- A visual servoing control method that performs as well as Gauss-Newton under ideal imaging circumstances and reduces error and convergence time in the presence of noise
- An automatic camera weighting method that requires neither robot nor camera parameters and is shown to further improve servoing performance

CHAPTER I

INTRODUCTION

1.1 Overview

This thesis introduces a control method for image-based robot guidance that exploits the advantages of multiple cameras. It provides system survivability in the event of image occlusion or camera failure and produces a control action based on statistically meaningful data weighting without any prior knowledge of robot or camera parameters. The research presents a novel control law that uses a Kalman filter and can track a moving target. Adaptive filtering improves filter performance and it is decentralized for robustness and distributed processing. System stability is shown and the control method is supported experimentally and by simulation.

1.2 Motivation

Robots are employed in a variety of modern applications for tasks requiring repetitive high precision motions. This typically requires foreknowledge of the environment in order to program the robot controller. For example, a conventional robotic system for welding an automobile chassis only functions properly if each new car is brought to the same position relative to the robot. This requirement is a major limitation to expanding robot usage in unstructured environments (those for which no knowledge can be assumed regarding the relative position between the robot and the object to be manipulated).

Integrating visual sensing is one way to enhance robot capabilities. In the welding example, if machine vision identifies the weld locations on the car frame then this information can be used to guide the welding tool. When done in a closed-loop form this control is referred to as visual servoing (VS). With recent increases in digital camera resolution and sensitivity and higher processing speed, VS offers ever-more potential for the placement of robots in unstructured environments. Applications include agricultural tasks, food processing, explosive ordinance disposal, and disaster-zone inspections.

In VS systems cameras can be either mounted on the robot end-effector (eye-in-hand) or at a distance such that the end-effector of the robot is in the field of view (eye-to-hand). The camera data is used to track either the robot end-effector, a target object for the robot to follow, or both by identifying a set of features in the images. The aim of VS is to actuate the robot so that the features $\boldsymbol{y}$ from the vision system match a desired set $\boldsymbol{y}^*$. In other words, the robot joint displacements $\boldsymbol{\theta}$ are controlled to minimize the error

$$\boldsymbol{f}(\boldsymbol{\theta}, t) \equiv \boldsymbol{y}(\boldsymbol{\theta}, t) - \boldsymbol{y}^*(\boldsymbol{\theta}, t) \tag{1}$$

There are two main components in a visual-servoing system, Jacobian estimation and the control law. The Jacobian relates robot velocity in joint space to the velocity of the end-effector in the image space. Once this relationship is estimated the control law determines an action that best reduces $\boldsymbol{f}$. Most VS literature focuses on the first component, Jacobian estimation. This research deals with the control law formulation.

VS has been an active field of study for decades and it is commonly thought of as having two branches: image-based and position-based. Image-based visual servoing (IBVS) forms $\boldsymbol{y}$ based only on data available at the camera level, coordinates in the image space. In position-based visual servoing (PBVS) $\boldsymbol{y}$ are Cartesian coordinates with respect to a global frame, thus requiring knowledge of the camera position. There are advantages and disadvantages to each method.

The primary benefit of PBVS is that it can provide optimized trajectories in Euclidean space. On the other hand, PBVS requires precise camera calibration. Here calibration here refers to both the extrinsic (location and orientation) and intrinsic (for example, focal length) camera parameters. A drawback to IBVS is that the robot trajectory in Euclidean space can be unpredictable. A meritorious aspect of IBVS is that it is more accurate in the presence of imprecisely estimated system parameters [12].

A different VS taxonomy is calibrated or uncalibrated. Calibration refers to kinematic and optical parameters, for example link lengths of the robot, location of the camera, or pitch of the sensor (pixels per mm). (By necessity then PBVS is calibrated visual servoing.) In a calibrated system these parameters are utilized to generate the Jacobian estimate and the

Table 1: Known and unknown parameters in uncalibrated visual servoing

Known	Unknown
Robot degrees of freedom	Robot kinematic parameters
Robot joint angles	Camera locations
Number of cameras	Camera intrinsic parameters (for example, focal length or sensor pitch)
	Target geometry

robot commands. Uncalibrated visual servoing (UVS) is able to control the robot without knowledge of these parameters. In UVS the effect of robot joint motions on image-plane coordinates is estimated through measurements of image feature coordinates and robot joint positions, rather than by analytical means. The known and unknown parameters for an uncalibrated system are listed in Table 1.

Uncalibrated visual servoing is well-suited to unstructured environments because determining robot and camera parameters can be difficult therein. Rough treatment (for example, of an explosive ordnace disposal robot) can degrade calibration accuracy. Alternatively, consider a mobile robot carrying a camera over irregular terrain. If this camera provides image data for servoing of another robot then accurate extrinsic camera parameters are not reasonable to expect.

In an unstructured environment it might be desirable to have a VS system that is generically applicable. With an uncalibrated approach the same system can in one instance guide a robot having prismatic joints and in another an anthropomorphic arm (or even a wheeled robot). In this way the same controller can be used for different robots that would fit a scenario better, without retooling between tasks.

When precise calibration is available, calibrated VS achieves better performance over UVS with metrics such as settling time and precision [6, 58]. However, two cameras can yield better performance than one so when uncalibrated VS is required its performance can be improved by adding a camera [41, 68, 43]. No studies have investigated the effects of more than two cameras in uncalibrated visual servoing. With the ubiquity of digital cameras there is merit in knowing if, for example, a dozen low-resolution units can provide better performance than two high-cost cameras.

In addition to performance metrics, handling the loss of feature measurements such as due to occlusion is a large concern in VS. In controlled scenarios it is possible to optimize camera placement with regards to visibility or to servo the robot to avoid occlusions. Unstructured environments, however, provide no guarantees about lines-of-sight. Multiple cameras can also hedge against occlusion by providing redundant data. Scant research has been done regarding occlusion handling or camera failure in UVS. One exception is Jägersand et al. [37] who use two cameras and track redundant features. This serves as a bulwark in case of occlusion and their approach can be extended to handling camera failure. It treats all data as equally valuable though, which might not be optimal in the case of heterogeneous cameras or considering that some cameras will give more accurate image feature coordinates due to lighting differences etc. No schemes for weighting or prioritizing data from multiple sources in uncalibrated visual servoing exist in the literature.

In summary:

- There is benefit to be gained by using robots in unstructured environments.
- Uncalibrated visual servoing is well-suited to unstructured environments.
- Using two cameras in visual servoing is known to provide reduced tracking error compared to using a single camera [41, 43, 68].
- Multiple cameras make a system more robust to occlusion or camera failure.
- There are no uncalibrated visual servoing examples in the literature simultaneously employing more than two cameras.
- No methods for multi-camera UVS exist that weight data from each camera.

Thus, there are significant voids in using multiple cameras for UVS.

1.3 Contributions

This thesis makes the following specific contributions:

- Introduction of a Kalman filter based VS control method that reduces noise sensitivity and uses a covariance matching technique to weight camera data
- Controller decentralization through established means to provide system survivability with retention of algebraic equivalence to a centralized approach
- Controller verification through experiments and simulation, and comparison with a traditional control method

The research is contingent on the following assumptions:

- Within the robot dynamic limits and workspace, the joint-level servo control can attain any desired position in one time step.
- The servoing path starts with the robot in a full column rank configuration (that is, not in a kinematic singularity) and passes through another full column rank configuration at least once during servoing.

1.4 Outline

The remainder of this thesis has the following organization:

Chapter 2 A survey of visual servoing literature is presented that pertains to uncalibrated and multi-camera visual servoing. It provides general information on visual servoing and establishes the context for the research. Most existing efforts focus on Jacobian estimation and use zero-memory methods for the control law. No UVS research efforts simultaneously use more than two cameras or weight data. The Kalman filter is used in VS, but either for Jacobian estimation in an uncalibrated system or in 3-D position estimation for calibrated visual servoing.

Chapter 3 A traditional control law is described that uses the Gauss-Newton method. A discussion of the Kalman filter follows with application to visual servo control. This recursive method provides probabalistic smoothing compared to traditional methods. The Kalman filter is compared to an existing closely related recursive approach, exponentially weighted recursive least squares. It does not require Kalman filter system descriptions, which is beneficial for uncalibrated systems, so it cannot achieve the Kalman filter's optimality. Stability of the Kalman filter method is assessed and the resulting system is shown to be observable and controllable.

Chapter 4 Adaptive filtering as means for updating system noise descriptions is discussed. The suitability of four adaptive filtering categories is assessed with respect to the current research. A *covariance matching* approach is selected, which adapts the measurement noise description so that a theoretical covariance is brought into alignment with its estimated value. The result is a positive definite covariance matrix describing measurement noise.

Chapter 5 To make allowance for data interruption, the Kalman filter control method is decentralized. A traditional approach to handling data interruption is first discussed, as is the general data fusion problem. Derivation of a fusion method for a decentralized architecture is presented; the resulting estimate is shown algebraically equivalent to the centralized estimate, thus retaining optimality.

Chapter 6 Pseudo-code algorithms for decentralized adaptive Kalman filter based visual servoing are presented. Descriptions of the experimental testbed are also provided.

Chapter 7 Simulations of the experimental setup are performed to investigate the effects of multiple cameras on uncalibrated VS, to test for stability with camera failure, and to compare the new control method to GN for different camera scenarios. Both moving- and static-targets are used for servoing. Special attention is given to the effects of adaptive filtering on data weighting and the resulting control. The data show that the Kalman filter method provides meaningful camera weighting, which improves VS performance. Additional conclusions are made about the control methods.

Chapter 8 Visual servoing experimental results are presented. Performance is assessed for the Kalman filter control method and GN in the following scenarios: stationary target, moving target, single fixed camera, multiple fixed cameras with and without noise added, multiple fixed cameras with episodic data interruption. Both control methods are shown to survive the event of camera failure. Results also show that the Kalman filter controller outperforms the traditional approach, especially for a stationary target. Results are in agreement with simulation.

Chapter 8 The final chapter summarizes contributions and results. It also provides suggestions for implementation and future work.

CHAPTER II

REVIEW OF PERTINENT VISUAL SERVOING LITERATURE

2.1 Visual Servoing Overview

Visual servoing for closed-loop robot guidance was introduced by Shirai and Inoue [70] in 1973. Their aim was to desensitize the robot-and-vision system to calibration errors compared to previous, open-loop architectures. They use image data to calculate the 3-D position of both a box and a robot-mounted square prism. Using robot inverse kinematics they move the joints to decrease the error between the measured and target positions of the prism. This imaging-and-movement cycle is performed iteratively until the prism reaches its desired state (inside the box). They demonstrate insertion with 2.5 mm of clearance. Subsequently, there have been many different approaches developed to visual servoing.

Four classifications for VS were introduced by Sanderson and Weiss [66]:

1. Dynamic look-and-move
2. Direct visual servo
3. Position-based
4. Image-based

Joint-level control is the subject of items 1 and 2. In the first VS calculates the desired joint positions for the robot joint angle servo control system. For the second, desired joint torques are calculated directly by VS. Items 3 and 4 differ in how the feature coordinates are computed. For position-based the feature Cartesian coordinates are given with respect to the robot base. The work of Shirai and Inoue is dynamic look-and-move position-based visual servoing. Another example is Ficocelli and Janabi-Sharifi [24] who use an analytical camera model and an adaptive extended Kalman filter to estimate the Cartesian target coordinates for visual servoing. In image-based visual servoing the measurements are traditionally the image-plane coordinates of a set of points [12].

The work of this thesis is classified as image-based VS since position-based requires system calibration.

2.2 Various Feature Types

Alternatives to using points as features have been studied [13]. In one departure from convention, Collewett et al. [16] circumvent image processing by using the intensity value of each pixel as the feature vector. Their approach can improve VS performance by sidestepping the requirements for feature identification and tracking but requires robot and camera calibration. The use of lines instead of points is furthered by Espiau et al. [23] and Andreff et al. [3] but require calibrated systems. The need for intrinsic camera parameters (focal length, for example) is obviated by Malis et al. [49] who use lines for features. These examples are all dynamic look-and-move types of VS. In Wang et al. [78] feature lines are used for a direct visual servo system without requiring intrinsic or extrinsic parameters for an eye-in-hand camera, the extrinsic parameters being a Cartesian transformation from the camera frame to the robot tool flange.

All of these approaches use the kinematic parameters of the robot. In an approach using no robot or extrinsic camera parameters, Marshall et al. [50] employ the relative Cartesian displacements of an eye-in-hand 3-D time-of-flight camera as feature measurements. The current research uses points as features.

2.3 Estimating the Composite Jacobian (Interaction Matrix)

In addition to the four classifications listed in §2.1, the use of physical system parameters in VS formulation serves as a taxonomic scheme so a system can be classified as either calibrated or uncalibrated (see Table 1). To guide a robot based on data from a camera the relationship between joint displacements and image-plane displacements must be modeled. The relationship is non-linear and can be high dimensional. A linearization of this map is called the (composite) Jacobian matrix, J, where

$$J \equiv \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}} \tag{2}$$

where $\boldsymbol{y}$ comprises the feature image coordinates and $\boldsymbol{\theta}$ is the robot joint displacement. Formulating this matrix can be done analytically [14] using the system physical parameters including robot kinematics, camera location, lens focal length and radial distortion parameter, and the size of the pixels. Alternatively the mapping can be estimated numerically.

Using a numerical approach constitutes uncalibrated visual servoing, introduced by Hosoda and Asada [34]. They estimate the Jacobian through recursive least squares (RLS) techniques and are limited to fixed targets and cameras. A planar case developed by Yoshimi and Allen [81] has a robot place a peg in a hole. The control is driven by data from a camera about which neither intrinsic nor extrinsic parameters are known but a model of the robot is assumed. Jägersand [36] shows improved reliability with a Broyden update and an adaptive trust region. Using a different RLS method to estimate the Jacobian, Piepmeier et al. [62] demonstrate the ability to track a moving target with a robot-mounted (thus moving) camera. Hao et al. [31] use the same algorithm but with an adaptive tuning factor. Rather than estimating it iteratively, Sebastián et al. [68] use only the most "reliable" measurements in their estimation of the Jacobian. They rank measurements based on the magnitude of the associated movement (so as to mitigate the effect of noise) and the proximity to the current position. A different method to obviate calibration is to use neural networks [32, 10]. These systems do not require foreknowledge of the hardware parameters. The controller learns the relationship between image data and robot coordinates via an extensive teaching phase. Qian and Su [64] define state variables formed from elements of the image Jacobian matrix and employ a Kalman-Bucy filter to estimate them online. Large performance losses are noted in this type of controller as system noise increases [69].

The algorithm in Piepmeier et al. [63] is used herein for the Jacobian estimation. It has been shown that the population-based method for Jacobian estimation presented by Bonkovic et al. [8] tracks a moving target better than the recursive approach of Piepmeier, though the latter performs better for stationary targets and yields smoother robot motion [30]. A reason for eschewing the population-based method for multi-camera uncalibrated visual servoing is that population length increases with the number of images and this leads to computational requirements comparable to the cycle time of the robotic system.

Once a map between robot motion and image motion is obtained, servoing requires the implementation of a control law. The majority of the work in VS centers on formulation of the Jacobian matrix and this is especially true for uncalibrated VS. The work herein uses the Jacobian estimation algorithm of Piepmeier et al. [63] and introduces a novel control law in Chapter 3 based on Kalman filtering.

2.4 Control Laws

The control law determines robot commands based on feedback and the state of the system. In visual servoing the state pertains to the positions (possibly also velocities and accelerations) of the robot and target as measured by the robot joint encoders and by machine vision. In a direct visual servo system commands are in the form of joint torques/forces with Liu et al. [46] and Wang et al. [78] as examples. The more typical dynamic look-and-move paradigm commands joint positions (or velocities) and there are many different such control laws in the literature.

The most common control law is the Gauss-Newton algorithm detailed in Chapter 3. Alternatives exist to achieve various ends. For example, classical control laws (see §3.1.1) fail with the feature types (image pixel intensity values) used by Collewet et al. [16], so the authors detail a new control law that minimizes a cost function using an approximation of its second derivative and a tuning parameter.

Some unique control laws that are applicable only to calibrated systems are included here for context. Dame and Marchand [18] servo a robot to maximize the mutual information between the current and goal image. This alignment is robust to occlusions and variations of illumination. Alternatively, Malis [48] introduces a second-order approximation for a quasi-Newton method (see Chapter 3) that uses only first-order terms, but it cannot be applied to uncalibrated VS since one of those terms is the Jacobian at the goal pose, requiring a model.

2.5 Occlusion Handling

Loss of sensor data is an important issue to consider for visual servoing because a feature can easily leave the field of view or have its line of sight to the camera blocked (occlusion). Triggs and Laugier [76] present a method for predetermining optimal camera position based on several factors, such as visibility. This is not feasible for unstructured environments, where foreknowledge of the situation is not guaranteed.

One approach to dealing with the specter of occlusions is to avoid conditions that lead to them. In a multi-robot work cell Lippiello et al. [43] predict self and mutual occlusions using CAD models of the robots and objects in conjunction with an extended Kalman filter. Use of such models is inconsistent with uncalibrated VS. Another approach to occlusion avoidance in calibrated visual servoing is Cowan et al. [17] using navigation functions to provide almost globally guaranteed convergence to a goal image while maintaining line of sight of all the feature points.

Another approach is to utilize redundant information, though means of handling extra data vary widely. As mentioned above, Collewet et al. [16] use the intensity of every pixel in an image as features and for loss of visibility minimize a cost function based on the un-occluded pixels. The Jacobian used in this approach is calculated analytically. Yoshihata et al. [80] also use an analytically-formed Jacobian matrix and redundant feature data. They track the image-plane position of four points on a miniature helicopter though only three are needed to determine its position and orientation. The system allows occlusion of only one point at a time. Early work by Jägersand et al. [37] discusses a method for dealing with occlusions in uncalibrated VS. The image feature vector $\boldsymbol{y}$ (and also the target feature vector $\boldsymbol{y}^*$) contains redundant information so that if one (or more) features becomes occluded that component of $\boldsymbol{y}$ is removed, as are the corresponding rows of the Jacobian. This technique will serve as a benchmark against which to compare the Kalman filter approach presented in Chapters 3, 4, and 5.

2.6 Multiple Cameras in VS

There are reasons beyond occlusion handling to use multiple cameras in visual servoing. Gao and Su [27] guide a wheeled robot through an area using three fixed cameras where the robot passes from one camera view to another. It is a kinematically uncalibrated system (only the camera intrinsic parameters are required). They develop a switching method that passes control off from one camera to the next as the robot moves through the different fields of view by creating a weighted combination of the control actions generated from each camera with the weights being dynamic. Gao and Su identify this as “control-level fusion” and make the claim that its counterpart, “sensor-level fusion,” is not possible for uncalibrated visual servoing but the centralized-architecture examples in Chapter 5 perform this type of fusion. In Kühnlenz and Buss [40] an optimal camera (as determined by performance- and task-related criteria) is selected from an available set, though in practice it was only applied to an eye-in-hand pair.

Obtaining a higher control frequency is the aim of Schuurman and Capson [67] in their application of multiple cameras to visual servoing. They interleave frames from multiple cameras to achieve a sampling rate high enough for direct control (as opposed to dynamic look-and-move).

Use of two cameras can improve tracking performance compared to systems with just one camera, as demonstrated by Lamiroy et al. [41], Lippiello et al. [43], and Sebastián et al. [68]. This thesis explores how far improvements extend with additional cameras.

A stereo rig is a common device used in two-camera visual servoing, as in the calibrated eye-in-hand case of Lamiroy et al. [41]. Sebastián et al. [69] use a pair of fixed cameras in an uncalibrated scheme. They reduce sensitivity to noise by using the epipolar geometry when estimating the Jacobian. Their approach relies on the ability to generate point correspondences between images.

Another common camera pair is the “eye-in-hand/eye-to-hand” arrangement, which is intended to take advantage of eye-in-hand camera maneuverability and high precision data and the wide eye-to-hand camera field of view. Usually their data are used for separate tasks rather than fusing them. For example, Flandin et al. [25] employ data from the eye-to-hand camera to govern the translational motions of the robot end effector and the eye-in-hand camera is used for orientation. Zhang et al. [82] safeguarded visibility with the eye-to-hand camera and precision is afforded by the eye-in-hand camera. By default servoing is done with the eye-in-hand camera unless the target is occluded, then servoing is switched to the eye-to-hand camera. There are also studies in (position based) visual servoing that use data from the two cameras simultaneously.

Data are completely integrated in an eye-in-hand/eye-to-hand scenario by Lippiello et al. [44]. They use the extended Kalman filter (EKF) to create a pose estimate from both cameras for position based (and thus calibrated) visual servoing. Assa et al. [4] also use the EKF to estimate pose in a calibrated eye-in-hand/eye-to-hand scenario but they fuse estimates from multiple EKF's via ordered weighted averaging aggregation operators.

Stavnitzky and Capson [71] use a simple weighted linear combination of pose vectors from two stationary cameras using a model-based system that is quasi-uncalibrated since the robot kinematics are not required. These are all instances of sensor-level fusion of camera data for visual servoing requiring calibration.

2.7 Summary

In summary, various feature types can be used in VS though all existing uncalibrated systems track points. The Jacobian matrix can be numerically estimated using many different techniques and a well-established algorithm is selected for this research. Few control law options are reported for UVS. Conversely, there exist numerous control laws for calibrated visual servoing. In the existing literature weighted sensor-level fusion of vision data is only performed for calibrated systems. It has been stated that this is not possible for uncalibrated visual servoing [27] but this thesis demonstrates otherwise. Much work has been done in VS with two cameras. No studies simultaneously employ more than two.

CHAPTER III

A KALMAN FILTER BASED CONTROL LAW

This chapter presents a visual servoing control law based on the celebrated Kalman filter, which estimates a changing state based on a process model and noisy measurements. In the current context the filter estimates the joint-space distance between the robot and the target. Given accurate statistical process and measurement error models the filter is known to provide optimal estimates, minimizing the expected estimate error. Using the Kalman filter also facilitates an adaptive decentralized system that automatically weights camera data and is robust to data interruption. The Kalman filter's recursive nature provides control less sensitive to noise than traditional VS methods.

For comparison, the most common approach to designing a VS control law is discussed. It uses a version of Newton's root finding method to minimize the error $\boldsymbol{f}$ between robot and target image-plane coordinates. The Gauss-Newton variation is favored in practice because the difficult Hessian matrix ($H = \frac{\partial^2 \boldsymbol{y}}{\partial^2 \boldsymbol{\theta}} = \frac{\partial J}{\partial \boldsymbol{\theta}}$) computation is not required.

3.1 Gauss-Newton Control Law

3.1.1 Newton's method

Newton's method is an iterative approach to solving the scalar equation $g(x) = 0$ and is often applied to VS. Making an affine approximation of g at the initial guess x_0 gives

$$g(x) \approx m_0(x) = g(x_0) + g_0'(x - x_0)$$

where $g_0' \equiv g'(x_0)$. An improved estimate solves the affine approximation $m_0(x) = 0$, as

$$g(x_0) + g_0'(x_1 - x_0) = 0$$

to yield the new estimate

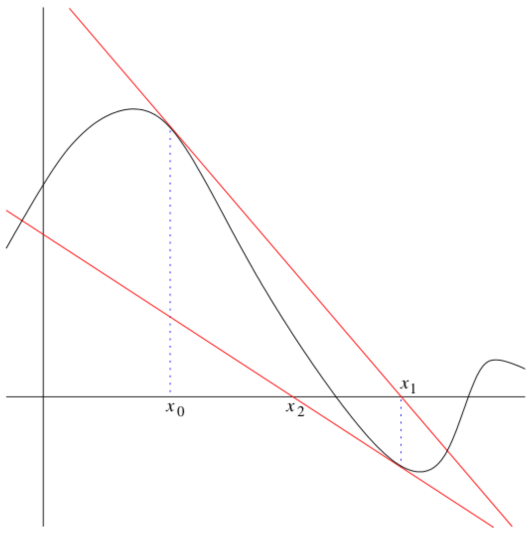


Figure 1: Two iterations of Newton's method

$$x_1 = x_0 - \frac{g(x_0)}{g'_0}$$

as the intersection of the x-axis and the tangent to $g(x)$ at x_0, see Figure 1.

To reach an acceptably accurate solution, the iteration continues until the difference between successive estimates is within some predetermined threshold value. The estimates are calculated as

$$\begin{aligned} x_2 &= x_1 - \frac{g(x_1)}{g'_1} \\ &\vdots \\ x_{k+1} &= x_k - \frac{g(x_k)}{g'_k} \end{aligned}$$

3.1.2 Newton's method in visual servoing

Visual servoing systems are often over constrained where the number of image features is greater than the robot degrees of freedom, and so in general (due to camera noise) there is no robot position that makes the error zero. Therefore, the computation of the desired joint positions is routinely treated as a minimization problem, usually of the error-squared objective function G,

$$G(\boldsymbol{\theta}, t) = \boldsymbol{f}^{\mathsf{T}}(\boldsymbol{\theta}, t) W \boldsymbol{f}(\boldsymbol{\theta}, t) \tag{3}$$

where the weighting matrix W typically is the identity matrix I and $\boldsymbol{f} \equiv \boldsymbol{y} - \boldsymbol{y}^*$ is the image-plane error defined in (1), $\boldsymbol{y}$ is the robot image features and $\boldsymbol{y}^*$ is the target image features.

The robot position that solves the minimization problem at a given time t_k is denoted $\boldsymbol{\theta}^*$,

$$G(\boldsymbol{\theta}^*, t_k) = \min G(\boldsymbol{\theta}, t_k) \tag{4}$$

Newton's method is useful in minimization since local minima (or maxima) of a function can be obtained by solving

$$g'(x) = 0$$

To find $\boldsymbol{\theta}^*$ using Newton's method for an unweighted system first represent the objective function (3) by its Taylor series about $\boldsymbol{\theta}$ and t,

$$G(\boldsymbol{\theta} + \boldsymbol{h}_\theta, t + h_t) = G(\boldsymbol{\theta}, t) + G_\theta \boldsymbol{h}_\theta + G_t h_t + \ldots$$

where G_θ and G_t are partial derivatives and $\boldsymbol{h}_\theta$ and h_t are small steps. Holding t fixed, G is minimized over $\boldsymbol{\theta}$ by solving

$$\frac{\partial G(\boldsymbol{\theta} + \boldsymbol{h}_\theta, t + h_t)}{\partial \boldsymbol{\theta}} = 0$$

via Newton's method,

$$\begin{aligned} G_\theta + G_{\theta\theta} \boldsymbol{h}_\theta + G_{t\theta} h_t &= 0 \\ \boldsymbol{h}_\theta &= -\left(G_{\theta\theta}\right)^{-1} \left(G_\theta + G_{t\theta} h_t\right) \end{aligned}$$

where

$$\begin{aligned}
G_\theta &= J_k^\mathsf{T} \boldsymbol{f}_k \\
G_{\theta\theta} &= J_k^\mathsf{T} J_k + \frac{\partial J_k^\mathsf{T}}{\partial \boldsymbol{\theta}} \boldsymbol{f}_k = J_k^\mathsf{T} J_k + S_k \\
G_{t\theta} &= J^\mathsf{T} \frac{\partial \boldsymbol{y}^*(t)}{\partial t} \quad \text{and} \\
J_k &= \frac{\partial \boldsymbol{y}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}
\end{aligned}$$

From Chapter 1, $\boldsymbol{y}(\boldsymbol{\theta}, t)$ and $\boldsymbol{y}^*(\boldsymbol{\theta}, t)$ are the image-plane coordinates of the robot end-effector and the target, respectively. This yields what Piepmeier et al. [60] dub the dynamic Newton's method

$$\widehat{\boldsymbol{\phi}}_k = \left(J_k^\mathsf{T} J_k + S_k\right)^{-1} \left(J_k^\mathsf{T} \boldsymbol{f}_k - J_k^\mathsf{T} \frac{\partial \boldsymbol{y}^*(t)}{\partial t} h_t\right) \tag{5}$$

where $\boldsymbol{\phi} \equiv \boldsymbol{\theta} - \boldsymbol{\theta}^*$ is the joint-space vector between the current robot position and the one that minimizes $\boldsymbol{f}^\mathsf{T} \boldsymbol{f}$, that is, the goal robot position $\boldsymbol{\theta}^*$ introduced in (4). The notation $\boldsymbol{h}_\theta$ is not used here because the distance $\boldsymbol{\theta} - \boldsymbol{\theta}^*$ can be large.

The term $\frac{\partial \boldsymbol{y}^*(t)}{\partial t} h_t$ in (5) facilitates servoing to moving targets. Leaving it out gives the more standard Newton's equation for visual servoing

$$\widehat{\boldsymbol{\phi}}_k = \left(J_k^\mathsf{T} J_k + S_k\right)^{-1} J_k^\mathsf{T} \boldsymbol{f}_k \tag{6}$$

With uncalibrated VS the Jacobian J is estimated as $\hat{J}$ by one of the methods outlined in Chapter 2. The S term in (6) and (5) is both difficult to compute and small when near the target so it is often left out, yielding the Gauss-Newton (GN) method.

$$\widehat{\boldsymbol{\phi}}_k = \left(J_k^\mathsf{T} J_k\right)^{-1} J_k^\mathsf{T} \boldsymbol{f}_k \tag{7}$$

An exception to this practice is the work of Fu et al. [26] who retain S by using a secant approximation. The use of approximations for both of these first- and second-order derivative terms qualifies their approach as *quasi-Newton.* Munnae [53] also retains the full Hessian matrix $G_{\boldsymbol{\theta}\boldsymbol{\theta}}$ and as the robot approaches its goal uses heuristic criteria to switch from an initial quasi-Newton method to a quasi-Gauss-Newton method (J is estimated using the algorithm of Piepmeier [59]).

Besides Newton and Gauss-Newton, other minimization-based control laws are described in the literature. Examples include the Jacobian Transpose control (or steepest descent) method,

$$\widehat{\boldsymbol{\phi}}_k = J_k^\mathsf{T} \boldsymbol{f}_k$$

and the Levenberg-Marquardt minimization method, which blends the Gauss-Newton and steepest descent methods and offers protection against problems with singularities,

$$\widehat{\boldsymbol{\phi}}_k = \left(J_k^\mathsf{T} J_k + \lambda D \right)^{-1} J_k^\mathsf{T} \boldsymbol{f}_k$$

where D comprises the diagonal elements of $J^\mathsf{T} J$. Also, a recursive Gauss-Newton is introduced by Gumpert [28] with details given in §3.2.6.2.

The Gauss-Newton control law is the most common in visual servoing. For example, Deng and Jägersand [19] use it to compare three different Jacobian estimation methods. It serves as the benchmark for the Kalman filter control law described below.

3.2 Kalman Filter Control Law

This section introduces a VS control law that mitigates the effects of noise and paves the way for a statistically meaningful method of sensor fusion discussed in Chapters 4 and 5. It uses the Kalman filter (KF) to efficiently estimate a changing state in the presence of noise. It provides the best linear unbiased estimate for systems with Gaussian noise. For non-Gaussian noise KF still provides the best linear estimator but a non-linear estimator might do even better. As discussed in Chapter 2, KF is used in visual servoing literature for Jacobian estimation in image-based visual servoing and 3-D pose estimation for the target in position-based VS. Here KF estimates the joint-space error $\boldsymbol{\phi} \equiv \boldsymbol{\theta} - \boldsymbol{\theta}^*$ (and possibly its velocity) where the goal robot position $\boldsymbol{\theta}^*$ is introduced in (4).

Derivations of the Kalman filter abound in technical literature (for example, [15, 72]) and are not repeated here. Instead, some of its salient features are discussed.

3.2.1 Background

The state estimate $\hat{\boldsymbol{x}}$ given by KF results from a prediction and a correction at each step in time. The prediction comes from a process equation and the correction is by means of a measurement $\boldsymbol{z}$. It is a one-step process where the new estimate only requires values from the previous step, since all the older values are built into the previous estimate. Also, the estimation is a linear function of the previous estimate and the current measurement,

$$\hat{\boldsymbol{x}}_{\text{new}} = L\hat{\boldsymbol{x}}_{\text{old}} + K\boldsymbol{z}_{\text{new}}$$

Alternatively, the estimate is based on the difference between the expected and current measurements,

$$\hat{\boldsymbol{x}}_{\text{new}} = \hat{\boldsymbol{x}}_{\text{old}} + K\left(\boldsymbol{z}_{\text{new}} - H_{\text{new}}\hat{\boldsymbol{x}}_{\text{old}}\right)$$

K is the "Kalman gain matrix" and depends on the relative certainty of the prediction and the measurement. The difference between the actual measurement and a predicted one $(\boldsymbol{z}_{\text{new}} - H_{\text{new}}\hat{\boldsymbol{x}}_{\text{old}})$, or the *innovation*, is multiplied by K to yield the correction to the last estimate.

The Kalman filter also provides a measure of the estimate's reliability in the form of the error covariance matrix,

$$P = \mathrm{E}\left[\left(\boldsymbol{x} - \hat{\boldsymbol{x}}\right)\left(\boldsymbol{x} - \hat{\boldsymbol{x}}\right)^{\mathsf{T}}\right]$$

where E denotes the expected value. This estimate is based on statistical properties of the system model, rather than any specific measurements $\boldsymbol{z}$ or state estimates $\hat{\boldsymbol{x}}$.

3.2.2 Static and dynamic recursive least squares

The KF is dynamic recursive least squares. That is, it provides estimates of a changing state $\boldsymbol{x}$ by referring to its previous estimate of the state. Before dealing with KF further, consider the non-recursive least squares approach to estimating a static $\boldsymbol{x}$

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} = \boldsymbol{x} \tag{8}$$

There are two reasons that regular (static) least squares is important to consider. First, it provides insight into KF and secondly, it enables comparison between a recursive least squares (RLS) control law [28] and the KF control law introduced here.

The goal of least squares is to estimate a static vector $\boldsymbol{x}$ through sequential noisy measurements

$$\boldsymbol{z}_i = H\boldsymbol{x} - \boldsymbol{v}_i$$

where the noise $\boldsymbol{v}_i$ is assumed to be zero-mean, Gaussian white noise with covariance R_i.

After an initial measurement the system is given by $H_0\boldsymbol{x} = \boldsymbol{z}_0 - \boldsymbol{v}_0$ and we want to estimate $\boldsymbol{x}$ as $\hat{\boldsymbol{x}}_0$. When the dimension of $\boldsymbol{z}_0$ is greater than that of $\boldsymbol{x}$ and H_0 has full rank then there is either no solution or there is one solution. The latter scenario is for an exact measurement ($\boldsymbol{v}_0 = \boldsymbol{0}$) so $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}$. When noise is present $\hat{\boldsymbol{x}}_0$ is calculated as a least squares solution weighted by the inverse of the covariance matrix R_0,

$$\hat{\boldsymbol{x}}_0 = \left(H_0^\mathsf{T} R_0^{-1} H_0\right)^{-1} H_0^\mathsf{T} R_0^{-1} \boldsymbol{z}_0$$

When a new measurement $\boldsymbol{z}_1$ arrives

$$\begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \end{bmatrix} - \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \end{bmatrix}$$

the new estimate $\hat{\boldsymbol{x}}_1$ reflects both $\boldsymbol{z}_0$ and $\boldsymbol{z}_1$ through the weighted least squares,

$$\hat{\boldsymbol{x}}_1 = \left(\begin{bmatrix} H_0^\mathsf{T} & H_1^\mathsf{T} \end{bmatrix} \begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix}^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \right)^{-1} \begin{bmatrix} H_0^\mathsf{T} & H_1^\mathsf{T} \end{bmatrix} \begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \end{bmatrix} \tag{9}$$

At time k the complete system is

$$\begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_k \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_k \end{bmatrix} - \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_k \end{bmatrix} \tag{10}$$

or

$$\mathfrak{A}\boldsymbol{x} = \mathfrak{z} - \mathfrak{e}$$

where the calligraphic script indicates the complete system up to time k. It follows that the latest estimate of $\boldsymbol{x}$ is just

$$\hat{\boldsymbol{x}}_k = \left(\mathfrak{A}^\mathsf{T}\Sigma^{-1}\mathfrak{A}\right)^{-1}\mathfrak{A}^\mathsf{T}\Sigma^{-1}\mathfrak{z} \tag{11}$$

where Σ is block diagonal comprising $R_0 \ldots R_k$ as in (9).

Static recursive least squares is able to generate the same estimate as (11) with reduced requirements for storage and computation [9],

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k-1} + K_k\left(\boldsymbol{z}_k - H_k\hat{\boldsymbol{x}}_{k-1}\right)$$

where $K_k = P_k H_k^\mathsf{T} R_k^{-1}$ and P_k is the covariance of the estimate,

$$P_k^{-1} = P_{k-1}^{-1} + H_k^\mathsf{T} R_k^{-1} H_k$$

For a dynamic $\boldsymbol{x}$ (the case for which KF was developed) there are measurements taken at each step

$$\boldsymbol{z}_k = H_k\boldsymbol{x}_k + \boldsymbol{v}_k \tag{12}$$

but (8) no longer applies because the state is changing over time. The change in state is predicted by the following linear equation,

$$\boldsymbol{x}_k = F_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_k \tag{13}$$

The error $\boldsymbol{w}$, like $\boldsymbol{v}$, is assumed to be zero-mean Gaussian white noise, this time with covariance Q_k. They are assumed to be uncorrelated,

$$\mathrm{E}\left(\boldsymbol{w}_i \boldsymbol{v}_j^{\mathsf{T}}\right) = 0$$

for all i, j.

Up to time k, the complete system defined by (13) and (12) is

$$\begin{bmatrix} H_0 & & & & \\ -F_0 & I & & & \\ & H_1 & & & \\ & -F_1 & I & & \\ & & \ddots & & \\ & & -F_{k-1} & I \\ & & & H_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{k-1} \\ \boldsymbol{x}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_0 \\ \mathbf{0} \\ \boldsymbol{z}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{z}_k \end{bmatrix} - \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{w}_1 \\ \boldsymbol{v}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_k \\ \boldsymbol{v}_k \end{bmatrix} \tag{14}$$

or

$$\mathfrak{A}\mathfrak{x} = \mathfrak{z} - \mathfrak{e}$$

The weighted least squares solution is given by (11) using $\mathfrak{A}$ and $\mathfrak{z}$ from (14) and

$$\Sigma = \begin{bmatrix} R_0 & & & & & \\ & Q_1 & & & & \\ & & \ddots & & & \\ & & & R_{k-1} & & \\ & & & & Q_k & \\ & & & & & R_k \end{bmatrix}$$

For this case the estimate $\hat{\mathfrak{x}}$ is the best estimate for the entire history up to time k.

$$\hat{\mathfrak{x}} = \begin{bmatrix} \hat{\boldsymbol{x}}_{0|k} \\ \vdots \\ \hat{\boldsymbol{x}}_{k-1|k} \\ \hat{\boldsymbol{x}}_{k|k} \end{bmatrix}$$

The second subscript shows that the computation used all information up to and including time k. Estimates $\hat{\boldsymbol{x}}_{0|k} \ldots \hat{\boldsymbol{x}}_{k-1|k}$ are called smoothed values; estimates of past states change with subsequent measurements. With static RLS each step adds a new (block) row to $\mathfrak{A}$ in (10) but for the dynamic case a new column is also added, (see (14)).

3.2.3 Kalman equations

The Kalman equations recursively solve weighted least squares for a system with the following process and observation equations,

$$\boxed{\begin{aligned} \boldsymbol{x}_k &= F_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_k, & \boldsymbol{w}_k &\sim N(\mathbf{0}, Q_k) \quad (15) \\ \boldsymbol{z}_k &= H_k\boldsymbol{x}_k + \boldsymbol{v}_k, & \boldsymbol{v}_k &\sim N(\mathbf{0}, R_k) \quad (16) \end{aligned}}$$

where $N(\boldsymbol{\mu}, \Sigma)$ refers to Gaussian noise with expected value $\boldsymbol{\mu}$ and covariance Σ.

At time k, the Kalman filter provides an estimate $\hat{\boldsymbol{x}}_{k|k}$ and its error covariance matrix $P_{k|k}$ according to

$$\boxed{\begin{aligned} \hat{\boldsymbol{x}}_{k|k-1} &= F_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} && (17) \\ P_{k|k-1} &= F_{k-1}P_{k-1|k-1}F_{k-1}^{\mathsf{T}} + Q_k && (18) \\ K_k &= P_{k|k-1}H_k^{\mathsf{T}}\left(H_kP_{k|k-1}H_k^{\mathsf{T}} + R_k\right)^{-1} && (19) \\ \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + K_k\left(\boldsymbol{z}_k - H_k\hat{\boldsymbol{x}}_{k|k-1}\right) && (20) \\ P_{k|k} &= \left(I - K_kH_k\right)P_{k|k-1} && (21) \end{aligned}}$$

The prediction step comprises (17) and (18), and the correction step comprises (19)–(21).

3.2.4 System definition

The state being estimated is based on the difference between the current and goal robot positions $\boldsymbol{\theta} - \boldsymbol{\theta}^*$ of (4). This joint-space error generally is a function of both $\boldsymbol{\theta}$ and time.

Following sections present four different filters: zeroth order representation without input, zeroth order representation with input, first order representation without input, and first order representation with input.

The measurement of an image-based visual servoing system is the error $\boldsymbol{f} \equiv \boldsymbol{y} - \boldsymbol{y}^*$ in image-plane coordinates defined by (1). It is necessary to relate the measurements to the state in order to use the Kalman filter. The measurement model uses the following approximation to (1)

$$\boldsymbol{f}_k \equiv \boldsymbol{y}_k - \boldsymbol{y}_k^* \approx J_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_k^*) \tag{22}$$

where J_k is the composite Jacobian at time k mapping robot velocity in joint space to the velocity of the end-effector in the image space. Due to non-linear robot kinematics the approximation (22) is generally valid in the area around $\boldsymbol{\theta}_k$.

3.2.4.1 System using zeroth order state space representation without input

Here the estimated state is the joint-space error

$$\boldsymbol{x} = \boldsymbol{\phi} \equiv \boldsymbol{\theta} - \boldsymbol{\theta}^* \tag{23}$$

The process equation predicts a constant error

$$\boldsymbol{x}_k = \boldsymbol{\phi}_k = \boldsymbol{\phi}_{k-1} + \boldsymbol{w}_k \tag{24}$$

which more faithfully represents tracking of a moving target than servoing to a stationary one. The error term $\boldsymbol{w}_k$ allows for the possibility that the joint-space error is changing with time. Using approximation (22) the prediction is updated via the measurement equation

$$\boldsymbol{z}_k = \boldsymbol{f}_k = J_k\boldsymbol{\phi}_k + \boldsymbol{v}_k \tag{25}$$

Once the noise covariance matrices Q and R are determined (see §3.2.7), equations (24) and (25) enable the use of the Kalman equations (17)–(21) with $F_k = I$ and $H_k = J_k$.

3.2.4.2 Zeroth order system with input

An input $\boldsymbol{u}$ can be applied to the process equation. The canonical form is then

$$\boldsymbol{x}_k = F_{k-1}\boldsymbol{x}_{k-1} + B_k\boldsymbol{u}_k + \boldsymbol{w}_k$$

with input matrix B and the new state prediction

$$\hat{\boldsymbol{x}}_{k|k-1} = F_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} + B_k\boldsymbol{u}_k \tag{26}$$

The rest of the Kalman equations (18)–(21) are unaltered.

A reformulation of the VS process equation (23) uses the change in joint angles as input

$$\boldsymbol{x}_k = \boldsymbol{\phi}_k = \boldsymbol{\phi}_{k-1} + \boldsymbol{h}_{\theta,k} + \boldsymbol{w}_k \tag{27}$$

where $\boldsymbol{h}_{\theta,k} \equiv \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$ and $B = I$.

This implements a stationary-target assumption, as shown by substituting for $\boldsymbol{\phi}$ and $\boldsymbol{h}_\theta$ in (27) to get

$$\boldsymbol{\theta}_k - \boldsymbol{\theta}^*_k = \boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}^*_{k-1} + \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1} + \boldsymbol{w}_k \tag{28}$$

which reduces to

$$\boldsymbol{\theta}^*_k = \boldsymbol{\theta}^*_{k-1} - \boldsymbol{w}_k \tag{29}$$

This assumption can improve estimation with static targets but it lessens the system generality. However, in practice it is often known beforehand whether or not the target is stationary so the trade-off is favorable. The measurement equation for this system is (25).

3.2.4.3 System using first order state space representation without input

A more detailed state description includes the error velocity as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\phi} \\ \dot{\boldsymbol{\phi}} \end{bmatrix} \tag{30}$$

Assuming a constant error velocity and no input, the process equation is

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} = \begin{bmatrix} I & h_t I \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{k-1} \\ \dot{\boldsymbol{\phi}}_{k-1} \end{bmatrix} + \boldsymbol{w}_k \tag{31}$$

where h_t is the elapsed time between iterations. Since there is no direct measurement of image-plane error velocity the measurement equation is

$$\boldsymbol{z}_k = \boldsymbol{f}_k = [J_k \; 0] \begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} + \boldsymbol{v}_k \tag{32}$$

The system modeled by (31) represents one where the robot is steadily closing on the target. The process error $\boldsymbol{w}$ allows for a varying rate of change. The measurement equation (32) provides means for updating the error velocity estimate. Using F and H from (31) and (32) the current estimate via (17), (19), and (20) is

$$\widehat{\dot{\boldsymbol{\phi}}}_{k|k} = \widehat{\dot{\boldsymbol{\phi}}}_{k-1|k-1} + p_{3,k|k-1} J_k^{\mathsf{T}} \left[J_k p_{1,k|k-1} J_k^{\mathsf{T}} + R_k \right]^{-1} \left[\boldsymbol{f}_k - J_k \widehat{\boldsymbol{\phi}}_{k|k-1} \right] \tag{33}$$

where p_3 and p_1 are blocks of P as

$$P = \begin{bmatrix} p_1 & p_2 \\ p_3 & p_4 \end{bmatrix}$$

To gauge the effects of different terms on the velocity estimate responsiveness to measurements expand $p_{3,k|k-1}$ and $p_{1,k|k-1}$ using (18)

$$p_{3,k|k-1} = p_{3,k-1|k-1} + h_t p_{4,k-1|k-1} + q_{3,k} \tag{34}$$

$$p_{1,k|k-1} = p_{1,k-1|k-1} + h_t \left(p_{2,k-1|k-1} + p_{3,k-1|k-1} \right) + h_t^2 p_{4,k-1|k-1} + q_{1,k} \tag{35}$$

where q_3 and q_1 are blocks of Q as

$$Q = \begin{bmatrix} q_1 & q_2 \\ q_3 & q_4 \end{bmatrix}$$

The salient points from (33)–(35) are that $\widehat{\dot{\boldsymbol{\phi}}}$ changes more rapidly with decreasing R and q_1 and increasing q_3.

3.2.4.4 First order system with input

Similar to §3.2.4.2, input u can be added to the process equation (31) imposing some assumptions on the independent motion of the target and possibly the robot. Three possible forms are

$$\boldsymbol{u}_k = \begin{bmatrix} 0 \\ \boldsymbol{h}_{\omega,k} \end{bmatrix}, \qquad \boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{h}_{\theta,k} \\ 0 \end{bmatrix}, \qquad \text{or} \qquad \boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{h}_{\theta,k} \\ \boldsymbol{h}_{\omega,k} \end{bmatrix}$$

where $\boldsymbol{h}_{\omega,k} \equiv \dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}_{k-1}$. Including $\boldsymbol{u}$ in (31) with the definitions of $\boldsymbol{\phi}$, $\boldsymbol{h}_\theta$ and $\boldsymbol{h}_\omega$ shows that only one of these forms is logical. The three resulting versions of the process equation are

$$\begin{bmatrix} \boldsymbol{\theta}_k - \boldsymbol{\theta}^*_k \\ \dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}^*_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}^*_{k-1} + h_t\dot{\boldsymbol{\theta}}_{k-1} - h_t\dot{\boldsymbol{\theta}}^*_{k-1} \\ \dot{\boldsymbol{\theta}}_{k-1} - \dot{\boldsymbol{\theta}}^*_{k-1} + \dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}_{k-1} \end{bmatrix} + \boldsymbol{w}_k$$
$$\begin{bmatrix} \boldsymbol{\theta}_k - \boldsymbol{\theta}^*_k \\ \dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}^*_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}^*_{k-1} + h_t\dot{\boldsymbol{\theta}}_{k-1} - h_t\dot{\boldsymbol{\theta}}^*_{k-1} + \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1} \\ \dot{\boldsymbol{\theta}}_{k-1} - \dot{\boldsymbol{\theta}}^*_{k-1} \end{bmatrix} + \boldsymbol{w}_k$$
$$\begin{bmatrix} \boldsymbol{\theta}_k - \boldsymbol{\theta}^*_k \\ \dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}^*_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}^*_{k-1} + h_t\dot{\boldsymbol{\theta}}_{k-1} - h_t\dot{\boldsymbol{\theta}}^*_{k-1} + \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1} \\ \dot{\boldsymbol{\theta}}_{k-1} - \dot{\boldsymbol{\theta}}^*_{k-1} + \dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}_{k-1} \end{bmatrix} + \boldsymbol{w}_k$$

which reduce to

$$\begin{bmatrix} \boldsymbol{\phi}_k \\ -\dot{\boldsymbol{\theta}}^*_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_{k-1} + h_t\dot{\boldsymbol{\phi}}_{k-1} \\ -\dot{\boldsymbol{\theta}}^*_{k-1} \end{bmatrix} + \boldsymbol{w}_k \tag{36}$$

$$\begin{bmatrix} -\boldsymbol{\theta}^*_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\theta}^*_{k-1} + h_t\dot{\boldsymbol{\phi}}_{k-1} \\ \dot{\boldsymbol{\phi}}_{k-1} \end{bmatrix} + \boldsymbol{w}_k \tag{37}$$

$$\begin{bmatrix} -\boldsymbol{\theta}^*_k \\ -\dot{\boldsymbol{\theta}}^*_k \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\theta}^*_{k-1} + h_t\dot{\boldsymbol{\theta}}_{k-1} - h_t\dot{\boldsymbol{\theta}}^*_{k-1} \\ -\dot{\boldsymbol{\theta}}^*_{k-1} \end{bmatrix} + \boldsymbol{w}_k \tag{38}$$

Process equation (36) assumes a target moving with constant velocity. In (37) the target is moving with velocity equal to the (constant) error velocity, thus the robot moves with twice this velocity. In order for both top and bottom blocks of (38) to hold it follows that $\dot{\boldsymbol{\theta}} = 0$ so the restriction is that the robot is not moving. This is an invalid assumption for visual servoing and (37) imposes an unlikely restriction so only the constant target velocity case (36) is feasible, where $\boldsymbol{u}^\mathsf{T} = \begin{bmatrix} 0 \; \boldsymbol{h}_{\omega,k}^\mathsf{T} \end{bmatrix}$, yielding the process equation

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} = \begin{bmatrix} I & h_t I \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{k-1} \\ \dot{\boldsymbol{\phi}}_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{h}_{\omega,k} \end{bmatrix} + \boldsymbol{w}_k \tag{39}$$

and the measurement equation is (32).

3.2.4.5 Summary of formulations

The system

$$\boldsymbol{x}_k = F_{k-1}\boldsymbol{x}_{k-1} + B_k \boldsymbol{u}_k + \boldsymbol{w}_k \tag{40}$$

$$\boldsymbol{z}_k = H_k \boldsymbol{x}_k + D_k \boldsymbol{u}_k + \boldsymbol{v}_k \tag{41}$$

has been formulated for image-based visual servoing in four ways:

1. zeroth order without input
2. zeroth order with input
3. first order without input
4. first order with input

In this section the vectors and matrices are identified for each formulation.

Zeroth order without input The system equation terms are

$$\boldsymbol{x}_k = \boldsymbol{\phi}_k \equiv \boldsymbol{\theta}_k - \boldsymbol{\theta}_k^*, \qquad \boldsymbol{u}_k = \boldsymbol{0}, \qquad \boldsymbol{z}_k = \boldsymbol{f}_k$$

$$F_k = I, \qquad B_k = I, \qquad H_k = J_k, \qquad \text{and} \qquad D_k = I$$

which yield the equations

$$\begin{aligned}\bar{\phi}_k &= \bar{\phi}_{k-1} + \bar{w}_k \\ \bar{f}_k &= J_k \bar{\phi}_k + \bar{v}_k\end{aligned}$$

Zeroth order with input The system equation terms are

$$\begin{aligned}&\boldsymbol{x}_k = \boldsymbol{\phi}_k \equiv \boldsymbol{\theta}_k - \boldsymbol{\theta}^*_k, \qquad \boldsymbol{u}_k = \boldsymbol{h}_{\theta,k} \equiv \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}, \qquad \boldsymbol{z}_k = \boldsymbol{f}_k \\ &F_k = I, \qquad B_k = I, \qquad H_k = J_k, \qquad \text{and} \qquad D_k = 0\end{aligned}$$

which yield the equations

$$\begin{aligned}\boldsymbol{\phi}_k &= \boldsymbol{\phi}_{k-1} + \boldsymbol{h}_{\theta,k} + \boldsymbol{w}_k \\ \boldsymbol{f}_k &= J_k \boldsymbol{\phi}_k + \boldsymbol{v}_k\end{aligned}$$

First order without input The system equation terms are

$$\begin{aligned}&\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix}, \qquad \boldsymbol{u}_k = \boldsymbol{0}, \qquad \boldsymbol{z}_k = \boldsymbol{f}_k \\ &F_k = \begin{bmatrix} I & h_t I \\ 0 & I \end{bmatrix}, \qquad B_k = I, \qquad H_k = [J_k \; 0], \qquad \text{and} \qquad D_k = I\end{aligned}$$

which yield the equations

$$\begin{aligned}\begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} &= \begin{bmatrix} I & h_t I \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{k-1} \\ \dot{\boldsymbol{\phi}}_{k-1} \end{bmatrix} + \boldsymbol{w}_k \\ \boldsymbol{f}_k &= [J_k \; 0] \begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} + \boldsymbol{v}_k\end{aligned}$$

First order with input The system equation terms are

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix}, \qquad \boldsymbol{u}_k = \begin{bmatrix} 0 \\ \boldsymbol{h}_{\omega,k} \end{bmatrix} = \begin{bmatrix} 0 \\ \dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}_{k-1} \end{bmatrix}, \qquad \boldsymbol{z}_k = \boldsymbol{f}_k$$

$$F_k = \begin{bmatrix} I & h_t I \\ 0 & I \end{bmatrix}, \qquad B_k = I, \qquad H_k = [J_k \; 0], \qquad \text{and} \qquad D_k = 0$$

which yield the equations

$$\begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} = \begin{bmatrix} I & h_t I \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{k-1} \\ \dot{\boldsymbol{\phi}}_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{h}_{\omega,k} \end{bmatrix} + \boldsymbol{w}_k$$

$$\boldsymbol{f}_k = [J_k \; 0] \begin{bmatrix} \boldsymbol{\phi}_k \\ \dot{\boldsymbol{\phi}}_k \end{bmatrix} + \boldsymbol{v}_k$$

3.2.5 Control action

The joint-space error $\boldsymbol{\phi} \equiv \boldsymbol{\theta}_k - \boldsymbol{\theta}_k^*$ estimated by the Kalman filter is used to set the robot joint offset command as

$$\Delta\boldsymbol{\theta}_{k+1} = \begin{cases} -\widehat{\boldsymbol{\phi}}_{k|k}, & \text{if} \quad \left\| \widehat{\boldsymbol{\phi}}_{k|k} \right\| < \mu \\ -\mu \dfrac{\widehat{\boldsymbol{\phi}}_{k|k}}{\left\| \widehat{\boldsymbol{\phi}}_{k|k} \right\|}, & \text{otherwise} \end{cases} \tag{42}$$

where μ is the maximum allowable joint offset norm, which is necessary to keep the robot motion confined to one VS control period. This also aids Jacobian estimation since too great a displacement renders the update less accurate. That is because the chosen Jacobian estimation algorithm [63] is recursive and the map is only locally valid so a previous estimate at distant $\boldsymbol{\theta}_{k-1}$ pulls the current estimate towards an invalid value.

3.2.5.1 Stability of Kalman filter control law

It is assumed that the joint-level servoing can attain any desired joint offset in one time step for sufficiently small μ. Thus, the state equation for the robot position is

$$\boldsymbol{\theta}_{k+1} = F\boldsymbol{\theta}_k + B\Delta\boldsymbol{\theta}_{k+1}$$

and the *reachability matrix* $C = \begin{bmatrix} B & FB & F^2B & \dots & F^{a-1}B \end{bmatrix}$ has rank a, the robot degrees of freedom, because both F and B are the $a \times a$ identity matrix. A full-rank reachability matrix is the criterion for a fully controllable discrete linear time-invariant system. Therefore the robot is completely controllable under the assumption above.

The Kalman filter acts as an observer of the state $\boldsymbol{x} = \boldsymbol{\phi}$ (or $\begin{bmatrix} \boldsymbol{\phi}^\mathsf{T} & \dot{\boldsymbol{\phi}}^\mathsf{T} \end{bmatrix}^\mathsf{T}$ for first order representation). A linear system is said to be *observable* at t_0 if $\boldsymbol{x}_0$ can be determined from the output sequence $\boldsymbol{z}_0 \dots \boldsymbol{z}_j$ for $t_0 \leq t_j$, where t_j is some finite time. If this is true for all t_0 and x_0 then the system is said to be *completely observable* [9]. This property ensures the convergence of P and bounded uncertainty (that is, error covariance).

In order for the system (40) and (41) to be completely observable at $k = 0$ there must exist some finite index N such that the square matrix

$$X = \sum_{k=0}^{N} \Phi_{k,0}^\mathsf{T} H_k^\mathsf{T} H_k \Phi_{k,0} \tag{43}$$

is positive definite, where Φ is the transition matrix. Provided the input sequence $\boldsymbol{u}_0 \dots \boldsymbol{u}_k$ is zero then given the state $\boldsymbol{x}_j$, the state at any other time k is given by the mapping $\boldsymbol{x}_k = \Phi_{k,j}\boldsymbol{x}_j$ [9].

For the formulations of §3.2.4.5 the process matrix F is constant. In such a case the transition matrix is $\Phi_{k,j} = F^{k-j}$ and (43) reduces to

$$X = \sum_{k=0}^{N} \left[\left(F^k \right)^\mathsf{T} H_k^\mathsf{T} H_k F^k \right]$$

This observability criterion is examined for the two state space representations of §3.2.4.5 and sufficient conditions are derived to ensure positive definite X for all $N \geq 1$.

Zeroth order representation In this case $F = I$ and $H_k = J_k$ so $X = \sum_{k=0}^{N} J_k^\mathsf{T} J_k$. To ensure positive definite X a sufficient condition is that J_l has full column rank for some l, where $0 \leq l \leq N$. The robot must be in a full column rank configuration at least once during servoing. In that case X is the sum of a positive definite matrix and N positive semidefinite matrices and is therefore positive definite.

First order representation In this case $F = \begin{bmatrix} I & h_t I \\ 0 & I \end{bmatrix}$ and $H_k = \begin{bmatrix} J_k & 0 \end{bmatrix}$, where $0 < h_t \leq 1$, which leads to the block matrix

$$X = \sum_{k=0}^{N} \begin{bmatrix} J_k^\mathsf{T} J_k & k h_t J_k^\mathsf{T} J_k \\ k h_t J_k^\mathsf{T} J_k & k^2 h_t^2 J_k^\mathsf{T} J_k \end{bmatrix}$$

X is positive definite if J_0 and some J_m, $0 < m \leq N$ are both full column rank.

Proof.

$$X = \begin{bmatrix} J_0^\mathsf{T} J_0 + J_m^\mathsf{T} J_m & mh J_m^\mathsf{T} J_m \\ mh J_m^\mathsf{T} J_m & m^2 h^2 J_m^\mathsf{T} J_m \end{bmatrix} + \sum_{k=1}^{m-1} \begin{bmatrix} J_k^\mathsf{T} J_k & k h_t J_k^\mathsf{T} J_k \\ k h_t J_k^\mathsf{T} J_k & k^2 h_t^2 J_k^\mathsf{T} J_k \end{bmatrix}$$
$$+ \sum_{k=m+1}^{N} \begin{bmatrix} J_k^\mathsf{T} J_k & k h_t J_k^\mathsf{T} J_k \\ k h_t J_k^\mathsf{T} J_k & k^2 h_t^2 J_k^\mathsf{T} J_k \end{bmatrix} \tag{44}$$

Let $M_k \equiv \begin{bmatrix} J_k & k h_t J_k \end{bmatrix}$, then the block matrices for $k \neq 0$, m of (44) are $M_k^\mathsf{T} M_k$ and thus are always positive semidefinite. Further, let

$$A = \begin{bmatrix} J_0^\mathsf{T} J_0 + J_m^\mathsf{T} J_m & mh J_m^\mathsf{T} J_m \\ mh J_m^\mathsf{T} J_m & m^2 h^2 J_m^\mathsf{T} J_m \end{bmatrix}$$

and subtract $1/mh$ times column 2 of A from the first column of A to get A'

$$A' = \begin{bmatrix} J_0^\mathsf{T} J_0 & mh J_m^\mathsf{T} J_m \\ 0 & m^2 h^2 J_m^\mathsf{T} J_m \end{bmatrix}$$

which has the same determinant as A. The determinant of a block triangular matrix is the product of the determinants of its diagonal entries, so

$$\det(A) = |J_0^\mathsf{T} J_0| \cdot |m^2 h^2 J_m^\mathsf{T} J_m| > 0$$

since $J_0^\mathsf{T} J_0$ and $J_m^\mathsf{T} J_m$ are both positive definite. Since A is positive definite and the block matrices in the summation are positive semidefinite then X is positive definite.

□

Regarding the formulations in §3.2.4.5, the zeroth order systems are completely observable if the robot passes through a full column rank configuration at least once during servoing. Complete observability of the first order systems necessitates the additional condition that servoing begin with the robot in a full column rank configuration.

3.2.6 Comparison of static and dynamic RLS control laws

There is a difference between using the Kalman equations for an assumed static state and using static RLS. An example using non-recursive forms shows a difference stemming from the error $\boldsymbol{w}_k$ in the process equation (15).

3.2.6.1 The steady model example

RLS and KF are used to estimate a scalar variable x based on scalar measurements $z_1 \dots z_k$.

$$z_k = x_k + v_k \qquad \text{(KF and RLS)}$$

Errors v_k are independent with variance $R_k = 1$. The difference in the approaches is that while both assume x is unchanging, KF allows for the possibility that it does change via the process error term w_k, also with variance $Q_k = 1$. For RLS the process equation is

$$x_k = x_{k-1} \qquad \text{(RLS)}$$

and the KF form is

$$x_k = x_{k-1} + w_k \qquad \text{(KF)}$$

The RLS estimate after n measurements is the average

$$\hat{x}_{n-1|n-1} = \frac{z_0 + z_1 + \ldots + z_{n-1}}{n}$$

as is seen in (11) with the $n \times 1$ matrix $\mathfrak{A} = [1\ 1\ \dots\ 1]^{\mathsf{T}}$ and $\Sigma = I$ with dimension $n \times n$. For KF, the complete system has the form of (14), where $H_k = F_k = [1]$.

As an illustration, consider applying KF to this system for cases where $n = 2$ and $n = 3$ (with $\Sigma = I$ of dimensions 3×3 and 5×5, respectively). The weighted normal equations $\mathfrak{A}^{\mathsf{T}}\Sigma\mathfrak{A}\hat{\mathfrak{x}} = \mathfrak{A}^{\mathsf{T}}\Sigma\mathfrak{z}$ are

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} \hat{x}_{0|1} \\ \hat{x}_{1|1} \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} \hat{x}_{0|2} \\ \hat{x}_{1|2} \\ \hat{x}_{2|2} \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} \tag{45}$$

and the inverses $\left(\mathfrak{A}^{\mathsf{T}}\Sigma\mathfrak{A}\right)^{-1}$ are

$$\frac{1}{3}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad \text{and} \qquad \frac{1}{8}\begin{bmatrix} 5 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 5 \end{bmatrix}$$

For $n = 2$ (45) yields two estimates,

$$\hat{x}_{0|1} = \frac{2z_0 + z_1}{3}$$
$$\hat{x}_{1|1} = \frac{z_0 + 2z_1}{3}$$

whereas regular least squares gives

$$\hat{x}_{1|1} = \frac{z_0 + z_1}{2}$$

The KF estimate responds to the possibility of change in x by weighting recent measurements more heavily than early ones. Also, the estimate $\hat{x}_0$ of the state at time 0 is refined to include the new measurement (though weighted less than the old one), by the smoothing process. Repeating this for the $n = 3$ case yields

$$\hat{x}_{0|2} = \frac{5z_0 + 2z_1 + z_2}{8}, \qquad \hat{x}_{1|2} = \frac{2z_0 + 4z_1 + 2z_2}{8}, \quad \text{and} \qquad \hat{x}_{2|2} = \frac{z_0 + 2z_1 + 5z_2}{8}$$

and least squares gives

$$\hat{x}_{2|2} = \frac{z_0 + z_1 + z_2}{3}$$

The treatment of data approaches an exponential weighting (the Fibonacci numbers, in fact). This example illustrates how RLS and Kalman filtering produce different results for the same static system.

3.2.6.2 Exponentially weighted RLS control law example

Static RLS is simpler to implement than Kalman filtering because it does not require estimation of process noise. This is advantageous because process noise can be difficult to quantify for complex systems. The downside of a noiseless process model is that the state estimate responds slowly to parameter variations. A compromise is to discard old data in static RLS estimation using a moving window. A popular method uses a weighting function, which "controls the way in which each measurement is incorporated relative to other measurements" [21].

Exponentially weighted recursive least squares (EWRLS) uses an exponential of a constant forgetting factor $0 < \lambda \leq 1$. The number of past measurements used for the estimate is approximately $\frac{1}{1-\lambda}$. Higher λ results in longer memory (a larger window), so "remembering ratio" might be a more informative alliterative term than the common "forgetting factor."

EWRLS is a special case of the Kalman Filter. Its equations can be written in the following Kalman form for comparison [21],

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k-1|k-1} + K_k \left(\boldsymbol{z}_k - H_k \hat{\boldsymbol{x}}_{k-1|k-1} \right) && (46) \\ K_k &= P_{k-1|k-1} H_k^{\mathsf{T}} \left(H_k P_{k-1|k-1} H_k^{\mathsf{T}} + \lambda I \right)^{-1} && (47) \\ P_{k|k} &= \frac{1}{\lambda} \left(I - K_k H_k \right) P_{k-1|k-1} && (48) \end{aligned}$$

The double subscripts now are superfluous because there is no prediction step (for example, $\hat{\boldsymbol{x}}_{k|k-1} = I\hat{\boldsymbol{x}}_{k-1|k-1}$) but they are left in for easy comparison with the Kalman equations

(17)–(21). Again, EWRLS assumes a static state ($F_i = I$) and makes no allowance for drift ($Q_i = 0$), the result of which is the smoothed value $\hat{\boldsymbol{x}}_{i|k} = \hat{\boldsymbol{x}}_{j|k}$. The other differences are that R_k in (19) is replaced by λI in (47), and the inclusion of $1/\lambda$ in the estimate of P.

EWRLS can be said to be a special case of KF because both provide a recursive solution to the weighted least squares problem,

$$\text{Minimize} \quad (\mathfrak{z} - \mathfrak{A}\mathfrak{x})^{\mathsf{T}} W (\mathfrak{z} - \mathfrak{A}\mathfrak{x})$$

where W is a weighting matrix. The components of this problem have different values for EWRLS and KF. They are given below for a system at time k with measurements $\boldsymbol{z}$ of dimension $n \times 1$ and a state vector $\boldsymbol{x}$ of dimension $m \times 1$. This form shows that KF with $F = I$, $Q = 0$, and $R = I$ is equivalent to EWRLS with $\lambda = 1$.

<u>EWRLS</u>

$$\mathfrak{z} - \mathfrak{A}\mathfrak{x} = \begin{bmatrix} \boldsymbol{z}_0 \\ \mathbf{0} \\ \boldsymbol{z}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{z}_k \end{bmatrix} - \begin{bmatrix} H_0 & & & & \\ -I & I & & & \\ & H_1 & & & \\ & -I & I & & \\ & & \ddots & & \\ & & -I & I & \\ & & & H_k & \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x} \\ \vdots \\ \boldsymbol{x} \end{bmatrix}$$

$$W = \begin{bmatrix} \lambda^{k-1} I_n & & & & & \\ & 0_m & & & & \\ & & \ddots & & & \\ & & & \lambda I_n & & \\ & & & & 0_m & \\ & & & & & I_n \end{bmatrix}$$

<u>KF</u>

$$\mathfrak{z} - \mathfrak{A}\mathfrak{x} = \begin{bmatrix} \boldsymbol{z}_0 \\ \mathbf{0} \\ \boldsymbol{z}_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{z}_k \end{bmatrix} - \begin{bmatrix} H_0 & & & & \\ -F_0 & I & & & \\ & H_1 & & & \\ & -F_1 & I & & \\ & & \ddots & & \\ & & -F_{k-1} & I & \\ & & & H_k & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_k \end{bmatrix}$$

$$W = \begin{bmatrix} R_0^{-1} & & & & & \\ & Q_1^{-1} & & & & \\ & & \ddots & & & \\ & & & R_{k-1}^{-1} & & \\ & & & & Q_k^{-1} & \\ & & & & & R_k^{-1} \end{bmatrix}$$

Solving the least squares problem with the EWRLS components $\mathfrak{z}$, $\mathfrak{A}$, $\mathfrak{x}$, and W, above, is equivalent to minimizing the cost function,

$$\varepsilon_k = \sum_{i=1}^{k} \left(\lambda^{k-i} \left\| \boldsymbol{z}_i - H_i \boldsymbol{x}_k \right\|^2 \right) \tag{49}$$

To gain a sense of what ε_k means for the current application, substitute from (23)–(25) and employ the approximation (22) to get

$$\varepsilon_k = \sum_{i=1}^{k} \left[\lambda^{k-i} \left\| \boldsymbol{f}_i - J_i \boldsymbol{\phi}_k \right\|^2 \right]$$

This is an exponentially weighted sum of the squares of the differences between the measured image plane error at time i and the expected value using the Jacobian at i and the *current* joint-space error.

Gumpert [28] introduces a visual-servoing control law using EWRLS by extending the dynamic Gauss-Newton method to have less sensitivity to noise and calls it a recursive Gauss-Newton algorithm (RGN). He defines the state being estimated as the relative joint offset necessary to minimize the weighted sum of the squares of the errors. In other words, he uses RGN to estimate the state $\boldsymbol{x}_k = \boldsymbol{h}_{\theta_k} = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$ so as to minimize (49). The observation model H_k in his formulation is the composite Jacobian J_k. The control action is

$$\begin{aligned}
\boldsymbol{\theta}_{k+1} &= 2\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1} - K_k \left(\boldsymbol{f}_k + \frac{\partial \boldsymbol{f}_k}{\partial t} (t_{k+1} - t_k) + J_k (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \right) \\
K_k &= P_{k-1} J_k^{\mathsf{T}} \left(J_k P_{k-1} J_k^{\mathsf{T}} + \lambda I \right)^{-1} \\
P_k &= \frac{1}{\lambda} (I - K_k J_k) P_{k-1}
\end{aligned}$$

Data exist comparing the performance of this control law with dynamic Gauss-Newton. Gumpert does so via a simulated two degree-of-freedom robot. Even without noise the Gauss-Newton controller is more susceptible to instabilities than RGN, but the latter requires slightly more time for convergence. With noise (± 1 pixel) added the average RMS error using RGN is lower than with the Gauss-Newton control law. Experiments performed by Piepmeier [61] for a 2-DoF planar robot show similar performances between the two controllers, yet her simulation of a 6-DoF Puma 560 manipulator tracking a target through a circular path with ± 0.5 pixel noise added to the image data gives slightly better results for the dynamic Gauss-Newton controller than for RGN [62].

These data provide some validation of the impetus behind the RGN but overall there is little gain is to be had over the classic Gauss-Newton control law. For this reason the control laws introduced in Chapters 3, 4, and 5 are compared only to Gauss-Newton.

3.2.7 Noise covariance

The covariance matrices for the process and measurement noise vectors characterize shortcomings in the models and noise in the sensors. Conceptually, the difference in magnitude between Q and R influences the filter in favor of either the process model or the measurements. The covariance matrices also play integral roles in the behavior of the error covariance of the state estimate. Influences on the process and measurement noise covariance matrices are now examined.

Process error w is due to unfit robot-target motion assumptions. For example, servoing to a stationary target using (23) and (24) merits larger Q elements than using (30) and (31) does since VS is designed to guide the robot closer to the target at every step. The covariance R of the measurement error pertains to the following sources:

- Inaccuracy of Jacobian estimate in the neighborhood of $\boldsymbol{\theta}_k$
- Inappropriateness of the linear approximation (22), which increases as robot gets farther from the goal position (that is, large $\boldsymbol{\phi}$)
- Noise in robot joint sensors and discretization errors
- Noise in camera sensors and discretization errors

These sources of noise are harder to quantify than in the process equation. This is especially true for an uncalibrated system where no prior information about sensors or kinematics is known.

The fact that these values for Q and R are static (they do not change during servoing) hinders performance of the filter, as discussed in Chapter 4. Furthermore, they are not in keeping with the spirit of uncalibrated visual servoing since they require knowledge about the system. Methods that address these issues are discussed in the next chapter

3.2.8 Initialization of the algorithm

The algorithm requires initialization of two quantities: $\hat{\boldsymbol{x}}_{0|0}$ (that is, $\hat{\boldsymbol{\phi}}_{0|0}$) and P_0. The initial state estimate $\hat{\boldsymbol{x}}_0$ is obtained from the Gauss-Newton method (7) using $\boldsymbol{\theta}_0$, $\boldsymbol{f}_0$, and J_0

$$\hat{\boldsymbol{x}}_{0|0} = \left(J_0^{\mathsf{T}} J_0\right)^{-1} J_k^{\mathsf{T}} \boldsymbol{f}_0$$

P_0 is set to a diagonal matrix βI and theory states that for a controllable and observable Kalman filter (see §3.2.5.1) the error covariance estimate converges regardless of its initial value.

3.3 Summary

A traditional VS control law is described. The Kalman filter is applied to VS, estimating the joint-space offset between the current robot position and that minimizing the image-plane error norm $\boldsymbol{f}^{\mathsf{T}}\boldsymbol{f}$. Four versions are introduced, each with a different robot-target motion assumption

- Constant joint-space error (zeroth order state space representation)
- Stationary target (zeroth order state space representation, with input)
- Constant joint-space error velocity (first order state space representation)
- Constant (joint-space) velocity target (first order state space representation, with input)

The stability of these four filters is studied, the conclusion being that the robot must begin servoing from a full column rank configuration and pass through at least one other full column rank configuration during servoing to guarantee stability. The control action resulting from the KF implementation is described, and the overall system is shown controllable.

The sources of the noise covariances Q and R are discussed. Further discussion of these matrices is presented in Chapter 4.

CHAPTER IV

ADAPTIVE KALMAN FILTER CONTROL LAWS

The Kalman filter control law of Chapter 3 is introduced to improve upon traditional control laws for visual servoing. It uses two assumptoins: that the covariances of the process and measurement noises are known and that they are constant (see §3.2.7). The present chapter discusses the shortcomings of this second assumption and alternatives to it.

The Kalman equations (17–21) constitute a linear, unbiased, and minimum error variance algorithm; they provide an optimal estimate in the sense that the standard deviation of the error probability density is minimized. To achieve optimality requires complete knowledge of the process and measurement noise statistics, but in practice determining satisfactory Q and R can be a struggle. Values for these matrices are often based on knowledge of system parameters and on experience. Determining these for uncalibrated applications or manual tuning can be difficult. Further, constant noise covariance matrices are problematic for time-varying error sources.

Poor characterization of Q and R can lead to either unreliable estimates or divergence of the filter, which typically means that the covariance matrix of the state estimate error P does not adequately reflect the true error. Therefore, adaptive Kalman filtering has been developed to adjust the noise covariance matrices as measurements become available. Research has been active in this field for more than forty years and numerous techniques exist. Mehra [52] describes the following four categories for adaptive filtering methods:

- Bayesian
- Maximum likelihood
- Correlation
- Covariance matching

Neither Bayesian nor maximum likelihood methods are favored in practice due to their sometimes excessive computational requirements. This chapter reviews the other two types of methods, provides references to implementations, and assesses the suitability of both categories for the VS control law presented in Chapter 3.

4.1 Correlation Approaches

Two correlation approaches to adaptive Kalman filtering are discussed here. The classical approach is presented by Mehra [51]. The more recent method comes from Odelson et al. [57] and is called ALS for *autocovariance least-squares*. Different terms, "autocorrelation" and "autocovariance," are used in the two works, but they refer to the same quantity: "the expectation of the data with some lagged version of itself" [57].

Mehra develops two different correlation methods, one based on the measurements $\boldsymbol{z}_k$ and the other on the innovation

$$\boldsymbol{\nu}_k \equiv \boldsymbol{z}_k - H_k \hat{\boldsymbol{x}}_{k|k-1} \tag{50}$$

The innovation method is more efficient because innovations are less correlated than measurements and is the preferred approach. "It is known from the theory of Kalman filtering that the innovation sequence is a zero-mean Gaussian white noise sequence for an optimal filter. However, for a suboptimal filter, the innovation sequence is correlated ..." [51]. The measurement method is less efficient because $\boldsymbol{z}_k$ does not represent totally new information, as a portion of it is in the prediction $H_k \hat{\boldsymbol{x}}_{k|k-1}$.

In both the classic approach and the ALS method the foundational quantity is the j-th lag autocovariance of the innovation $\boldsymbol{\nu}_k$,

$$C_{j,k} \equiv \mathrm{E}\left[\boldsymbol{\nu}_k \boldsymbol{\nu}_{k-j}^{\mathsf{T}}\right]$$

This expectation is estimated as

$$\widehat{C}_{j,k} = \frac{1}{N-j} \sum_{i=0}^{N-j-1} \boldsymbol{\nu}_{k-i} \boldsymbol{\nu}_{k-i-j}^{\mathsf{T}} \tag{51}$$

where N is the sample size.

The classic approach, and those based on it, take three steps to estimate Q and R. The first step uses (51) in a least squares problem to estimate PH^{T}. The second calculates

$$\widehat{R} = \widehat{C}_k - H\left(\widehat{PH^{\mathsf{T}}}\right) \tag{52}$$

and the final step determines Q by employing the estimates of PH^{T} and R.

Neethling and Young [56] point out that this results in statistical estimates with large variances. One cause, according to Odelson et al. [57], is splitting the estimation of Q and R into two stages. They improve upon the above method with the ALS technique.

ALS is a single-step procedure and, for a reported example, results in estimates with at least an order of magnitude less variance than the classical approach. It is possible, though, that the estimates of Q and R may not be positive semidefinite. Since such estimates are without physical meaning, the authors add constraints to guarantee positive semidefinite covariances. Åkesson et al. [1] introduce a specialized algorithm to solve this semidefinite programming problem.

The ALS method is used by Wu et al. [79] to characterize the effects of plant mismatch and unmodeled disturbances on estimates of zone temperatures for use with Model Predictive Control of the ventilation system in a large scale livestock barn. They report significant performance improvement with the adaptive filter compared to the traditional manually-tuned system. This is one of only a handful of implementations of ALS reported in the literature.

4.2 Covariance Matching Approaches

This category is the most widely used, most intuitive, and provides acceptable results (see Chapters 7 and 8). These methods work to adjust either Q or R (or both) so that estimated covariances are consistent with the theoretical values.

For example, the estimated covariance of the innovations is

$$\widehat{C}_{\nu,k} = \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{\nu}_{k-i}\boldsymbol{\nu}_{k-i}^{\mathsf{T}} \tag{53}$$

Note the difference between this and (51) that here there is no lag between the data samples in the outer product.

The theoretical covariance of the innovations can be obtained by applying the *law of covariance propagation* to the innovation (50). This law states that if $\boldsymbol{Y} = B\boldsymbol{X} + \boldsymbol{l}$ and $\boldsymbol{l}$ is known then the covariances of $\boldsymbol{Y}$ and $\boldsymbol{X}$ are related by

$$\Sigma_Y = B\,\Sigma_X B^{\mathsf{T}}$$

Proof.

$$\begin{aligned}
\Sigma_Y &= \mathrm{E}\left[(\boldsymbol{Y} - E[\boldsymbol{Y}])(\boldsymbol{Y} - E[\boldsymbol{Y}])^{\mathsf{T}}\right] \\
&= \mathrm{E}\left[(B\boldsymbol{X} + \boldsymbol{a} - BE[\boldsymbol{X}] - \boldsymbol{a})(B\boldsymbol{X} + \boldsymbol{a} - BE[\boldsymbol{X}] - \boldsymbol{a})^{\mathsf{T}}\right] \\
&= \mathrm{E}\left[(B\boldsymbol{X} - BE[\boldsymbol{X}])(B\boldsymbol{X} - BE[\boldsymbol{X}])^{\mathsf{T}}\right] \\
&= B\mathrm{E}\left[(\boldsymbol{X} - E[\boldsymbol{X}])(\boldsymbol{X} - E[\boldsymbol{X}])^{\mathsf{T}}\right] B^{\mathsf{T}} = B\,\Sigma_X B^{\mathsf{T}}
\end{aligned}$$

□

For the case where the vector $\boldsymbol{l}$ is subject to zero-mean noise (independent of $\boldsymbol{X}$) its covariance is included

$$\Sigma_Y = B\,\Sigma_X B^{\mathsf{T}} + \Sigma_l$$

Applying this to the innovation $\left(\boldsymbol{\nu}_k = \boldsymbol{z}_k - H_k\hat{\boldsymbol{x}}_{k|k-1}\right)$ yields

$$E[\boldsymbol{\nu}_k\boldsymbol{\nu}_k^{\mathsf{T}}] = (-H_k)\Sigma_{x_{k|k-1}}(-H_k)^{\mathsf{T}} + \Sigma_{z_k} = H_k P_{k|k-1} H_k^{\mathsf{T}} + R_k \tag{54}$$

or, using (18) for $P_{k|k-1}$,

$$E[\boldsymbol{\nu}_k\boldsymbol{\nu}_k^{\mathsf{T}}] = H_k\left(F_{k-1}P_{k-1|k-1}F_{k-1}^{\mathsf{T}} + Q_k\right)H_k^{\mathsf{T}} + R_k$$

Note that these two equations are only approximate, since for a suboptimal filter (inaccurate Q and R) $P_{k|k-1}$ and $P_{k-1|k-1}$ are not true representations of the error covariance of the state estimate.

A covariance matching technique using the innovation yields an estimate of the measurement noise covariance by combining (53) with (54) as

$$\widehat{C}_{\nu,k} = \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{\nu}_{k-i}\boldsymbol{\nu}_{k-i}^{\mathsf{T}} \approx E[\boldsymbol{\nu}_k\boldsymbol{\nu}_k^{\mathsf{T}}] = H_k P_{k|k-1} H_k^{\mathsf{T}} + R_k \tag{55}$$

and subtracting HPH^{T} from both sides yields the estimate for R at time k

$$\widehat{R}_{\nu,k} = \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{\nu}_{k-i}\boldsymbol{\nu}_{k-i}^{\mathsf{T}} - H_k P_{k|k-1} H_k^{\mathsf{T}} \tag{56}$$

This is the same form as (52) except here $P_{k|k-1}$ comes from the Kalman filter. The value of N is chosen empirically to provide balance between statistical significance (large N) and reactiveness (small N). One consolation about the need to manually tune this parameter is given by Almagbile et al. [2], who report little effect on adaptive filtering performance with changes in the window size.

Myers and Tapley [55] introduce an approach to estimate both Q and R at every iteration, whereas the above method estimates only R. The Myers-Tapley method for R adaptation is similar to (56) except that instead of assuming a zero-mean it uses the sample mean of $\boldsymbol{\nu}$

$$\hat{\boldsymbol{\nu}}_k = \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{\nu}_{k-i}$$

as the expected value in the estimated covariance of the innovation

$$\widehat{C}_{\nu,k} = \frac{1}{N}\sum_{i=0}^{N-1} \left(\boldsymbol{\nu}_{k-i} - \hat{\boldsymbol{\nu}}\right)\left(\boldsymbol{\nu}_{k-i} - \hat{\boldsymbol{\nu}}\right)^{\mathsf{T}}$$

The resulting estimate of the noise covariance is

$$\widehat{R}_k = \frac{1}{N}\sum_{i=0}^{N-1} \left[\left(\boldsymbol{\nu}_{k-i} - \hat{\boldsymbol{\nu}}\right)\left(\boldsymbol{\nu}_{k-i} - \hat{\boldsymbol{\nu}}\right)^{\mathsf{T}} - H_{k-i} P_{k-i|k-i-1} H_{k-i}^{\mathsf{T}}\right]$$

To estimate Q they follow the same line, except with the following approximation for the process error $\boldsymbol{w}_k$ instead of the innovation

$$\boldsymbol{q}_k \equiv \hat{\boldsymbol{x}}_{k|k} - F_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1}$$

with the result that

$$\widehat{Q}_k = \frac{1}{N}\sum_{i=0}^{N-1}\Big[\left(\boldsymbol{q}_{k-i}-\hat{\boldsymbol{q}}\right)\left(\boldsymbol{q}_{k-i}-\hat{\boldsymbol{q}}\right)^{\mathsf{T}} - \Big(F_{k-i}P_{k-i-1|k-i-1}F_{k-i}^{\mathsf{T}} - P_{k-i|k-i}\Big)\Big]$$

It is noted by Blanchet et al. [7] that using this as a basis to estimate both covariance matrices might give poor performance, "since it is not easy to distinguish between errors in Q and R." Furthermore, the best results with the covariance matching approach are obtained for the case in which Q is known but R is unknown [52]. The Myers-Tapley algorithm has nevertheless been successfully implemented and remains highly cited.

For example, Lippiello et al. [45] update both Q and R for an EKF estimating the position and orientation of a moving object. Despite the warning by Blanchet et al. cited above about simultaneously updating the two noise covariance matrices, Lippiello et al. report smaller errors in the estimate by doing this than by updating only one or the other (or neither).

Another example using covariance matching to update both matrices comes from Han et al. [29] who by doing so decrease the error in estimating the charge state of lead-acid batteries compared to static covariances. They do not test the filter when estimating only one of the covariances.

Ficocelli and Janabi-Sharifi [24] partially implement Myers-Tapley. They update Q in their extended Kalman filter estimation of target pose for visual servoing. They estimate the 6-D position of a target moving relative to a camera and compare the adaptive extended Kalman filter with an EKF using static Q and R. For both filters, they perform two trials. In one trial the object goes from rest to moving at constant velocity and then back to rest. In the other it undergoes a variety of maneuvers. The dynamic model (F) for their filter predicts constant-velocity motion of the target. For their standard EKF experiments they set the values of Q to be low (zero, in fact) for the first trial and conservatively high for the second trial. They find that the adaptive EKF outperforms the non-adaptive filter in both instances.

The biggest concern [5] about the covariance matching approaches described is that they do not guarantee positive semidefinite estimates of Q and R. The methods of the following section always generate positive semidefinite estimates.

4.2.1 Fading and scaling factors

A covariance matching technique introduced by Jwo and Weng [39] combines conventional KF with an adaptive tuning system that generates two scalar multipliers (λ_P and λ_R) for updating $P_{k|k-1}$ and R_k.

$$\begin{aligned} P_{k|k-1} &= \lambda_P \left(F_k P_{k-1|k-1} F_k^\mathsf{T} + Q_k \right), \quad \lambda_P \geq 1 \\ R_k &= \lambda_R R_{k-1} \end{aligned}$$

The purpose of the multipliers is to create a unity ratio from the traces of the estimated value of the innovation covariance (53) and its theoretical value (54).

Multiplier λ_P is the *fading factor* and increasing it influences the filter towards measurements (and away from the process model) during periods of high dynamic change in the actual state. The authors test its efficacy alone and in conjunction with the measurement noise *scaling factor* λ_R. Simulated navigation trials fusing data from the Global Positioning System (GPS) and an inertial navigation system (INS) show that the adaptive filter is nearly ten times more accurate when using both λ_P and λ_R than when using only the fading factor λ_P.

A similar work by Almagbile et al. [2] compares the performance of two covariance-matching based adaptive filters. One assumes R is known and applies a scaling factor to Q, and the other assumes Q is known and uses (56) to update R. The authors observe that the Q-scaling method is less sensitive to inaccuracies in the assumed R value than the method of estimating R is to errors in the assumed value of Q. On the other hand, the Q-scaling method provides less accuracy in the state estimate than does the R adaptation.

The scaling approaches are based on the traces of covariance matrices, which discards some information contained in (53) and (54). They are not used in the present work.

4.2.2 Fuzzy logic

Covariance matching is used as the basis for many fuzzy logic systems that tune Kalman filter noise covariances (for example, [47, 11, 77, 42, 38, 35]). A recent and illustrative instance is the work of Sung et al. [74], who use an adaptive Kalman filter as part of an ultra-precision positioning controller to provide a better position estimate than a capacitive displacement sensor alone yields. They define the "degree of matching" (DoM) as the difference between the theoretical and estimated covariances of the innovation.

$$DoM_k = H_k P_{k|k-1} H_k^\mathsf{T} + R_k - \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{\nu}_{k-i} \boldsymbol{\nu}_{k-i}^\mathsf{T}$$

This is a scalar since the filter estimates a 1-D state. It is the input to a single-input-single-output (SISO) fuzzy inference system (FIS) that adjusts R_k using the rule base

1. IF $DoM_k = 0$ THEN No change in R_k
2. IF $DoM_k > 0$ THEN Decrease R_k
3. IF $DoM_k < 0$ THEN Increase R_k

The updated value of the measurement noise covariance is $R_k = R_{k-1} + \Delta R_k$.

The membership functions used for DoM and ΔR are shown in Figure 2. The shapes and boundary layers for these are tuned through trial and error.

This is true of most fuzzy controllers. The shapes of the membership functions are typically designed by using knowledge of the system [75, 83]. Since one goal of uncalibrated visual servoing is to allow operation with little foreknowledge of the system (cameras, robot, etc.) then the fuzzy-logic approach to adaptive filtering is not suitable for UVS.

4.3 Adaptive Measurement Noise Covariance in Uncalibrated Visual Servoing

This section discusses how adaptation of the Kalman noise covariance matrices is treated for the present research. The questions to be answered when choosing an adaptive Kalman filtering technique are "Which covariance matrices are estimated?" and "What is the estimation method?"

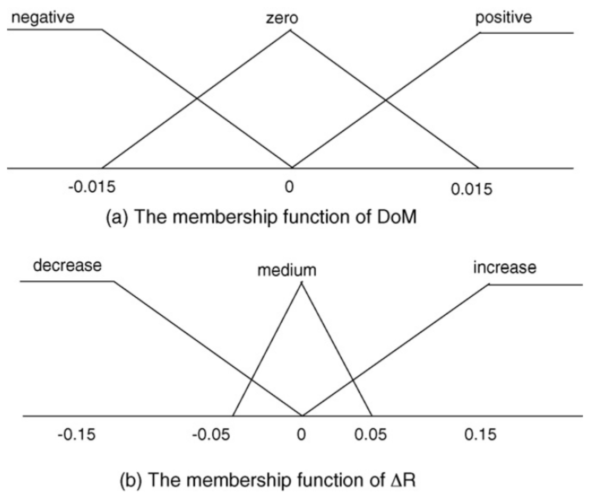


Figure 2: Membership functions of DoM and ΔR for the "adaptive fuzzy Kalman filter" of Sung et al.

Since it is difficult to differentiate between effects of errors in Q and errors in R, one of these matrices is assumed to be known. There are several reasons to choose Q as the assumed covariance. Mehra states that better results are obtained for this case than for the assumed-R-adaptive-Q case. Further, the process equation (24), for example, has fewer sources of error than the corresponding measurement equation (25) (see §3.2.7). Finally, one of the research goals is VS improvement by prioritizing data from accurate cameras over noisy cameras. Therefore, in this work Q is assumed known and R is updated adaptively. Q is set to a diagonal matrix βI, where β can be treated as a tuning parameter.

Of the four estimation categories listed at the outset, Bayesian and maximum likelihood are rejected due to excessive computational requirements. In the realm of correlation methods, the new autocovariance least-squares method is promising but relatively untested compared to the number of covariance-matching implementations. Covariance matching is the basis for many systems and it is used in the present research to estimate the measurement noise covariance.

It is common in covariance matching techniques to use ad-hoc methods for guaranteeing positive semidefinite estimates. Myers and Tapley advocate always resetting the diagonal elements of $\widehat{R}$ (and $\widehat{Q}$) to the absolute values of their estimates. While this is an instinctive step, consideration of (55) and (56) reveals it to be unsound. According to those equations

$\widehat{R}_{\nu,k}$ is negative when the estimated innovation covariance $\widehat{C}_{\nu,k}$ is less than the contribution made by the covariance of the prediction error $H_k P_{k|k-1} H_k^\mathsf{T}$ to the theoretical innovation covariance. The theoretical value is bigger than the estimate even if $R = 0$; using $R = \left|\widehat{R}_{\nu,k}\right|$ only exacerbates the problem.

The adaptation technique for this research resembles the covariance matching method of Mehra with two alterations made to (56). First, R is diagonalized by setting the off-diagonal terms to zero to improve filter stability [33] while yielding comparable accuracy as retaining the off-diagonal elements [7]. Second, a diagonal element is set equal to that of $\widehat{R}_{\nu,k}$ from (56) only if it is positive, otherwise it reverts to the value in R_0. Thus the positive definiteness of R is assured.

The result is the following update equation for the estimate of R,

$$\widehat{R}_k(i,j) = \begin{cases} \delta_{i,j}\widehat{R}_{\nu,k}(i,j), & \text{if} \quad \widehat{R}_{\nu,k}(i,j) > 0 \\ R_0(i,j), & \text{otherwise} \end{cases} \tag{57}$$

where $R_0 = \kappa I$. Like β for Q, κ can be treated as a tuning parameter. Also,

$$\widehat{R}_{\nu,k} = \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{\nu}_{k-i}\boldsymbol{\nu}_{k-i}^\mathsf{T} - H_k P_{k|k-1} H_k^\mathsf{T}$$

and

$$\boldsymbol{\nu}_k = \boldsymbol{z}_k - H_k \hat{\boldsymbol{x}}_{k|k-1}$$

4.4 Summary

An adaptive filtering technique is selected and modified to ensure positive semidefinite estimates of the measurement covariance matrix. Four categories of adaptive filtering are discussed. The first two, comprising Bayesian and maximum likelihood techniques, are disregarded because of possibly high computational loads. The third, correlation methods, have recently been improved with the ALS approach but successful applications are few compared to the fourth category which is covariance matching. Covariance matching techniques abound. Of those, scaling-factor approaches forfeit information present in the innovation covariance and fuzzy logic methods require system experience to design the membership functions that is not compatible with uncalibrated visual servoing.

Some covariance matching techniques adapt only R, while others update estimates of both Q and R. It is reported in the literature that accurately estimating both Q and R can be difficult and that better results are obtained when assuming Q and adapting R than for the opposite scenario. Furthermore, a goal of the research being to apply relative weighting to cameras it is logical to adapt R. Chapter 5 presents a decentralized version of the Kalman filter and discusses another reason to adapt R and not Q. Chapter 7 gives data from simulations comparing VS performance with adaptive and non-adaptive Kalman filter methods.

CHAPTER V

DECENTRALIZED ADAPTIVE KALMAN FILTER FOR UVS

Results from experimentation and simulation (Chapters 8 and 7) demonstrate that the Kalman filter based control law of Chapter 3 yields improved tracking performance over the conventional Gauss-Newton approach in the presence of noise, and that further gains are to be had by employing the adaptive technique of §4.3. However, a goal of the present research is multiple-camera visual servoing to deliver system survivability and improved tracking over one camera. This chapter realizes that goal by extending the adaptive Kalman filter control law to multiple-camera systems. For comparison, the Gauss-Newton control law (7) is also extended for multiple cameras.

5.1 Data Fusion

Using multiple sensors to monitor a state has several advantages. The system can be robust to failure of any given sensor whereas a single-sensor system collapses. Likewise, inaccurate readings from one of many sensors may have an insignificant effect on system performance. Finally, sensor selection becomes more flexible as multiple units can act more effectively than a single one. This means that cheaper and less precise sensors can be employed en masse compared to a single expensive sensor.

When multiple sensors are used to monitor the state of a system they provide varied and sometimes inconsistent information. In order to maximize the potential benefits of multiple sensors the information from each must be combined in an intelligent way. This is referred to as *data fusion.* In multisensor fusion the data from many measurement devices are combined to form a coherent description of the state being observed. Fusion methods are either quantitative or qualitative. Quantitative approaches are numerically based, often stochastically, with particle filtering and Kalman filtering as examples. Qualitative techniques use abstractions to describe the state being measured, such as expert systems.

The quantitative sensor fusion method utilized herein for uncalibrated visual servoing is referred to as the *decentralized Kalman filter*. More generally, there are three arrangements for sensors and filters: *centralized*, *hierarchical*, and *decentralized*.

In the *centralized* architecture information from all sensors is directly incorporated in a single fusion process, see Figure 3. For example, it is implemented for a KF with zeroth order state space representation (see §3.2.4.1) using C cameras by stacking the measurement vector and observation matrix of (25),

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{f}_k^{(1)} \\ \boldsymbol{f}_k^{(2)} \\ \vdots \\ \boldsymbol{f}_k^{(C)} \end{bmatrix} \qquad \text{and} \qquad H_k = \begin{bmatrix} J_k^{(1)} \\ J_k^{(2)} \\ \vdots \\ J_k^{(C)} \end{bmatrix} \tag{58}$$

A centralized Kalman filter provides an optimal estimate, but at the cost of fragility (the system is dependent a single fusion site) and computational load (multiplication and inversion of a $2mC \times 2mC$ matrix, where $2m$ is the number of image feature coordinates being tracked in each camera).

The imposition of such a high computational burden is addressed by a *hierarchical* architecture, exemplified in Figure 4. This design employs low-level fusion nodes to generate estimates with sensor subsets. Higher-level processors combine these estimates and possibly provide feedback to lower-level nodes. A hierarchical architecture retains a central processor but its load is less than for a centralized architecture. The hierarchical scheme lends itself to systems with sets of different sensor types. For example, in a mobile robot one node could fuse data from sonar sensors and another could generate an estimate using data from optical ranging devices, with both estimates then fused at the central fusion site.

The *decentralized* architecture (Figure 5) mitigates problems associated with centralized and hierarchical designs. There is a processing node associated with each sensor that generates a *local estimate* of the state, which is shared among nodes according to the topology of the system and fused at every node. This improves system survivablity (it's not dependent on a global filter), reduces communication requirements (for a centralized multisensor system the state variable is generally of smaller dimension than a measurement),

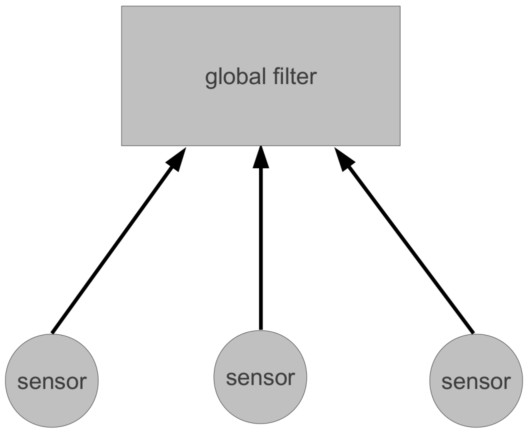


Figure 3: Centralized architecture example

and parallelizes the computation.

5.2 Gauss-Newton Control Law for Multiple Cameras

The traditional multi-camera approach in the UVS literature is a centralized architecture using two cameras. The image-plane coordinate vectors are thus concatenated,

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}^{(1)} \\ \boldsymbol{y}^{(2)} \end{bmatrix}$$

where $\boldsymbol{y}^{(1)}$ is the feature coordinate vector from camera 1 and $\boldsymbol{y}^{(2)}$ is from camera 2. Assuming that the same number of features are tracked in both cameras, the dimension of $\boldsymbol{y}$ is double that of single-camera VS. The Jacobian grows accordingly,

$$J = \begin{bmatrix} J^{(1)} \\ J^{(2)} \end{bmatrix}$$

where $J^{(1)}$ is the Jacobian mapping robot velocity in joint space to the velocity of the end-effector in the image space of camera 1 and $J^{(2)}$ corresponds to camera 2.

The control law for this system is often obtained via Guss-Newton (7). For C cameras the control law is

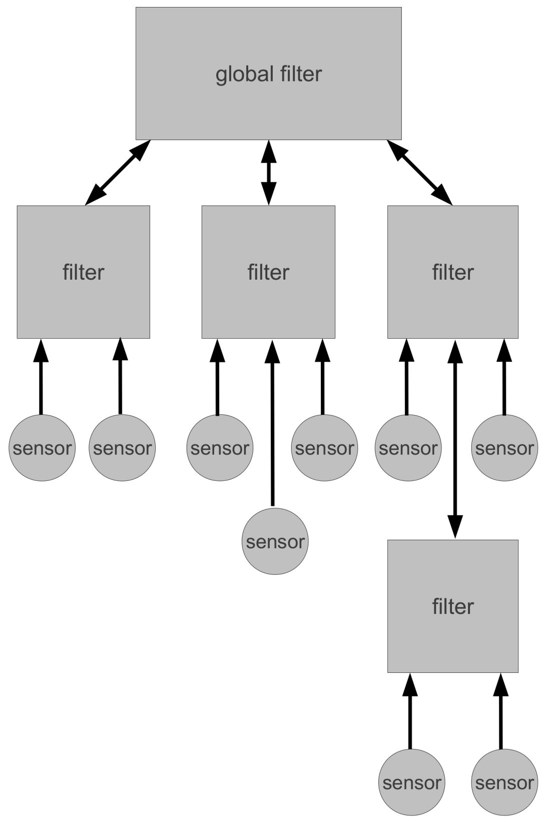


Figure 4: Hierarchical architecture example

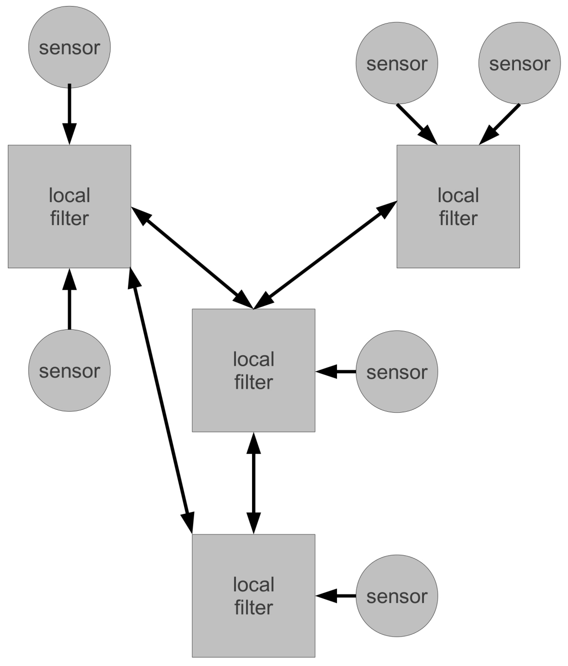


Figure 5: Decentralized architecture example

$$\widehat{\boldsymbol{\phi}}_k = \left(\begin{bmatrix} J^{(1)} \\ J^{(2)} \\ \vdots \\ J^{(C)} \end{bmatrix}^\mathsf{T} \begin{bmatrix} J^{(1)} \\ J^{(2)} \\ \vdots \\ J^{(C)} \end{bmatrix} \right)^{-1} \begin{bmatrix} J^{(1)} \\ J^{(2)} \\ \vdots \\ J^{(C)} \end{bmatrix}^\mathsf{T} \begin{bmatrix} \boldsymbol{f}^{(1)} \\ \boldsymbol{f}^{(2)} \\ \vdots \\ \boldsymbol{f}^{(C)} \end{bmatrix} \tag{59}$$

where $\boldsymbol{\phi} \equiv \boldsymbol{\theta} - \boldsymbol{\theta}^*$ per §3.1.2 and $\boldsymbol{f}^{(i)}$ is the error in the image plane of camera i,

$$\boldsymbol{f}_k^{(i)} = \boldsymbol{y}_k^{(i)} - \boldsymbol{y}_k^{*(i)}$$

This constitutes a control law that minimizes $\boldsymbol{f}^\mathsf{T}\boldsymbol{f}$, the norm of concatenated of error vectors from all cameras. This method requires no tuning or robot-target motion assumptions but it treats data from all cameras as equally valid and does not provide noise filtering.

Another reported approach for UVS with multiple cameras is hierarchical, applying Gauss-Newton (7) to data from each camera then averaging these solutions,

$$\widehat{\boldsymbol{\phi}}_k = \frac{1}{C} \sum_{i=1}^{C} \left(\hat{J}_k^{(i)\mathsf{T}} \hat{J}_k^{(i)} \right)^{-1} \hat{J}_k^{(i)\mathsf{T}} \boldsymbol{f}_k^{(i)} \tag{60}$$

A weighted version is employed by Gao and Su [27] in UVS of a mobile robot using three fixed cameras (but only two at a time). For weights $\eta^{(i)}$, $i = 1 \ldots C$ the resulting control law is

$$\widehat{\boldsymbol{\phi}}_k = \frac{1}{\sum_{i=1}^{C} \eta^{(i)}} \sum_{i=1}^{C} \eta^{(i)} \left(\hat{J}_k^{(i)\mathsf{T}} \hat{J}_k^{(i)} \right)^{-1} \hat{J}_k^{(i)\mathsf{T}} \boldsymbol{f}_k^{(i)}$$

Camera weighting is based on the proximity of the target image features to the optical axis, the camera resolution, and the focal length. The resolution and focal length are unavailable for UVS, therefore this approach is not used in the present research. The unweighted version (60) is not used either because in general it does not satisfy the error squared objective function (3). Since the Kalman filter satisfies (3) the centralized Gauss-Newton approach is used for comparison in experiments and simulations.

5.2.1 Dealing with sensor failure

The probability of either the robot end-effector or the target becoming occluded or leaving the field of view of at least one camera increases with the number of cameras. So does the chance of hardware or communication failure. Therefore the ability to handle interruption of camera data is an important feature for a robust multiple-camera VS system.

Jägersand et al. [37] describe a method of truncating the stacked Jacobian in the event of data loss. When information from a camera becomes unavailable its corresponding rows in J and $\boldsymbol{f}$ are removed, and the control law is thence carried out as before. For example, if the target becomes occluded in camera 2 then (59) becomes

$$\widehat{\boldsymbol{\phi}}_k = \left(\begin{bmatrix} J^{(1)} \\ J^{(3)} \\ \vdots \\ J^{(C)} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} J^{(1)} \\ J^{(3)} \\ \vdots \\ J^{(C)} \end{bmatrix} \right)^{-1} \begin{bmatrix} J^{(1)} \\ J^{(3)} \\ \vdots \\ J^{(C)} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{f}^{(1)} \\ \boldsymbol{f}^{(3)} \\ \vdots \\ \boldsymbol{f}^{(C)} \end{bmatrix}$$

If data from a new camera becomes available during servoing or if a new camera is added to the system then this information can be added to the control loop after first allowing the system to generate a suitably accurate estimate of its Jacobian matrix. Since this is an iterative technique (see Algorithm 4, Chapter 6) it can require more iterations for a new camera to be incorporated than for one returning from an offline state where the last calculated estimate of J can be stored. This old value is used as $\hat{J}_{k-1}$ in Algorithm 4 when the camera returns to operation. For a new camera $\hat{J}_{k-1}$ can be initialized arbitrarily with proper dimension since the estimate eventually converges [59]. Some criterion must be chosen for judging when the estimate of a camera Jacobian has reached a sufficient level of fidelity to be included in the control loop.

One approach is to compare the control law generated by the erstwhile set of cameras using the concatenated GN (59) with that of the new camera using the individual GN (7). When the similarity between the directions of the two control vectors reaches some threshold value then the Jacobian estimate and error vector from the new camera are concatenated with the rest. For example, given a system with C established cameras and a new one $C+1$

the current Jacobian estimate is

$$\hat{J}_k = \begin{cases} \begin{bmatrix} \hat{J}_k^{(1)} \\ \vdots \\ \hat{J}_k^{(C+1)} \end{bmatrix}, & \text{if} \quad \dfrac{\widehat{\boldsymbol{\phi}}_k^{(1\ldots C)}}{\left\| \widehat{\boldsymbol{\phi}}_k^{(1\ldots C)} \right\|} \cdot \dfrac{\widehat{\boldsymbol{\phi}}_k^{(C+1)}}{\left\| \widehat{\boldsymbol{\phi}}_k^{(C+1)} \right\|} > \alpha \\ \begin{bmatrix} \hat{J}_k^{(1)} \\ \vdots \\ \hat{J}_k^{(C)} \end{bmatrix}, & \text{otherwise} \end{cases} \tag{61}$$

where $\widehat{\boldsymbol{\phi}}_k^{(1\ldots C)}$ is from (59) using cameras $1\ldots C$ and $\widehat{\boldsymbol{\phi}}_k^{(C+1)}$ is from (7) using camera $C+1$. The threshold value α is some constant less than unity.

This method is implemented with multiple cameras in simulation and experimentally for comparison with a *decentralized adaptive Kalman filter* control law introduced below.

5.3 Decentralized Kalman Filter Derivation

Rao and Durrant-White [65] derive a decentralized Kalman filter algorithm for a fully connected system, one where each local estimate is broadcast to every other node (see Figure 6). The estimate after fusion is the same at each node, is called the *global estimate*, and is equivalent to a centralized Kalman filter estimate. The derivation follows and application is made to the Kalman filter VS control method in §5.4.

First the prediction and update equations for the local Kalman filter are presented. Next the *information matrix* is introduced and its update equation is derived as background for the subsequent derivation of the fusion equations that yield the same optimal estimate as a centralized Kalman filter.

5.3.1 Local equations

Each local filter in a decentralized system computes local state and error covariance predictions then updates them using the local measurement $\boldsymbol{z}^{(i)}$. The estimated information is then communicated to other nodes (and other local estimates are received) for the fusion step. In this section the local prediction and update equations are presented.

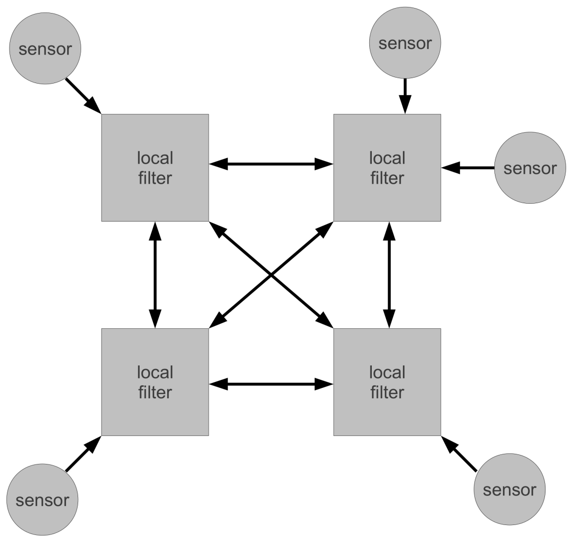


Figure 6: Fully connected decentralized architecture example

For a centralized filter the measurement vector $\boldsymbol{z}$, the measurement model H, and the measurement noise v are partitioned by the C sensors,

$$\boldsymbol{z}_k = \left[\boldsymbol{z}_k^{(1)\mathsf{T}} \quad \boldsymbol{z}_k^{(2)\mathsf{T}} \quad \ldots \quad \boldsymbol{z}_k^{(C)\mathsf{T}}\right]^{\mathsf{T}} \tag{62}$$

$$H_k = \left[H^{(1)\mathsf{T}} \quad H^{(2)\mathsf{T}} \quad \ldots \quad H^{(C)\mathsf{T}}\right]^{\mathsf{T}} \tag{63}$$

$$\boldsymbol{v}_k = \left[\boldsymbol{v}_k^{(1)\mathsf{T}} \quad \boldsymbol{v}_k^{(2)\mathsf{T}} \quad \ldots \quad \boldsymbol{v}_k^{(C)\mathsf{T}}\right]^{\mathsf{T}} \tag{64}$$

The partitions of the noise vector are assumed to be uncorrelated so

$$\mathrm{E}\left[\boldsymbol{v}_k^{\mathsf{T}}\boldsymbol{v}_k\right] = R_k = \mathrm{diag}\left\{R_k^{(1)},\, R_k^{(2)},\, \ldots\,,\, R_k^{(C)}\right\}$$

The process model F and process noise covariance Q are the same for every node in a fully-connected decentralized Kalman filter. The prediction step

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k|k-1}^{(i)} &= F_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1}^{(i)} \\
P_{k|k-1}^{(i)} &= F_{k-1}P_{k-1|k-1}^{(i)}F_{k-1}^{\mathsf{T}} + Q_k
\end{aligned}$$

is performed locally before communication with other nodes. Here $\hat{\boldsymbol{x}}^{(i)}_{k-1|k-1}$ and $P^{(i)}_{k-1|k-1}$ are the global estimates at time $k-1$. They are the same at every node of a fully-connected decentralized system so

$$\begin{aligned}\hat{\boldsymbol{x}}^{(i)}_{k|k-1} &= \hat{\boldsymbol{x}}^{(j)}_{k|k-1}\\ P^{(i)}_{k|k-1} &= P^{(j)}_{k|k-1}\end{aligned}$$

for all i and j.

For filters with identical predictions the local correction step is

$$\begin{aligned}K^{(i)}_k &= P_{k|k-1}H^{(i)\mathsf{T}}_k\left(H^{(i)}_k P_{k|k-1} H^{(i)\mathsf{T}}_k + R^{(i)}_k\right)^{-1}\\ \tilde{\boldsymbol{x}}^{(i)}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + K^{(i)}_k\left(\boldsymbol{z}^{(i)}_k - H^{(i)}_k \hat{\boldsymbol{x}}_{k|k-1}\right) && (65)\\ \tilde{P}^{(i)}_{k|k} &= \left(I - K^{(i)}_k H^{(i)}_k\right)P_{k|k-1} && (66)\end{aligned}$$

where $(\tilde{\cdot})$ indicates an estimate that is based on the current measurement of only the i-th node.

After the local estimates are computed communication takes place between all nodes. Fusion of the entire set is carried out at each node. The fusion equations are derived in the following section. First consider the following equality for diagonal matrix B and $m \times 1$ column vectors $\boldsymbol{a}$ and $\boldsymbol{c}$.

$$\boldsymbol{a}^{\mathsf{T}} B \boldsymbol{c} = \sum_{i=1}^{m} \boldsymbol{a}(i)^{\mathsf{T}} B(i,i)\boldsymbol{c}(i) \tag{67}$$

This also holds for a block diagonal matrix and two block vectors such as the partitioned R_k, H_k, and $\boldsymbol{z}_k$ above. It might then seem acceptable to obtain a global estimate by summing the updates from all sensors as,

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \sum_{i=1}^{C} K^{(i)}_k\left(\boldsymbol{z}^{(i)}_k - H^{(i)}_k \hat{\boldsymbol{x}}_{k|k-1}\right) \tag{68}$$

since K_k and $\boldsymbol{\nu}_k \equiv \boldsymbol{z}_k - H\boldsymbol{x}_{k|k-1|}$ are $C \times 1$ block column vectors. Doing so is not possible however, due to the fact that the local innovations $\boldsymbol{\nu}_k^{(i)} \equiv \boldsymbol{z}_k^{(i)} - H_k^{(i)}\hat{\boldsymbol{x}}_{k|k-1}$ are correlated between different sensors. This is because every local innovation contains the same prediction $\hat{\boldsymbol{x}}_{k|k-1}$ and is visible in the off-diagonal terms of the theoretical innovation covariance

$$C_\nu = HPH^\mathsf{T} + R = \begin{bmatrix} H^{(1)} \\ H^{(2)} \\ \vdots \\ H^{(C)} \end{bmatrix} P \begin{bmatrix} H^{(1)} \\ H^{(2)} \\ \vdots \\ H^{(C)} \end{bmatrix}^\mathsf{T} + \begin{bmatrix} R^{(1)} & 0 & \cdots & \\ 0 & R^{(2)} & & \\ \vdots & & \ddots & \\ & & & R^{(C)} \end{bmatrix}$$

$$= \begin{bmatrix} H^{(1)}PH^{(1)\mathsf{T}} + R^{(1)} & H^{(1)}PH^{(2)\mathsf{T}} & \cdots & \\ H^{(2)}PH^{(1)\mathsf{T}} & H^{(2)}PH^{(2)\mathsf{T}} + R^{(2)} & & \\ \vdots & & \ddots & \\ & & & H^{(C)}PH^{(C)\mathsf{T}} + R^{(C)} \end{bmatrix}$$

Since the expected innovation covariance C_ν of the centralized filter is not block diagonal (and the local innovations are correlated) then (68) is an impermissible method of unstacking the Kalman equations [54]. A different fusion method is therefore required, one that properly unstacks the centralized equations using (67).

5.3.2 Information matrix

The decentralized Kalman filter fusion equations use the *information matrix* update equation. The information matrix is the inverse of the covariance matrix, and its product with the state vector is called the information state vector. When a Kalman filter uses the information matrix it is referred to as an Information filter.

The Information filter is algebraically equivalent to the Kalman filter assuming Gaussian noise and optimal estimation, but has advantages in a decentralized architecture [54]. It requires inversion of smaller matrices and is simpler to initialize. Mutambara [54] derives a non-fully connected decentralized Information filter that provides the same results as a centralized filter and minimizes communication.

The transformation from the Kalman filter to the Information filter requires the Woodbury matrix identity.

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1}+VA^{-1}U\right)^{-1}VA^{-1} \tag{69}$$

Inverting the Kalman covariance update equation $P_{k|k} = (I - K_k H_k)\, P_{k|k-1}$ with the definition of the Kalman gain K,

$$\begin{aligned}
P_{k|k}^{-1} &= \left[(I - K_k H_k)\, P_{k|k-1}\right)^{-1} \\
&= \left(P_{k|k-1} - K_k H_k P_{k|k-1}\right)^{-1} \\
&= \left[P_{k|k-1} - P_{k|k-1} H_k^{\mathsf{T}} \left(H_k P_{k|k-1} H_k^{\mathsf{T}} + R_k\right)^{-1} H_k P_{k|k-1}\right]^{-1}
\end{aligned}$$

and using

$$\begin{aligned}
A &= P_{k|k-1} \\
U &= -P_{k|k-1} H_k^{\mathsf{T}} \\
C &= \left(H_k P_{k|k-1} H_k^{\mathsf{T}} + R_k\right)^{-1} \\
V &= H_k P_{k|k-1}
\end{aligned}$$

in (69) yields the information matrix update

$$\begin{aligned}
P_{k|k}^{-1} &= P_{k|k-1}^{-1} + P_{k|k-1}^{-1} P_{k|k-1} H_k^{\mathsf{T}} \times \\
&\quad \left(H_k P_{k|k-1} H_k^{\mathsf{T}} + R_k - H_k P_{k|k-1} P_{k|k-1}^{-1} P_{k|k-1} H_k^{\mathsf{T}}\right)^{-1} H_k P_{k|k-1} P_{k|k-1}^{-1} \\
&= P_{k|k-1} + H_k^{\mathsf{T}} \left(H_k P_{k|k-1} H_k^{\mathsf{T}} + R_k - H_k P_{k|k-1} H_k^{\mathsf{T}}\right)^{-1} H_k \\
&= P_{k|k-1}^{-1} + H_k^{\mathsf{T}} R_k^{-1} H_k
\end{aligned} \tag{70}$$

From §5.3.1, the local covariance update equation (66) becomes the local form of the information matrix update (70)

$$\tilde{P}_{k|k}^{(i)^{-1}} = P_{k|k-1}^{-1} + H_k^{(i)\mathsf{T}} R_k^{(i)^{-1}} H_k^{(i)} \tag{71}$$

5.3.3 Derivation of fusion equations

Now that the information matrix update equation is available valid fusion equations can be developed. The partitioning of (62–64) yields

$$H_k^{\mathsf{T}} R_k^{-1} H_k = \sum_{i=1}^{C} H_k^{(i)\mathsf{T}} R_k^{(i)^{-1}} H_k^{(i)} \tag{72}$$

Rearranging (70) and (71),

$$\begin{aligned} H_k^{\mathsf{T}} R_k^{-1} H_k &= P_{k|k}^{-1} - P_{k|k-1}^{-1} && (73) \\ H_k^{(i)\mathsf{T}} R_k^{(i)^{-1}} H_k^{(i)} &= \tilde{P}_{k|k}^{(i)^{-1}} - P_{k|k-1}^{-1} && (74) \end{aligned}$$

and substituting in (72) gives the centralized error covariance update

$$P_{k|k}^{-1} = P_{k|k-1}^{-1} + \sum_{i=1}^{C} \left(\tilde{P}_{k|k}^{(i)^{-1}} - P_{k|k-1}^{-1} \right)$$

with the equivalent decentralized form

$$P_{k|k}^{(i)^{-1}} = P_{k|k-1}^{-1} + \sum_{i=1}^{C} \left(\tilde{P}_{k|k}^{(i)^{-1}} - P_{k|k-1}^{-1} \right)$$

Here the *variance error information* is introduced as $E^{(i)} \equiv \tilde{P}_{k|k}^{(i)^{-1}} - P_{k|k-1}^{-1}$.

Fusion of the error covariance estimate is complete. To obtain a fusion equation for the state estimates the covariance update equation (21) is rearranged as

$$K_k = H_k^{-1} - P_{k|k} P_{k|k-1}^{-1} H_k^{-1} \tag{75}$$

Next, premultiplying the update equation of the state estimate (20) by $P_{k|k}^{-1}$ and using K from (75) gives

$$\begin{aligned}
P_{k|k}^{-1}\hat{\boldsymbol{x}}_{k|k} &= P_{k|k}^{-1}\hat{\boldsymbol{x}}_{k|k-1} + P_{k|k}^{-1}\left(H_k^{-1} - P_{k|k}P_{k|k-1}^{-1}H_k^{-1}\right)\left(\boldsymbol{z}_k - H_k\hat{\boldsymbol{x}}_{k|k-1}\right) \\
&= P_{k|k}^{-1}\hat{\boldsymbol{x}}_{k|k-1} + \left(P_{k|k}^{-1}H_k^{-1} - P_{k|k-1}^{-1}H_k^{-1}\right)\left(\boldsymbol{z}_k - H_k\hat{\boldsymbol{x}}_{k|k-1}\right) \\
&= P_{k|k}^{-1}\hat{\boldsymbol{x}}_{k|k-1} + P_{k|k}^{-1}H_k^{-1}\boldsymbol{z}_k - P_{k|k}^{-1}\hat{\boldsymbol{x}}_{k|k-1} - P_{k|k}^{-1}H_k^{-1}\boldsymbol{z}_k + P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1} \\
&= \left(P_{k|k}^{-1} - P_{k|k-1}^{-1}\right)H_k^{-1}\boldsymbol{z}_k + P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1}
\end{aligned}$$

and using (73) yields

$$H_k^{\mathsf{T}}R_k^{-1}\boldsymbol{z}_k = P_{k|k}^{-1}\hat{\boldsymbol{x}}_{k|k} - P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1} \tag{76}$$

In like fashion, combining (65), (66), and (74) produces the local form of (76)

$$H_k^{(i)\mathsf{T}}R_k^{(i)^{-1}}\boldsymbol{z}_k^{(i)} = \tilde{P}_{k|k}^{-1}\tilde{\boldsymbol{x}}_{k|k}^{(i)} - P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1} \tag{77}$$

The final step in obtaining a fusion equation for the local state estimates is made possible by the partitioning (62–64), which results in

$$H_k^{\mathsf{T}}R_k^{-1}\boldsymbol{z}_k = \sum_{i=1}^{C} H_k^{(i)\mathsf{T}}R_k^{(i)^{-1}}\boldsymbol{z}_k^{(i)} \tag{78}$$

Substituting (78) into (76) and (77) and rearranging gives the state fusion equation for a central node,

$$\hat{\boldsymbol{x}}_{k|k} = P_{k|k}\left[P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1} + \sum_{i=1}^{C}\left(\tilde{P}_{k|k}^{(i)^{-1}}\tilde{\boldsymbol{x}}_{k|k}^{(i)} - P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1}\right)\right]$$

with the equivalent decentralized form

$$\hat{\boldsymbol{x}}_{k|k}^{(i)} = P_{k|k}^{(i)}\left[P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1}^{(i)} + \sum_{i=1}^{C}\left(\tilde{P}_{k|k}^{(i)^{-1}}\tilde{\boldsymbol{x}}_{k|k}^{(i)} - P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1}\right)\right]$$

Here the *state error information* is introduced as $e^{(i)} \equiv \tilde{P}_{k|k}^{(i)^{-1}}\tilde{\boldsymbol{x}}_{k|k}^{(i)} - P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1}$.

5.3.4 Summary of decentralized Kalman equations

C local filters begin each iteration with the same global error covariance and state estimates. Every filter has identical local predictions since they share a process model F and also Q.

Each filter updates the predictions via the local measurement $\boldsymbol{z}^{(i)}$ to generate the local estimates $\tilde{\boldsymbol{x}}_{k|k}^{(i)}$ and $\tilde{P}_{k|k}^{(i)}$. The proper method for fusing these estimates is derived in §5.3.3. The steps are summarized equations.

Every node begins an iteration by generating predictions from the common previous estimates

$$\boxed{\begin{aligned} P_{k|k-1} &= F_{k-1}P_{k-1|k-1}F_{k-1}^{\mathsf{T}} + Q_k \qquad (79) \\ \hat{\boldsymbol{x}}_{k|k-1} &= F_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} \qquad (80) \end{aligned}}$$

With the measurement $\boldsymbol{z}_k^{(i)}$ each node updates the predictions

$$\boxed{\begin{aligned} K_k^{(i)} &= P_{k|k-1}H_k^{(i)\mathsf{T}}\left(H_k^{(i)}P_{k|k-1}H_k^{(i)\mathsf{T}} + R_k^{(i)}\right)^{-1} \qquad (81) \\ \tilde{P}_{k|k}^{(i)} &= \left(I - K_k^{(i)}H_k^{(i)}\right)P_{k|k-1} \qquad (82) \\ \tilde{\boldsymbol{x}}_{k|k}^{(i)} &= \hat{\boldsymbol{x}}_{k|k-1} + K_k^{(i)}\left(\boldsymbol{z}_k^{(i)} - H_k^{(i)}\hat{\boldsymbol{x}}_{k|k-1}\right) \qquad (83) \end{aligned}}$$

The variance error information $E^{(i)}$ and state error information $e^{(i)}$ are transmitted from each node to all the other nodes

$$\boxed{\begin{aligned} E^{(i)} &= \tilde{P}_{k|k}^{(i)^{-1}} - P_{k|k-1}^{-1} \qquad (84) \\ e^{(i)} &= \tilde{P}_{k|k}^{(i)^{-1}}\tilde{\boldsymbol{x}}_{k|k}^{(i)} - P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1} \qquad (85) \end{aligned}}$$

Fusion is then carried out at each node

$$\boxed{\begin{aligned} P_{k|k}^{-1} &= P_{k|k-1}^{-1} + \sum_{j=1}^{C} E^{(j)} \qquad (86) \\ \hat{\boldsymbol{x}}_{k|k} &= P_{k|k}\left(P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1} + \sum_{j=1}^{C} e^{(j)}\right) \qquad (87) \end{aligned}}$$

A fully-connected system results in identical estimates at every node.

5.4 Decentralized Adaptive Kalman Filter For VS Control

The previous section describes a method for fully-connected decentralized estimation with the Kalman filter. This technique allows for sensor failure since missing data are left out of the summations in fusion equations (86) and (87).

Besides improving the survivability of a multi-camera UVS system, another goal is to improve tracking performance by prioritizing measurements from the most precise cameras. To that end the adaptive Kalman filter of Chapter 4 is combined with the decentralized Kalman filter. The following reasons help elucidate the choice between providing adaptation of the covariance matrix Q or R.

For the fully-connected decentralized Kalman filter Q is identical at all nodes; there are global values for the process model F and for the process error covariance Q. Adaptation of Q can provide performance improvements by increasing the accuracy of the process noise description but does not give any means of data weighting. On the other hand $R^{(i)}$ is a measure of confidence for the i-th sensor. Adapting $R^{(i)}$ generates a description of the error $\boldsymbol{v}_i$ associated with camera i. This provides a means of data weighting since the Kalman equations recursively solve weighted least squares with the block diagonal R_k as part of the weighting matrix. Adaptation of $R_k^{(i)}$ transpires before the update equations (81–83) according to the covariance matching technique of (57).

5.4.1 Alternate examples of weighted data fusion

From the literature, four examples are presented of weighted data fusion. The first two are similar to the local R adaptation method used here in that they adapt individual measurement covariances. The third and fourth examples employ somewhat ad-hoc methods for fusion of local estimates, methods that make no claims as to optimality.

Escamilla-Ambrosio and Mort [22] use a fuzzy inference system similar to that of Sung et al. [74] (see §4.2.2) for adaptation of the local measurement covariance $R^{(i)}$. The DKF equations of §5.3.4 are used in order to obtain an estimate, resulting in what they term a fuzzy logic-based adaptive decentralized Kalman filter. By means of simulation, they compare this approach with two other fusion methods and find that it yields the most accurate estimates.

Subramanian et al. [73] make use of a "fuzzy logic enhanced Kalman filter" for autonomous navigation of citrus grove rows. Theirs is a centralized Kalman filter, which provides estimates identical to those of a fully-connected decentralized Kalman filter. It does not allow for failure of any sensors. They assume a diagonal R as,

$$R = \begin{bmatrix} \sigma_{Xc} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{Xl} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\theta c} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\theta imu} & 0 \\ 0 & 0 & 0 & 0 & \sigma_V \end{bmatrix}$$

where the different variances σ are determined experimentally prior to operation. The fuzzy system is used to alter the variances for a machine vision system and a LIDaR sensor (σ_{Xc} and σ_{Xl}) in order to dictate on which sensor the filter relies for its estimate of the vehicle's lateral position relative to the trees on either side of a grove row. However, in their fuzzy inference system the resulting variance can have one of only three values: 0, 0.5, or 1.

Drolet et al. [20] perform data fusion for the positioning of an underwater vehicle. Their technique is to create a bank of Kalman filters, each of which employs a different combination of the available sensors. The fusion tool is an averaging of the low-level state estimates weighted according to the local information matrices as,

$$\hat{\boldsymbol{x}}_{k|k} = \left(\sum_{j=1}^{C} \tilde{P}_{k|k}^{(j)^{-1}} \right)^{-1} \sum_{j=1}^{C} \tilde{P}_{k|k}^{(j)^{-1}} \tilde{\boldsymbol{x}}_{k|k}^{(j)}$$

except that their application estimates only a one-dimensional state and so they report the following fusion equation

$$\hat{\boldsymbol{x}}_{k|k} = \frac{\sum_{j=1}^{C} \tilde{\boldsymbol{x}}_{k|k}^{(j)} \,/\, \tilde{P}^{(j)}}{\sum_{j=1}^{C} 1 \,/\, \tilde{P}^{(j)}} \tag{88}$$

This estimate might then be fused with another, making this an example of the hierarchical architecture. The fusion equation (88) has the problem of only being applicable to one dimensional problems. Furthermore, the optimality of this method is not investigated.

The fourth example is from Stavnitzky and Capson [71] who generate local estimates of the Cartesian position and orientation of an object in a multi-camera quasi-uncalibrated visual servoing system. There is no Kalman filter in their system. Instead, they employ a nonlinear variation of Gauss-Newton optimization to get local estimates. The global estimate is a weighted linear sum of the local values where each weight comes from an ad-hoc measure of confidence in the local estimate. They verify the approach through experimentation with two cameras.

They are concerned with the task of mating two parts, one attached to the robot end-effector and the other stationary. Pose estimates of both parts are generated at every time step. This allows the system to simultaneously estimate the mating pose of the mobile part (based on that of the stationary object) and the Jacobian relating motion of the end-effector in Cartesian space to its motion in joint space. The system is not uncalibrated because even though it estimates the Jacobian without needing the robot kinematic parameters, the 6-D pose estimation uses camera parameters and CAD models of the object.

5.4.2 Dealing with sensor failure

In the event that data from a sensor in the decentralized adaptive Kalman filter (DAKF) become unavailable then the corresponding local filter is suspended and the summations in the fusion equations (87) and (86) exclude that data. If the sensor comes back online then its local filter is resumed, it receives the latest global estimate and calculates its local predictions and estimates, but it is not returned to the control loop for W time steps (the window size for the approximated innovation covariance $\widehat{C}_\nu$ of (53)).

5.5 Summary

Three sensor fusion architectures are discussed. An example of each is applied to visual servoing.

The traditional multi-camera VS paradigm employs a centralized architecture and the Gauss-Newton method for the concatenated matrices as (59). It satisfies the objective function (3) for the concatenated error vector and requires no tuning or robot-target motion assumptions but it treats data from all cameras as equally valid and does not provide noise filtering. A method of handling camera failure with this scheme is described.

A hierarchical version of the Gauss-Newton control law with camera weighting is discussed. The weighting method is deemed unsuitable for UVS and the hierarchical fusion equation does not satisfy (3). The centralized Gauss-Newton method is therefore chosen as the benchmark for the Kalman filter approach developed in this research.

Decentralized Kalman equations are derived and the adaptive technique of §4.3 is applied to the local filters to yield a multi-camera visual servoing method that is robust to sensor failure and provides statistically meaningful camera weighting in an uncalibrated setting. The decentralized Kalman filter provides global estimates that are equivalent to the (optimal) estimates of a centralized filter and has the benefits of parallelized processing and protection against sensor failure.

Chapter 6 provides a DAKF algorithm implementing the methods of Chapters 3–5 and describes the setup used for visual servoing experiments and simulations that employ these methods and also the traditional Gauss-Newton method.

CHAPTER VI

ALGORITHM AND SETUP FOR EXPERIMENTS AND SIMULATIONS

The decentralized adaptive Kalman filter based visual servoing control method developed in Chapters 3–5 is supported through simulation and experimentation and compared to the traditional Gauss-Newton approach. Results of simulations and experiments are presented in Chapters 7 and 8. The present chapter discusses the methods and means for performing these trials.

Visual servoing pseudo-code with the zeroth order KF formulation (24) and (25) is provided in Algorithm 1. The other three KF formulations of Chapter 3 can be implemented with minor changes to the algorithm.

The physical setup for experiments is described in this chapter. For simulations this setup is idealized using the robot manipulator's kinematic model and a pinhole camera model, see §7.1. The simulations differ from experiments in that they implement GN and all four of the KF formulations, whereas the experiments employ GN and just the zeroth order KF formulation (24) and (25). Also, more camera scenarios are tested in simulation than in experiments, varying the camera parameters (both intrinsic and extrinsic) and the number of cameras.

6.1 Algorithm Summary

The following system parameters are used in the visual servoing algorithm:

- a — robot degrees of freedom
- C — number of cameras
- λ — remembering ratio for Jacobian estimation $\hat{J}$
- N — window size for estimating the innovation covariance

- Q — process covariance matrix
- R_0 — initial measurement covariance matrix
- μ — maximum allowable norm for a joint offset command to the robot
- $\Delta\theta_{\text{init}}$ — displacement to use for measuring the initial Jacobian estimate

The matrices $Q = \beta I$ and $R_0^{(i)} = \kappa I$ for camera i with β and κ as tuning parameters per §4.3. The remembering ratio for the Jacobian-estimation algorithm $\lambda = 0.95$. The sample size for R adaptation is $N = 12$.

Algorithm 1 presents pseudo-code for an uncalibrated visual servoing system using a decentralized adaptive Kalman filter with zeroth order state space representation. Several subroutines are presented in Algorithms 2–6.

Algorithm 2, a subroutine for generating initial Jacobian estimates $\hat{J}_0^{(i)}$, is not required for servoing. The local Jacobians can be initialized as arbitrary matrices with proper dimension and the online estimator for $J^{(i)}$ eventually converges [59] but in the process the robot is liable to move erratically. For safety concerns with the physical robot and because the purpose of the present research is not to evaluate Jacobian estimation schemes, Algorithm 2 is carried out prior to servoing.

The subroutine of Algorithm 3 imposes the joint offset command limitation discussed in §3.2.5. Algorithm 4 implements the iterative Jacobian estimation technique of Piepmeier [59]. The local measurement error covariance matrix is updated via the covariance matching technique of §4.3 in Algorithm 5, which verifies that a statistically significant number of innovation samples are available before adapting $R^{(i)}$. The final subroutine, shown in Algorithm 6, computes the local variance error information and state error information (see §5.3.3) for fusion.

6.2 Setup for Experiments

A Windows-based computer (PC) acts as the visual-servoing controller. It extracts image feature coordinates from the camera data and receives robot joint coordinates from the robot

Algorithm 1 Visual servoing algorithm using DAKF control law

```
1: function MAIN($a$, $C$, $N$, $\lambda$, $P_0^{(i)}$, $Q$, $R_0^{(i)}$, $\mu$, $\Delta\theta_{\text{init}}$)
2:     $\left(\hat{J}_0^{(1)} \dots \hat{J}_0^{(C)}\right)$ = initializeJacobians($a, C, \Delta\theta_{\text{init}}$)
3:     for $i = 1 \to C$ do
4:         get $\boldsymbol{f}_0^{(i)}$
5:         $\boldsymbol{f}_0$ = concatenate$\left(\boldsymbol{f}_0, \boldsymbol{f}_0^{(i)}\right)$        ▷ Append to global error vector
6:         $\hat{J}_0$ = concatenate$\left(\hat{J}_0, \hat{J}_0^{(i)}\right)$        ▷ Append to global Jacobain
7:         $covarianceSamples^{(i)}$ = new list        ▷ Each filter has a list of up to $N$ samples
8:     end for
9:     get $\boldsymbol{\theta}_0$
10:    $\hat{\boldsymbol{x}}_{0|0} = \left(J_0^{\mathsf{T}} J_0\right)^{-1} J_k^{\mathsf{T}} \boldsymbol{f}_0$        ▷ Initial state estimate from Gauss-Newton
11:    moveRobotTowardsTarget($\hat{\boldsymbol{x}}_{0|0}$, $\mu$)        ▷ Command robot towards estimate
12:    $k = 1$
13:    while servoing do        ▷ Main loop
14:        get $\boldsymbol{\theta}_k$
15:        $\hat{\boldsymbol{x}}_{k|k-1} = \hat{\boldsymbol{x}}_{k-1|k-1}$        ▷ Global state prediction
16:        $P_{k|k-1} = P_{k-1|k-1} + Q$        ▷ Global error covariance prediction
17:        for $i = 1 \to C$ do        ▷ Go through each local filter
18:            if camera "i" data is available then
19:                get $\boldsymbol{y}_k^{(i)}$ and $\boldsymbol{y}_k^{*(i)}$
20:                $\left(J_k^{(i)}, D_k^{(i)}\right)$ = updateJEstimate$\left(\boldsymbol{y}_k^{(i)}, \boldsymbol{y}_{k-1}^{(i)}, \boldsymbol{\theta}_k, \boldsymbol{\theta}_{k-1}, \hat{J}_{k-1}^{(i)}, D_{k-1}^{(i)}\right)$
21:                $\boldsymbol{z}_k^{(i)} = \boldsymbol{y}_k^{(i)} - \boldsymbol{y}_k^{*(i)}$
22:                $\boldsymbol{\nu}_k^{(i)} = \boldsymbol{z}_k^{(i)} - J_k^{(i)} \hat{\boldsymbol{x}}_{k|k-1}^{(i)}$
23:                $covarianceSamples^{(i)}$.add$\left(\boldsymbol{\nu}_k^{(i)} \boldsymbol{\nu}_k^{(i)\mathsf{T}}\right)$
24:                $R_k^{(i)}$ = updateREstimate$\left(covarianceSamples^{(i)}, J_k^{(i)}, R_0, P_{k|k-1}, N\right)$
25:                if ($covarianceSamples^{(i)}$.length $= N$ or $k < N$ ) then
26:                    $\left(E^{(i)}, e^{(i)}\right)$ = getErrorInformation$\left(\hat{J}_k^{(i)}, P_{k|k-1}, R_k^{(i)}, \hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{z}_k^{(i)}\right)$
27:                    $e \overset{+}{=} e^{(i)}$
28:                    $E \overset{+}{=} E^{(i)}$
29:                    $covarianceSamples^{(i)}$.remove(1)        ▷ Discard oldest sample in list
30:                end if
31:            else
32:                $covarianceSamples^{(i)}$ = new list        ▷ $R$ adaptation starts anew
33:            end if
34:        end for        ▷ End loop through every local filter
35:        $P_{k|k}^{-1} = P_{k|k-1}^{-1} + E$        ▷ Global information matrix estimate
36:        $\hat{\boldsymbol{x}}_{k|k} = P_{k|k}\left(P_{k|k-1}^{-1} \hat{\boldsymbol{x}}_{k|k-1} + e\right)$        ▷ Global state estimate
37:        moveRobotTowardsTarget($\hat{\boldsymbol{x}}_{k|k}$, $\mu$)
38:        $k \overset{+}{=} 1$
39:    end while
40: end function
```

Algorithm 2 Exploratory moves to populate initial Jacobian estimates

```
1:  function INITIALIZEJACOBIANS($a$, $C$, and $\Delta\theta_{\text{offset}}$)
2:      for $j = 1 \to a$ do
3:          for $i = 1 \to C$ do
4:              get $\boldsymbol{y}^{(i)}_{\text{home}}$
5:          end for
6:          $\boldsymbol{\theta}(j) \overset{+}{=} \Delta\theta_{\text{offset}}$                                  ▷ Jog the $j$-th joint
7:          for $i = 1 \to C$ do
8:              get $\boldsymbol{y}^{(i)}_{\text{new}}$
9:              $\hat{J}^{(i)}_0(:,j) = \frac{1}{\Delta\theta_{\text{offset}}}\left(\boldsymbol{y}^{(i)}_{\text{new}} - \boldsymbol{y}^{(i)}_{\text{home}}\right)$          ▷ Populate the $j$-th column
10:         end for
11:         $\boldsymbol{\theta}(j) \overset{-}{=} \Delta\theta_{\text{offset}}$                            ▷ Return to home position
12:     end for
13:     return $\hat{J}^{(1)}_0 \dots \hat{J}^{(C)}_0$
14: end function
```

Algorithm 3 Send (possibly limited) joint offset command to robot

```
1: function MOVEROBOTTOWARDSTARGET($\Delta\boldsymbol{\theta}_{\text{desired}}$, $\mu$)
2:     if $\|\Delta\boldsymbol{\theta}_{\text{desired}}\| < \mu$ then
3:         $\Delta\boldsymbol{\theta}_{\text{command}} = \Delta\boldsymbol{\theta}_{\text{desired}}$
4:     else
5:         $\Delta\boldsymbol{\theta}_{\text{command}} = \mu\frac{\Delta\boldsymbol{\theta}_{\text{desired}}}{\|\Delta\boldsymbol{\theta}_{\text{desired}}\|}$
6:     end if
7:     moveRobot($\Delta\boldsymbol{\theta}_{\text{desired}}$)                    ▷ Send command to robot controller
8: end function
```

Algorithm 4 Sub-routine to compute newest local Jacobian estimate

```
1: function UPDATEJESTIMATE($\boldsymbol{y}_k$, $\boldsymbol{y}_{k-1}$, $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_{k-1}$, $\hat{J}_{k-1}$, $D_{k-1}$)
2:     $\Delta\boldsymbol{y} = \boldsymbol{y}_k - \boldsymbol{y}_{k-1}$
3:     $\boldsymbol{h}_\theta = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$
4:     $\hat{J}_k = \hat{J}_{k-1} + \left(\lambda + \boldsymbol{h}_\theta^\top D_{k-1}\boldsymbol{h}_\theta\right)^{-1}\left(\Delta\boldsymbol{y} - \hat{J}_{k-1}\boldsymbol{h}_\theta\right)\boldsymbol{h}_\theta^\top D_{k-1}$
5:     $D_k = \frac{1}{\lambda}\left[D_{k-1} - \left(\lambda + \boldsymbol{h}_\theta^\top D_{k-1}\boldsymbol{h}_\theta\right)^{-1}\left(D_{k-1}\boldsymbol{h}_\theta^\top\boldsymbol{h}_\theta D_{k-1}\right)\right]$
6:     return $\hat{J}_k$, $D_k$
7: end function
```

Algorithm 5 Perform adaptation of measurement covariance matrix

```
1: function UPDATEREstimate($innovationOuterProductList$, $J_k$, $R_0$, $P_{k|k-1}$, $N$)
2:     if $innovationOuterProductList$.length $= N$ then        ▷ Check number of samples
3:         $n = \text{numrows}(R_0)$
4:         $R = 0_{n\times n}$                                    ▷ Initialize estimate
5:         $\widehat{C}_{\nu_k} = \frac{1}{N}\sum_{j=1}^{N} innovationOuterProductList.\text{get}(j)$
6:         $\widehat{R}_{\nu_k} = \widehat{C}_{\nu_k} - H_k P_{k|k-1} H_k^\intercal$
7:         for $i = 1 \to n$ do                                 ▷ $R$ to be diagonal and positive
8:             if $\widehat{R}_{\nu_k}(i,i) > 0$ then
9:                 $R(i,i) = \widehat{R}_{\nu_k}(i,i)$
10:            else
11:                $R(i,i) = R_0(i,i)$
12:            end if
13:        end for
14:        return $R$
15:    else
16:        return $R_0$                                         ▷ Not enough samples for adaptation
17:    end if
18: end function
```

Algorithm 6 Compute variance error information and state error information

```
1: function GETERRORINFORMATION($\hat{J}_k$, $P_{k|k-1}$, $R_k$, $\hat{\boldsymbol{x}}_{k|k-1}$, $\boldsymbol{z}_k$)
2:     $K_k = P_{k|k-1} J_k^\intercal \left(J_k P_{k|k-1} J_k^\intercal + R_k\right)^{-1}$
3:     $\tilde{P}_{k|k} = (I - K_k J_k)\, P_{k|k-1}$
4:     $\tilde{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + K_k \left(\boldsymbol{z}_k - J_k \hat{\boldsymbol{x}}_{k|k-1}\right)$
5:     $E = \tilde{P}_{k|k}^{-1} - P_{k|k-1}^{-1}$
6:     $e = \tilde{P}_{k|k}^{-1}\tilde{\boldsymbol{x}}_{k|k} - P_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1}$
7:     return $E$, $e$
8: end function
```

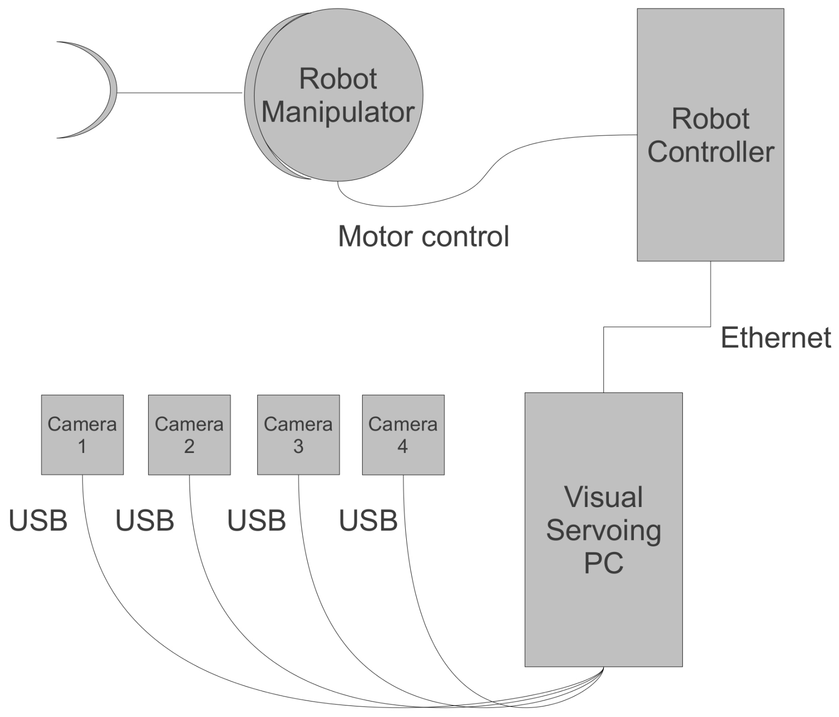


Figure 7: Overall system layout

controller via Ethernet, see Figure 7. It computes the desired joint offset and transmits this command to the robot controller, which then executes the joint-level servoing.

The robot manipulator is a KUKA KR 15 SL pictured in Figure 8. The maximum reach of its wrist point is 909 mm. It has a 15 kg payload capacity and maximum joint speeds ranging from 156 deg/s (axis 1) to 609 deg/s (axis 6). It is observed that the joint level accuracy is approximately ± 0.01 deg.

6.2.1 Image data

Four stationary cameras observe the manipulator, as shown in Figure 9. They are Chameleon USB cameras from Point Grey Research with 1280×960 pixel resolution. Image processing for all four cameras as well as visual servoing control is carried out on the PC, which has one USB bus.

The maximum frame rate of a Chameleon camera operating at its highest resolution is 18 Hz. Including image processing reduces the frequency to 10 Hz. Since the visual servoing

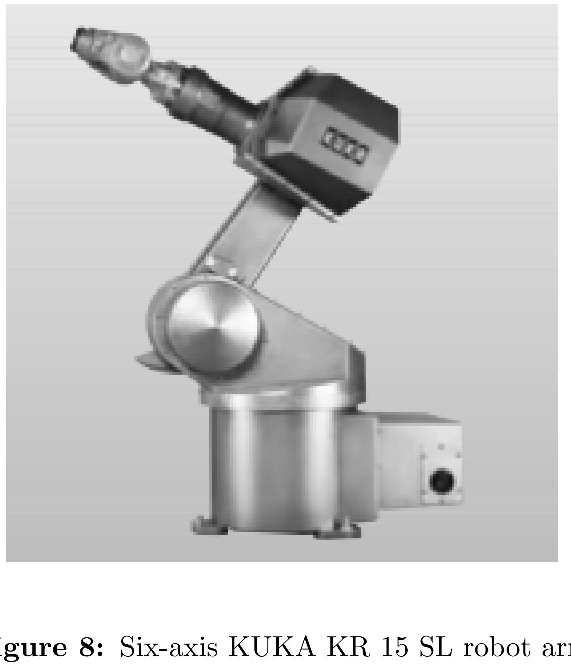


Figure 8: Six-axis KUKA KR 15 SL robot arm

computer has a single USB bus, simultaneously employing four cameras at the maximum resolution requires that the data transmission speed of each one must be lowered, resulting in an overall frequency of just 3 Hz.

The cameras are labeled Camera 1, Camera 2, Camera 3, and Camera 4 referring to their locations on the bus. One camera is placed in each of the four quadrants of the robot base coordinate system with Camera 3, Camera 2, Camera 4, and Camera 1 in quadrants I-IV, respectively as in Figure 9.

Figure 10 shows a checkerboard pattern mounted to the robot tool flange. Twelve vertices are tracked using OpenCV software and are shown circled. Figure 11 shows images from all four cameras at one instant during servoing. In each image note the feature detection and another camera in the view (circled). The image-plane coordinates of the four outermost vertices make up the image feature vector of the i-th camera

$$\boldsymbol{y}^{(i)} = [u_1^{(i)} \; v_1^{(i)} \; u_2^{(i)} \; v_2^{(i)} \; u_3^{(i)} \; v_3^{(i)} \; u_4^{(i)} \; v_4^{(i)}]^\mathsf{T}$$

The checkerboard squares are 27 mm wide and the rectangle described by the four outermost vertices is 81×52 mm (see Figure 10). The focal length for every camera is approximately 10 mm and all are placed about 2.5 m from the starting position of the robot

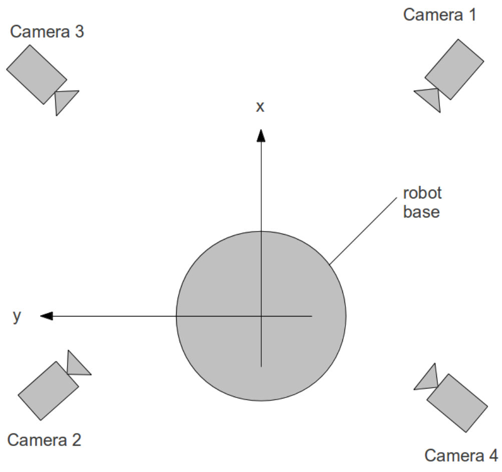


Figure 9: Fixed camera locations in experiments

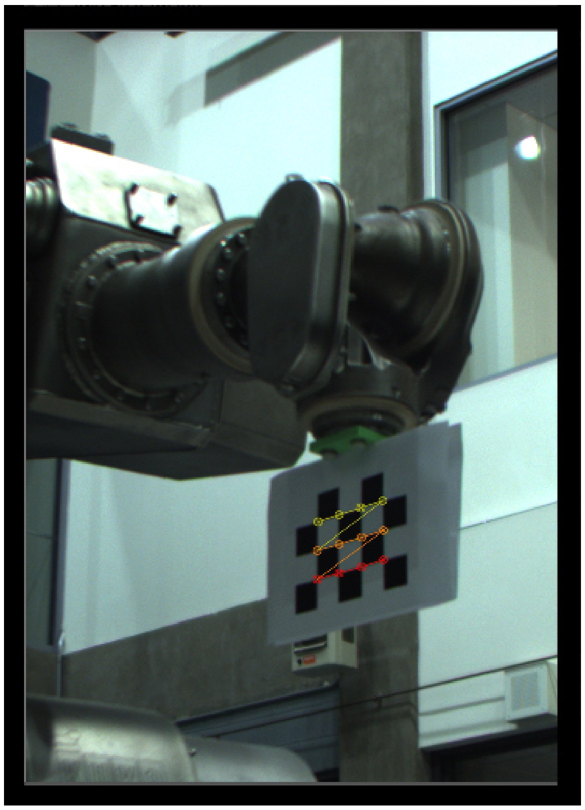


Figure 10: Image feature extraction comprises twelve checkerboard vertices.

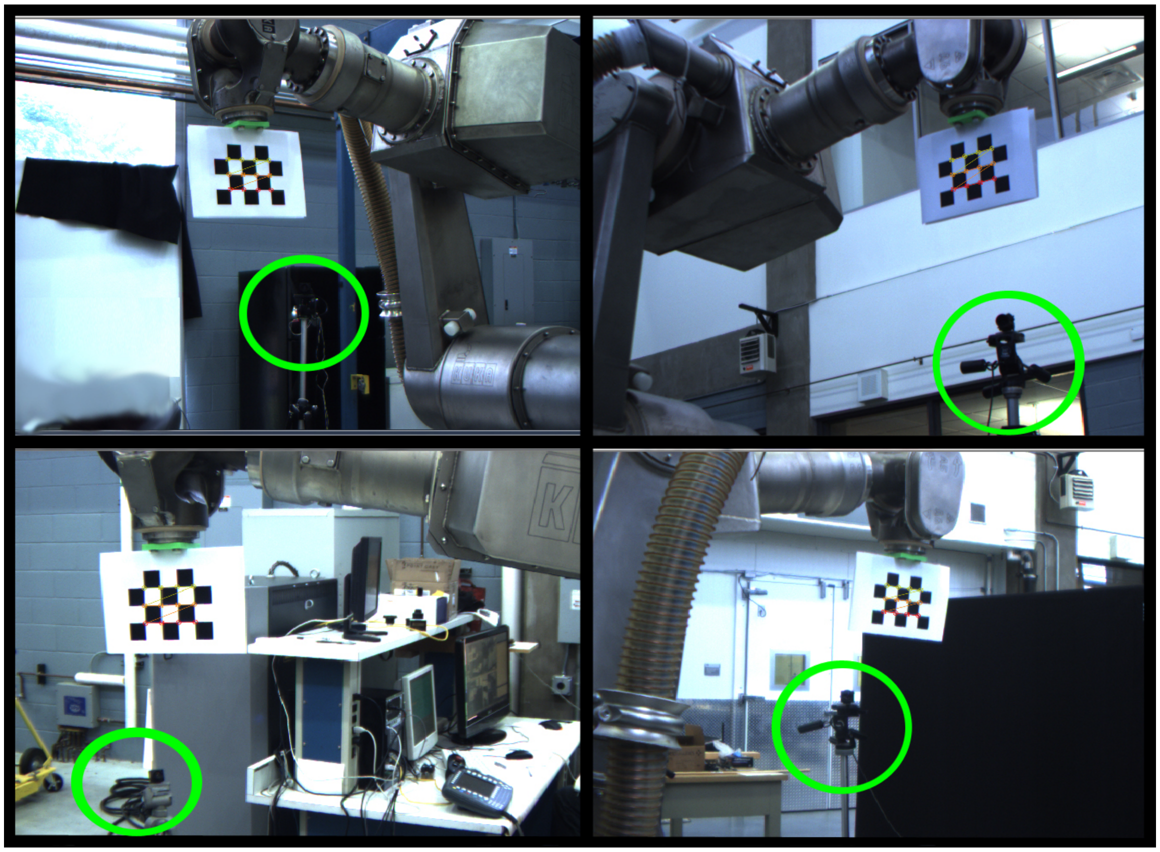


Figure 11: Images from the four cameras; counterclockwise from top left, view from Camera 3 (circled in the image is Camera 1), view from Camera 2 (circled in the image is Camera 4), view from Camera 4 (circled in the image is Camera 2), and view from Camera 1 (circled in the image is Camera 3)

tool with their optical axes pointed towards the servoing volume. The resulting dimensions of the rectangle in the images range from approximately 100×65 to 220×140 pixels during servoing. VS provides a control action so the vertices approach a target position $\boldsymbol{y}^{*(i)}$ in the image plane.

6.2.1.1 Pre-recorded target coordinates

Two different tasks are performed requiring the visual servoing system to either servo to a stationary target or to track a moving one. Each trial begins with the robot at a starting position: $\boldsymbol{\theta}_{0,\text{static}}$ for the static-target trials and $\boldsymbol{\theta}_{0,\text{moving}}$ for the moving-target trials. The target values at time $j = 0 \ldots S$ (where S is the total number of iterations in a trial) for cameras $i = 1 \ldots 4$ denoted $\boldsymbol{y}_j^{*(i)}$ are determined by recording the coordinates of the checkerboard corners from the camera while the robot is in the position $\boldsymbol{\theta}_j^*$ prior to servoing.

Pre-recorded data are used since having the robot position the target checkerboard provides accurate Cartesian and joint-space coordinates for analyzing VS performance. Doing so also means that different VS controllers are compared using identical target data. Furthermore, since using prerecorded data results in only one checkerboard in a camera view it simplifies the image processing task, which is outside the scope of this research.

Goal data for moving target The target joint positions for the moving-target case are determined by the inverse kinematics of the robot controller such that the checkerboard traces the following path:

1. Translate 320 mm in the world z direction
2. Translate 320 mm in the world y direction
3. Translate 320 mm in the world x direction
4. Translate -320 mm in the world z direction while rotating 15 degrees about an axis through the checkerboard center, parallel to world z
5. Translate -320 mm in the world y direction while rotating -15 degrees about an axis through the checkerboard center, parallel to world y

6. Translate -320 mm in the world x direction while rotating 15 degrees about an axis through the checkerboard center, parallel to world x

7. Simultaneously translate 320 mm in each of the principal directions while rotating 15 degrees about the three axes

The initial robot position $\boldsymbol{\theta}_{0,\text{moving}}$ places the checkerboard coincident with the start of this path. The linear displacement of the checkerboard center (the *tool center point*, or TCP) between each image acquisition is 20 mm for the first six sections of the path and 34.6 mm in the final section.

Goal data for static target For the static target the robot is left motionless at $\boldsymbol{\theta}^*$ and a few hundred images are acquired, enough to assure convergence during trials. The starting position $\boldsymbol{\theta}_{0,\text{static}}$ for the static-target case is offset from the goal position $\boldsymbol{\theta}^*$ by $[-11.79 \quad 18.66 \quad -37.60 \quad -5.6 \quad 24.66 \quad -6.34]^\mathsf{T}$ deg. In Cartesian space the change between these two positions for the checkerboard is a 200 mm translation in each of the three principal directions of the ground coordinate system (346.4 mm total) and a 5.63 deg rotation in three directions: first about the world z-axis, then about the new checkerboard y-axis, and finally about the new checkerboard x-axis.

6.3 Process Flow and Termination

Prior to servoing, an initial Jacobian estimate is measured for each camera according to Algorithm 2.

Once the Jacobians are initialized servoing commences following Algorithm 1 with $\boldsymbol{y}_j^{*(i)}$ from the j-th pre-recorded image and $\boldsymbol{y}_j^{(i)}$ from the real-time image of camera i. The control action comes from either the Gauss-Newton method or decentralized adaptive Kalman filter developed in Chapters 3–5. The joint offset command sent to the robot is limited as in §3.2.5. The offset norm limit is $\mu = 8$ deg for moving-target trials and $\mu = 1$ deg for a static-target.

The static- and moving-target trials also have different termination criteria. Static-target trials are concluded when either the image-plane error norm in each camera is less than some threshold value ϵ

$$\left\| \boldsymbol{f}^{(i)} \right\| < \epsilon \qquad \text{for } i = 1 \ldots C$$

or the number of iterations reaches a preset value, which is considered failure. Servoing for the moving-target lasts 112 steps, the number of images recorded for the path.

6.4 Summary

Algorithms for VS with DAKF are given in pseudo-code. An experimental setup is described that comprises a six-axis industrial robot arm, its controller, four fixed cameras, and a PC that executes image processing and Algorithms 1–6. The target data for trials are pre-recorded feature image coordinates. The Cartesian-space descriptions of the target robot positions for both moving- and static-target trials are described.

Results of from visual servoing with simulations of the described setup are presented in Chapter 7. Experimental data is given in Chapter 8.

CHAPTER VII

SIMULATION RESULTS

DAKF effectiveness is investigated. Simulations for a wide array of scenarios using the setup discussed in Chapter 6, with a select few tested experimentally in Chapter 8.

Briefly, the important results are:

- For a moving target, using more than three cameras provides little reduction in tracking error.
- No number of low-cost cameras can match the performance of two high-cost cameras for a moving target.
- With regard to positional accuracy for static targets, the point of diminishing returns is two high-cost cameras or six low-cost cameras.
- Six or more low-cost cameras perform as well as any number of high-cost cameras.
- DAKF and GN are shown to be robust to camera failure.
- If covariance Q is chosen well then the Kalman filter approach outperforms Gauss-Newton for both moving and static targets, especially as noise increases or mixtures of high- and low-cost cameras are used.
- DAKF outperforms the non-adaptive Kalman filter technique

The results are presented in the following sections:

High- and low-cost camera performance Section 7.3 details simulations to test multiple-camera effectiveness in VS. Moving and stationary target trials are performed with $C = 1 \ldots 10$ cameras. In each scenario all cameras are either high-cost or low-cost models. The relationships between performance and number of cameras is explored and comparison is made between multi-sensor systems using high-cost cameras and those using low-cost cameras. Tests are performed with three different controllers: the Gauss-Newton method and two DAKF formulations (zeroth order with input and first order without input). The latter two are used twice (window sizes N =12, 20).

Camera-failure handling Section 7.4 presents moving and stationary target simulations with four high-cost cameras, each of which has a 75% chance of failure during the trial. Three controllers are tested: GN, 0th order DAKF with input, and 1st order DAKF. All are shown to be robust to camera failure. Average moving-target tracking error is better than for two uninterrupted high-cost cameras but worse than three.

Heterogeneous camera sets Section 7.5 uses four sets of three cameras, all possible combinations of high- and low-cost cameras. Every Kalman filter formulation is tested (see §3.2.4) with adaptive and non-adaptive $R^{(i)}$. Different values of Q, $R_0^{(i)}$, and window size N are tested. Gauss-Newton is also used for comparison.

7.1 Setup for Simulations

Visual servoing simulations are performed for a six-axis manipulator with a rectangle mounted to the tool flange that is observed by fixed cameras similar to the experimental setup except: more cameras are simulated, they are placed in different locations, and the focal lengths are varied.

7.1.1 Robot model

The robot kinematic parameters are for the Kuka robot in §6.2. The feature points are the same checkerboard corners mounted on the laboratory robot in §6.2.1. Robot joint positioning errors are modeled by 0.01 degree white noise for each joint.

7.1.2 Camera model

Image data is obtained using a pinhole camera model with

$$\begin{bmatrix} u_p \\ v_p \end{bmatrix} = \frac{l}{z_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix}$$

where l is the focal length and $(u_p,\, v_p)$ are the image-plane coordinates of a point located at $[x_c \quad y_c \quad z_c]^\mathsf{T}$ relative to the pinhole camera. White noise is added to u_p and v_p in order to more closely approximate a real camera,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} p_x & 0 \\ 0 & p_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \Upsilon \begin{bmatrix} \sin(\gamma) \\ \cos(\gamma) \end{bmatrix}$$

where p_x and p_y are the horizontal and vertical pitches (pixels per unit length) of the image sensor, Υ is a random scalar that determines the magnitude of added noise, and γ is a random angle between 0 and 2π.

In a physical system feature image coordinate noise depends on several factors. Image noise plays a factor; for the same lighting conditions image noise increases with decreasing sensor size or reduction in lighting. Noise is influenced by feature type, for example, the coordinates for the centroid of a sphere are noisier than those of a checkerboard vertex due to color artifacts and lighting effects at the edge of the sphere image. Physical pixel size (or focal length, or depth) directly affects accuracy.

Sensor size (and thus image noise) and focal length both affect camera cost. The Chameleon from Point Grey Research serves as the high-cost camera model with a 10 mm focal length and a maximum additive noise magnitude of 0.5 pixels. A low-cost camera likely has a wide-angle lens and a smaller sensor and is modeled as $l = 3.95$ mm with 2 pixels of noise. The pitches are equal for both camera models because the relationship between physical pixel size and camera cost is uncertain compared to that of sensor size and camera cost. To illustrate the effect of the camera model differences on the image data Figure 12 shows coordinates from twenty images of a static target for the different camera models.

7.2 General Description of Simulations

Simulations are performed for a moving target and a stationary one. The target coordinates are the same as used for the experiments, the pre-recorded robot positions $\boldsymbol{\theta}_j^*$ of Chapter 6

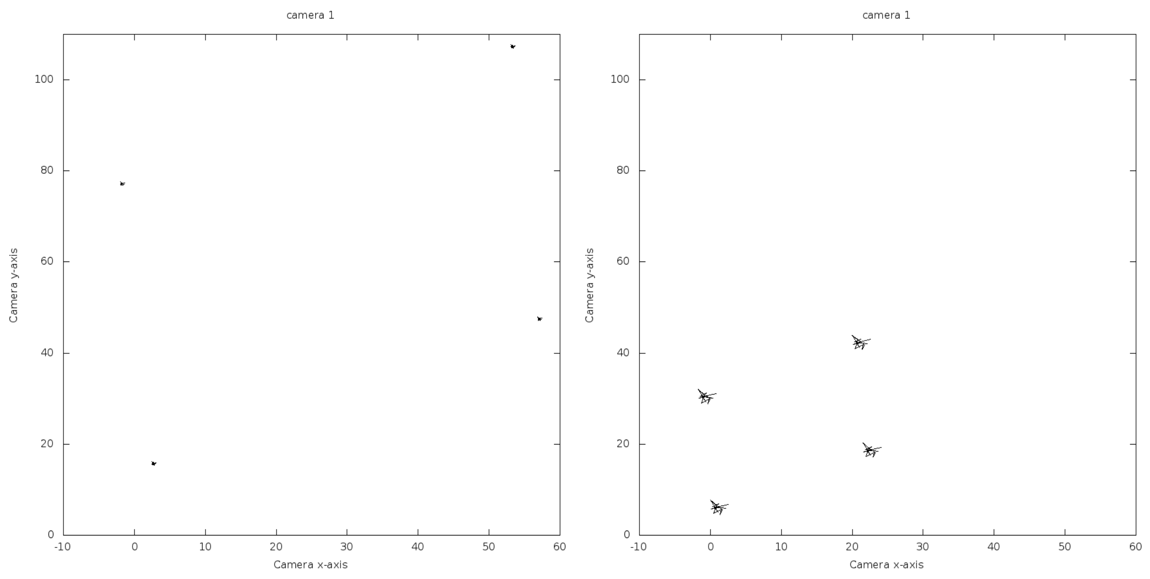


Figure 12: Simulation image data from twenty frames of a static target acquired by the high-cost camera model (left) and the low-cost model (right) placed at identical positions

are used as input to the simulated system. Various camera scenarios are tested. For each scenario a number of trials are run with a different random-number-generator seed that is used for:

- The noise added to robot joint commands
- The noise added to the pixel coordinates
- The position and orientation of each camera
- If modeling camera failures, then
  - Selecting which of the C cameras experience failure during a trial
  - Choosing the iterations for which a camera fails

When comparing controllers, the same seed is used. For example, Trial 10 of the two-camera, static-target case using zeroth order DAKF has the same noise seed as Trial 10 of the two-camera, static-target case using Gauss-Newton. Ramifications include identically positioned cameras for both trials and identical target pixel coordinates.

Moving target trials last 112 iterations. For static-target scenarios the threshold value for termination is ϵ pixels; once the image-plane error norm is less than ϵ in all C cameras servoing is terminated,

$$\text{Halt when} \quad \left\| \boldsymbol{f}_k^{(i)} \right\| < \epsilon, \quad \text{for } i = 1 \ldots C$$

7.3 Comparing High-Cost and Low-Cost Cameras for Visual Servoing

Using multiple sensors can improve state estimates. Part of the impetus behind a multi-sensor system is to get a state estimate from many low-cost sensors comparable in fidelity to an estimate from one high-cost sensor. This section details simulations intended to determine

- How VS performance changes with increasing camera number
- If numerous low-cost cameras can rival performance of a few high-cost cameras

Moving- and static-target trials are performed for twenty camera scenarios: $C_H = C = 1 \ldots 10$ and $C_L = C = 1 \ldots 10$, where C_H is the number of high-cost cameras in the system, C_L is the number of low-cost cameras, and C is the total number of cameras in the system.

7.3.1 Moving target

For a moving target twenty-five trials are performed for each of the twenty camera scenarios. The performance metric for a scenario is the Cartesian error norm for the checkerboard corners, averaged over all $T \times S$ iterations, where T is the number of trials and S is the number of iterations in a given trial

$$\bar{e}_{\text{check},T,S} = \frac{1}{T \times S} \sum_{l=1}^{T} \sum_{j=1}^{S} \left\| \boldsymbol{p}_{\text{check},l,j} - \boldsymbol{p}^*_{\text{check},j} \right\| \tag{89}$$

where $\boldsymbol{p}_{\text{check},l,j} = \begin{bmatrix} \boldsymbol{p}_{1,l,j} & \boldsymbol{p}_{2,l,j} & \boldsymbol{p}_{3,l,j} & \boldsymbol{p}_{4,l,j} \end{bmatrix}^{\mathsf{T}}$ comprises the $\boldsymbol{p}_{i,l,j}$ position vectors to the the $i = 1 \ldots 4$ corners of the checkerboard at time step j of trial l. In the current case $T = 25$ and $S = 112$.

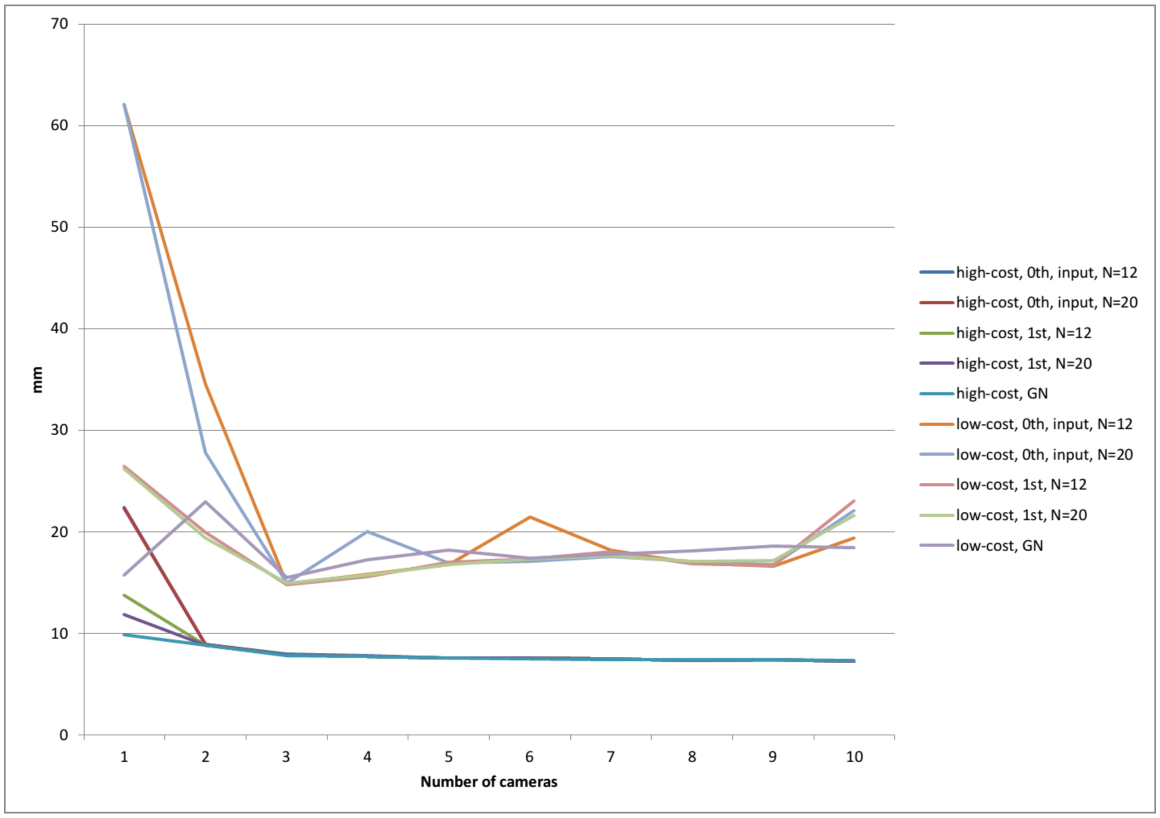


Figure 13: Average checkerboard-corners error norm using either high-cost or low-cost cameras and different control laws ($Q = 5I$ for DAKF). The top five lines represent Low-cost camera trials and the bottom five lines are for high-cost camera trials.

Five controller variations are tested in each scenario: zeroth order with input DAKF ($N = 12$), zeroth order with input DAKF ($N = 20$), first order DAKF ($N = 12$), first order DAKF ($N = 20$), and GN. The resulting values of $\bar{e}_{\text{check}}$ are plotted in Figure 13, the most pertinent aspects of which are the diminishing returns for systems with more than two high-cost cameras or three low-cost cameras and the inability of low-cost cameras to achieve errors as low as a single high-cost camera.

It is also notable that for three or more cameras the control-law utilized has little impact on performance. Below that number the zeroth order with input DAKF grossly under-performs the other controllers.

Image-plane data are plotted for a trial using the first order DAKF with two high-cost cameras in Figure 14 and the corresponding 3-D data are shown in Figure 15. The target points ($\boldsymbol{y}^{(i)*}$ for 2-D and $\boldsymbol{p}^*_{\text{check}}$ for 3-D) are shown in black while the actual points ($\boldsymbol{y}^{(i)}$

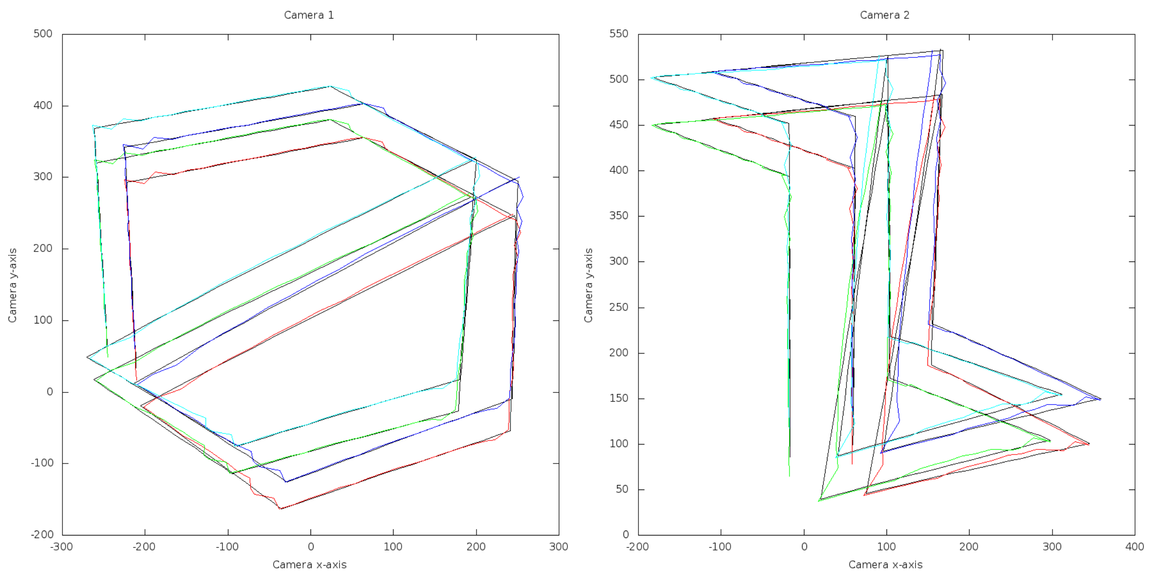


Figure 14: Image plane data from two high-cost cameras, moving target trial

and $\boldsymbol{p}_{\text{check}}$) are colored. The target image-plane data are noisy, while the 3-D target data are the pre-recorded points given by the robot forward kinematics.

Similar plots for a trial with four low-cost cameras are provided in Figures 16 and 17. The error is $\bar{e}_{\text{check},1,112} = 8.7$ mm for the trial with two high-cost cameras and $\bar{e}_{\text{check},1,112} = 15.1$ mm with four low-cost cameras. The higher error is evident when comparing the proximity of the servoed path to the ideal path in Figures 15 and 17.

Comparing Figures 14 and 16 shows the relative value of data from high- and low-cost cameras. The signal-to-noise ratio (SNR) is higher for the high-cost camera. There the rectangle size is approximately 50×100 pixels and for the low-cost camera it is about 20×30. This directly translates to a higher sensitivity to robot motions in the high-cost camera, a higher precision measurement. The problem with the low sensitivity of the low-cost camera is compounded by the higher noise in its data. It is the relatively low SNR of the low-cost camera that causes poorer tracking error than with high-cost cameras.

7.3.2 Static target

The metric for evaluating static-target trials is the average final tool center point (TCP) error

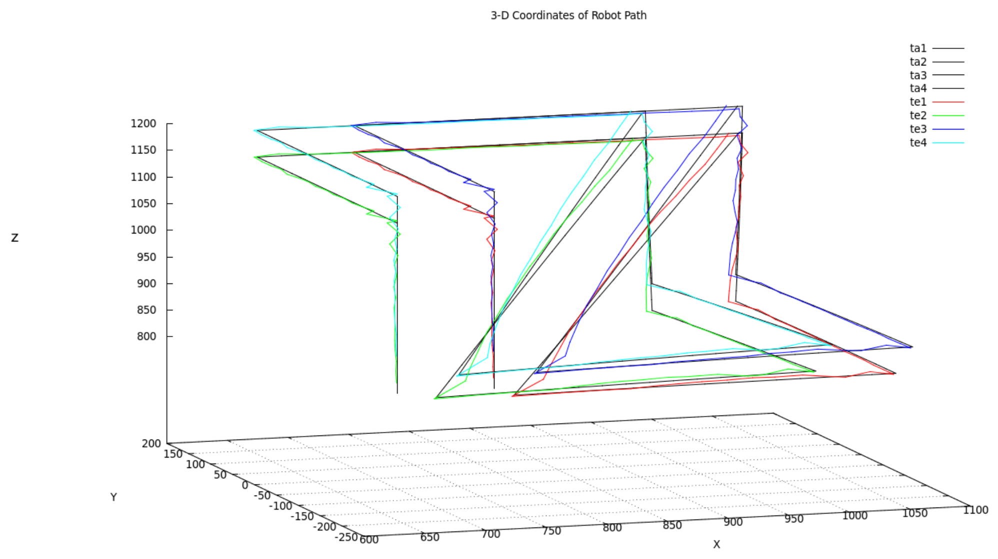


Figure 15: 3-D data using two high-cost cameras, moving target trial

$$\bar{e}_{\text{TCP},T} = \frac{1}{T}\sum_{l=1}^{T} \|\boldsymbol{p}_{\text{TCP},l,S} - \boldsymbol{p}^*_{\text{TCP}}\|$$

where $\boldsymbol{p}_{\text{TCP},l,S}$ is the position vector from the origin of the robot base frame to the center of the checkerboard at the final iteration of trial l (when $\left\|\boldsymbol{f}_k^{(i)}\right\| < \epsilon, \quad \text{for } i = 1 \dots C$). This error is a function of camera geometry at the final position so the control law has no bearing on the matter.

Meeting the termination criterion becomes more difficult with increasing camera count. For these trials the termination threshold ϵ is higher for low-cost cameras because of their noisier data. For high-cost camera trials $\epsilon = 0.5$ pixels and $\epsilon = 1$ pixel for low-cost camera trials. Resulting $\bar{e}_{\text{TCP},T}$ values for $C_H = 1 \dots 10$ and $C_L = 1 \dots 10$ are plotted in Figure 18.

With the checkerboard and cameras used the minimum positional error is about 0.5 mm. The point of diminishing returns for high-cost cameras comes at two cameras while for low-cost cameras improvement is seen up to six cameras, at which point positional error is comparable to that of high-cost camera systems. Reaching this position requires more iterations with low-cost cameras. For example, one trial with ten high-cost cameras required

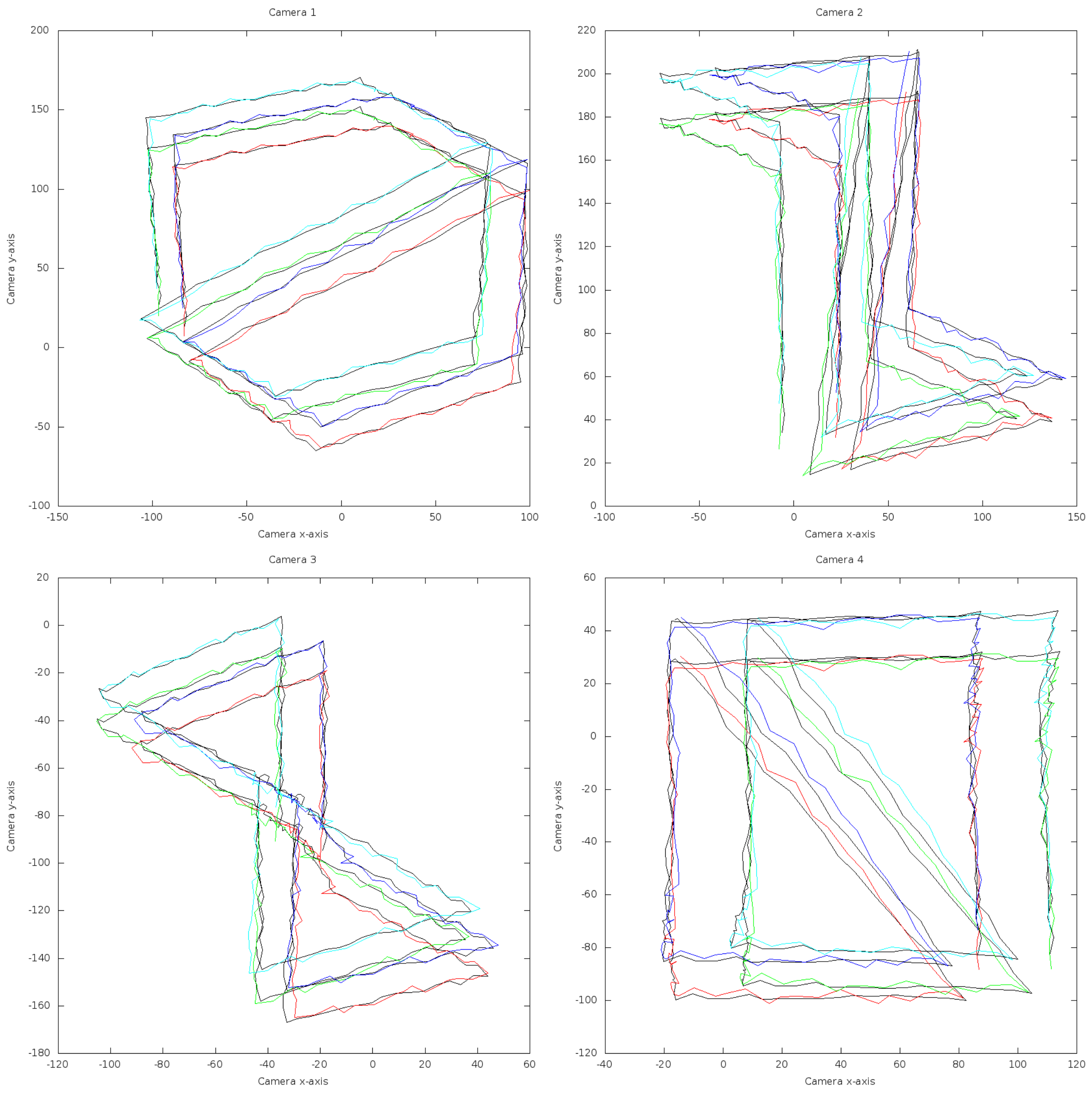


Figure 16: Image plane data from four low-cost cameras, moving target trial

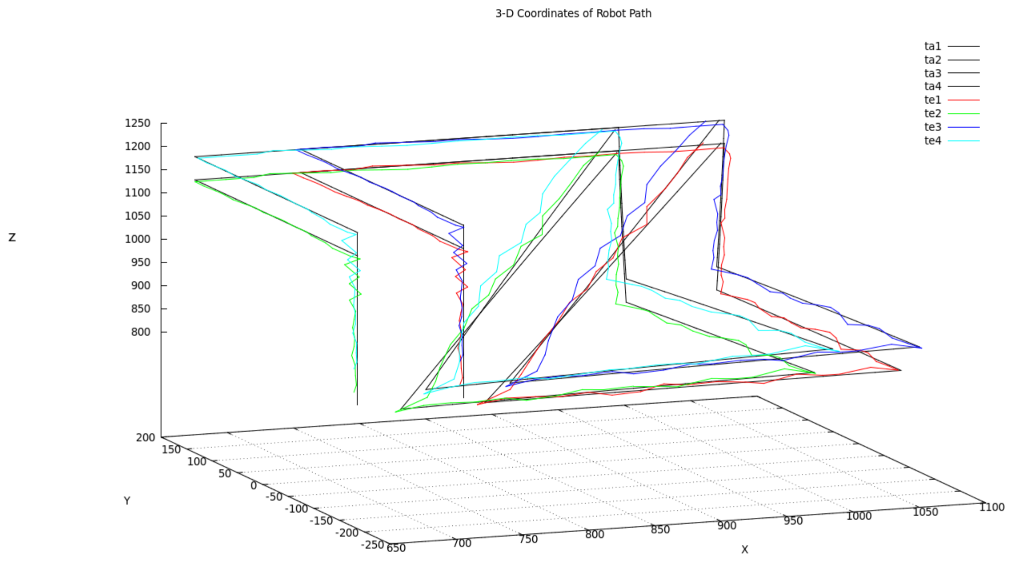


Figure 17: 3-D data using four low-cost cameras, moving target trial

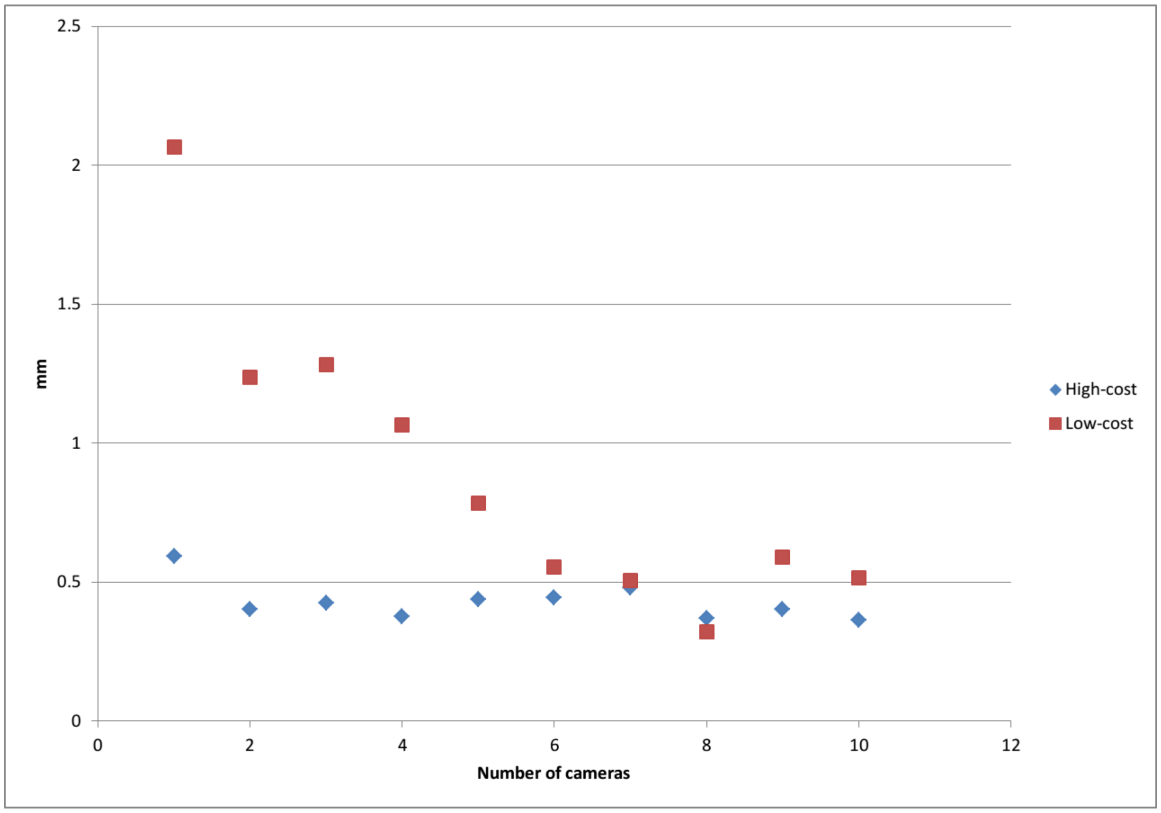


Figure 18: Average ending position error norm using either high-cost or low-cost cameras

69 steps for convergence and a trial with ten low-cost cameras took 108 steps.

For moving-target visual servoing multiple low-cost cameras are unlikely to provide tracking error comparable to two high-cost cameras. If servoing to a static target multiple low-cost cameras can be a feasible alternative to high-cost cameras.

7.4 Handling Camera Failure

To assess whether or not the methods for handling camera failure discussed in Sections 5.2.1 and 5.4.2 are functional they are utilized in both moving- and static-target simulations. Four high-cost cameras are used, each with a 75% chance of experiencing a failure episode during servoing. Each camera undergoes at most one episode of failure during a trial. Which cameras fail and during what iterations they fail are chosen randomly. For example, the following camera outages are possible for some trial:

- camera 1 — always on
- camera 2 — off for $k = 41 \ldots 51$
- camera 3 — off for $k = 1 \ldots 13$
- camera 4 — off for $k = 44 \ldots 54$

The failure episodes are identical for the control laws so that comparison is as meaningful as possible.

Two DAKF formulations as well as GN are tested. The DAKF controllers and the centralized Gauss-Newton (59) method all continue servoing through camera failure. For DAKF the window size, which determines when to reintroduce a camera to the control loop, is $N = 12$. For Gauss-Newton the camera inclusion threshold of (61) is $\alpha = 0.7$.

Seventy-five trials are performed with each controller. The average checkerboard error in the moving-target trials is shown in Table 2 for all three controllers. Also shown are $\bar{e}_{\text{check},T,112}$ values for three high-cost camera scenarios of §7.3 to evaluate the intermittent-data effect on performance. The average tracking error using four intermittent cameras is better than with two constant cameras but worse than with three. This shows that both the DAKF and GN methods provide safeguards against camera failure.

Table 2: Average checkerboard-corners error norm (mm) for different moving target, high-cost camera scenarios and different controllers. The notation for the DAKF filters is "state space representation order", "Y or N using input", "β", "κ", and "N".

C	$\text{DAKF}_{0,\text{Y},5,0.1,12}$	$\text{DAKF}_{1,\text{N},5,0.1,12}$	GN	Remarks
1	22.4	13.8	9.9	Always on
2	8.9	8.8	8.8	Always on
3	7.9	7.9	7.8	Always on
4	8.3	8.3	8.2	Intermittent

The static-target trials display a similar resilience, exemplified in Figure 19 where image plane data are plotted for one trial using zeroth order with input DAKF $Q = 5I$ $N = 12$. Image plane data are plotted as points in this figure instead of lines to make the camera outages (in cameras 1, 3, and 4) apparent. Seventy-five iterations are required for convergence in this trial.

The number of iterations to convergence with four intermittent high-cost cameras are $\bar{S} = 57.1$ for $\text{DAKF}_{0,\text{Y},5,0.1,12}$, 57.1 for $\text{DAKF}_{1,\text{N},5,0.1,12}$, and 58.6 for GN. Here the convergence speed is faster than reported in §7.3 because of a more relaxed termination threshold, $\epsilon = 10$ pixels.

The multiple-camera VS control methods of Chapter 5 are shown to provide continued servoing in the event of camera failure.

7.5 Using Heterogeneous Sets of Cameras

Though camera models in these simulations are termed "high-cost" and "low-cost" they can be thought of as high-SNR and low-SNR cameras. It is shown in §7.3 that low camera signal-to-noise ratio degrades VS performance. The simulations thus far use homogeneous camera sets (either all high-cost or all low-cost) but this can be unrealistic for unstructured environments, where camera placement options and lighting conditions vary. For given lighting and camera model, a camera far from the robot will have a lower SNR than one close. Poor lighting also degrades the SNR. Thus it is important to simulate using sets combining high- and low-cost camera models.

Moving- and static-target simulations in this section use all possible combinations of

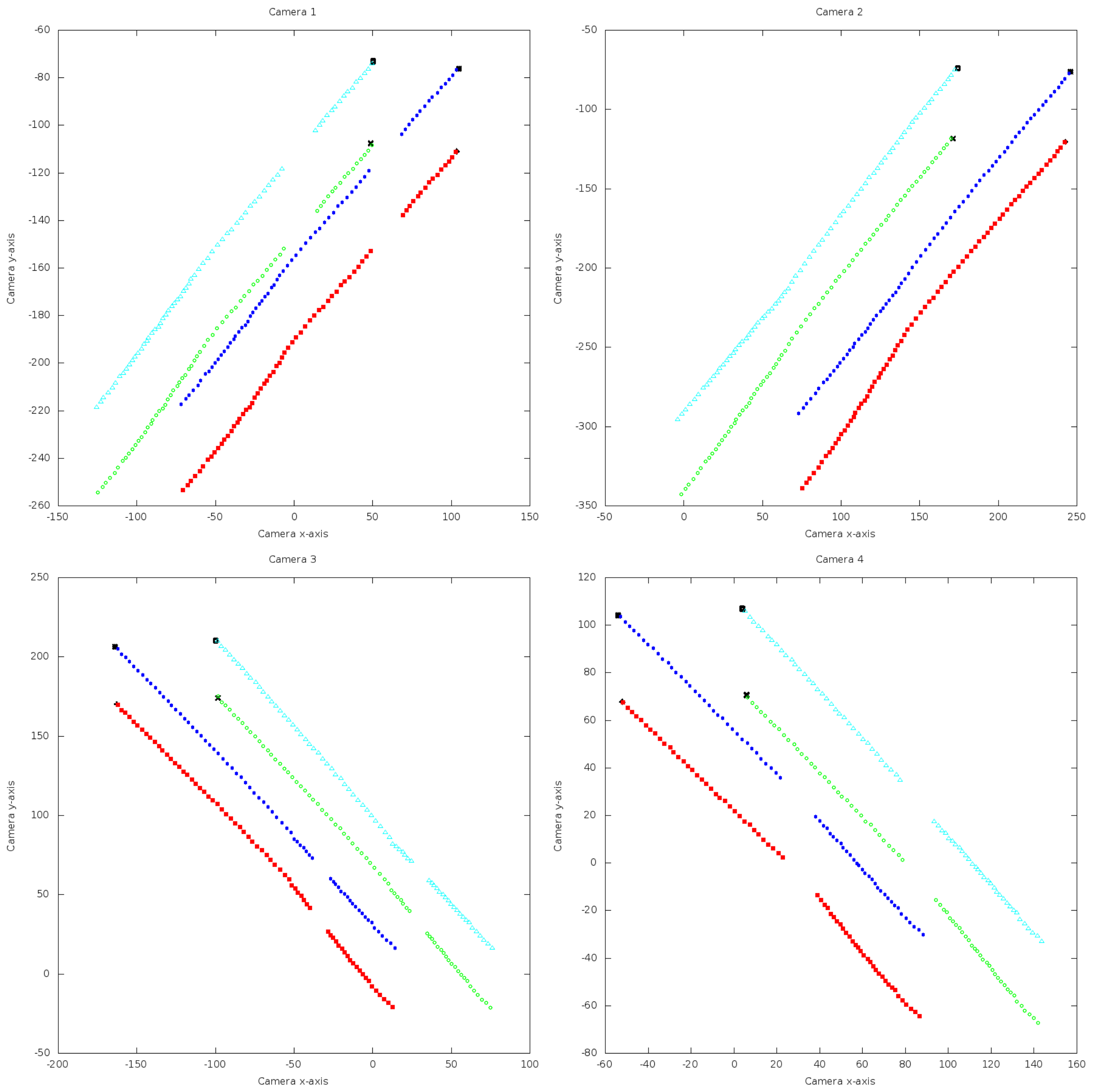


Figure 19: Image-plane coordinates using DAKF controller with four high-cost cameras, each given 75% chance of failing during servoing — static target

Table 3: Kalman filter control law variations used in heterogeneous camera sets simulations

state space order	input	β	κ	N
0, 1	with, w/o	1, 3, 5	0.1	12, 20
			1, 5	∞

high- and low-cost cameras for a three-camera system. That is, $C_H + C_L = C = 3$ with $C_H = 0 \ldots 3$. In addition to Gauss-Newton, a wider range of DAKF parameters are tested than in previous sections. Non-adaptive versions are also used. Table 3 summarizes the different Kalman filter settings used, where $Q = \beta I$ and $R_0 = \kappa I$. All combinations of parameters from the rows of that table are used. The adaptive filters use $\kappa = 0.1$ with two values of window size $N = 12$, 20. For the non-adaptive filters ($N = \infty$) two different values of κ are used, 1 and 5. One hundred trials are run for each combination of camera scenario and controller.

This section presents average performance metrics for both moving- and static-target three-camera scenarios using forty-nine controllers: Gauss-Newton and those of Table 3. The moving-target data is given first, followed by discussion, and more data for three of the best controllers are presented. Plots of the robot path in camera views and 3-D are presented for comparison between different controllers tracking a moving target. The same treatment is repeated for the static-target simulations. The section concludes with discussion of adaptive-R effectiveness and tuning considerations for DAKF.

7.5.1 Moving target

Figure 20 plots $\bar{e}_{\text{check},100,112}$ (average checkerboard-corners error over 100 moving-target trials, each lasting 112 iterations) for GN and the twenty-four non-adaptive decentralized Kalman filter formulations of Table 3. Figure 21 does the same for GN and the twenty-four DAKF controllers of Table 3. These figures show $\bar{e}_{\text{check},100,112}$ (mm) along the vertical axis and different camera scenarios along the horizontal. "HHH" stands for $C_H = 3$, "HHL" is for $C_H = 2$, "HLL" is for $C_H = 1$, and "LLL" is for $C_H = 0$.

Figure 20 shows that using input has negligible effect on the non-adaptive KF versions, as evidenced by the identical heights in the column pairs "0, N"-"0, Y" and "1, N"-"1, Y". Also, the "$Q = 5I$, $R = I$" plot shows that a large Q and small R combination performs

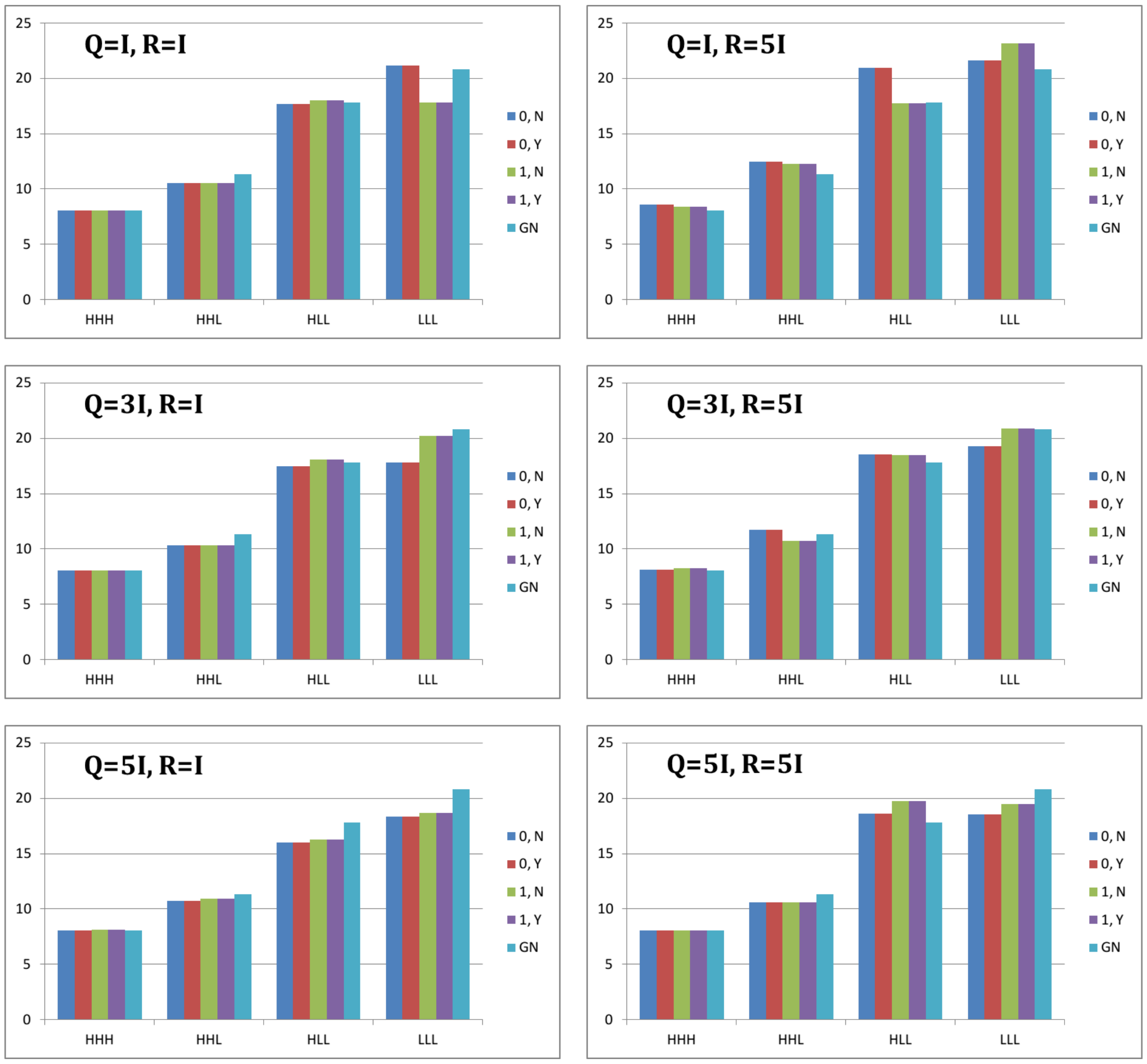


Figure 20: Average checkerboard-corners error norm using combinations of high- and low-cost cameras with GN and 24 variations of DAKF, "0" or "1" refers to the state space representation order and "Y" or "N" refers to with or without input

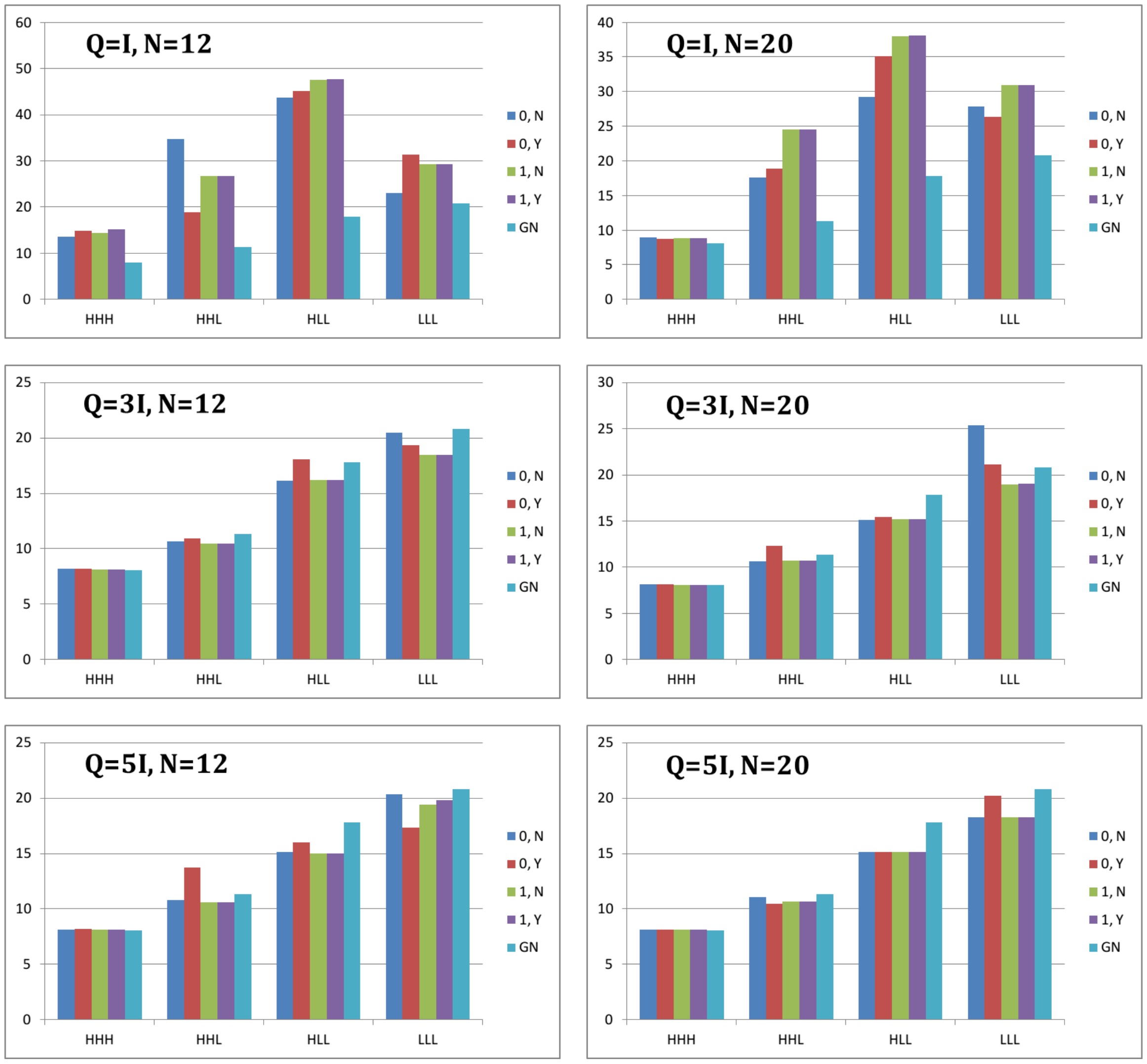


Figure 21: Average checkerboard-corners error norm using combinations of high- and low-cost cameras with GN and 24 variations of non-adaptive decentralized Kalman filter, “0” or “1” refers to the state space representation order and “Y” or “N” refers to with or without input

best, prioritizing camera data over the process model. For a system with such well chosen Q and R the non-adaptive Kalman filter formulation performs as well or better than the Gauss-Newton method in all camera scenarios, demonstrating its capacity to reject sensor noise. For poorly chosen noise covariances (for example, $Q = I$ and $R = 5I$) GN outperforms the non-adaptive Kalman filter method in all camera scenarios.

Figure 21 displays some traits of the DAKF that are similar to the non-adaptive versions and some traits that are different. Like non-adaptive KF, DAKF does poorly when Q is chosen too low and the best performances are for $Q = 5I$ cases. The "$Q = I$, $N = 12$" and "$Q = I$, $N = 20$" plots in this and Figure 20 show that too-low Q has more effect on DAKF than on the non-adaptive version. Reasons for this are explored in §7.5.3.

It is also apparent that including input affects DAKF performance more than the non-adaptive cases because the adaptive system responds to predictions via the innovation. This is seen most with the zeroth order state space representation, there are dramatic differences within some "0, N"-"0, Y" column pairs.

The larger window size $N = 20$ improves tracking error over the smaller size $N = 12$. Larger N also improves resiliency to poorly chosen Q, as shown in the changes between $Q = I$ plots and $Q = 5I$ plots for the two window sizes.

Both Figures 20 and 21 show that in general the zeroth order and first order formulations have about equal tracking error.

For all camera scenarios tested the first order DAKF with $Q = 5I$ and $N = 20$ has lower tracking error than Gauss-Newton and the non-adaptive KF variations. Table 4 gives the average error over all trials $\bar{e}_{\text{check},100,112}$, the smallest average error of all trials $\bar{e}_{\text{check, min}}$, and the largest average error of all trials $\bar{e}_{\text{check, max}}$ for all four camera scenarios with the best controller tested from each category (zero-memory, non-adaptive, and adaptive): Gauss-Newton, non-adaptive zeroth order Kalman filter with $Q = 5I$ and $R = I$, and first order DAKF with $Q = 5I$ and $N = 20$. The biggest performance gain is with DAKF for the scenario with one high-cost camera and two low-cost cameras, a 15% improvement over Gauss-Newton and 5% over non-adaptive Kalman filter.

A representative scenario is depicted for GN and a DAKF. Figure 22 shows image-plane

Table 4: Minimum, average, and maximum average checkerboard-corners error (mm) for best controllers tested from three categories. The notation for the DAKF filters is "state space representation order", "Y or N using input", "β", "κ", and "N".

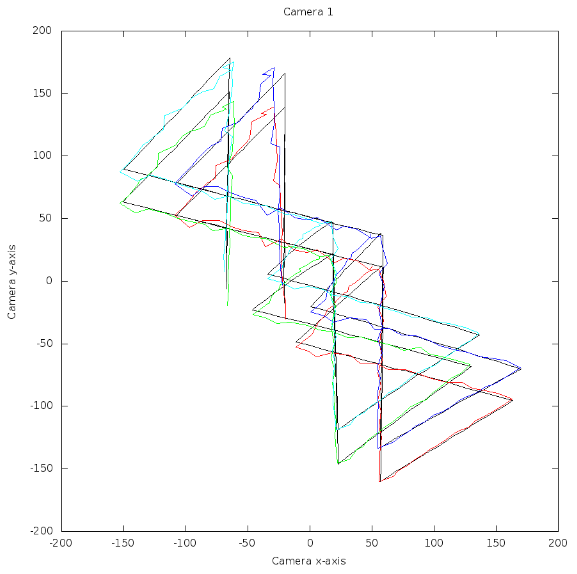
	$C_H = 3$, $C_L = 0$			$C_H = 2$, $C_L = 1$			$C_H = 1$, $C_L = 2$			$C_H = 0$, $C_L = 3$		
	$\bar{e}_{\min}$	$\bar{e}$	$\bar{e}_{\max}$	$\bar{e}_{\min}$	$\bar{e}$	$\bar{e}_{\max}$	$\bar{e}_{\min}$	$\bar{e}$	$\bar{e}_{\max}$	$\bar{e}_{\min}$	$\bar{e}$	$\bar{e}_{\max}$
GN	6.1	8.0	14.9	7.3	11.3	80.5	9.7	17.8	87.5	10.4	20.8	113.6
$\text{DAKF}_{0,\text{N},5,1,\infty}$	6.1	8.1	16.9	7.3	10.8	54.6	9.2	16.0	137.5	11.6	18.3	103.9
$\text{DAKF}_{1,\text{N},5,0.1,20}$	6.2	8.1	16.2	7.5	10.6	38.1	9.5	15.1	41.0	11.2	18.3	72.4

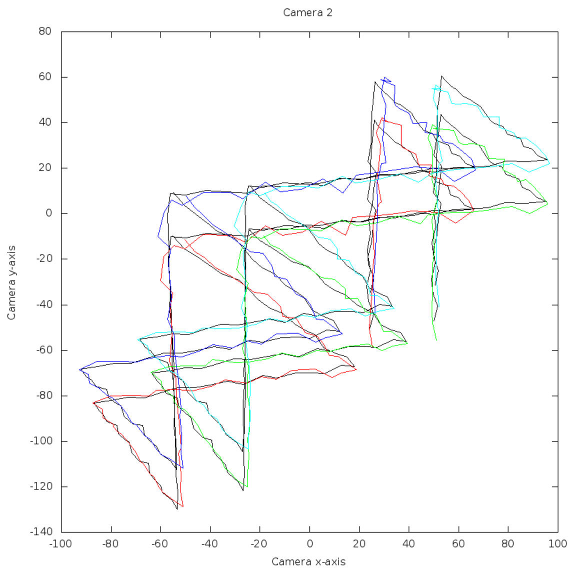
coordinates with GN and $C_H = 1$. The target robot coordinates, shown with black lines, provide a sense of the relative noisiness of Camera 1 to Cameras 2 and 3. The 3-D path is depicted in Figure 23. Similar plots are given in Figures 24 and 25 for a first order DAKF with $Q = 5I$ and $N = 20$. The paths resulting from the two controllers appear similar but the average checkerboard-corners error norm for this trial is 10% better with DAKF.

DAKF improves moving-target visual servoing by more than lowering the average error, it yields a system less sensitive to disturbances compared to GN or DKF. This is shown by the maximum error values $\bar{e}_{\text{check, max}}$ for the three controllers in Table 4: for scenarios with at least one low-cost camera the maximum average error with DAKF is at most 64% of the maximum average error with Gauss-Newton. Sensitivity to variations can be seen in Figures 26 and 27, which plot the $\bar{e}_{\text{check}}$ for each of the 100 trials with first-order DAKF and GN. The same number of cameras is used in each trial but the placemenet varies, as do the image noise and joint noise. Figure 26 is for trials with three high-cost cameras and the two controllers have nearly equivalent performance. Figure 27, on the other hand, is for trials with one high-cost camera and two low-cost cameras and the two controllers have similar average performance but GN results in higher outliers than DAKF.

7.5.2 Static target

Identical camera scenarios are used ($C_H = 0 \ldots 3$) as the moving-target simulations and one hundred trials using each controller of Table 3 are performed for each scenario using a static target. The performance metric for static-target servoing is the average number of iterations to convergence





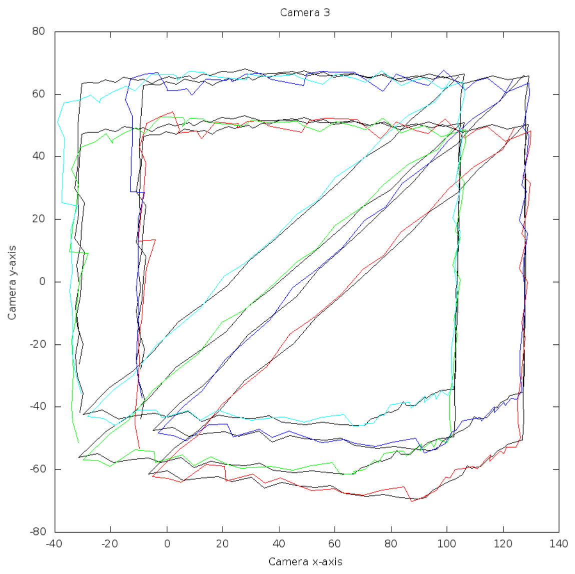


Figure 22: Image-plane coordinates using GN controller with one high-cost camera (Camera 1) and two low-cost cameras — moving target

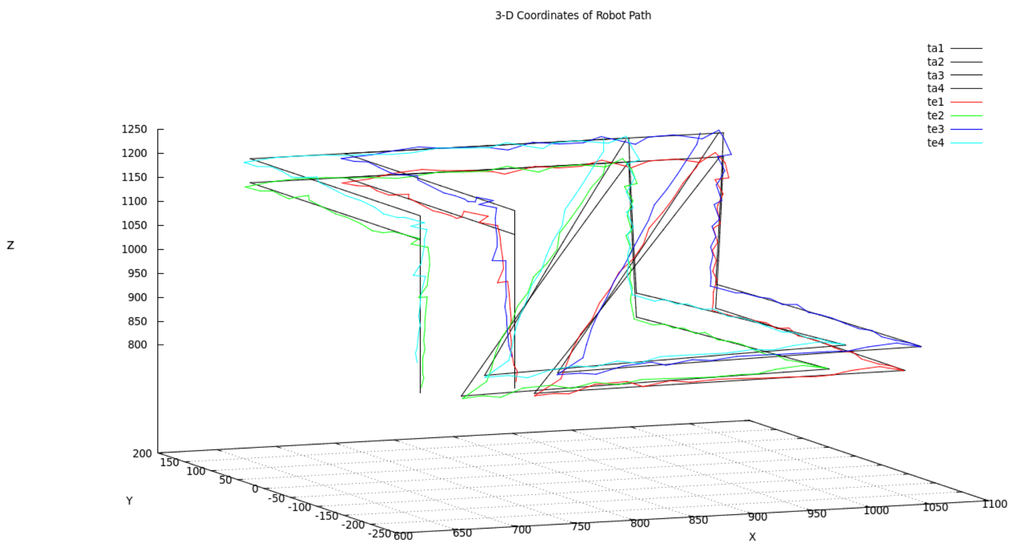


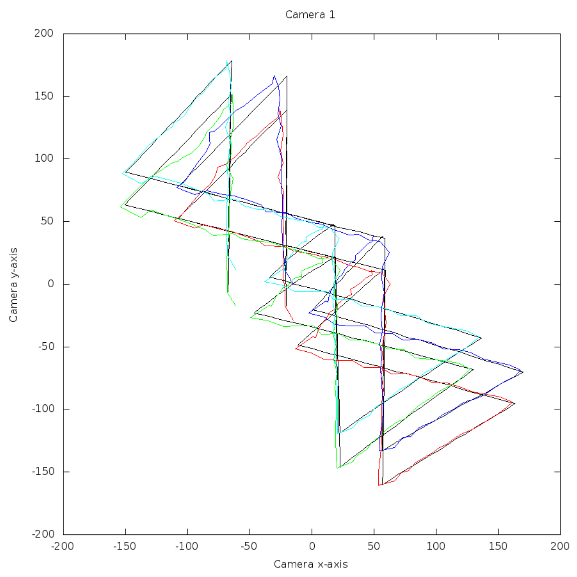
Figure 23: 3-D coordinates using GN controller with one high-cost camera (Camera 1) and two low-cost cameras — moving target

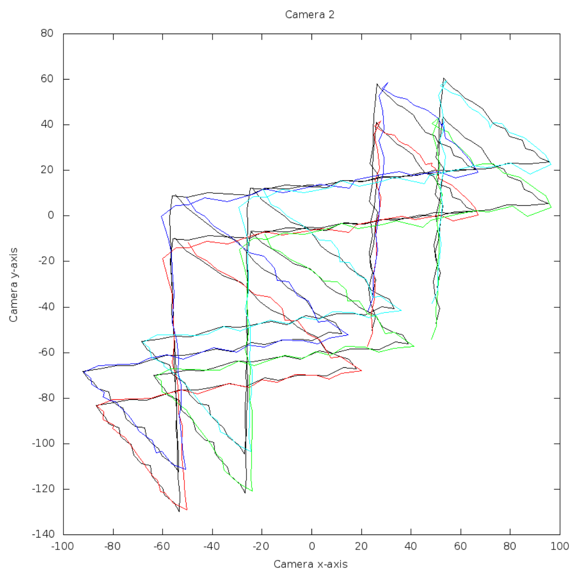
$$\bar{S} = \frac{1}{T}\sum_{l=1}^{T} S_l$$

Here the number of trials is $T = 100$. Convergence means the image-plane error norm is less than ϵ in all $C = 3$ cameras. For these trials the threshold is $\epsilon = 10$ pixels. Figures 28 and 29 show $\bar{S}$ for the forty-eight controllers of Table 3 plus Gauss-Newton. A missing column in those figures indicates that in one trial the termination threshold was not reached within six hundred iterations, discussed further in §7.5.3.1.

Examining Figure 28 reveals that non-adaptive KF is less sensitive to Q values for a static target than for a moving one; when servoing with the first order non-adaptive formulation using $R = I$ for $C_H = 1$, $\bar{S}$ varies 1% with changing Q while $\bar{e}_{\text{check},T,S}$ of the moving-target trials changes 10% with different Q values (see Figure 20). All tested non-adaptive KF methods perform better than GN for all values of Q and R, which is also different from the moving-target case where some Q and R combinations fare worse than GN.

A difference between moving-target and static-target VS with DAKF is that a smaller window size works better for a static target. This is seen comparing the "$Q = 5I$, $N = 12$"





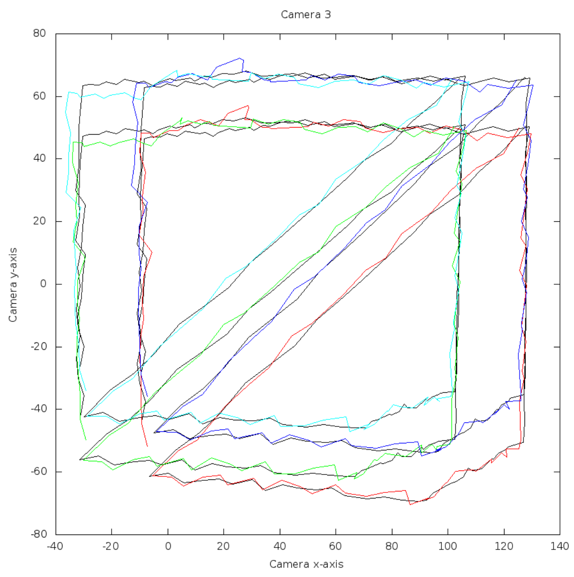


Figure 24: Image-plane coordinates using DAKF controller with one high-cost camera (Camera 1) and two low-cost cameras — moving target

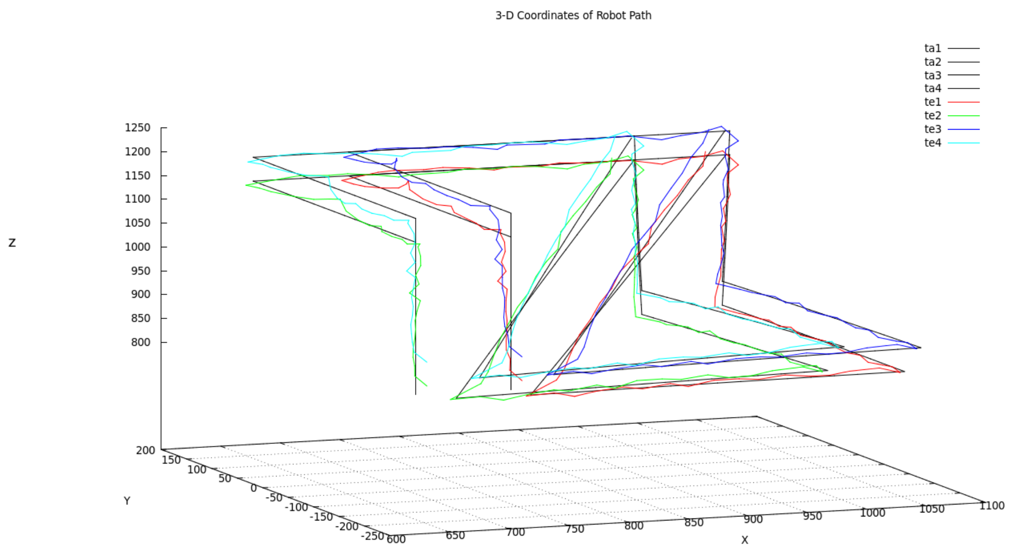


Figure 25: 3-D coordinates using DAKF controller with one high-cost camera (Camera 1) and two low-cost cameras — moving target

and "$Q = 5I$, $N = 20$" plots in Figure 29. A probable reason for this behavior is that the innovation outer product at time k,

$$\boldsymbol{\nu}_k \boldsymbol{\nu}_k^{\mathsf{T}} = \left(\boldsymbol{z}_k - H_k \hat{\boldsymbol{x}}_{k|k-1}\right)\left(\boldsymbol{z}_k - H_k \hat{\boldsymbol{x}}_{k|k-1}\right)^{\mathsf{T}}$$

fluctuates more with the moving-target path described in §6.2.1.1 than when servoing to a static target. This is evident comparing Figures 30 and 31. Spikes appear in the former at every eighteen iterations, the number of pre-recorded images per section of the moving-target path. Larger window size helps smooth the estimated innovation covariance $\widehat{C}_{\nu,k}$ of (53) when there are such rapid state changes. For the static-target case such smoothing is not necessary and thus the more responsive adaptation with $N = 12$ provides better VS performance.

The moving- and static-target cases are also different in that the adaptive technique yields larger improvements for a static target than a moving one. Table 5 gives $\bar{S}$ for all four camera scenarios with the best controller tested from each category (zero-memory, non-adaptive, and adaptive): Gauss-Newton, non-adaptive zeroth order with input Kalman filter

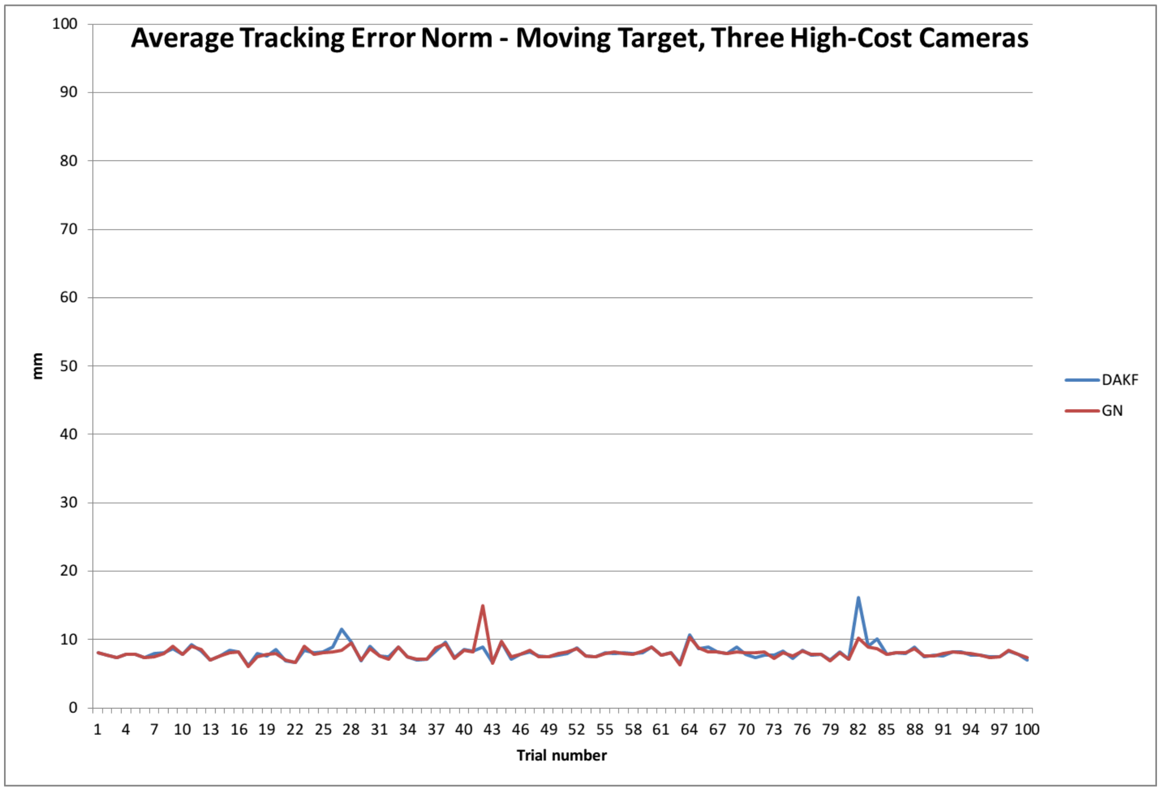


Figure 26: Average checkerboard-corners error (mm) for each of 100 trials with three high-cost cameras using first order DAKF and GN, moving target

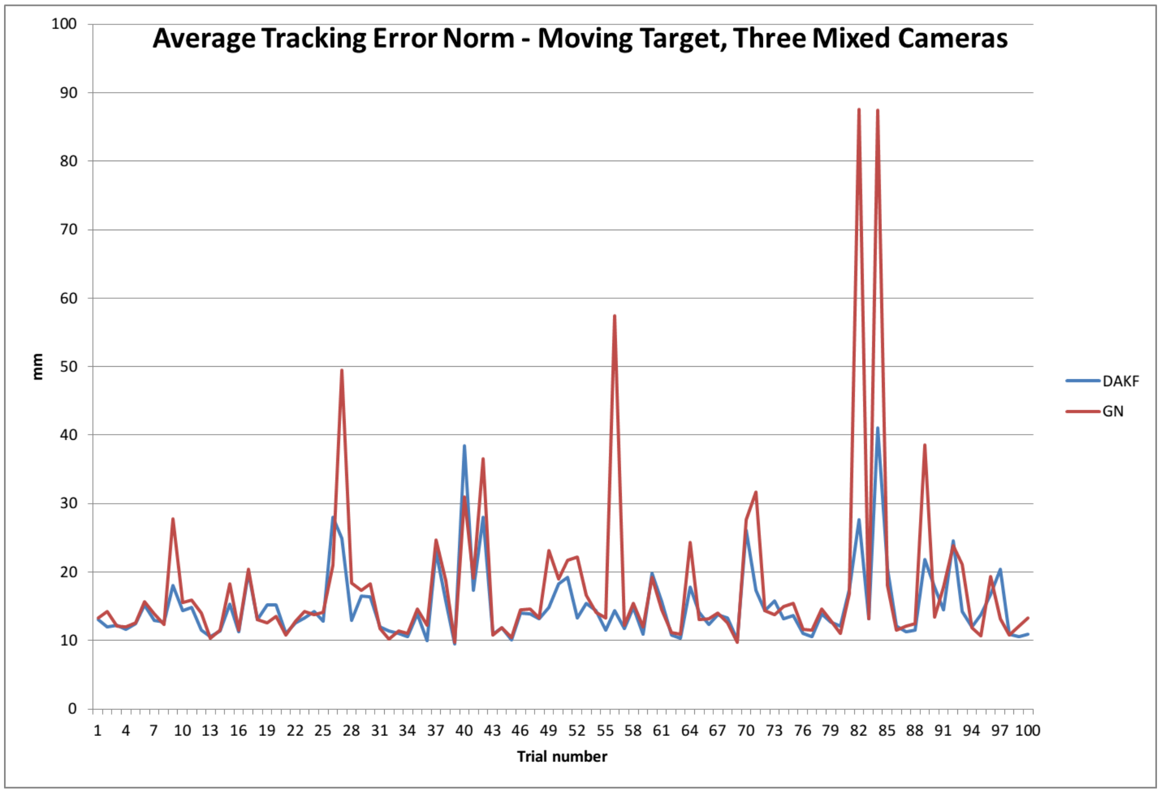


Figure 27: Average checkerboard-corners error norm (mm) for each of 100 trials with one high-cost camera and two low-cost cameras using first order DAKF and GN, moving target

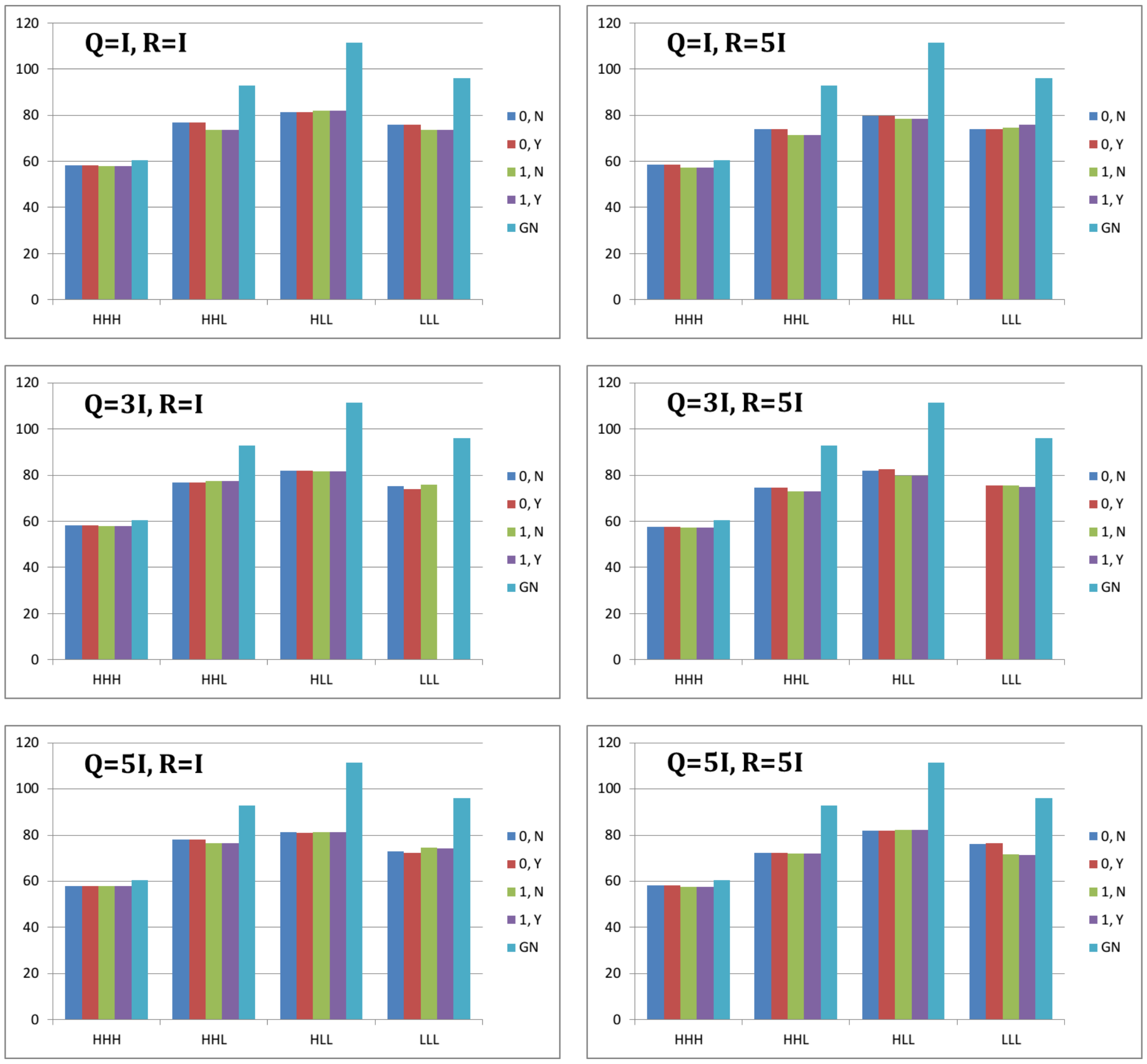


Figure 28: $\bar{S}$ using combinations of high- and low-cost cameras with GN and 24 variations of DAKF, "0" or "1" refers to the state space representation order and "Y" or "N" refers to with or without input

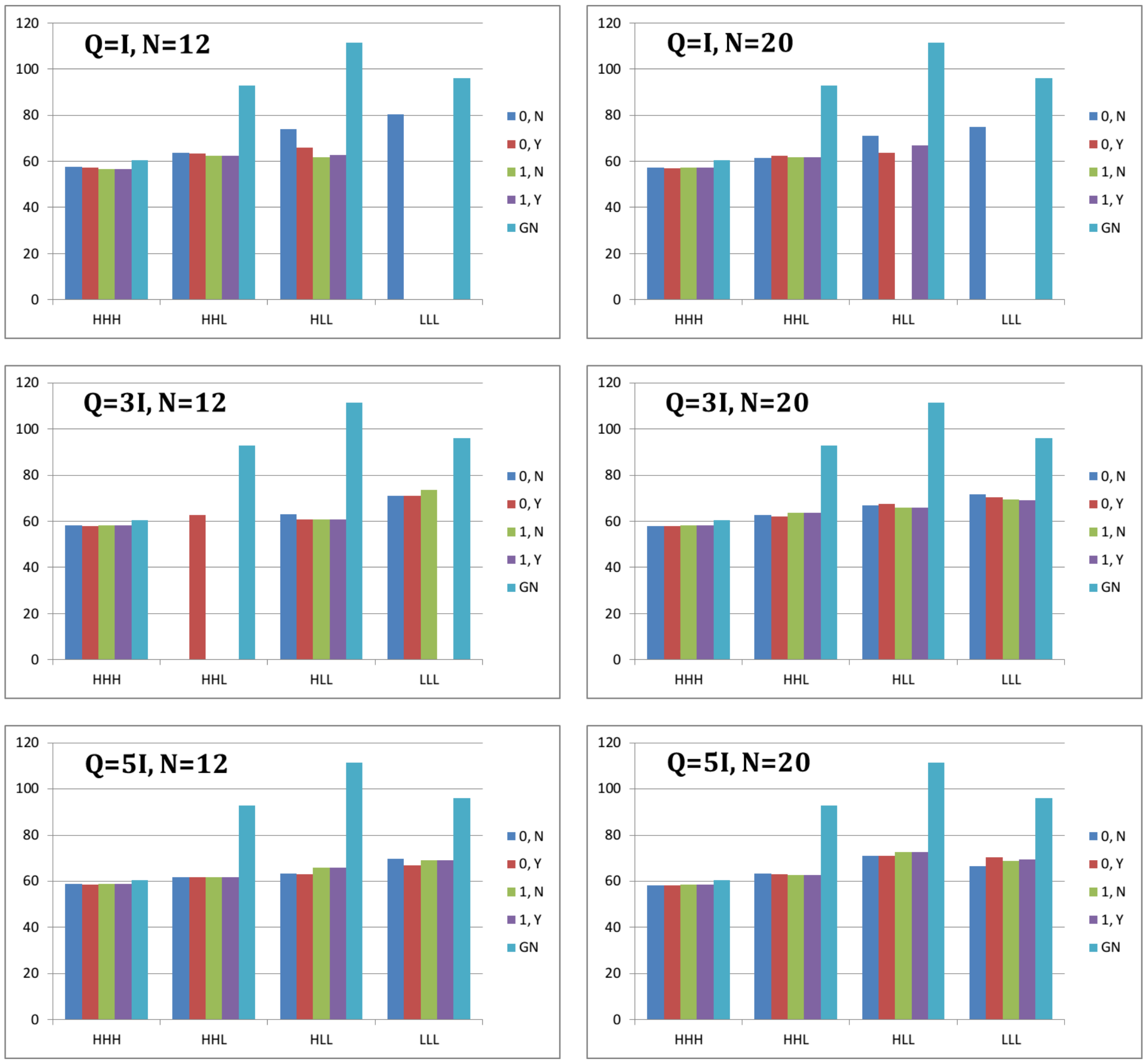


Figure 29: $\bar{S}$ using combinations of high- and low-cost cameras with GN and 24 variations of non-adaptive decentralized Kalman filter, "0" or "1" refers to the state space representation order and "Y" or "N" refers to with or without input

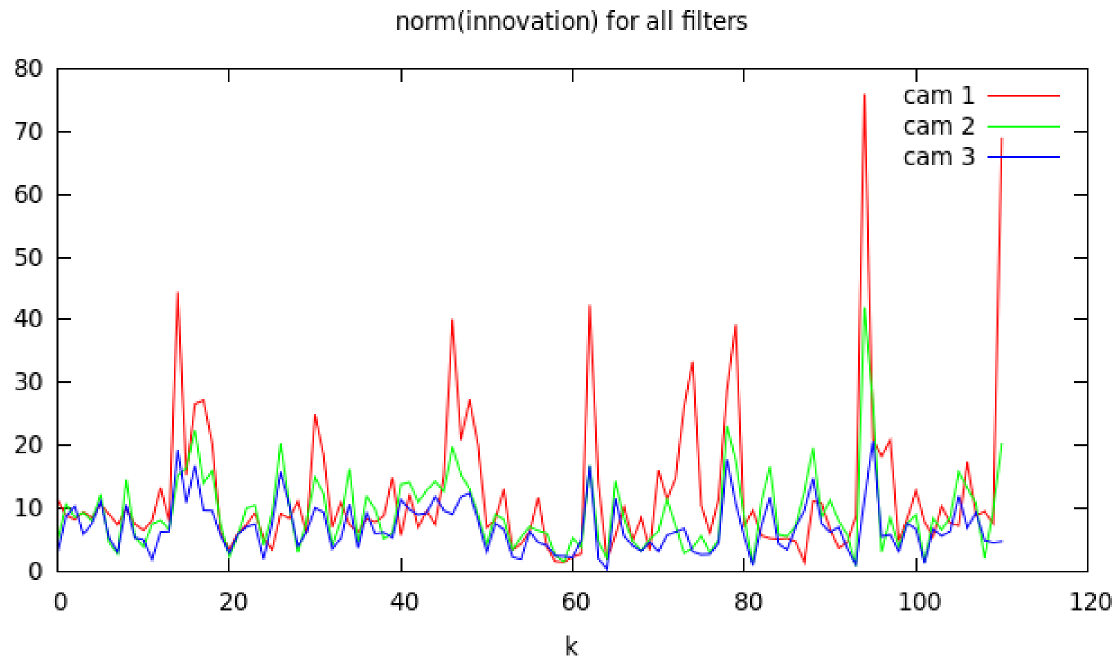


Figure 30: Innovation (pixels) for all three filters at every iteration of a moving-target, three-camera trial with $C_H = 1$ ("cam 1")

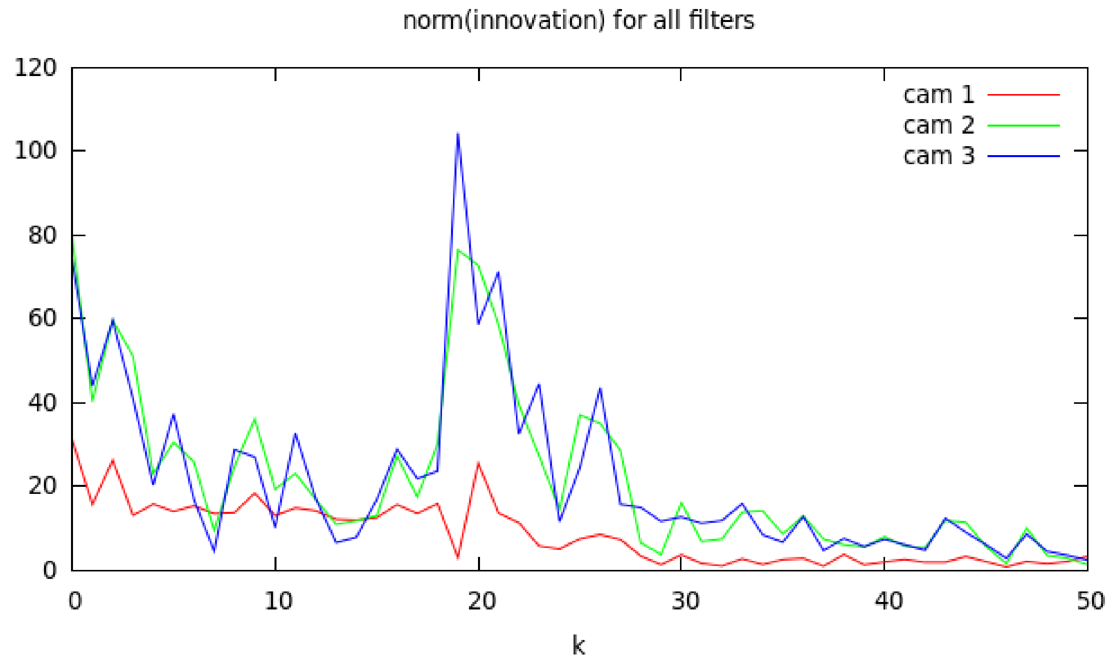


Figure 31: Innovation (pixels) for all three filters at every iteration of a static-target, three-camera trial with $C_H = 1$ ("cam 1")

Table 5: Minimum, average, and maximum iterations to convergence S for best controllers tested from three categories. The notation for the DAKF filters is "state space representation order", "Y or N using input", "β", "κ", and "N".

	$C_H = 3$, $C_L = 0$			$C_H = 2$, $C_L = 1$			$C_H = 1$, $C_L = 2$			$C_H = 0$, $C_L = 3$		
	$S_{\min}$	$\bar{S}$	$S_{\max}$	$S_{\min}$	$\bar{S}$	$S_{\max}$	$S_{\min}$	$\bar{S}$	$S_{\max}$	$S_{\min}$	$\bar{S}$	$S_{\max}$
GN	50	60.5	169	49	92.8	274	48	111.3	436	48	96.0	461
$\text{DAKF}_{0,\text{Y},5,1,\infty}$	50	58.0	106	50	78	288	47	81.1	305	46	72.4	273
$\text{DAKF}_{0,\text{Y},5,0.1,12}$	50	58.4	121	50	61.6	138	47	62.9	136	47	67.0	303

using $Q = 5I$ and $R = I$, and zeroth order with input DAKF using $Q = 5I$ and $N = 12$. Also presented are the minimum and maximum number of iterations of any trial $\bar{S}_{\min}$ and $\bar{S}_{\max}$ in each scenario.

The largest gains from the adaptive Kalman filter method again come for the $C_H = 1$ scenario: 43% improvement over Gauss-Newton and 23% over non-adaptive KF. Furthermore, the DAKF yields a more stable system as evidenced by the fact that of the 100 trials simulated for the $C_H = 1$ case with each controller, the largest number of iterations required for convergence $\bar{S}_{\max}$ is only 136 for DAKF compared to 436 for Gauss-Newton and 293 for non-adaptive KF. This sensitivity to variations can be seen in Figures 32 and 33, which plot the S for each of the 100 trials with first-order DAKF and GN. Figure 32 is for trials with three high-cost cameras and the two controllers have nearly equivalent performance. Figure 33 is for trials with one high-cost camera and two low-cost cameras and the two controllers have significantly different average performance and GN also has much higher outliers than DAKF.

The robot path for a static trial using GN is shown in Figures 34 (image-plane coordinates) and 35 (3-D). The black dots in Figure 34 are the target image coordinates. The same trial is run using DAKF of Table 5 and the resulting path is shown in Figures 36 and 37. Examining Figures 34–37 reveals that DAKF yields a more direct path to the target: the GN trajectory lasts 368 steps, while DAKF servos the robot to the target in just 59 steps. The adaptive measurement covariance technique engenders this improvement, as evidenced by the fact that the DKF of Table 5 requires 305 steps for convergence.

For the static-target case the DAKF method shows pronounced insensitivity to camera

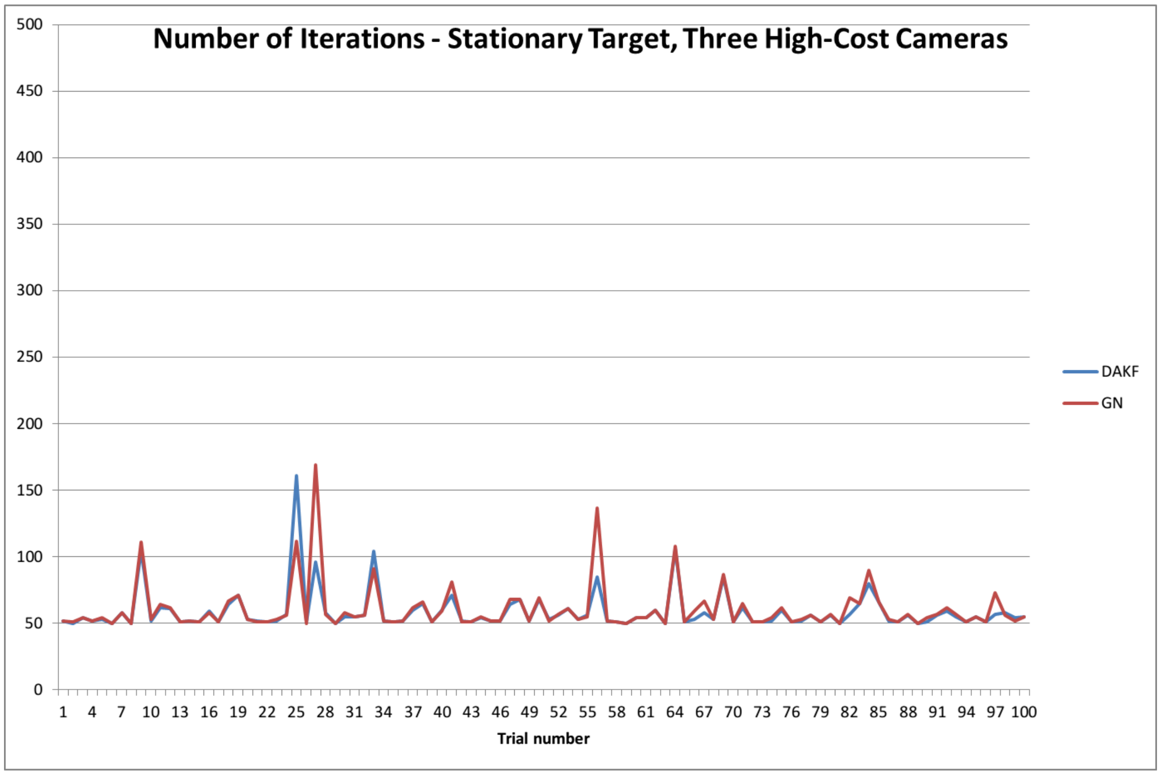


Figure 32: Number of iterations to convergence S for each of 100 trials with three high-cost cameras using first order DAKF and GN, static target

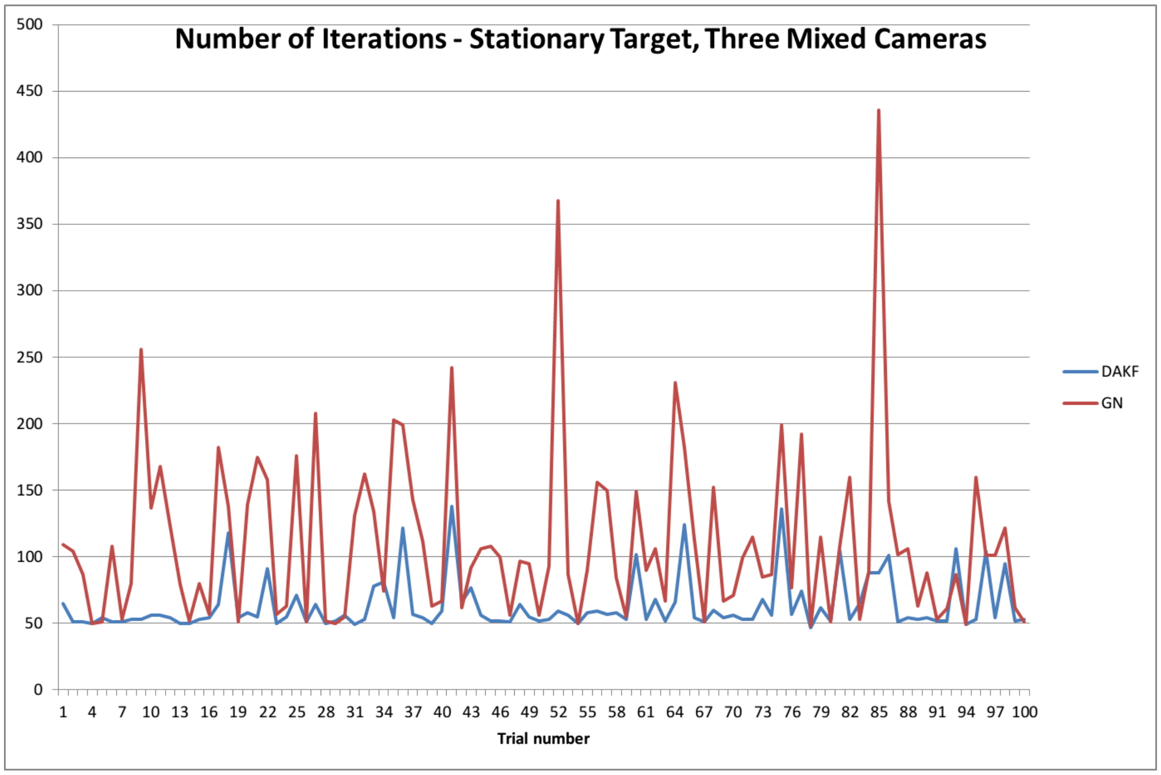


Figure 33: Number of iterations to convergence S for each of 100 trials with one high-cost camera and two low-cost cameras using first order DAKF and GN, static target

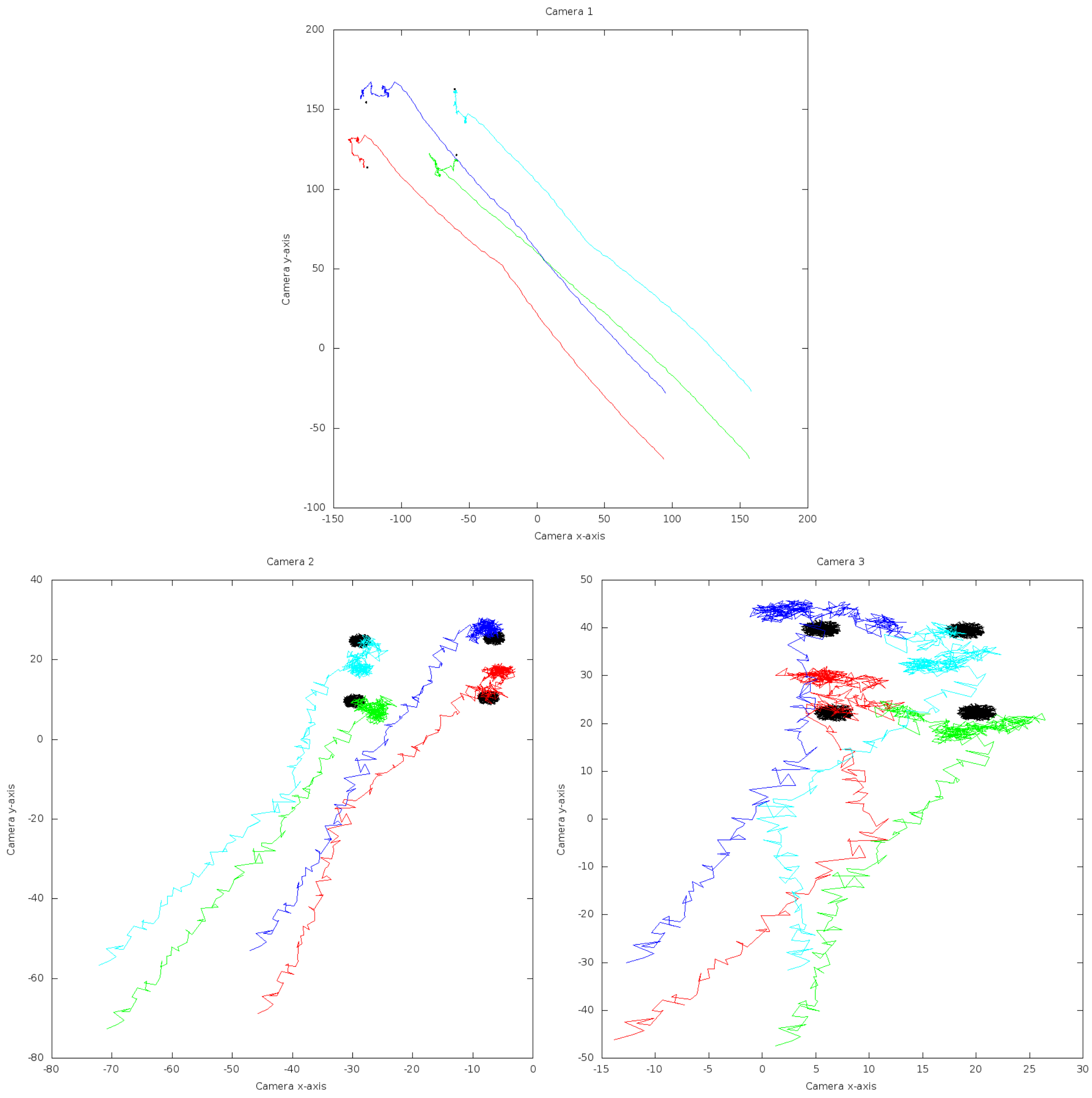


Figure 34: Image-plane coordinates using GN controller with one high-cost camera (Camera 1) and two low-cost cameras — static target

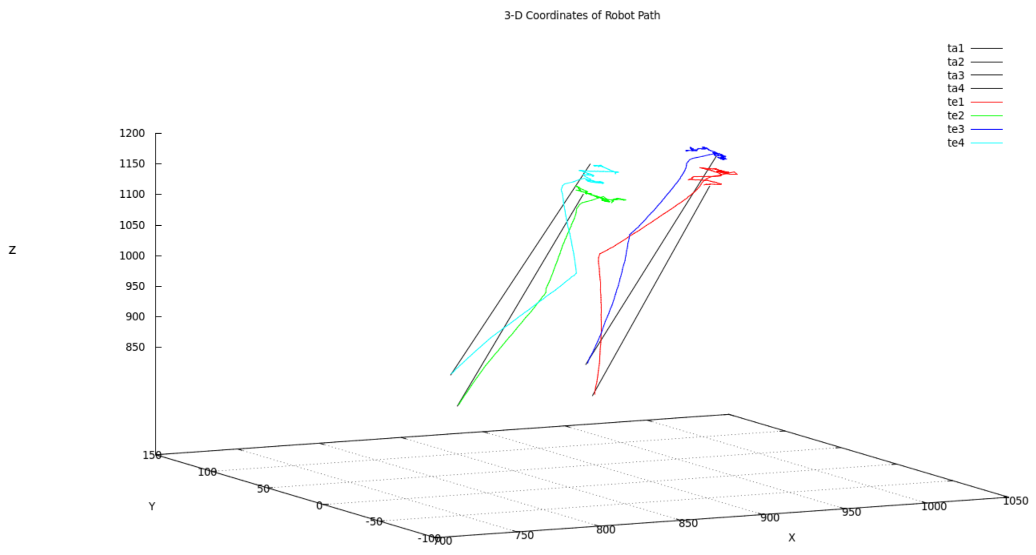


Figure 35: 3-D coordinates using GN controller with one high-cost camera (Camera 1) and two low-cost cameras — static target

noise compared to GN. For example, using a zeroth order with input DAKF having $Q = 5I$ and $N = 12$, $\bar{S}$ changes only 8% for $C_H = 0 \ldots 3$, whereas using Gauss-Newton $\bar{S}$ varies 84% (see Figure 29).

Data show that when using cameras of various signal-to-noise ratios DAKF can provide improved performance over non-adaptive KF and the traditional Gauss-Newton approach. The mechanism responsible for this improvement, local measurement noise covariance adaptation, is explored in the following section.

7.5.3 Evaluation of camera weighting

It is reasonable to expect $\widehat{R}^{(i)}$ of (57) to have larger elements for a low-cost camera than for a high-cost camera, but this is not borne out by Figure 38 which for each camera scenario of the moving-target simulations shows the average trace of $\widehat{R}^{(i)}$ in the three local first order DAKF filters with $Q = 5I$ and $N = 20$. If the expectation were realized then the third column of the "HHL" scenario would be higher than the other two and the first column in "HLL" would be the shortest. A corresponding plot for static-target simulations is given

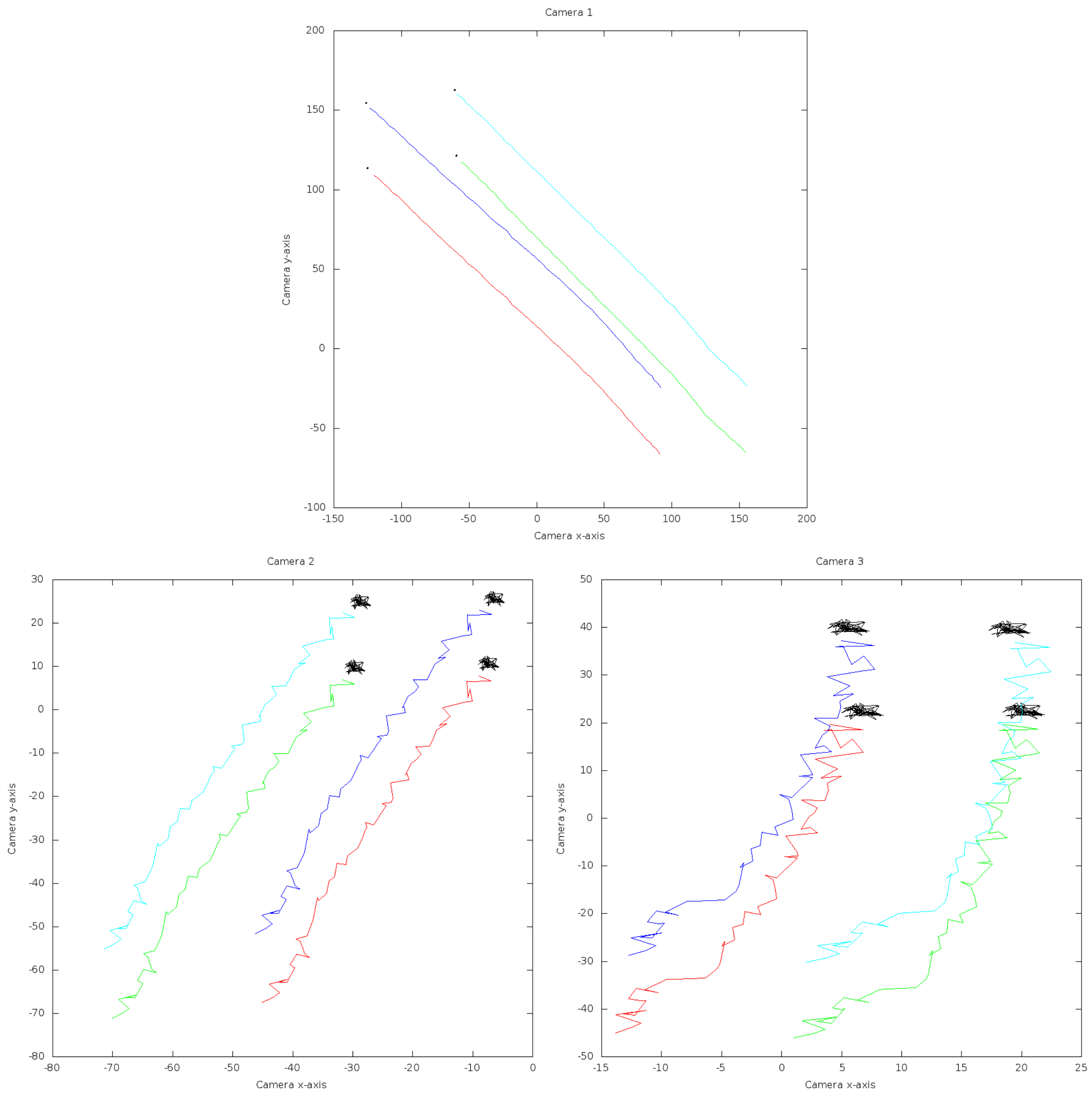


Figure 36: Image-plane coordinates using DAKF controller with one high-cost camera (Camera 1) and two low-cost cameras — static target

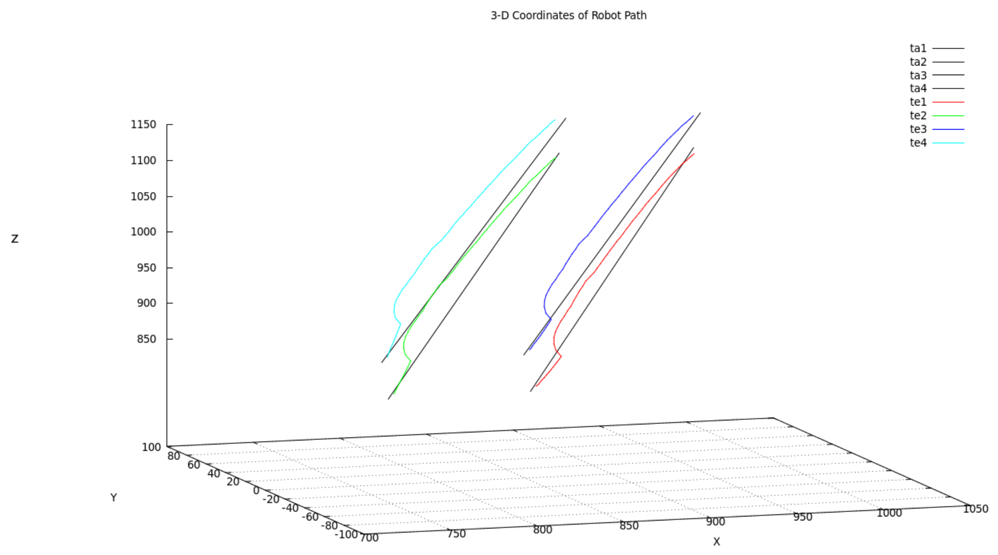


Figure 37: 3-D coordinates using DAKF controller with one high-cost camera (Camera 1) and two low-cost cameras — static target

in Figure 39, this time for zeroth order with input DAKF having $Q = 5I$ and $N = 12$. In that figure the "HLL" group appears as expected but "HHL" does not.

These figures show $\widehat{R}^{(i)}$ is not always dependent on camera type. Camera location is influential too. For example, consider two cameras observing a robot describe a circle in Cartesian space. The optical axis of one camera is in the plane of motion, the other camera is placed above the circle with its optical axis perpendicular to said plane. While the robot end-effector moves along the circle, the linear model for the first camera $\hat{J}^{(1)}$ does a better job predicting image coordinate changes than $\hat{J}^{(2)}$ since the motion appears more linear in image-plane 1 than in image-plane 2. The innovation for the first camera $\boldsymbol{\nu}_k^{(1)}$ will be smaller than for the second camera, which bears directly on the measurement covariance estimates $\widehat{R}^{(1)}$ and $\widehat{R}^{(2)}$.

Even small changes in $\widehat{R}^{(i)}$ can affect servoing performance. The moving target trials of Figures 22–25 have average error $\bar{e}_{\text{check}} = 19.1$ mm for Gauss-Newton and 17.3 mm for DAKF, for which the trace of $\widehat{R}^{(i)}$ is plotted in Figure 40. With $R_0^{(i)} = 0.1I$ adaptation only occurs at three iterations for $\widehat{R}^{(2)}$ (trace$\left(R_0^{(i)}\right) = 0.8$ pixels^2), nonetheless DAKF has

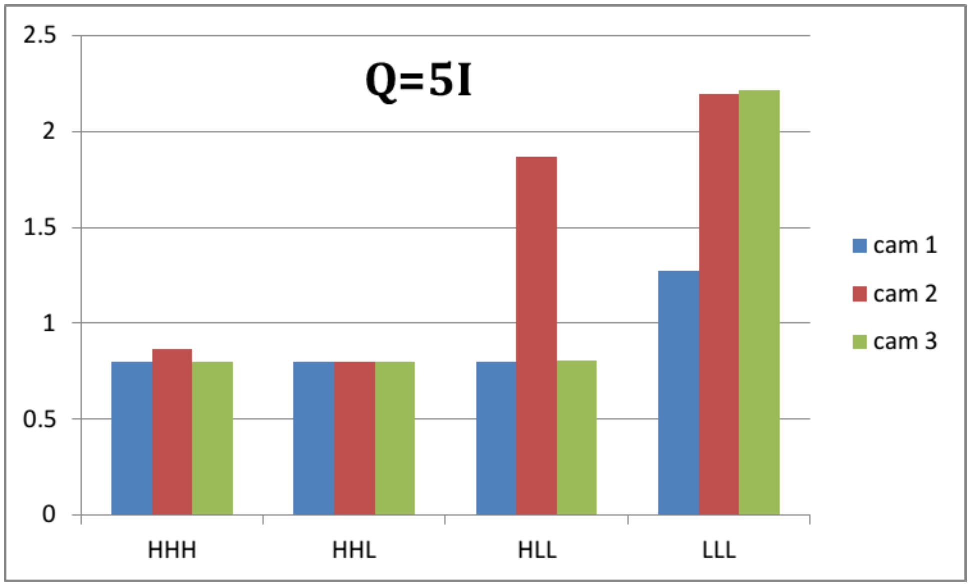


Figure 38: Average trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = 5I$ and $N = 20$ — moving target

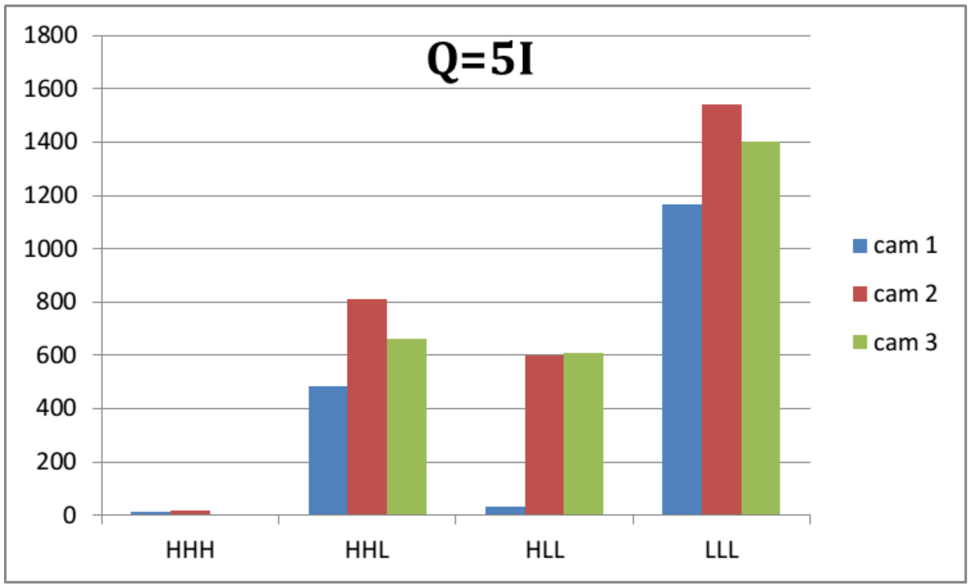


Figure 39: Average trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using zeroth order with input DAKF $Q = 5I$ and $N = 12$ — static target

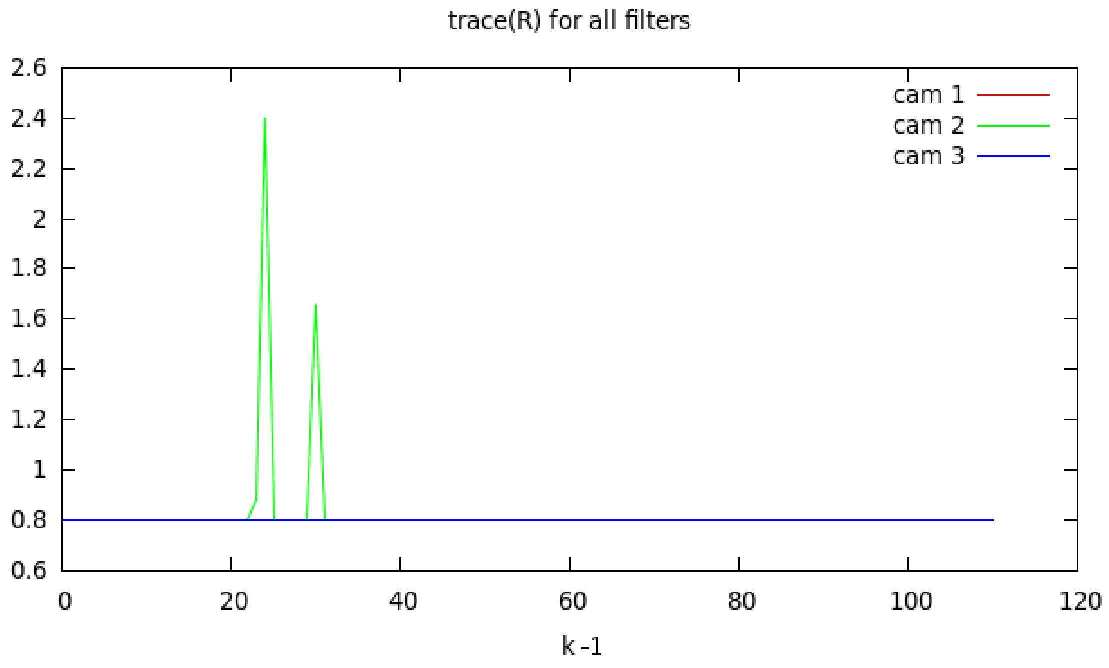


Figure 40: trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = 5I$ and $N = 20$ with $C_H = 1$ ("cam 1") — moving target

significantly lower tracking error for this trial than non-adaptive KF with $Q = 5I$ and $R = I$: 17.3 mm compared to 137.5 mm. The error for non-adaptive KF is reduced to 19.0 mm by setting $R = 5I$ but DAKF provides this adjustment automatically.

As stated above, the static-target trials depicted in Figures 34–37 have convergence times $S = 368$ iterations for Gauss-Newton and $S = 59$ for the DAKF. The non-adaptive KF in Table 5 requires 305 steps to converge to the goal. Changing the covariance matrix to $R = 5I$ reduces this to 251 steps, which combined with the 59 steps for DAKF indicates that automatic R adjustment is responsible for the improved performance. The trace of $\widehat{R}^{(i)}$ is plotted in Figure 41 for the DAKF trial, illustrating the adjustments during servoing. The large initial spike at $k = 12$ is due to high innovation $\boldsymbol{\nu}_1^{(i)} = \hat{\boldsymbol{z}}_1^{(i)} - H_1^{(i)}\hat{\boldsymbol{x}}_{1|0}$, visible in Figure 42.

Examining this history of $\widehat{R}_k^{(i)}$ and comparing DAKF and DKF performances shows that adaptive R simplifies parameter selection since, as noted above, window size N has little effect — it's easier to choose N than R. Choosing an appropriate value for Q is still

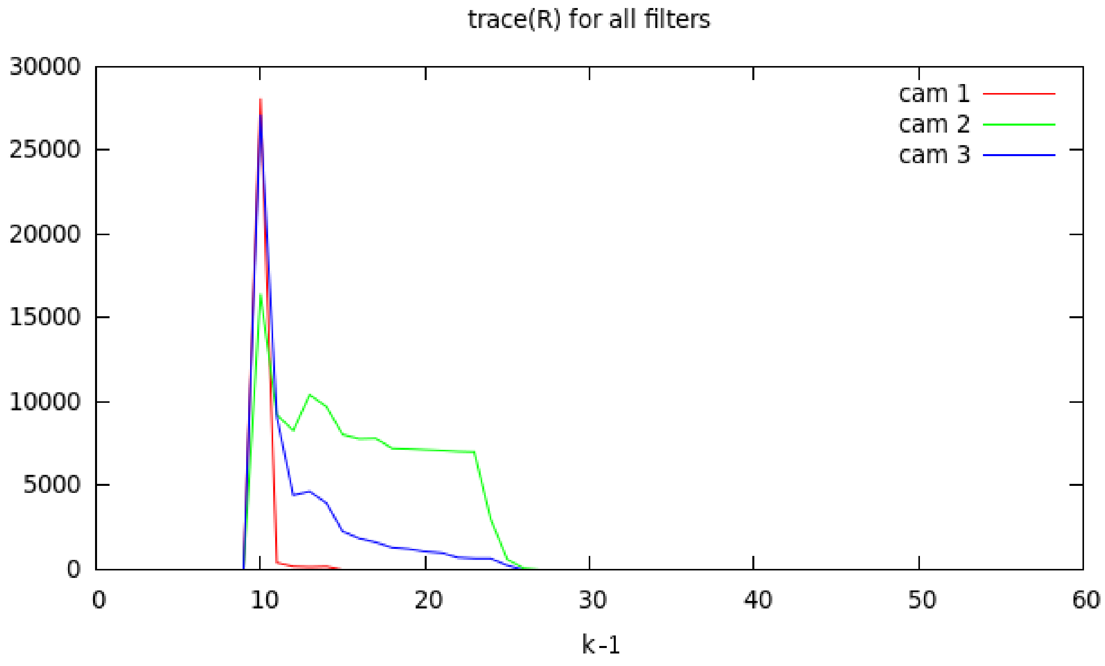


Figure 41: $\text{trace}\left(\widehat{R}^{(i)}\right)$ (pixels^2) using zeroth order with input DAKF $Q = 5I$ and $N = 12$ with $C_H = 1$ ("cam 1") — static target

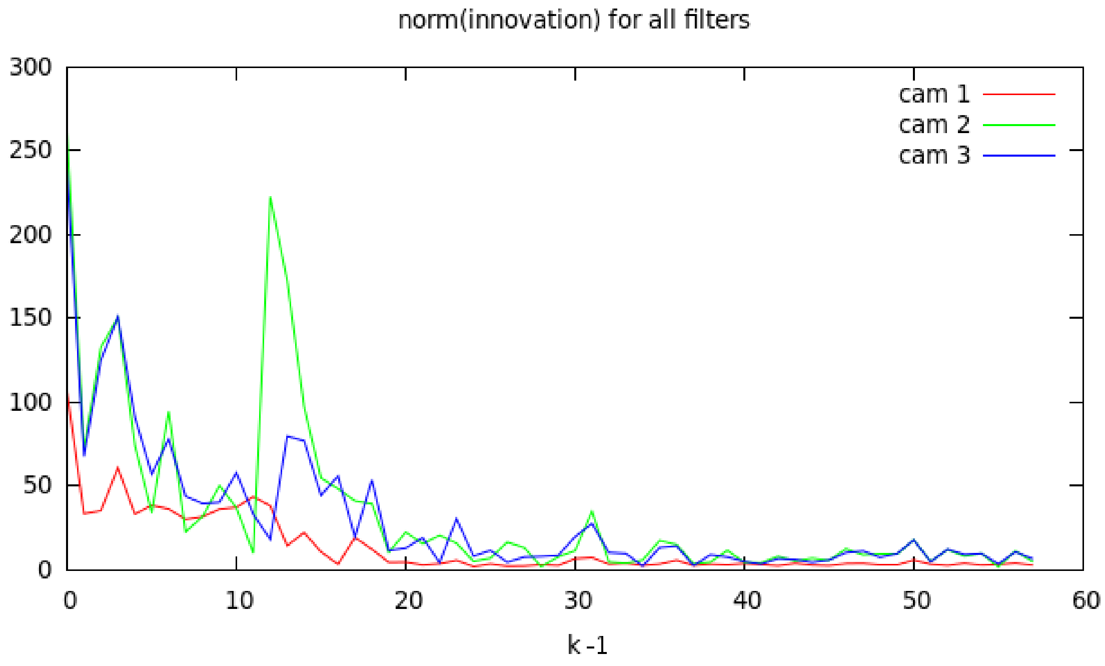


Figure 42: Innovation $\boldsymbol{\nu}_k^{(i)}$ (pixels) using zeroth order with input DAKF $Q = 5I$ and $N = 12$ with $C_H = 1$ ("cam 1") — static target

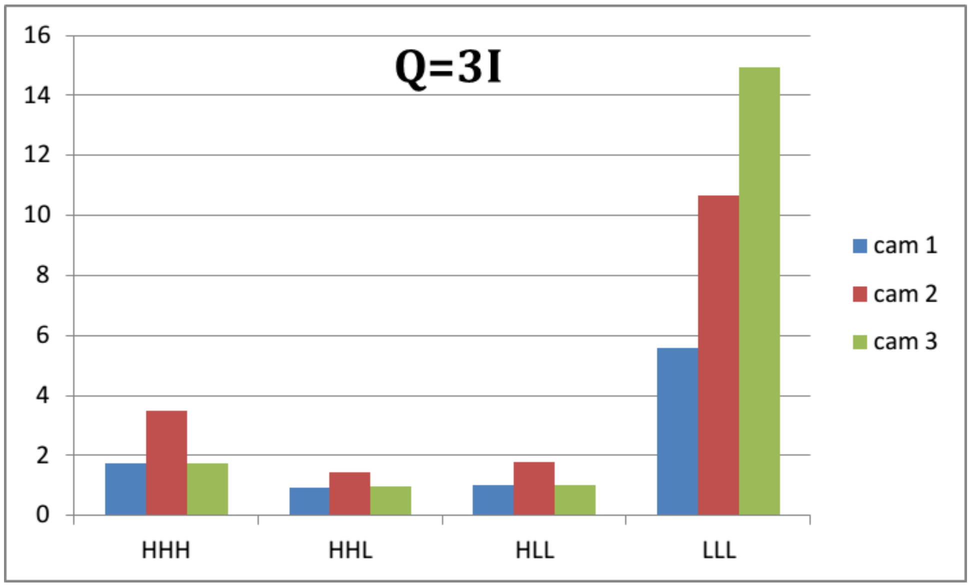


Figure 43: Average trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = 3I$ and $N = 20$ — moving target

required. The next section discusses the effect of this choice on R adaptation.

7.5.3.1 Tuning DAKF

Figures 43 and 44 extend the data of Figure 38 by providing the average trace of $\widehat{R}^{(i)}$ for trials with different process noise covariances $Q = 3I$ and $Q = I$. The trends of Figure 38 are exaggerated in Figure 43. The local filters retain their rankings for highest, middle, and lowest average trace$\left(\widehat{R}^{(i)}\right)$ in each camera scenario, but the magnitudes of and differences between these averages are greater with $Q = 3I$. That is, this change in Q makes R more adaptive. The trends do not continue to $Q = I$. In fact, Figure 44 shows that the trends are reversed with the high-cost cameras receiving higher average $\widehat{R}^{(i)}$ than the low-cost cameras. The R adaptation behavior is dramatically changed with such low Q.

The traces of $\widehat{R}^{(i)}$ for the moving-target scenario of Figures 24, 25, and 40 are plotted for $Q = 3I$ in Figure 45 and for $Q = I$ in Figure 46. The increased adaptiveness in Figure 45 improves tracking error compared to the $Q = 5I$ trial of Figure 40: $\bar{e}_{\text{check}} = 15.6$ mm versus 17.3 mm. Lowering the process noise covariance to $Q = I$ makes adaptation (Figure 46) inaccurate and harms performance: $\bar{e}_{\text{check}} = 19.0$ mm. The paths in Cartesian space

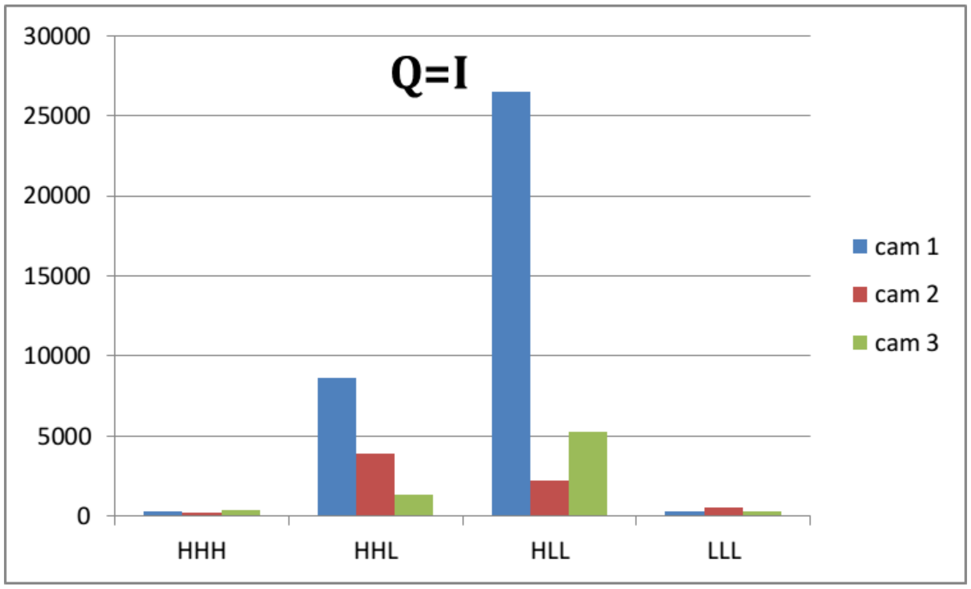


Figure 44: Average trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = 1I$ and $N = 20$ — moving target

for these three trials ($Q = 5I$, $3I$, and I) are shown in Figures 47–49, examination of which shows most accurate tracking in the $Q = 3I$ case.

Figures 50 and 51 extend the static-target data of Figure 39 by providing the average trace of $\widehat{R}^{(i)}$ for different process noise covariances $Q = 3I$ and $Q = I$. Comparison of these three figures is similar to those of the moving-target case in that when $Q = 5I$ or $3I$ the high-cost cameras have lower covariance than the low-cost cameras and this is reversed for $Q = I$. The traces of $\widehat{R}^{(i)}$ for the static-target scenario of Figures 36, 37, 41, and 42 are plotted for $Q = 3I$ in Figure 52 and for $Q = I$ in Figure 53. The effect of varying Q in these trials is that $S = 59$ with $Q = 5I$, $S = 60$ with $Q = 3I$, and $S = 179$ with $Q = I$ (still lower than the 368 steps required for GN and 305 for non-adaptive KF with $Q = 5I$ and $R = I$).

Plots for a $C_H = 1$ static-target trial in which the first order DAKF with $Q = I$ and $N = 20$ fails to converge are presented in Figures 54–57. The high SNR data of Camera 1 is ignored due to the deleterious effects of having Q too small, thus the controller gets stuck ignoring this data despite the large image-plane error in that camera. Measurements from Camera 3 are prioritized, as evidenced by the low trace$\left(\widehat{R}^{(3)}\right)$ values and small image-plane

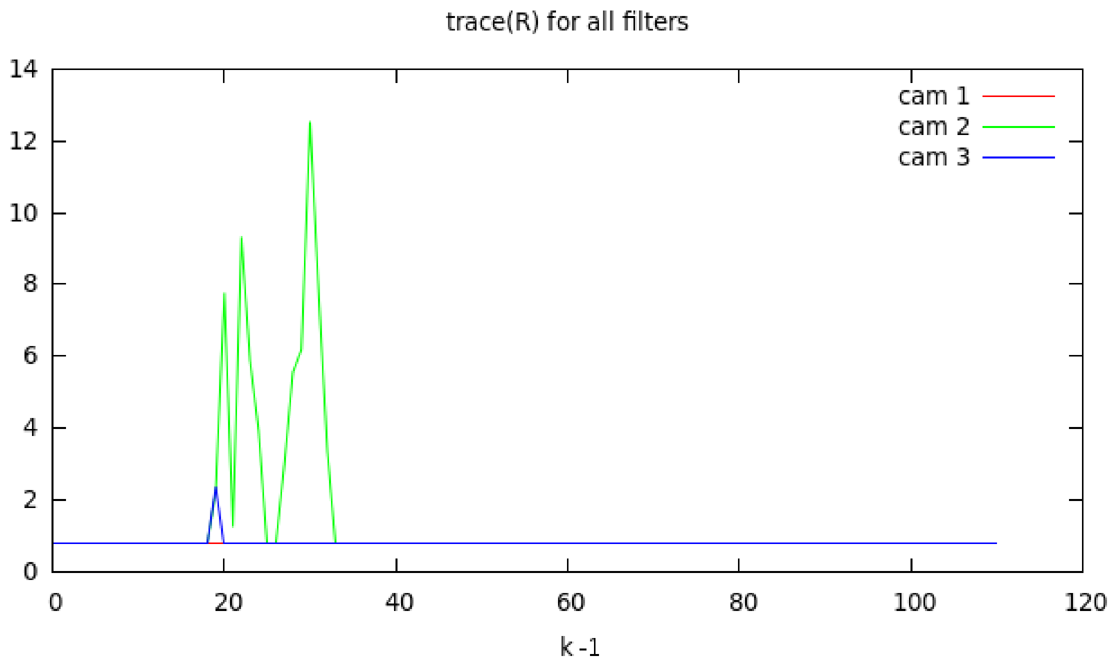


Figure 45: trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = 3I$ and $N = 20$ with $C_H = 1$ ("cam 1") — moving target

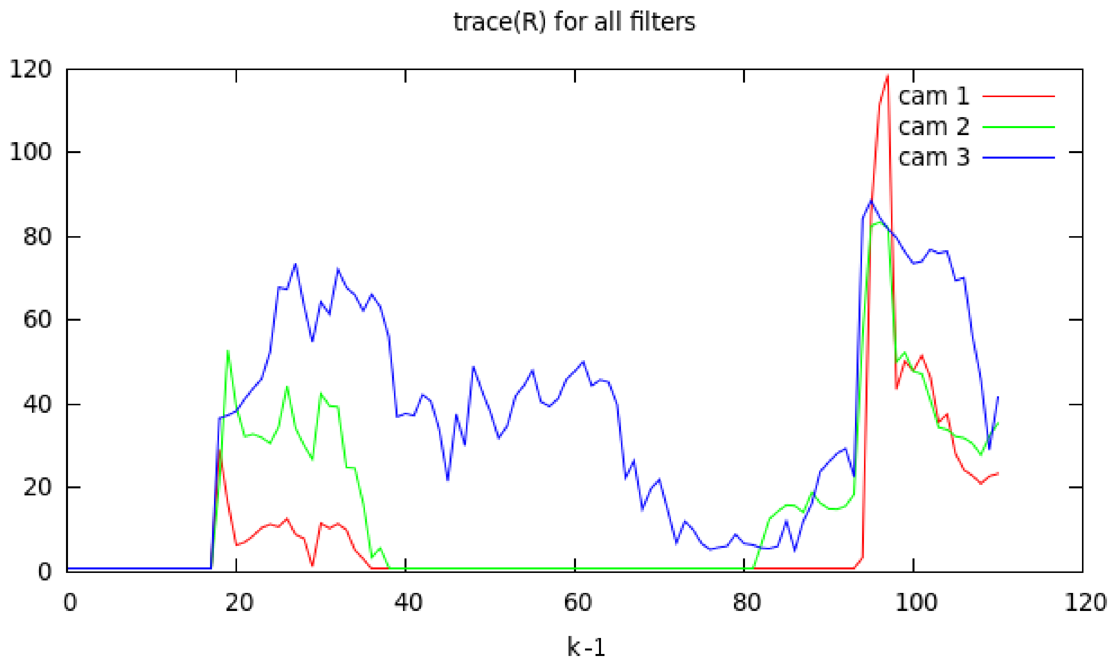


Figure 46: trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = I$ and $N = 20$ with $C_H = 1$ ("cam 1") — moving target

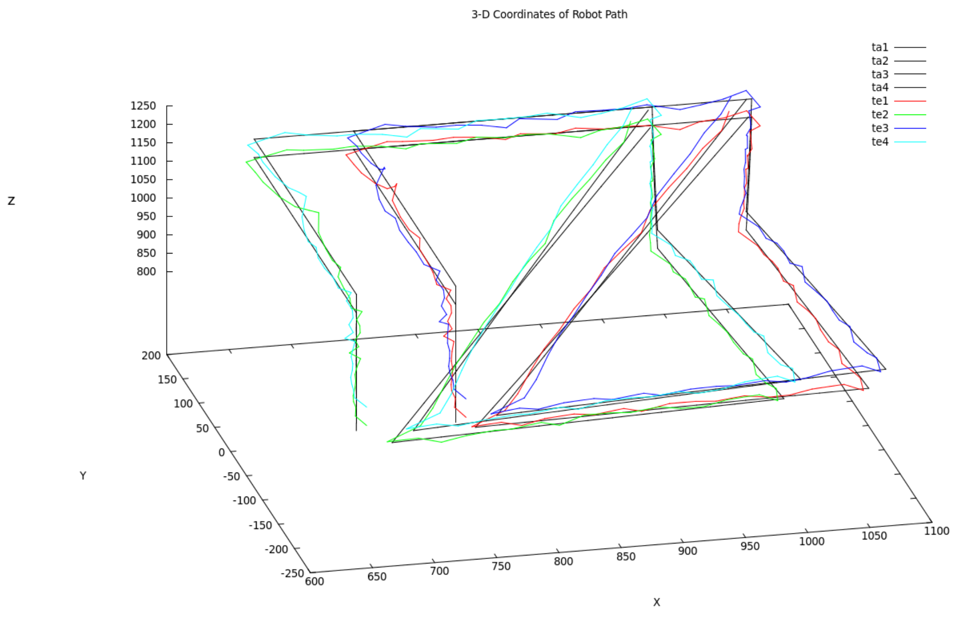


Figure 47: 3-D coordinates using DAKF controller $Q = 5I$ with one high-cost camera (Camera 1) and two low-cost cameras — moving target

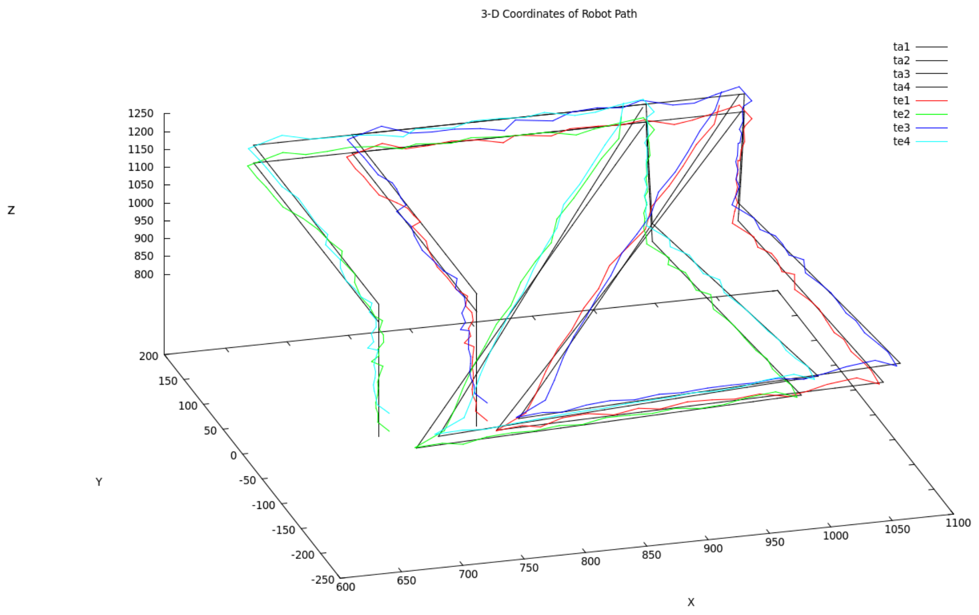


Figure 48: 3-D coordinates using DAKF controller $Q = 3I$ with one high-cost camera (Camera 1) and two low-cost cameras — moving target

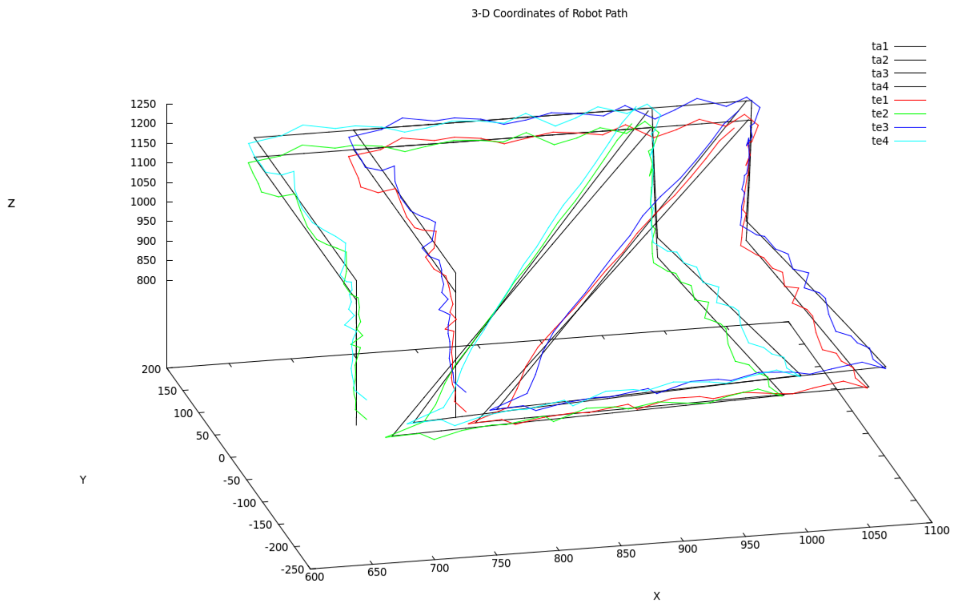


Figure 49: 3-D coordinates using DAKF controller $Q = I$ with one high-cost camera (Camera 1) and two low-cost cameras — moving target

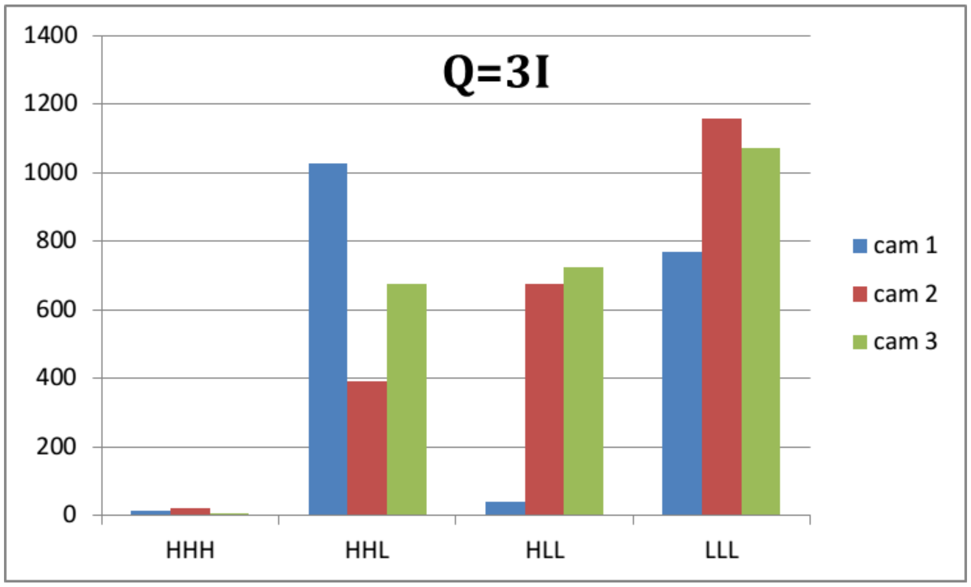


Figure 50: Average trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using zeroth order with input DAKF $Q = 3I$ and $N = 12$ — static target

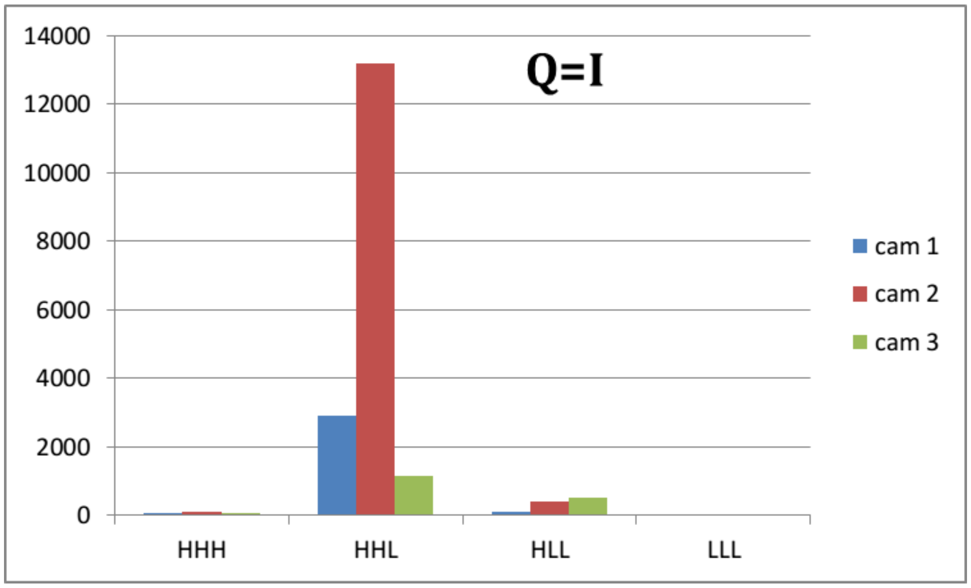


Figure 51: Average trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using zeroth order with input DAKF $Q = 1I$ and $N = 12$ — static target

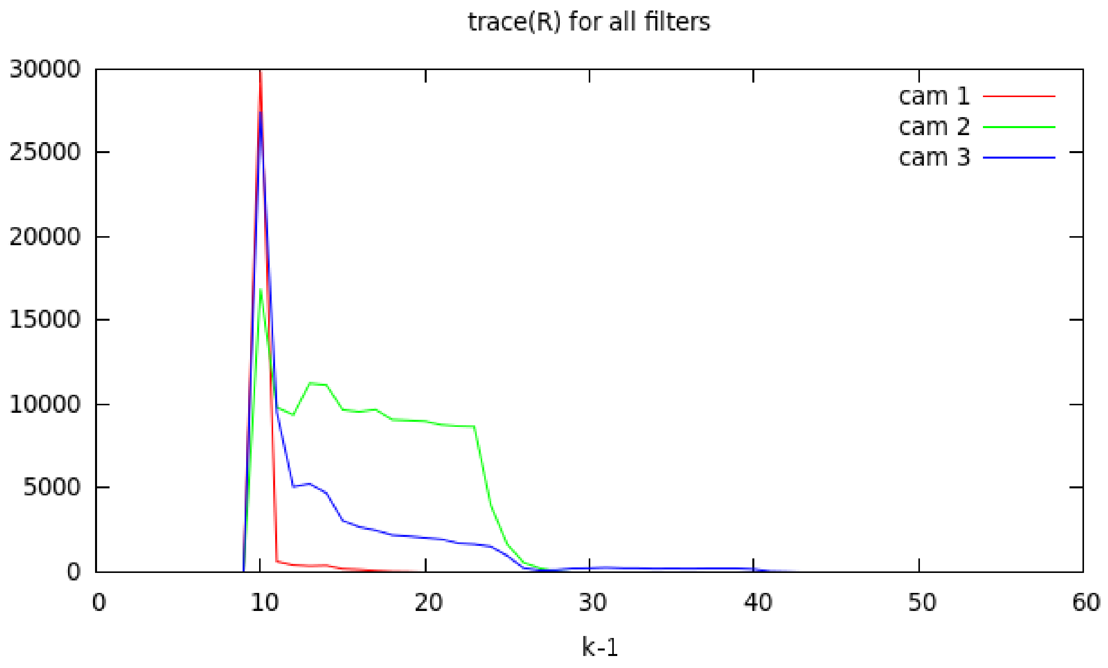


Figure 52: trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = 3I$ and $N = 20$ with $C_H = 1$ ("cam 1") — static target

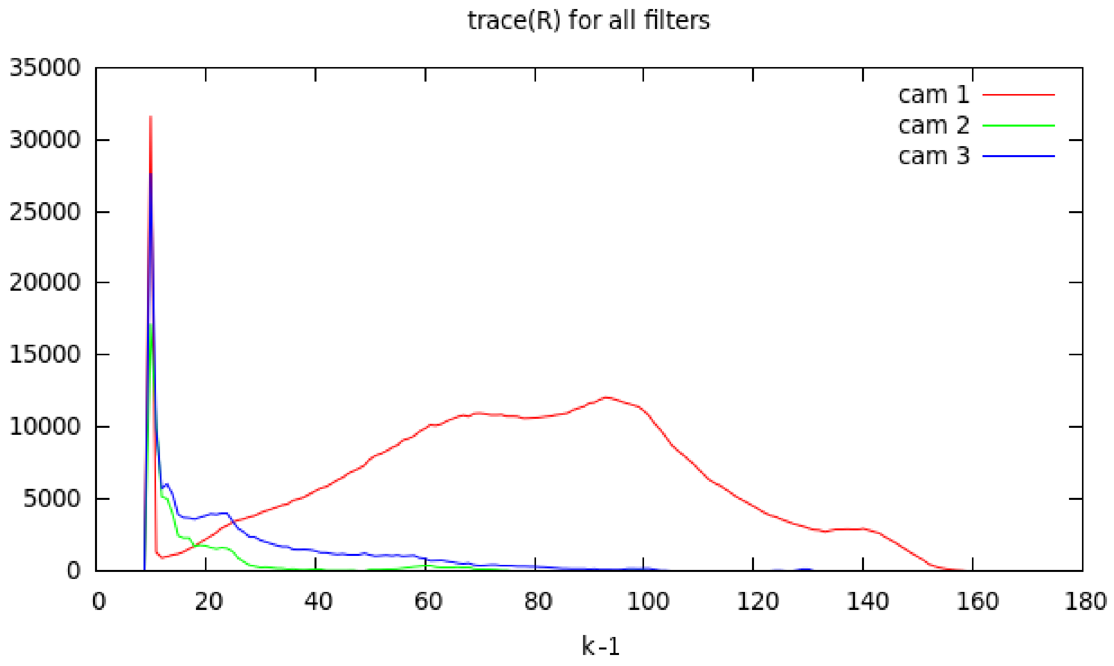


Figure 53: trace$\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = I$ and $N = 20$ with $C_H = 1$ ("cam 1") — static target

error in that camera view. Inaccurate Q degrades controller performance but the penalty is much higher when Q is too low than when it is too high. Therefore Q ought to be assumed conservatively high.

7.6 Summary

Several conclusions can be drawn from the simulation data presented:

- §7.3, "Comparing High-Cost and Low-Cost Cameras for Visual Servoing":
  - For moving targets Figure 13 shows diminishing returns for more than three cameras and an inability of low-cost cameras to achieve errors as low as a single high-cost camera.
  - For static targets Figure 18 shows a minimum positional error of about 0.5 mm and diminishing returns with more than two high-cost cameras or six low-cost cameras, at which point positional error is comparable to that of high-cost camera systems.

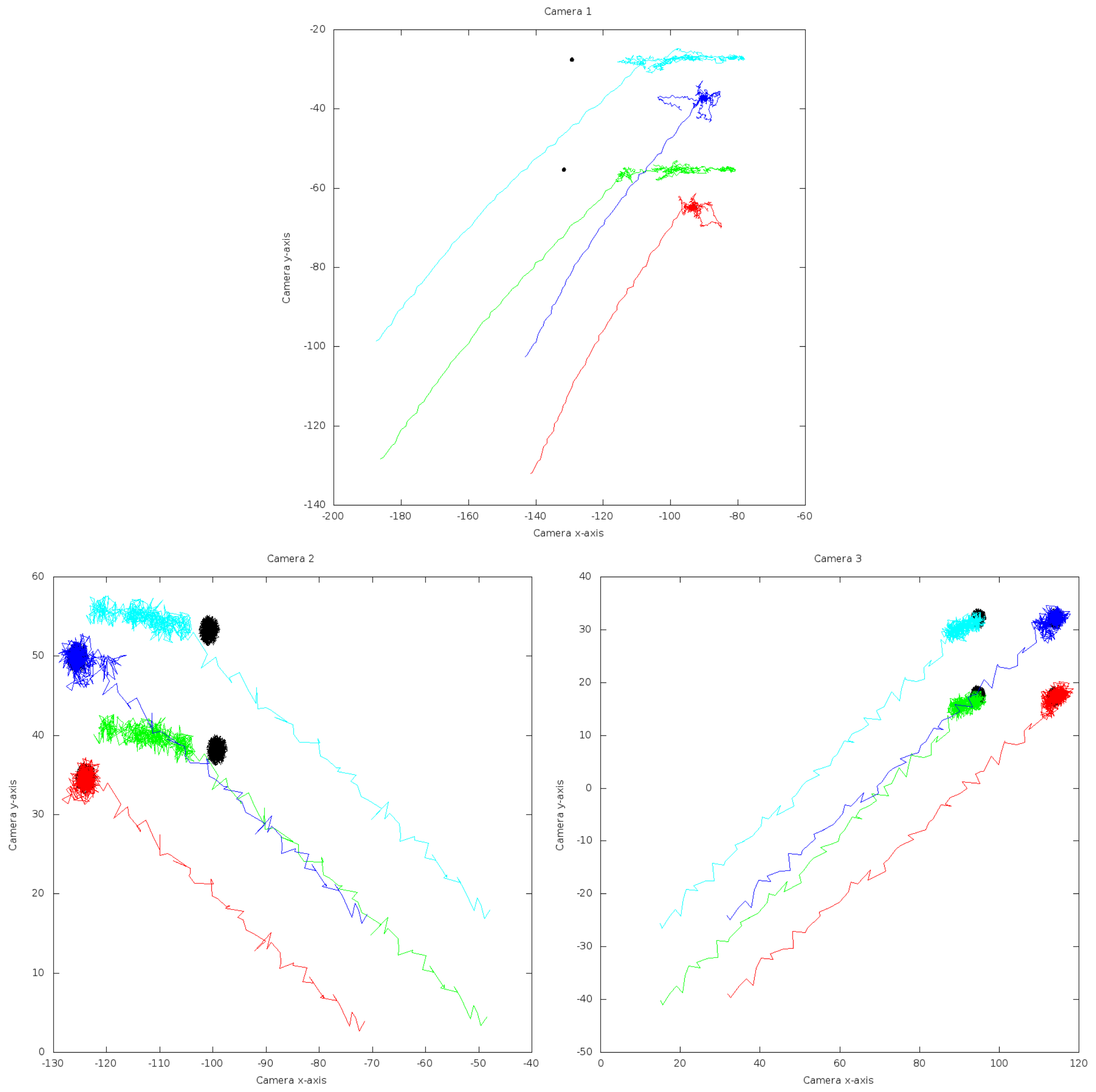


Figure 54: Image-plane coordinates using DAKF controller $Q = I$ with one high-cost camera (Camera 1) and two low-cost cameras — failure to converge to static target

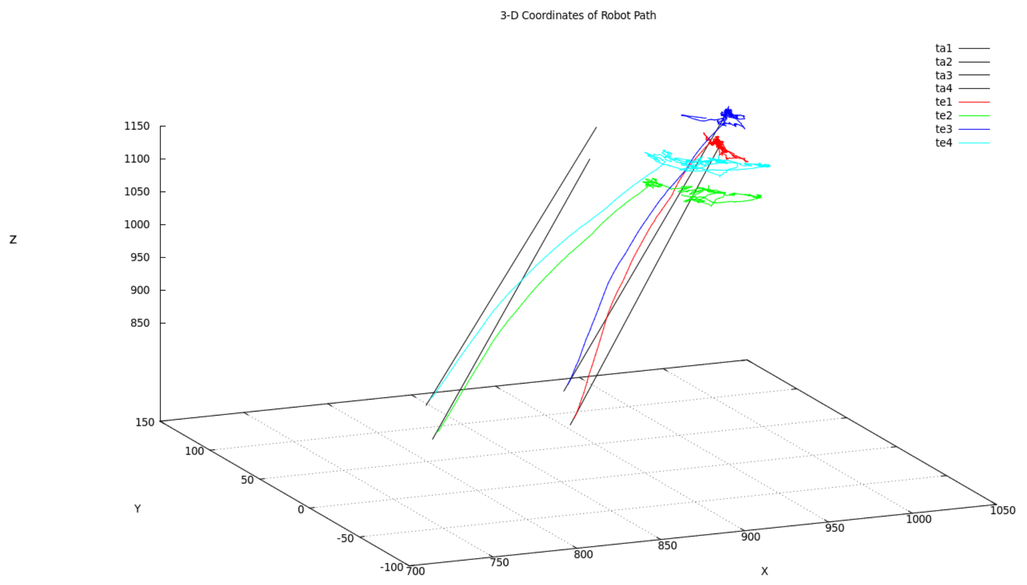


Figure 55: 3-D coordinates using DAKF controller $Q = I$ with one high-cost camera (Camera 1) and two low-cost cameras — failure to converge to static target

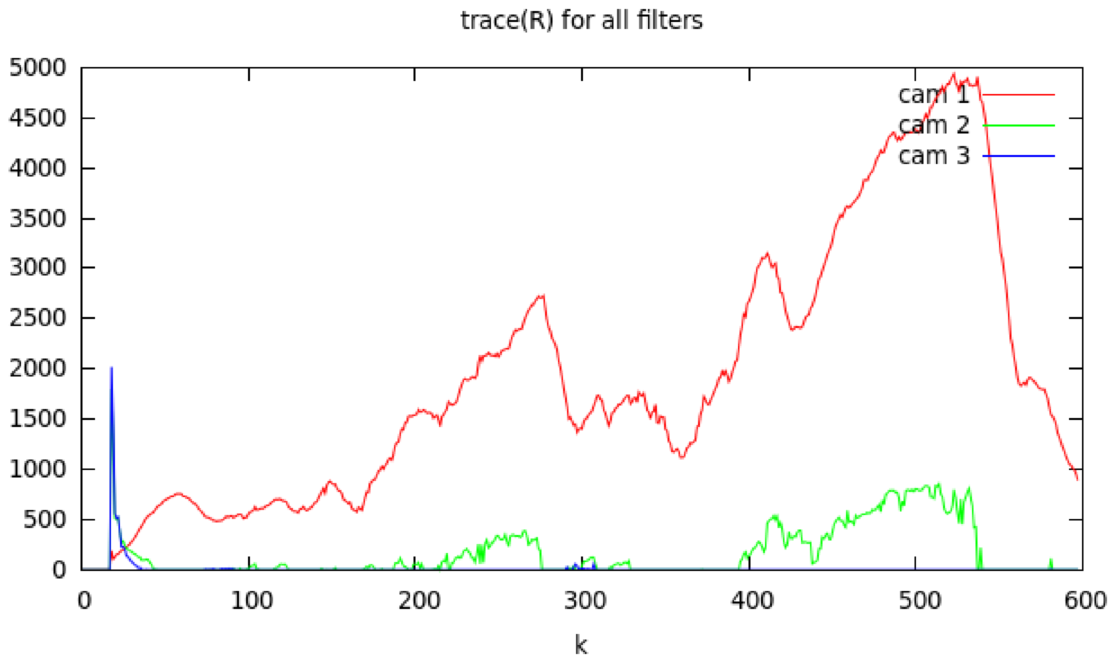


Figure 56: $\text{trace}\left(\widehat{R}^{(i)}\right)$ (pixels^2) using first order DAKF $Q = I$ and $N = 20$ with $C_H = 1$ ("cam 1") — failure to converge to static target

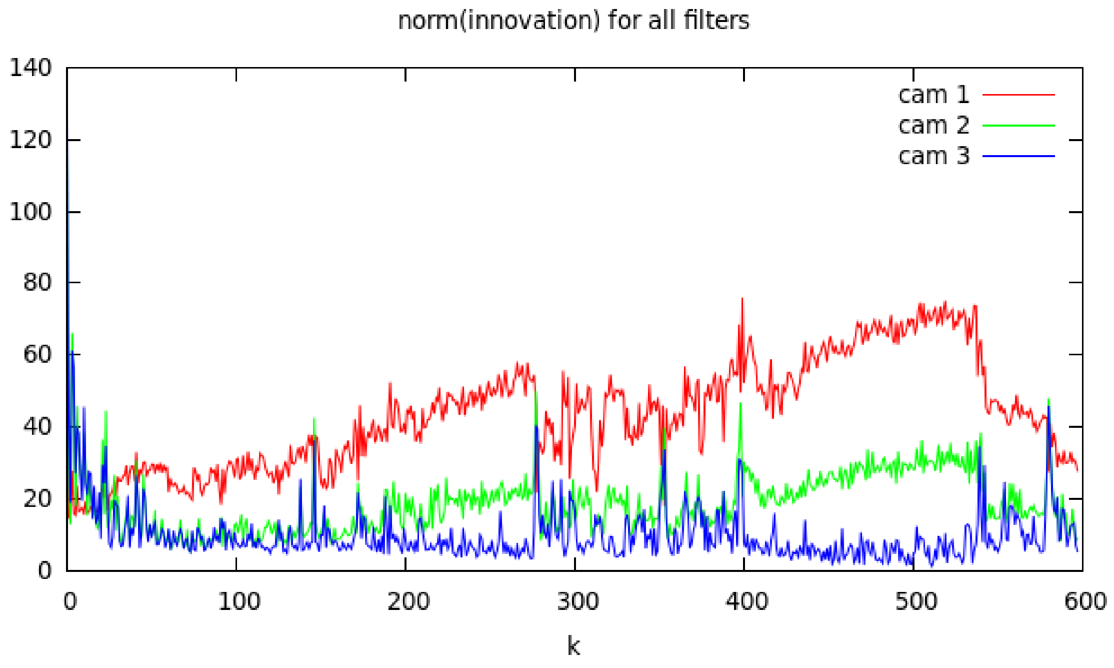


Figure 57: Innovation $\boldsymbol{\nu}_k^{(i)}$ (pixels) using first order DAKF $Q = I$ and $N = 20$ with $C_H = 1$ ("cam 1") — failure to converge to static target

- §7.4 demonstrates that the multiple-camera VS control methods of Chapter 5 continue servoing in the event of camera failure.
- Using input has little effect on the KF method (for example, Figure 20).
- For heterogeneous camera systems tracking a moving target:
  - Figure 20 shows that larger Q and smaller R yield lower tracking error.
  - Figure 20 shows that DAKF is more sensitive to too-low Q than DKF and exhibits greater effects from using input.
  - Zeroth and first order systems perform about equally as well.
  - First order DAKF with $Q = 5I$ and $N = 20$ has lower tracking error than GN and DKF in all camera scenarios tested.
- For heterogeneous camera systems tracking a static target:
  - Figure 28 shows that DKF is less sensitive to Q than with a moving target.

  - Figure 29 demonstrates that DAKF with small N converges faster than with large N and that DAKF is insensitive to camera noise compared to GN.
  - DKF and DAKF formulations with $Q = 5I$ perform better than Gauss-Newton in all camera scenarios tested.

- Tables 4 and 5 show that DAKF improves average performance and also improves overall system stability, exhibited by smaller outliers.
- Greater gains are had using DAKF with static targets than with moving targets and when $C_H = 1$ than for other camera scenarios.
- It can be seen from Figures 20, 30, and 31 that large window size N is good for a highly dynamic system.
- Figure 38 of §7.5.3 indicates that $\widehat{R}^{(i)}$ is not always dependent on camera model.
- §7.5.3 also shows that using DAKF simplifies parameter selection compared to DKF. Since Gauss-Newton has no parameters it is the simplest to use.
- In §7.5.3.1 it is observed that small Q can improve performance up to a point, beyond which servoing quality rapidly degrades. Conservative estimates of Q are therefore recommended.

These data show visual servoing performance improves with DAKF over a traditional method, though at the price of greater complexity. Experiments are performed in order to obtain further validation, results of which are presented in Chapter 8.

CHAPTER VIII

EXPERIMENTAL RESULTS

Simulation data in Chapter 7 indicate that DAKF can improve visual servoing performance over the traditional Gauss-Newton technique. Experiments are performed using the setup in §6.2. Comparison is made between zeroth order with input DAKF ($Q = 3I$) and GN for several moving- and static-target camera scenarios.

There are some differences between the simulations and experiments. First, the static-target maximum allowable norm for a joint offset command to the robot is $\mu = 2.5$ deg instead of 1.0 deg as in simulations. Another difference is that the physical cameras are identical and located somewhat symmetrically. To better model an unstructured environment, where neither identical cameras nor optimal camera locations might be available, three different levels of added noise are used in the three-heterogeneous-cameras trials: 1 pixel for the first camera, 2 pixels for the second, and 4 pixels for the third camera. The simulations and experiments are different for the intermittent camera trials in that for simulations the outages are random, while for experiments they are fixed (see Table 6). The last difference is that for a given simulation camera scenario (for example, $C_H = 1$ and $C_L = 2$) numerous trials are run with those cameras positioned differently in each, while for an experimental camera scenario five trials are run with the cameras identically positioned.

Simulation and experimental results generally agree: the zeroth order with input DAKF $Q = 3I$ is outperformed by GN for a moving target and DAKF converges more quickly to a static target than GN. Experimental tracking error and iterations to convergence are also on the same scale as in simulations.

The experiments explore three topics:

1. Verification that the hypothesis that accuracy improves with increasing number of cameras

2. Validation of the camera-failure-handling methods of §5.2.1 and §5.4.2

Table 6: Experimental scenarios

	Moving target
1.	One camera, constant operation, no added noise
2.	Three cameras, constant operation, 1 pixel noise added to one camera, 2 pixels for the second camera, and 4 pixels for the third
3.	Four cameras, three of them failing at certain times: iterations 16–65 for one camera, 32–81 for the second, and 48–97 for the third
	Static target
4.	One camera, constant operation, no added noise
5.	Two cameras, constant operation, no added noise
6.	Three cameras, constant operation, 1 pixel noise added to one camera, 2 pixels for the second camera, and 4 pixels for the third

3. Comparison between a decentralized adaptive Kalman filter control law and the Gauss-Newton method

Six different scenarios are used for each of the two control methods and are summarized in Table 6. Scenarios 1 and 4 are visual servoing with one fixed camera. Scenario 5 uses two fixed cameras. Scenarios 2 and 6 have three cameras operating with white noise superimposed on the feature coordinates to model an effect of using a low-cost camera since a smaller (cheaper) sensor has higher noise. The maximum possible magnitude of additive noise is constant for each camera but varies from one camera to the next. Scenario 3 uses four cameras with data from three of them becoming unavailable at different times during servoing. Three cameras are simultaneously offline for several iterations during servoing.

For illustration, the image-plane and Cartesian data from a scenario-4 trial are plotted in Figures 58 and 59. The coordinates used for Figure 58 come directly from the image processing. The 3-D coordinates are from a forward kinematic analysis of the manipulator using the measured $\boldsymbol{\theta}_j$. The sensed paths of the checkerboard corners are shown with colored lines. The target points are black in Figure 58, but they are practically obscured by the colored lines of the servoing path. In Figure 59 black lines represent the checkerboard path that is ideal in both image space and Cartesian space. The legend refers to the four target

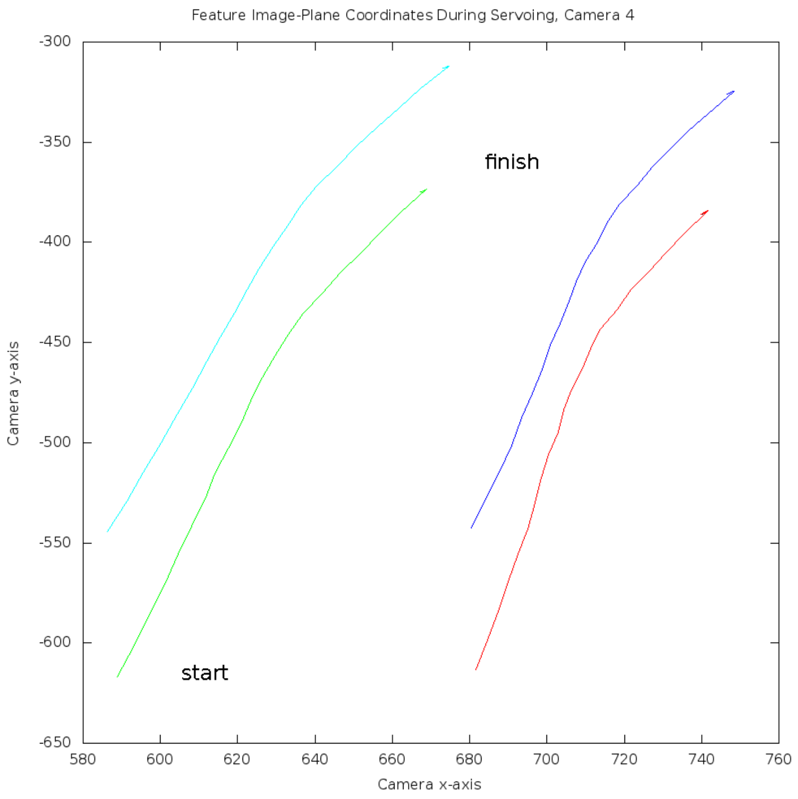


Figure 58: Image feature coordinates for the DAKF controller with one camera (Camera 4) — stationary target

points as ta1–ta4 and to the four points on the robot end-effector as te1–te4.

Five trials are performed for each of the scenarios in Table 6 and the performance metrics are averaged for each of the scenarios. Results are presented and discussed for each.

8.1 Moving-Target Trials

Figure 60 shows the paths traveled by the checkerboard corners for scenario 1 with the Gauss-Newton controller. The largest errors are visible just after the corners of the target path, where the Jacobian estimate becomes less accurate. A simulated trial using two high-cost cameras shown in Figure 15 looks similar, indicating fidelity in the simulation.

For the moving-target scenario the error norms for the checkerboard corners $\bar{e}_{\text{check}}$ (89) are averaged over the iterations of the five trials. There are a fixed number of iterations per trial so repeatable periods of camera outages are possible over different trials for verifying the camera-failure-handling methods of the Chapter 5. The camera inclusion threshold α

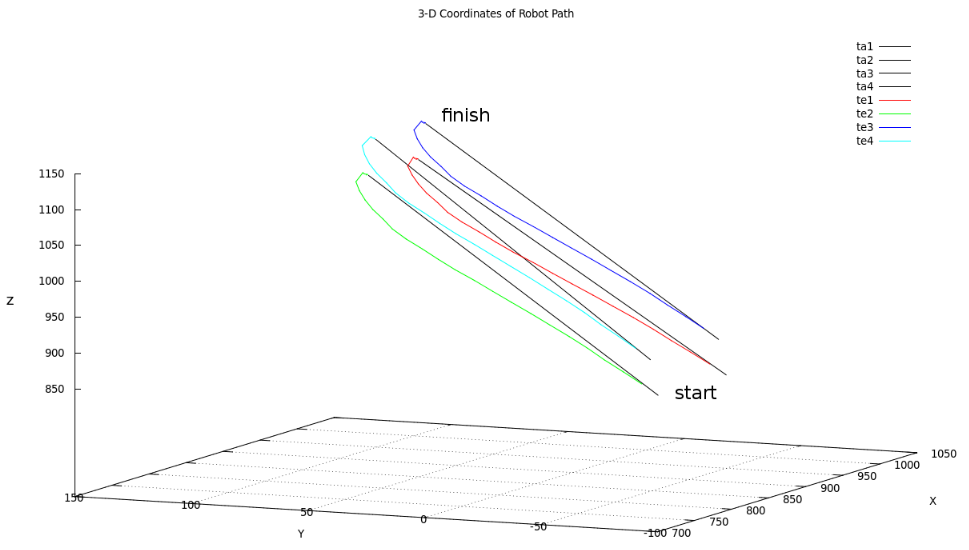


Figure 59: 3-D coordinates for the DAKF controller, scenario 1 (one camera — Camera 4) — stationary target

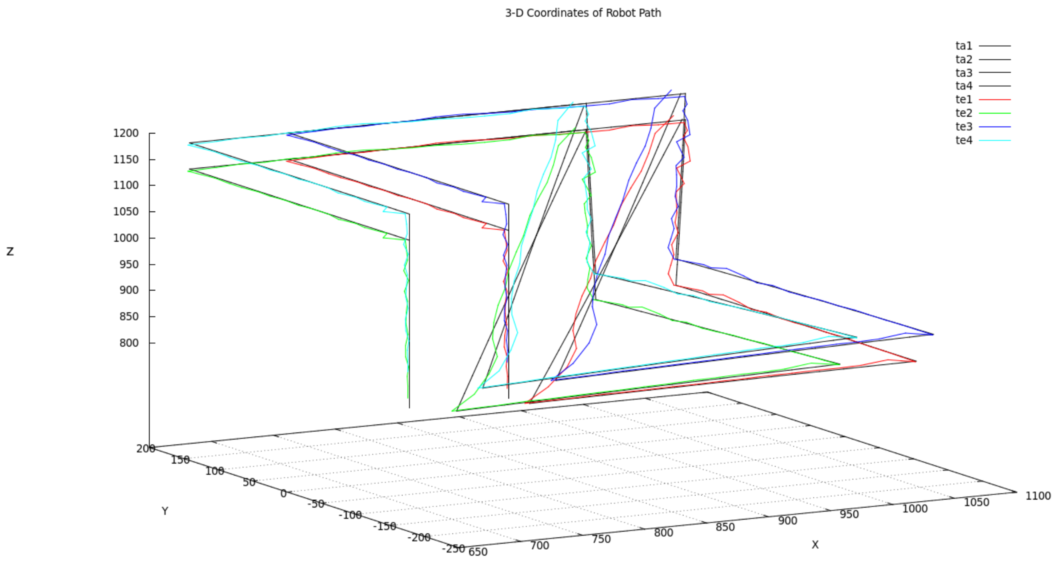


Figure 60: 3-D coordinates using GN controller, scenario 1 (one camera — Camera 4) — moving target

Table 7: Experimental results for moving target

	Scenario 1		Scenario 2		Scenario 3	
	DAKF	GN	DAKF	GN	DAKF	GN
Checkerboard error norm (mm)	18.2	10.7	21.7	18.3	18.3	15.5

in (61) is set to 0.98.

The results for the camera scenarios with a moving target are presented in Table 7. These data do not facilitate conclusions about the relationship between tracking error and number of cameras because the multiple-camera scenarios have extra variables, namely added noise and camera failure episodes.

Comparing scenarios 1 and 3, Table 7 shows that the addition of intermittent cameras slightly improves the Cartesian tracking error for the DAKF controller, while it degrades the performance of the Gauss-Newton controller. This shows that the methods described in §5.2.1 and §5.4.2 for handling camera failure provide stable servoing to a moving target. The image-plane data from all four cameras are shown in Figure 61 for a scenario-3 trial. The pre-recorded paths of the feature points are drawn with black lines and the locations of the checkerboard corners during servoing are shown as colored points. In this way it is possible to see the times during servoing that data are unavailable from Camera 2, Camera 3, and Camera 4. The corresponding 3-D path is shown in Figure 62. No extraordinary disturbances in the path are visible from camera failure, indicating system reliability.

It is expected from theory and simulations that zeroth order with input DAKF (27)–(29) is less capable than GN for moving targets since it imposes a stationary-target assumption on the process equation. Direct comparison between the two controllers in Table 7 shows that GN outperforms DAKF for each of the camera scenarios tested ($C = C_H = 1$; $C = C_H + C_L = 3$, $C_H = 1$; and $C_H = 4$, intermittent), which agrees with theory and simulation.

Table 8 shows average tracking error from simulations and experiments for their common moving-target camera scenarios: one high-cost camera (scenario 1) and three heterogeneous cameras (scenario 2). Differences between simulation and experiment range from 2% to 19%. Since the simulation data are averages over 100 different camera layouts while experimental

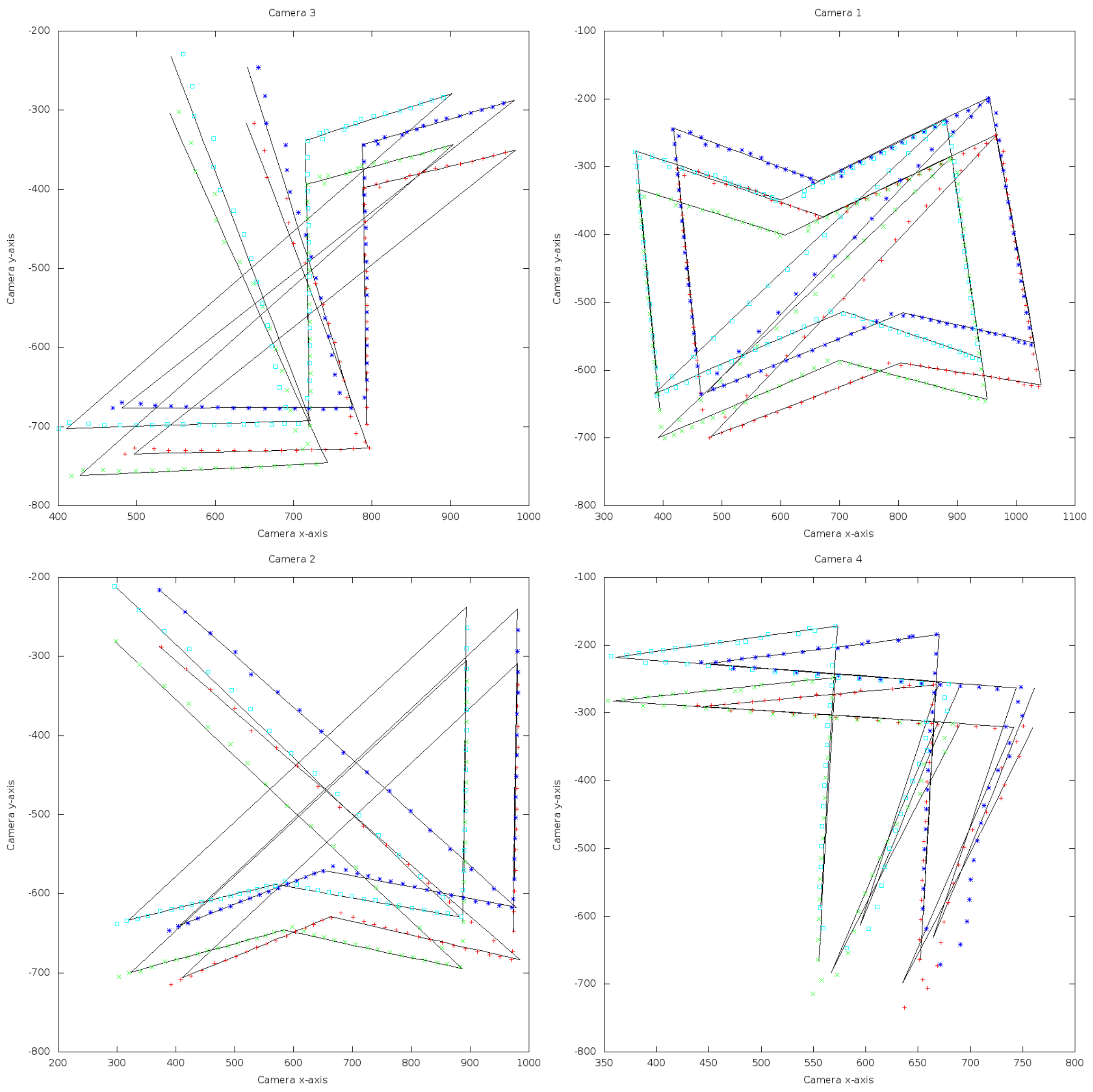


Figure 61: Feature image coordinates for the DAKF controller, scenario 3 (four cameras, three of which — Camera 2, Camera 3, and Camera 4 — go offline for portions of servoing) — moving target

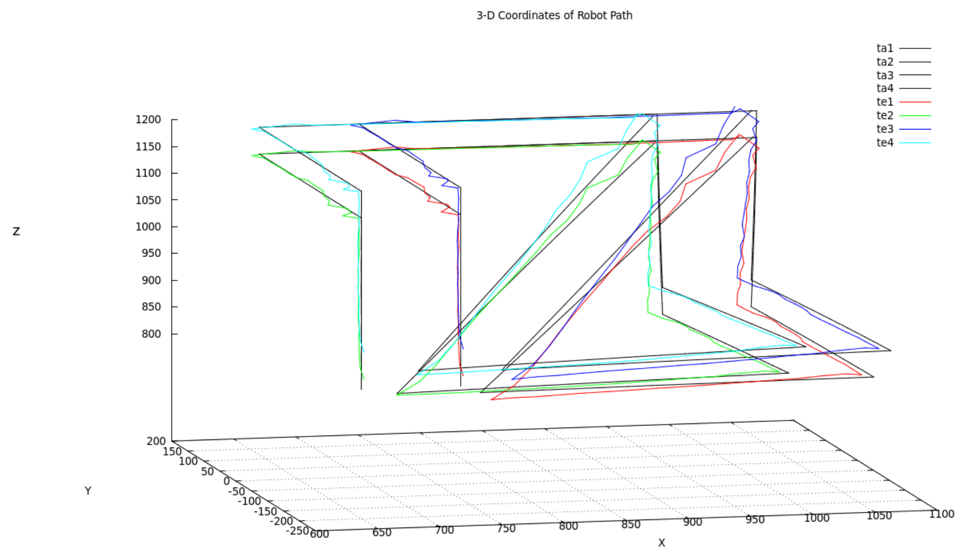


Figure 62: 3-D coordinates for the DAKF controller, scenario 3 (four cameras, three of which — Camera 2, Camera 3, and Camera 4 — go offline for portions of servoing) — moving target

Table 8: Simulated and experimental average tracking error (mm) for similar moving-target scenarios

	Scenario 1		Scenario 2	
	DAKF	GN	DAKF	GN
Simulation	22.4	9.9	18	17.8
Experiment	18.2	10.7	21.7	18.3

data are averages over five trials in a single camera layout and since Table 4 indicates that controller performance varies greatly with camera layout a 19% difference is reasonable. This, combined with the small differences for other cases suggests simulation validity.

8.2 Static-Target Trials

To gauge the error reduction from multiple cameras the final Cartesian error norm is calculated as the distance between the checkerboard center when the image-plane stopping criterion is met (at that point the robot position is $\boldsymbol{\theta} = \boldsymbol{\theta}_S$) and at the pre-recorded location ($\boldsymbol{\theta} = \boldsymbol{\theta}^*$). The final error norm is compared for the one- and two-camera cases

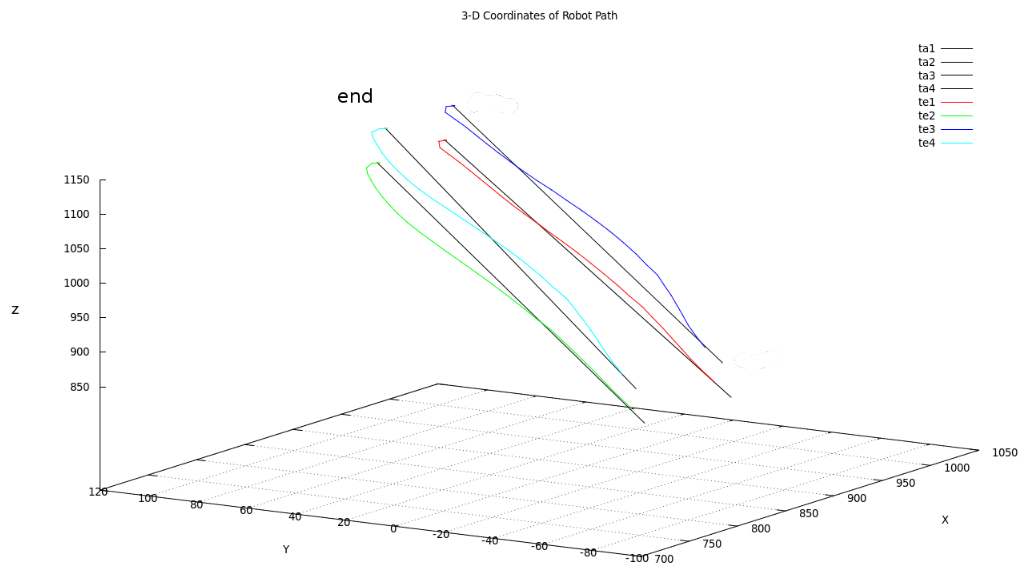


Figure 63: 3-D coordinates for the DAKF controller, scenario 5 (two cameras — Camera 2 and Camera 4) — stationary target

(scenarios 4 and 5) with the stopping threshold $\epsilon = 1$ pixel. The results are 2.0 mm for scenario 4 in Figure 59 and 0.5 mm for scenario 5 in Figure 63. The figures show that two cameras yield a servoing path closer to linear, which is shortest in both image space and Cartesian space.

The performance metrics for comparing the DAKF and Gauss-Newton for a stationary target are the number of iterations S required for convergence (that is, image-plane error norm below the threshold value $\epsilon = 10$ pixels) and d, the norm of the absolute distance traveled in joint space.

$$d = \sum_{j=1}^{N} \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j-1}\| \tag{90}$$

The results in Table 9 show comparable performance between the controllers in scenarios 4 and 5 and a marked improvement with DAKF over Gauss-Newton for scenario 6. This, too agrees with simulations.

The image-plane coordinates for scenario-6 trials are plotted in Figures 64 and 65 for the Gauss-Newton and DAKF controllers. Examining $\boldsymbol{y}_j^*$ (shown in black) for each camera

Table 9: Experimental results for stationary target

	Scenario 4		Scenario 5		Scenario 6	
	DAKF	GN	DAKF	GN	DAKF	GN
Iterations to convergence	23.8	24.6	28.6	27.2	25.8	35.6
Total joint distance norm (deg)	52.8	52.7	61.7	58.9	60	83.2

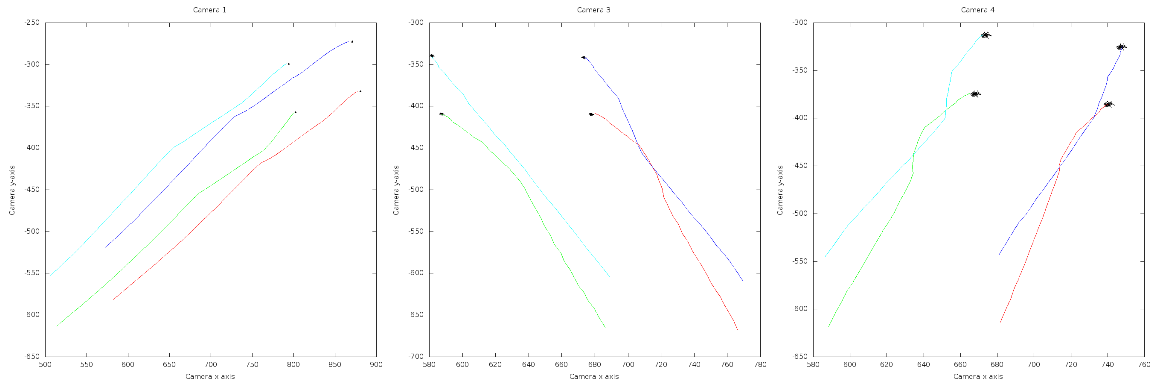


Figure 64: Feature image coordinates using GN controller with three noisy cameras (Camera 1, Camera 3, and Camera 4) — stationary target

illustrates the different magnitudes of the noise added (Camera 4 is noisier than Camera 3, which is noisier than Camera 1). The 3-D data are depicted in Figures 66 and 67. To allow for the additive noise in this scenario $\epsilon = 10$ pixels, otherwise the stopping criterion is unlikely to be met. This relatively high threshold value explains the gap between the goal position and final robot position evident in the figures.

Comparing Figures 64 and 66 with 65 and 67 reveals benefits of DAKF for multiple-camera static-target VS in that the GN path strays much farther from the ideal (straight line) than does the DAKF path.

8.3 Summary

Experimental results are in general agreement with simulation data:

- Zeroth order with input DAKF servos worse than Gauss-Newton for a moving target but better than GN for a stationary target.
- Positional accuracy increases with additional cameras for a static target.

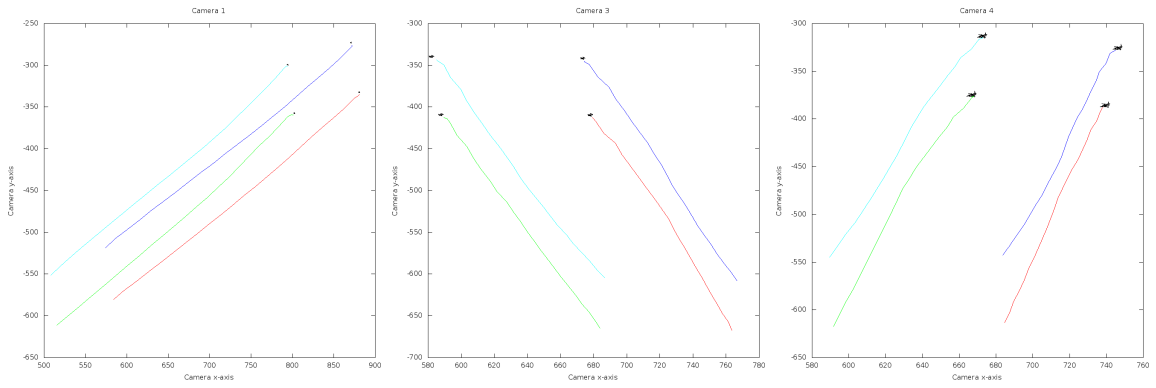


Figure 65: Feature image coordinates for the DAKF controller with three noisy cameras (Camera 1, Camera 3, and Camera 4) — stationary target

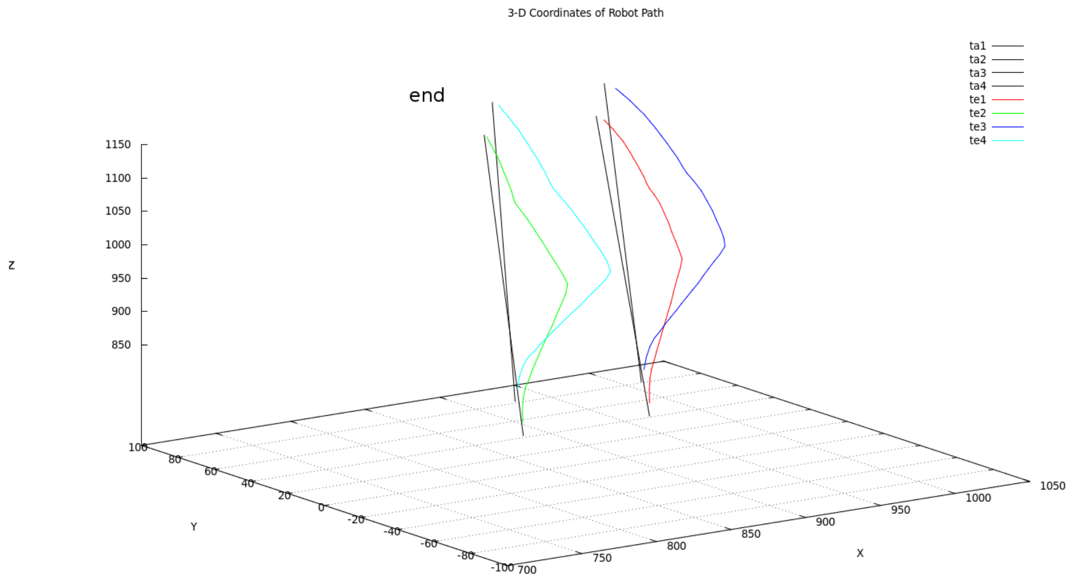


Figure 66: 3-D coordinates using GN controller with three noisy cameras (Camera 1, Camera 3, and Camera 4) — stationary target

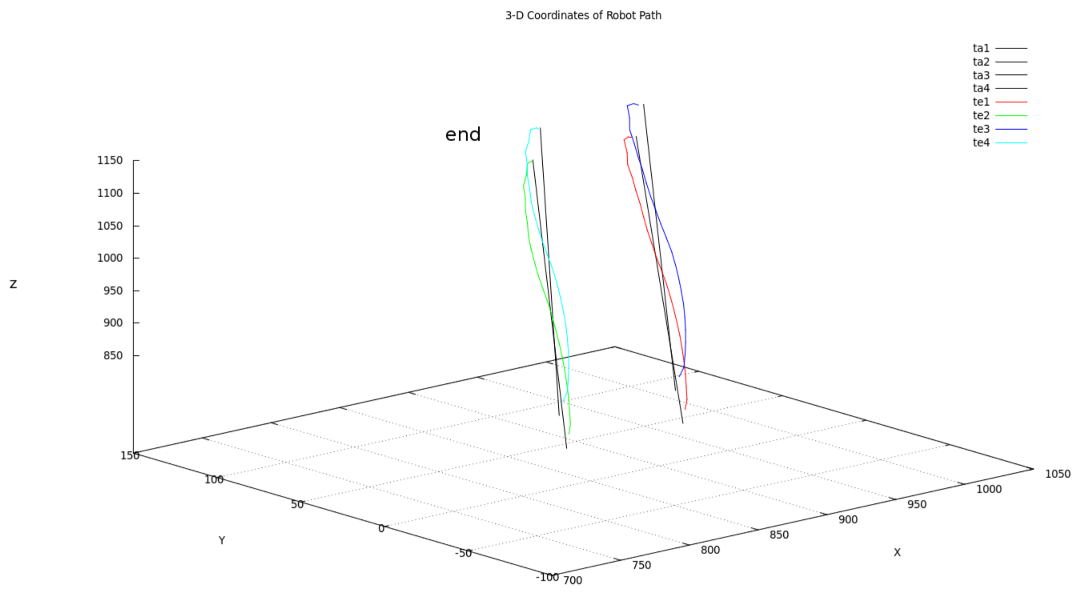


Figure 67: 3-D coordinates for the DAKF controller with three noisy cameras (Camera 1, Camera 3, and Camera 4) — stationary target

- The camera-failure handling methods of Chapter 5 engender system survivability.

Particular experimental data points based on a single camera layout offer poor correspondence to simulation data averaged over numerous layouts but VS performance can be sensitive to camera placement so this is unsurprising. Other data points in Table 8 show close correspondence between experiment and simulation. These results serve to bolster the conclusions drawn in Chapter 7.

CHAPTER IX

CONCLUDING REMARKS

A Kalman filter based visual servo control method is developed and analyzed. Four formulations are discussed and tested in simulation that vary in state space order and the use of input. The Kalman approach employs an adaptive filtering technique to weight multi-camera data to simplify tuning and provide superior performance over non-adaptive KF. The controller is decentralized for system survivability in the event of camera occlusion or failure.

Numerous control laws exist in the uncalibrated VS literature. Few options are available for uncalibrated VS, with a Gauss-Newton based method being the de facto standard. Comparison is made between this conventional VS control law and the decentralized adaptive Kalman filter approach, and with sufficiently high estimates of the covariance matrix Q, the first order adaptive Kalman filter is shown to outperform GN in all camera and target scenarios tested.

The Kalman filter has previously been applied in position based visual servoing for target pose estimation and in uncalibrated VS for Jacobian estimation. In this work Jacobian estimation is assumed. The recursive least squares technique of Piepmeier [59] is used in the simulations and experiments.

9.1 Major Contributions

This thesis makes the following specific contributions:

1. Developing an uncalibrated visual servoing control method using the Kalman filter that outperforms Gauss-Newton for both moving and static targets in the presence of noise

2. Applying an adaptive filter technique to visual servoing allowing camera weighting without any prior knowledge of robot or camera parameters

3. Implementing the above in a decentralized architecture to allow continuous servoing with occlusions or camera failure
4. Experimentally verifying the decentralized adaptive Kalman filter control method and comparing it to the Gauss-Newton technique
5. Exploring the relationship between uncalibrated VS performance and number of cameras

9.2 Controllability and Observability

The Kalman filter application to uncalibrated visual servoing is shown to be stable (§3.2.5.1). It estimates the joint-space offset between the current robot position and the position where the image plane error is minimized. The system is completely controllable and observable under the following assumptions:

- Within the robot dynamic limits and workspace, the joint-level servo control can attain any desired position in one time step.
- The servoing path starts with the robot in a full column rank configuration (that is, not in a kinematic singularity) and passes through another full column rank configuration at least once during servoing.

9.3 Noise Estimates

To generate optimal estimates the Kalman filter requires accurate descriptions of the noise covariances. The process noise covariance Q represents errors in the robot-target motion assumptions while the measurement noise R is due to many sources such as inaccuracy of the Jacobian estimate in the neighborhood of $\boldsymbol{\theta}_k$. In order to achieve improved performance and to provide camera weighting an adaptive filtering technique is applied to the Kalman filter visual servoing method. It is shown to yield lower tracking error, faster convergence, and simplified filter design compared to a non-adaptive Kalman filter.

Four adaptive filtering approaches are mentioned in Chapter 4, with close attention paid to covariance-matching methods. The suitability to VS of several such methods is explored.

The chosen technique is from Mehra [52], with modifications made to assure positive definite matrices. Rather than attempting to adapt Q and R simultaneously, adaptation is provided only for R. This is well suited to a decentralized architecture with a common Q and local adaptation of the covariance matrix $R^{(i)}$ associated with each camera.

9.4 Decentralized Architecture

The final step in the Kalman filter uncalibrated visual servoing control method is to implement a decentralized architecture for robustness. Assuming measurement noise that is uncorrelated between cameras, optimal fusion equations allow for camera failure during servoing.

9.5 Simulation Results

From simulation results it is concluded that:

- The Kalman filter formulations using input (whether joint offset $\boldsymbol{h}_\theta$ or joint velocity change $\boldsymbol{h}_\omega$) perform, on average, no better than those without.
- For moving targets, low-cost cameras are unable to provide tracking comparable to just two high-cost cameras.
- For static targets, maximum positional accuracy is possible with either two high-cost cameras or six low-cost cameras.
- Camera occlusion is successfully handled via the decentralized architecture.
- DAKF provides performance superior to the conventional Gauss-Newton approach in the presence of noise, especially if there is a high-fidelity camera in the system.
  - Tables 4 and 5 and Figures 27 and 33 show that DAKF improves average performance and also improves overall system stability, exhibited by smaller outliers.
- The DAKF controller is simpler to design and yields better servoing than a non-adaptive version.

- The adapted local measurement covariance $R^{(i)}$ depends not only on image noise and focal length but also on camera location.

9.6 Experimental Results

Experiments are performed on the setup described in Chapter 6 that comprises a 6 DoF robot arm, the industrial robot controller, four cameras, and one personal computer implementing image processing and VS control. This thesis reports the first uncalibrated visual servoing experiments using more than two cameras simultaneously. Experimental data are in general agreement with simulation data:

- Zeroth order with input DAKF servos worse than Gauss-Newton for a moving target (Table 7) but better than GN for a stationary target (Table 9).
- Positional accuracy increases with additional cameras for a static target: 2.0 mm with a single high-cost camera and 0.5 mm with two (for the stopping threshold $\epsilon = 1$ pixel). This result is a function only of camera geometry and so is independent of control method.

Thus credence is lent to the conclusions from Chapter 7.

9.7 Summary

In summary, uncalibrated visual servoing can potentially extend robot usage to unstructured environments where calibrated systems can completely fail after reconfiguration such as a change in camera position [59]. Multiple cameras are known to improve accuracy of uncalibrated VS. The decentralized adaptive Kalman filter formulation in this thesis exploits the advantages of multiple cameras by weighting cameras and allowing for occlusion.

9.8 Recommendations

The following practices are recommended for DAKF implementation:

- Do not use input in process equation because it imposes assumptions on target and/or robot motion with little performance gain even when those assumptions are valid.

- Regarding filter design, a recommendation is that Q be set conservatively high to avoid poor servoing results. For example, $\beta \geq 5$ for $Q = \beta I$.
- Increasing target dynamis should be treated with larger window size N in the adaptive technique, ranging between $N = 12$ and $N = 20$.

9.9 Future work

In order to assure a positive definite measurement noise covariance matrix an ad-hoc technique is presented here. Further performance improvements could be seen by instead implementing the new autocovariance least-squares (ALS) method of Odelson et al. [57]. This solves a constrained ALS problem to ensure positive semidefinite estimates and it estimates both Q and R with more accurate covariances than classical methods. This would obviate the need for DAKF tuning and could yield better performance since it is shown in this thesis that Q selection is important. ALS has not been applied to a multisensor system so adapting it to a decentralized architecture where local filters share a common process model and Q but have individual R matrices represents a challenge.

The adaptive measurement covariances (camera weighting) could help determine optimal camera placement.

The VS control method developed in this thesis uses a fully-connected decentralized Kalman filter. Using the Information filter instead would allow for a non-fully-connected system with the same estimate and minimal communication. Adaptive methods for the Information filter formulation could be explored in future work.

The decentralized architecture currently allows for camera failure or occlusion by removing that filter from the summations in (87) and (86). This does not consider the possibility of partial occlusion. If redundant image features are tracked in a camera then with partial occlusion the associated filter may still be able to compute a local joint error estimate ($\hat{\boldsymbol{\phi}}$) using some technique for Kalman filtering with intermittent data. Applying such a technique could further improve system reliability.

The reported gains from DAKF are more pronounced when servoing to a static target than when tracking a moving one. A possible reason is that static-target trials use a smaller joint offset limit (μ) than moving-target trials. Reducing the control period could improve moving-target performance by having smaller robot and target motions between steps, providing more measurements for the state estimate.

Multiple cameras have been shown to improve accuracy of uncalibrated visual servoing and though further improvements are possible the Kalman filter based approach presented here goes further in maximizing this multi-camera potential than a traditional VS control method.

REFERENCES

[1] Akesson, B. M., Jrgensen, J. B., Poulsen, N., and Jorgensen, S. B., "A generalized autocovariance least-squares method for kalman filter tuning," *Journal of Process Control*, vol. 18, no. 78, pp. 769 – 779, 2008.

[2] Almagbile, A., Wang, J., and Ding, W., "Evaluating the performances of adaptive kalman filter methods in gps/ins integration," *Journal of Global Positioning Systems*, vol. 9, pp. 33–40, 2010.

[3] Andreff, N., Espiau, B., and Horaud, R., "Visual servoing from lines," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 3, pp. 2070 –2075 vol.3, 2000.

[4] Assa, A., Janabi-Sharifi, F., Moshiri, B., and Mantegh, I., "A data fusion approach for multi-camera based visual servoing," in *Optomechatronic Technologies (ISOT), 2010 International Symposium on*, pp. 1 –7, oct. 2010.

[5] Bavdekar, V. A., Deshpande, A. P., and Patwardhan, S. C., "Identification of process and measurement noise covariance for state and parameter estimation using extended kalman filter," *Journal of Process Control*, vol. 21, no. 4, pp. 585 – 601, 2011.

[6] Bilen, H., Hocaoglu, M., Ozgur, E., Unel, M., and Sabanovic, A., "A comparative study of conventional visual servoing schemes in microsystem applications," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 1308 –1313, 29 2007-nov. 2 2007.

[7] Blanchet, I., Frankignoul, C., and Cane, M., "A comparison of adaptive kalman filters for a tropical pacific ocean model," *Monthly Weather Review*, vol. 125, pp. 40–58, 1997.

[8] Bonkovic, M., Hace, A., and Jezernik, K., "Population-based uncalibrated visual servoing," *Mechatronics, IEEE/ASME Transactions on*, vol. 13, pp. 393 –397, june 2008.

[9] Brogan, W. L., *Modern Control Theory Third Edition.* Prentice Hall, 1991.

[10] Carusone, J. and D'Eleuterio, G., "The feature cmac: a neural-network-based vision system for robotic control," *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 2959–2964, 16-20 1998.

[11] Chang, C.-F., Tsai, C.-C., Hsu, J.-C., and Lin, C.-C., "Laser pose tracking for a mobile robot using fuzzy adaptive extended information filtering," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, pp. 2471 – 2476 vol.3, june 2003.

[12] Chaumette, F. and Hutchinson, S., "Visual servo control, part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 82–90, Dec. 2006.

[13] Chaumette, F. and Hutchinson, S., “Visual servo control, part II: Advanced approaches,” *IEEE Robotics and Automation Magazine*, vol. 14, pp. 109–118, Mar. 2007.

[14] Chaumette, F. and Rives, P., “Vision-based-control for robotic tasks,” in *Intelligent Motion Control, 1990. Proceedings of the IEEE International Workshop on*, vol. 2, pp. 395 –400, aug 1990.

[15] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S., *Principles of Robot Motion Theory, Algorithms, and Implementations.* MIT, 2005.

[16] Collewet, C., Marchand, E., and Chaumette, F., “Visual servoing set free from image processing,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 81 –86, may 2008.

[17] Cowan, N., Weingarten, J., and Koditschek, D., “Visual servoing via navigation functions,” *Robotics and Automation, IEEE Transactions on*, vol. 18, pp. 521 – 533, aug 2002.

[18] Dame, A. and Marchand, E., “Improving mutual information-based visual servoing,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 5531 –5536, may 2010.

[19] Deng, Z. and Jagersand, M., “Evaluation of model independent image-based visual servoing,” in *Computer and Robot Vision, 2004. Proceedings. First Canadian Conference on*, pp. 138 – 144, 17-19, 2004.

[20] Drolet, L., Michaud, F., and Cote, J., “Adaptable sensor fusion using multiple kalman filters,” in *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 2, pp. 1434–1439 vol.2, 2000.

[21] El-Hawary, F., “A comparison of recursive weighted least squares estimation and kalman filtering for source dynamic motion evaluation,” in *OCEANS ’89. Proceedings*, vol. 4, pp. 1082 –1086, sep 1989.

[22] Escamilla-Ambrosio, P. and Mort, N., “Hybrid kalman filter-fuzzy logic adaptive multisensor data fusion architectures,” in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 5, pp. 5215 – 5220 Vol.5, dec. 2003.

[23] Espiau, B., Chaumette, F., and Rives, P., “A new approach to visual servoing in robotics,” *Robotics and Automation, IEEE Transactions on*, vol. 8, pp. 313 –326, jun 1992.

[24] Ficocelli, M. and Janabi-Sharifi, F., “Adaptive filtering for pose estimation in visual servoing,” in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, pp. 19 –24 vol.1, 2001.

[25] Flandin, G., Chaumette, F., and Marchand, E., “Eye-in-hand/eye-to-hand cooperation for visual servoing,” in *Robotics and Automation, 2000. Proceedings. ICRA ’00. IEEE International Conference on*, vol. 3, pp. 2741 –2746 vol.3, 2000.

[26] Fu, Q., Zhang, Z., and Shi, J., "Uncalibrated visual servoing using more precise model," in *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, pp. 916 –921, sept. 2008.

[27] Gao, Z. and Su, J., "Switch images based on fusion in uncalibrated visual servoing," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IEEE Cat. No. 06CH37780D)*, pp. 3803–8, 2006.

[28] Gumpert, B. A., "A recursive gauss-newton method for model independent eye-in-hand visual servoing," Master's thesis, Georgia Institute of Technology, 2001.

[29] Han, J., Kim, D., and Sunwoo, M., "State-of-charge estimation of lead-acid batteries using an adaptive extended kalman filter," *Journal of Power Sources*, vol. 188, no. 2, pp. 606 – 612, 2009.

[30] Hao, M. and Sun, Z., "A universal state-space approach to uncalibrated model-free visual servoing," *Mechatronics, IEEE/ASME Transactions on*, vol. PP, no. 99, pp. 1 –14, 2011.

[31] Hao, M., Deuflhard, P., Sun, Z., and Fujii, M., "Model-free uncalibrated visual servoing using recursive least squares," *Journal of Computers*, vol. 3, no. 11, 2008.

[32] Hashimoto, H., Kubota, T., Kudou, M., and Harashima, F., "Self-organizing visual servo system based on neural networks," *Control Systems Magazine, IEEE*, vol. 12, pp. 31–36, Apr 1992.

[33] Hide, C., Moore, T., and Smith, M., "Adaptive kalman filtering algorithms for integrating gps and low cost ins," in *Position Location and Navigation Symposium, 2004. PLANS 2004*, pp. 227 – 233, april 2004.

[34] Hosoda, K. and Asada, M., "Versatile visual servoing without knowledge of true jacobian," *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, vol. 1, pp. 186–193, 12-16 1994.

[35] Ip, Y. L., Rad, A. B., Wong, Y. K., Liu, Y., and Ren, X. M., "A localization algorithm for autonomous mobile robots via a fuzzy tuned extended kalman filter.," *Advanced Robotics*, vol. 24, no. 1/2, pp. 179 – 206, 2010.

[36] Jagersand, M., "Visual servoing using trust region methods and estimation of the full coupled visual-motor jacobian," *IASTED Applications of Robotics and Control*, 1996.

[37] Jagersand, M., Fuentes, O., and Nelson, R., "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4, pp. 2874 –2880 vol.4, apr 1997.

[38] Jwo, D.-J. and Chang, F.-I., "A fuzzy adaptive fading kalman filter for gps navigation," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues* (Huang, D.-S., Heutte, L., and Loog, M., eds.), vol. 4681 of *Lecture Notes in Computer Science*, pp. 820–831, Springer Berlin / Heidelberg, 2007.

[39] Jwo, D.-J. and Weng, T.-P., "An adaptive sensor fusion method with applications in integrated navigation," *Journal of Navigation*, vol. 61, pp. 705–721, 2008.

[40] Khnlenz, K. and Buss, M., "On sensor switching visual servoing," *International Journal of Optomechatronics*, vol. 2, no. 3, pp. 233–256, 2008.

[41] Lamiroy, B., Espiau, B., Andreff, N., and Horaud, R., "Controlling robots with two cameras: how to do it properly," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2100–2105, 2000.

[42] Lin, H.-H., Tsai, C.-C., Hsu, J.-C., and Chang, C.-F., "Ultrasonic self-localization and pose tracking of an autonomous mobile robot via fuzzy adaptive extended information filtering," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, pp. 1283 – 1290 vol.1, sept. 2003.

[43] Lippiello, V., Siciliano, B., and Villani, L., "Eye-in-hand/eye-to-hand multi-camera visual servoing," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pp. 5354 – 5359, dec. 2005.

[44] Lippiello, V., Siciliano, B., and Villani, L., "Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration," *Robotics, IEEE Transactions on*, vol. 23, pp. 73 –86, feb. 2007.

[45] Lippiello, V., Siciliano, B., and Villani, L., "Adaptive extended kalman filtering for visual motion estimation of 3d objects," *Control Engineering Practice*, vol. 15, no. 1, pp. 123 – 134, 2007.

[46] Liu, Y.-H., Wang, H., Wang, C., and Lam, K. K., "Uncalibrated visual servoing of robots using a depth-independent interaction matrix," *Robotics, IEEE Transactions on*, vol. 22, pp. 804 –817, aug. 2006.

[47] Lv, X. and Huang, X., "Fuzzy adaptive kalman filtering based estimation of image jacobian for uncalibrated visual servoing," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 2167 –2172, oct. 2006.

[48] Malis, E., "Improving vision-based control using efficient second-order minimization techniques," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, pp. 1843 – 1848 Vol.2, 26-may 1, 2004.

[49] Malis, E., Borrelly, J.-J., and Rives, P., "Intrinsics-free visual servoing with respect to straight lines," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, pp. 384 – 389 vol.1, 2002.

[50] Marshall, M., Matthews, M., Hu, A.-P., McMurray, G., and Lipkin, H., "Uncalibrated visual servoing for intuitive human guidance of robots," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 4463 –4468, may 2012.

[51] Mehra, R., "On the identification of variances and adaptive kalman filtering," *Automatic Control, IEEE Transactions on*, vol. 15, pp. 175 – 184, apr 1970.

[52] Mehra, R., "Approaches to adaptive filtering," *Automatic Control, IEEE Transactions on*, vol. 17, pp. 693 – 698, oct 1972.

[53] Munnae, *Uncalibrated Robotic Visual Servo Tracking For Large Residual Problems.* PhD thesis, Georgia Institute of Technology, 2010.

[54] Mutambara, A. G., *Decentralized Estimation and Control for Multisensor Systems.* CRC Press, 1998.

[55] Myers, K. and Tapley, B., "Adaptive sequential estimation with unknown noise statistics," *Automatic Control, IEEE Transactions on*, vol. 21, pp. 520 – 523, aug 1976.

[56] Neethling, C. and Young, P., "Comments on "identification of optimum filter steady-state gain for systems with unknown noise covariances"," *Automatic Control, IEEE Transactions on*, vol. 19, no. 5, pp. 623–625, 1974.

[57] Odelson, B. J., Rajamani, M. R., and Rawlings, J. B., "A new autocovariance least-squares method for estimating noise covariances," *Automatica*, vol. 42, no. 2, pp. 303 – 308, 2006.

[58] Pari, L., Sebastian, J., Traslosheros, A., and Angel, L., "A comparative study between analytic and estimated image jacobian by using a stereoscopic system of cameras," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 6208 –6215, oct. 2010.

[59] Piepmeier, J., *A Dynamic Quasi-Newton Method for Model Independent Visual Servoing.* PhD thesis, Georgia Institute of Technology, 1999.

[60] Piepmeier, J., McMurray, G., and Lipkin, H., "A dynamic quasi-newton method for uncalibrated visual servoing," vol. 2, pp. 1595 –1600 vol.2, 1999.

[61] Piepmeier, J., "Experimental results for uncalibrated eye-in-hand visual servoing," pp. 335 – 339, 2003.

[62] Piepmeier, J. and Lipkin, H., "Uncalibrated eye-in-hand visual servoing," *International Journal of Robotics Research*, vol. 22, no. 10-11, pp. 805–19, 2003.

[63] Piepmeier, J., McMurray, G., and Lipkin, H., "Uncalibrated dynamic visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 143–147, feb. 2004.

[64] Qian, J. and Su, J., "Online estimation of image jacobian matrix by kalman-bucy filter for uncalibrated stereo vision feedback," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 562–567, 2002.

[65] Rao, B. and Durrant-Whyte, H., "Fully decentralised algorithm for multisensor kalman filtering," *Control Theory and Applications, IEE Proceedings D*, vol. 138, pp. 413 –420, sep 1991.

[66] Sanderson, A. and Weiss, L., *Robot Vision*, ch. Adaptive Visual Servo control of Robots, pp. 107–116. Springer-Verlag, 1983.

[67] Schuurman, D. and Capson, D., "Robust direct visual servo using network-synchronized cameras," *Robotics and Automation, IEEE Transactions on*, vol. 20, pp. 319 – 334, april 2004.

[68] Sebastian, J., Pari, L., Gonzalez, C., and Angel, L., "A new method for the estimation of the image jacobian for the control of an uncalibrated joint system," *Proceedings of the 2005 IbPRIA Conference on Pattern Recognition and Image Analysis, Part I (Lecture Notes in Computer Science)*, vol. 3522, pp. 631–8, 2005.

[69] Sebastin, J., Pari, L., Angel, L., and Traslosheros, A., "Uncalibrated visual servoing using the fundamental matrix," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 1–10, 2009.

[70] Shirai, Y. and Inoue, H., "Guiding a robot by visual feedback in assembling tasks," *Pattern Recognition*, vol. 5, no. 2, pp. 99–106, IN3, 107–108, 1973.

[71] Stavnitzky, J. and Capson, D., "Multiple camera model-based 3-d visual servo," *Robotics and Automation, IEEE Transactions on*, vol. 16, pp. 732 –739, dec 2000.

[72] Strang, G. and Borre, K., *Linear Algebra, Geodesy, and GPS.* Wellesley-Cambridge, 1997.

[73] Subramanian, V., Burks, T., and Dixon, W. E., "Sensor fusion using fuzzy logic enhanced kalman filter for autonomous vehicle guidance in citrus groves," *Transactions of the ASABE*, vol. 52, pp. 1411–1422, 2009.

[74] Sung, W., Lee, S., and You, K., "Ultra-precision positioning using adaptive fuzzy-kalman filter observer," *Precision Engineering*, vol. 34, no. 1, pp. 195 – 199, 2010. ¡ce:title¿CIRP-CAT 2007¡/ce:title¿.

[75] Takagi, T. and Sugeno, M., "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-15, pp. 116 –132, jan.-feb. 1985.

[76] Triggs, B. and Laugier, C., "Automatic camera placement for robot vision tasks," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 2, pp. 1732 –1737 vol.2, may 1995.

[77] Tsai, C.-C. and Lin, H.-H., "Improved global localization and pose tracking of an autonomous mobile robot via fuzzy adaptive extended information filtering," in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pp. 1813 –1818, oct. 2006.

[78] Wang, H., Liu, Y.-H., and Zhou, D., "Adaptive visual servoing using point and line features with an uncalibrated eye-in-hand camera," *Robotics, IEEE Transactions on*, vol. 24, pp. 843 –857, aug. 2008.

[79] Wu, Z., Rajamani, M., Rawlings, J., and Stoustrup, J., "Application of an autocovariance least - squares method for model predictive control of hybrid ventilation in livestock stables," in *American Control Conference, 2007. ACC '07*, pp. 3630 –3635, july 2007.

[80] Yoshihata, Y., Watanabe, K., Iwatani, Y., and Hashimoto, K., "Multi-camera visual servoing of a micro helicopter under occlusions," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 2615 –2620, 29 2007-nov. 2 2007.

[81] Yoshimi, B. and Allen, P., "Active, uncalibrated visual servoing," *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 156–161, 8-13 1994.

[82] Zhang, G., Wang, B., Wang, J., and Liu, H., "A hybrid visual servoing control of 4 dofs space robot," in *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pp. 3287 –3292, aug. 2009.

[83] Zhang, S.-T. and Wei, X.-Y., "Fuzzy adaptive kalman filtering for dr/gps," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 5, pp. 2634 – 2637 Vol.5, nov. 2003.

VITA

Matthew Marshall was born in DeLand, Florida where he graduated from high school as a National Merit Scholarship finalist. Matthew received a B.S. in Mechanical Engineering from the University of Florida and then earned an M.S. for work in tensegrity mechanisms. Matthew and his wife, Laura, moved to Atlanta, Georgia where he embarked upon Ph.D. work at the Georgia Institute of Technology. In the nonce he worked for GTRI FPTD, Plastic Omnium, and Phoenix Engineering and Consulting, Inc. and taught as an adjunct professor at Southern Polytechnic State University. He's thankful for so much.