Implementation, Modeling, and Exploration of Precision Visual Servo Systems

Zhenyu Ye

2020

This research is supported by the Netherlands Organisation for Scientific Research (NWO).

A catalogue record is available from the Eindhoven University of Technology Library.

Implementation, Modeling, and Exploration of Precision Visual Servo Systems/ by Zhenyu Ye. – Eindhoven : Technische Universiteit Eindhoven, 2020
Proefschrift. – ISBN: 978-94-028-1985-4

Copyright © 2020 by Zhenyu Ye.

This thesis was prepared with the pdfLaTeX documentation system.
Cover Design: Zhenyu Ye
Reproduction: IPSKAMP printing, Enschede, The Netherlands

Implementation, Modeling, and Exploration of Precision Visual Servo Systems

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. F.P.T. Baaijens,
voor een commissie aangewezen door het College voor Promoties,
in het openbaar te verdedigen
op donderdag 26 may 2020 om 16:00 uur

door

Zhenyu Ye

geboren te Huizhou, China

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. L.P.H. de Goey
1^e promotor:	prof.dr. H. Nijmeijer
2^e promotor:	prof.dr.ir. P.P. Jonker
leden:	prof.dr. H. Corporaal
	prof.dr. B. Nelson (ETH Zurich)
	prof.dr. R. Babuska (Technische Universiteit Delft)
	prof.dr.ir. H. Bruyninckx

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Contents

Societal summary

Automation and improvements of efficiency are key enablers of economic growth, a higher standard of living, and well-being of individuals. As we have witnessed in the third industrial revolution, computing and communication technology have significantly improved the productivity of our society. However, growth cannot be sustained without new innovations. More specifically, innovations are needed for tackling certain daunting challenges of automation that were technically infeasible in the past but are ripe for investigation in the short term future. One of these innovations needed is to build **smart machines that can see, think, and act, with a high speed and a high accuracy**. Several key challenges that hinder this innovation are investigated and resolved in this thesis.

These challenges are induced by constraints in the speed of camera, the speed of computation, the speed and accuracy of vision algorithms, and a lack of methods to effectively compensate for these constraints and to find out a good overall solution within these constraints. In this thesis, methods are provided to design high-speed vision systems, to evaluate delay and accuracy of vision algorithms, to design control laws that compensate for these constraints, and to find out among a large number of design options which is most suitable for a specific use case. These methods are applied to an industrial use case, in which they are demonstrated to be effective and offer significant improvements over an ad-hoc design.

By overcoming these challenges, this thesis has demonstrated that it is feasible to build a smart machine with a high speed and a high accuracy, and has provided a set of methods that can help designers achieve this goal. Perhaps such a smart machine is still ahead of its time. However, designers can use the methods provided in this thesis to make incremental improvements on existing machines, and to explore new possibilities in their future products. In such a way, this thesis has made the eventual realization of smart machines one step closer. Hopefully, my fellow researchers, engineers, and entrepreneurs will carry on with this journey and make it part of the next industrial revolution. That is when this thesis brings its full value for our society.

Summary

Implementation, Modeling, and Exploration of Precision Visual Servo Systems

Visual sensing is widely used in mechatronic and robotic systems to observe online dynamics of a system in its environment, which is otherwise difficult, if not impossible, to observe using other sensors. However, despite decades of research, it remains challenging to integrate visual sensing into precision mechatronic and robotic systems.

The challenges are multifold, and several open problems remain. First, visual sensing is computation-intensive and data-intensive, which necessarily involves a trade-off between sample rate, delay, and accuracy. Second, control laws need to be designed to utilize additional information provided by visual feedback and to alleviate drawbacks induced by it. Third, high-speed visual sensing is often achieved by co-design of software and hardware, which hinders an early estimate of its delay prior to implementation. Last but not least, to explore design trade-offs across multiple domains in visual servo systems, a cross-domain model that is comprehensive and analyzable is required.

A set of methods are proposed in this thesis to overcome each of these challenges. First, design methods are proposed for high-speed and high-precision vision systems, and are subsequently applied to an implementation of a prototype. Second, control laws are proposed to utilize online measurements of visual feedback and are optimized for sample rate, delay, and quantization error of visual feedback. Third, algorithmic patterns and architectural templates are used to obtain an early estimation of delay in customized vision systems. Fourth, an axiomatic design method is applied to visual servo systems, which explicitly models cross-domain couplings and enables explorations of cross-domain trade-offs with regard to overall system performance.

Tackling the aforementioned challenges leads to multiple contributions and findings in this thesis. These contributions include methods to design high-speed vision systems for control purposes, methods to design and optimize control laws

for visual feedback, methods to evaluate algorithmic and architectural choices, and methods to model and explore cross-domain trade-offs. These methods are applied to implementations of prototypes, which leads to multiple findings. It is found that, even for a baseline example and for a small set of design parameters, there are non-trivial cross-domain trade-offs, and more importantly, there are significant improvements in system performance that can only be achieved by cross-domain exploration.

In summary, this thesis provides a tool to understand and evaluate fundamental design choices in high-precision and high-speed visual servo systems. This tool can be used to analyze and optimize existing systems, to set up a product family of systems with various trade-offs, and to explore a road map for future generations of systems.

Chapter 1

Introduction

Measure what is measurable, and make measurable what is not so.

Galileo Galilei's rule of measurement

Abstract. *This chapter introduces visual servoing systems, and motives their applications in semi-structured environments. The challenges of designing precision visual servo systems are discussed. To overcome these challenges, research objectives and research methods are defined. Subsequently, the structure of this thesis is summarized.*

1.1 Visual servo systems

Visual servoing is a servo mechanism that uses machine vision to provide measurements for closed-loop motion control (Chaumette and Hutchinson, 2006; Hutchinson et al., 1996). Visual servo systems are commonly defined as robotics or automation systems that apply the method of visual servoing. A typical configuration of a visual servo system is illustrated in Figure 1.1, and explained as follows.

Visual servo systems, using Figure 1.1 as an example, consist of several components that differ from other servo systems. First, image sensors are used to

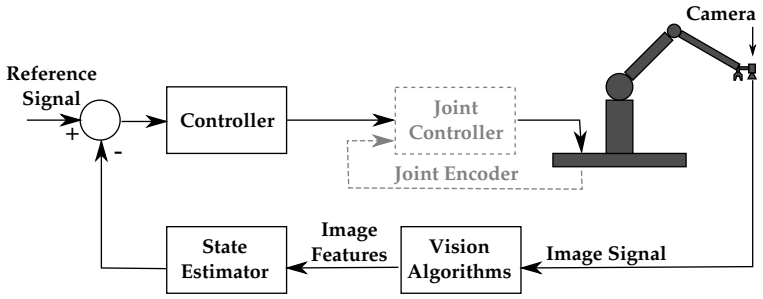


Figure 1.1: Diagram of a typical visual servo system. Some systems can be designed to use only visual feedback (without using joint encoders), which is called direct visual servoing.

Environment	Optical features	Adaptability	Speed
Unstructured	Natural features	High	Low
Semi-structured	Predefined patterns	Medium	Medium
Structured	Grating	Low	High

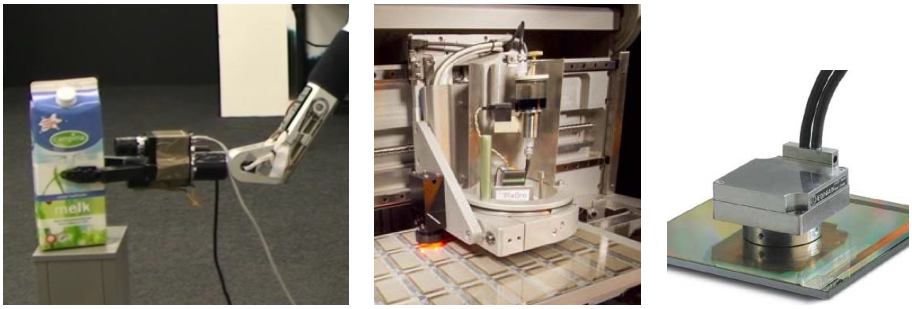
Table 1.1: A comparison of visual sensing in different environments. The semi-structured environment is the focus of this thesis.

provide visual feedback to the control loop. Other sensors like motor encoders can be used as well, but they are not strictly required in a wide range of visual servo systems. Second, compute-intensive processing is performed to extract appropriate feedback from image signals. Third, controllers are designed according to the characteristics of visual feedback, typically characterized by a coarse quantization, a low sample rate, and a large delay.

Historically, the usage of optical sensing in control systems arose from both structured and unstructured environments. In structured environments, optical sensing with little vision processing can provide high-speed and high-precision measurements for mechatronic systems (Gao et al., 2015). In unstructured environments, optical sensing followed by compute-intensive vision processing allows robotic applications to adapt to the environments (Hutchinson et al., 1996). In addition, there is a middle ground where the environment is semi-structured, where both adaptivity and performance are required (Goldberg, 2012). Examples of these three cases are illustrated in Figure 1.2.

The three types of environments mentioned above impose constraints on the optical features that can be used in visual sensing, which require different levels of adaptability and lead to different speeds in processing, as summarized in Table 1.1.

A wide range of applications exists in unstructured environments, such as mobile robots and autonomous vehicles. In these applications, visual sensing needs to be adaptable to and robust against variations in the environment. Machine learning



(a) (b) (c)

Figure 1.2: Optical sening in (a) home robot[1], (b) product manufacturing[2], and (c) surface encoder[3]. They represent optical sening in unstructured, semi-structured, and structured environments, respectively.

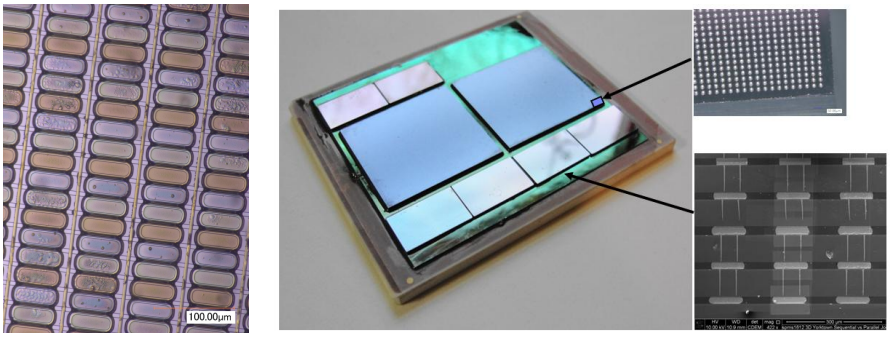
[1] An AMOR robot with eye-in-hand configuration.
[2] A ROTH & RAU PiXDRO LP50 inkjet printer with integrated vision systems.
[3] A HEIDENHAIN PP 281 R exposed linear encoder.

based methods are increasingly used for visual sensing in these applications, with examples in robots (Bateux et al., 2018) and autonomous vehicles (Lu et al., 2019). Integration of machine learning based visual sensing into these applications remains a research challenge and is being actively worked on.

The focus of this thesis, on the other hand, is on semi-structured environments, in which there is similarly a wide range of applications (Watanabe et al., 2014). In semi-structured environments, common in industrial applications, predefined patterns can be used as optical features. A typical example of these predefined patterns is the repetitive patterns on semiconductor products, as shown in Figure 1.3.

One of the benefits of applying visual servoing in semi-structured environments is to compensate certain dynamic errors induced by the system or its environment, which is otherwise difficult, if not impossible, to observe using other sensors. Dynamic errors are fast varying with time, which include geometric errors of machine components and structures, errors induced by thermal distortion, among other minor ones (Ramesh et al., 2000a,b). The dynamic errors that are hard to obtain by traditional sensing techniques are becoming limiting factors in precision motion control in semiconductor manufacturing. For example, flexible dynamics are becoming major limiting factors of light-weight motion stages (van Herpen et al., 2014). Visual sensing is capable of measuring these dynamic errors and then enabling dynamic compensation for them, as demonstrated in the case study of (de Best et al., 2012). Figure 1.4 illustrates the aforementioned dynamic errors that can be measured and compensated by visual sensing.

Besides measuring dynamic errors, another benefit of visual servoing is to allow



(a) (b)

Figure 1.3: Examples of repetitive patterns on semiconductor products: (a) pixel patterns of a display substrate[1], and (b) micro-bumps on three dimensional chip stacks[2].

[1]Sample provided by ROTH & RAU, imaged by a Keyence microscopy.
[2]Source: 3D Semiconductor & Packaging Technology for Systems, IBM.

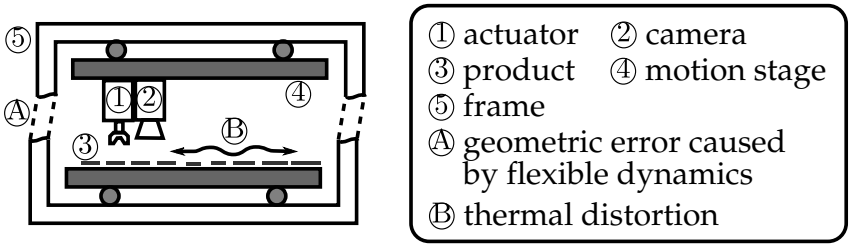


Figure 1.4: An example setup illustrating dynamic errors that can be measured and compensated by visual sensing. In the setup, an actuator (①) and a co-located camera (②) are mounted at the top, while the semiconductor product (③) is mounted at the bottom. Either one or both of them can be mounted on a motion stage (④), and both of them are connected to the frame (⑤). In the setup, geometric errors induced by flexible dynamics (Ⓐ) and thermal distortion (Ⓑ) are present, which can be measured by visual sensing but are hard to be measured by other sensors.

built-in flexibility in the system. For example, pick-and-place machines with visual sensing can handle components at non-predefined locations with high speed and high precision, as shown in Figure 1.5. Another example is vision-enabled printing machines for manufacturing flexible electronics, which is further investigated in subsequent chapters. Without visual sensing, more restrictions will be required on these applications, which incurs greater design efforts and higher costs.

To achieve the aforementioned benefits, several challenges need to be overcome when integrating visual sensing in precision mechatronic and robotic systems.

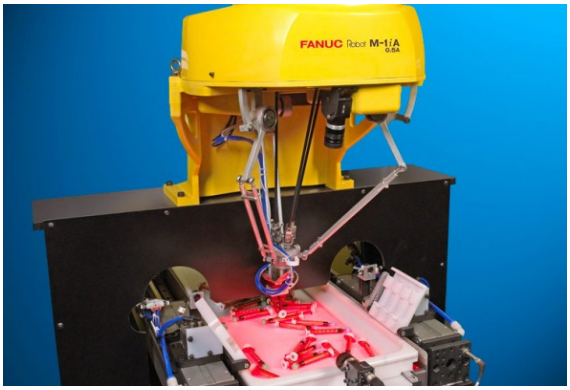


Figure 1.5: An example of a vision-enabled robot for flexible manufacturing. It is a pick-and-place machine (FANUC M-1iA) operating on components at non-predefined locations.

These challenges motivate the research work of this thesis, and will be elaborated next.

1.2 Research Challenges

While visual sensing has multiple benefits compared to other sensing techniques, it is challenging to integrate visual sensing into precision mechatronic systems. Designing vision algorithms and the corresponding processing systems are challenges by themselves, but they are well addressed by the research community. On the other hand, the integration of visual sensing into mechatronic systems imposes bigger challenges, which are not yet fully addressed despite decades of research. It motivates this thesis to focus on the challenges described as follows.

One of the biggest challenges is that visual sensing is computational-intensive and data-intensive, which necessarily involves trade-offs between sample rate, delay, and accuracy. This challenge exists since the inception of visual servoing more than two decades ago, described by early research of Corke and Good (1996) as "*a relatively low sample rate, significant latency (one or more sample intervals) and coarse quantization*". Despite that the speed of processing and the accuracy of vision algorithms has improved over the past two decades, the design trade-offs remain. The most accurate vision algorithms are typically not fast enough, while the fastest vision algorithms are often not accurate enough. Such trade-offs will be further discussed and quantified using a case study in the subsequent chapters.

The second challenge is to design control laws that utilize the additional information provided by visual feedback, while alleviating the drawbacks induced by visual feedback. Visual feedback provides online and direct measurements on the

objects of interests, thus requiring an online generation of reference and a high-performance controller for tracking such a reference, whithout which the major benefit of visual feedback is underutilized. On the other hand, visual feedback is often inferior than other types of sensors in terms of sample rate, delay, and quantization error, thus requiring the controller to be explicitly designed and tuned for these parameters. Despite recent developments of high-speed vision system, it remains a challenge to utilize visual feedback in the control systems (Watanabe et al., 2014).

The third challenge is the difficulty of having an early estimation of delays in high-speed vision systems prior to implementation. Due to the computational-intensive and data-intensive nature of visual sensing, high-speed visual sensing is often achieved by co-design and co-optimization of the vision algorithms and the processing systems, which makes it non-trivial to have an early estimation of its delay prior to implementation. In addition, there exist a wide range of vision algorithms, each having multiple variations, which provide different accuracy and delay trade-offs that need to be evaluated. A promising approach to tackle this challenge is to perform high-level explorations of design choices by synthesizing and evaluating these choices, as demonstrated by (Caarls, 2008). Yet it remains an open problem to perform design space exploration for algorithmic choices in combination with architectural choices of customized processing systems.

Last but not least, to explore the design trade-offs across multiple domains of visual servo systems, a cross-domain model that balances comprehensiveness and analyzability is required. The model is required to be comprehensive, because excluding one or more domains from the model leads to sub-optimal or even infeasible designs. For example, excluding certain nonlinear dynamics of the plant, like friction in the mechanics, leads to a preference for accurate but slow visual feedback that turns out to be far from optimal in practice. The model is required to be analyzable because it enables analytical methods to derive optimal parameters, which is more efficient than alternative methods such as heuristic or brute-force search. Comprehensiveness and analyzability are often contradicting requirements. Designing cross-domain models for design space exploration remains an open problem (Haveman and Bonnema, 2013), and balancing the comprehensiveness and analyzability of the model is a major challenge.

The aforementioned research challenges motivate this thesis to formulate them into research problems and to choose appropriate methods to approach them. This will be elaborated next.

1.3 Problem statement and research methods

Motivated by the benefits and challenges of integrating visual sensing into precision mechatronic systems, the research problem of this thesis is formulated as follows.

Research challenge	Research method
Low sample rate, significant latency, and coarse quantization.	Proposing design methods of vision systems for speed and accuracy, and applying them to a case study.
Control laws that utilize advantages and alleviate disadvantages of visual feedback.	Online trajectory generation and controller synthesis for visual feedback.
Difficulty of having an early estimation of delay in customized vision systems.	Using algorithmic patterns and architectural templates to perform high-level synthesis of the system.
Cross-domain coupling.	Using axiomatic design method to decouple design parameters.

Table 1.2: Research challenges and the corresponding research methods to tackle them.

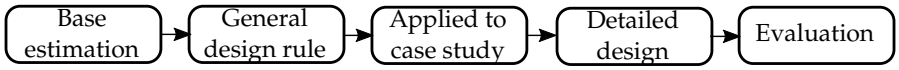


Figure 1.6: Design methods of vision system are proposed as general rules, and then applied to a case study and evaluated.

> *Propose methods to design high-speed vision systems for control purposes, to holistically model the cross-domain coupling within visual servo systems, and to explore cross-domain trade-offs with regard to the overall performance of visual servo systems. Evaluate the effectiveness of the proposed methods by prototyping and case studies.*

Tackling the problem stated above requires solving the research challenges as described in the previous section. A set of research methods are chosen to overcome each of theses challenges. The research methods deployed for these challenges are summarized in Table 1.2 and explained in the remaining part of this section.

First, methods are proposed to design vision systems for speed and accuracy, which is one of the biggest challenges of using visual sensing in control systems. General rules are proposed for the selection of camera systems and processing systems that can achieve different scales of sample rate with a delay of no more than two sample periods. After that, these general rules are applied to a case study, which is evaluated for speed and accuracy. Figure 1.6 provides an overview on this method.

Second, control laws are proposed to utilize the advantages of visual feedback and to alleviate its disadvantages. One of the major benefits of visual feedback is providing online and direct measurements on the objects of interest. Accordingly,

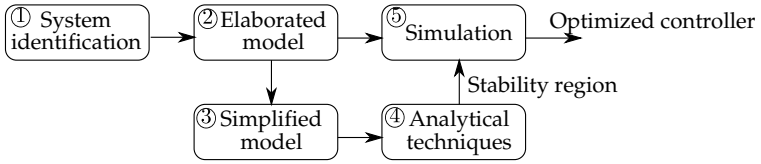


Figure 1.7: Method of designing and optimizing controllers for visual feedback. The method uses analytical techniques to quickly derive a feasible region of controller parameters, and subsequently uses simulation to optimize controller parameters within the feasible region.

a vision-based online trajectory generator is used to perform continuous motion based on the online dynamics of the objects. To alleviate the disadvantages of visual feedback, that is, inferior sample rate, delay, and quantization error, controllers are designed and optimized based on these parameters. Figure 1.7 provides an overview of this method.

Third, to enable an early estimation of the delay of customized vision systems, algorithmic patterns and architectural templates are proposed to support high-level synthesis of vision systems, as illustrated in Figure 1.8. The algorithmic patterns are common data access patterns in image processing algorithms, which can be composed to form an image processing pipeline. Architectural templates are parameterized building blocks of a processing system, with each architectural template optimized for one or multiple algorithmic patterns. Given an image processing pipeline, designers can extract the algorithmic patterns in the pipeline, and accordingly choose suitable architectural templates to instantiate a customized vision system via high-level synthesis. The delay of the synthesized vision system can be obtained either by simulation or by running it on a programmable device. Subsequently, designers can rapidly co-optimize the algorithm and the architecture to reduced delay, improve accuracy, eliminate timing jitter, etc.

Fourth, an axiomatic design method (Suh, 1998) is applied to visual servo systems to explicitly model cross-domain coupling and to explore cross-domain trade-offs with regard to the overall performance of the visual servo system. The axiomatic design method represents the coupling between design parameters (DP) and functional requirements (FR) in a matrix form, as shown in (1.1), where '$\star$' and '0' represents whether or not there is a dependency between a functional requirement and a design parameter.

$$\begin{bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{bmatrix} = \begin{bmatrix} \star & 0 & 0 \\ \star & \star & 0 \\ \star & \star & \star \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{bmatrix}. \tag{1.1}$$

Design rules are enforced to constrain cross-domain coupling and to shape the coupling matrix to be triangular, such that cross-domain design parameters can be explored in a structural way. This method is demonstrated using the model of a precision visual servo system, and by exploring design parameters that are most representative in the cross-domain coupling of visual servo systems.

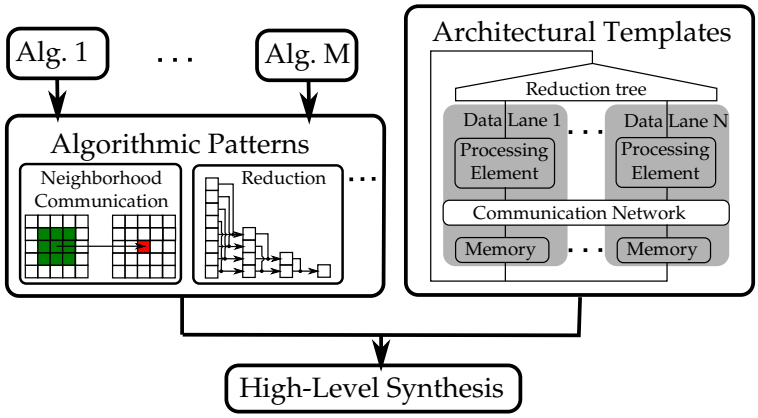


Figure 1.8: The design method using algorithmic patterns and architectural templates to perform high-level synthesis of customized vision systems.

Using the aforementioned research methods, this thesis research contributes towards a framework for solving the research challenges of integrating vision systems into precision mechatronic systems. Despite that the aforementioned research methods are based on previous research, applying and adapting these methods to the application domains of precision visual servo systems is nontrivial and not yet systematically addressed by previous research. By applying a framework of methods to tackle these research challenges, several contributions are made in this thesis, as described below.

1.4 Contributions

The contributions of this thesis are in both methodologies and implementations. Methods are provided to overcome the research challenges of integrating visual sensing into precision mechatronic systems. The design methods are applied to the implementation of prototypes, which in turn provide a realistic baseline for the modeling of precision visual servo systems and the exploration of design trade-offs.

Design methods of high-speed vision system

A generic model is proposed to estimate the sample rate and delay of high-speed vision system. The model is used to evaluate various technologies of implementation for their achievable sample rate and delay. Guided by the generic model, a high-speed vision system is implemented for a representative use case, and subsequently validated against the model.

Controller design and optimization for visual feedback

A high-performance visual servo system is implemented that utilizes the benefits and alleviates the drawbacks of visual feedback. More specifically, a vision-based online trajectory generator is used to update the reference signal according to the online dynamics of the targets, and is demonstrated to be more effective than traditional motor encoders. In addition, a control law is designed and optimized taking into account three major limitations of visual feedback, that is, sample rate, delay, and quantization errors.

Method of evaluating algorithmic and architectural choices

Different algorithmic choices and architectural choices are modelled and implemented using algorithmic patterns and architectural templates that are suitable for high-level synthesis. There are prior research on algorithmic patterns and architectural templates, but implementing them in a way that is suitable for high-level synthesis and can achieve a high sample rate and a low latency is a contribution of this thesis.

Methods for explorations of cross-domain trade-offs

The axiomatic design method is applied to reduce the cross-domain coupling in precision visual servo systems, and to facilitate the exploration of design trade-offs across multiple domains. Using the given method, the overall performance of a precision visual servo system can be efficiently derived from a wide range of design parameters across the domains of image processing, processing architecture, and control systems.

1.5 Outline of the thesis

As illustrated in Figure 1.9, this thesis consists of four major chapters besides the Introduction, Background, and Conclusions. Each of these chapters tackles one of the research challenges stated previously. Chapter 3 provides design methods for high-speed and high-precision vision systems, which is needed at the first place before tackling other challenges. The vision system is subsequently integrated into a prototype visual servo system in Chapter 4, which provides direct measurements needed for the construction of models. Using the aforementioned results, Chapter 5 constructs a baseline model of visual servo systems, which links design parameters across multiple domains to the overall performance of the visual servo system. The baseline model is subsequently used for the exploration of cross-domain trade-offs in Chapter 6, in which the challenge of cross-domain coupling is tackled. The outlines of the four major chapters are as follows.

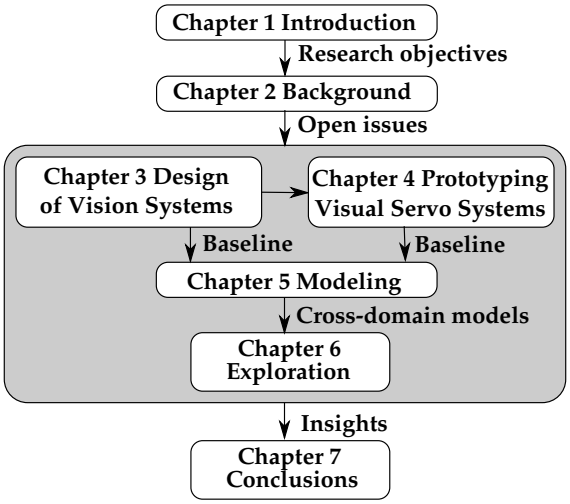


Figure 1.9: Outline of this thesis.

In Chapter 3, design methods for high-speed vision systems are proposed. Investigation is performed on existing technologies that are suitable for the implementation of high-speed vision systems, followed by proposals of generic design principles. The generic design principles are applied to a case study, which requires a sample rate of 1 KHz and an accuracy at micrometer scale. A vision algorithm is subsequently designed for the case study, balancing accuracy and speed. The accuracy of the algorithm is evaluated by simulation, and the speed of the vision system is evaluated by implementing the algorithm on a combination of dedicated hardware and programmable processor. The result of the case study confirms that the implemented vision system, guided by the design principles, achieves a high sample rate and a low delay as targeted.

In Chapter 4, the vision system implemented in Chapter 3 is integrated into a prototype visual servo system. The integration requires the control system to take into account the delay and quantization error induced by the visual measurement. The control system is subsequently designed and tuned accordingly. In addition, to fully utilize the benefits of visual feedback, a vision-based trajectory generator is used to improve the accuracy of the control task. In the end, the system is evaluated using a set of experiments, and the result confirms that the prototype satisfies the requirements of the case study. The prototype is providing a baseline for the modeling of a visual servo system.

In Chapter 5, a baseline model is constructed for the prototype visual servo system implemented for the case study. The model incorporates design parameters across multiple domains and links them to the overall performance of the visual servo system. As an example, the model includes four cross-domain parameters that are tightly coupled, that is, image size, vision algorithm, processing system, and the plant. Given these design parameters, the model derives the sample rate,

measurement error, delay, and an optimized controller gain of the system, which are subsequently used to obtain the overall performance of the system.

In Chapter 6, the choices of design parameters across multiple domains are explored using the aforementioned baseline model. To overcome the complexity induced by cross-domain coupling of the design parameters, the exploration of design choices is guided by the axiomatic design method, which imposes an order in which design parameters are tuned. The design parameters are subsequently optimized for representative requirements of precision visual servo systems, including accuracy, bandwidth, and cost. The case study indicates that the absence of cross-domain modeling and exploration leads to a significantly sub-optimal design. In addition, when multiple requirements are present, the optimal configuration for one set of requirements could be significantly sub-optimal for another set of requirements. It confirms and makes a case for the importance of cross-domain modeling and exploration and the effectiveness of the proposed method.

Before proceeding to the major chapters mentioned above, the background of related subjects is provided in Chapter 2. It reviews prior research, identifies open issues, and relates this thesis to the prior research and open issues. It provides the contexts in which major chapters are developed, and is presented next.

Chapter 2

Background

Science never solves a problem without creating ten more.

George Bernard Shaw

Abstract. *Previous work is reviewed and compared to this thesis. Previous work is organized into four areas, that is, design of high-speed vision systems, controller design for visual feedback, methods of exploring algorithmic and architectural choices, and cross-domain modeling and optimization. They correspond to four major research challenges in visual servo systems and four contributions of this thesis, as described in Chapter 1. Despite recent advances, there are open issues in each area when combining the methods of these areas into the design and implementation of visual servo systems. Open issues are identified by reviewing related work in each area and putting them in the context of visual servo systems. By comparing related work to this work, contributions of this thesis are clarified.*

2.1 Design of high-speed vision systems

Speeds of image sensors and their interfaces have been improving over the years, which brings opportunities as well as challenges to the design of high-speed vision

systems, especially for control purposes. With regard to opportunities, commercial off-the-shelf industrial cameras can reach sample rates at the scale of 1,000 to 10,000 frames-per-second, which is comparable to the sample rates of other sensors commonly used in high-performance control applications. However, it remains challenging to design vision processing systems that can process image data with a high throughput and a low latency, because visual sensing is data-intensive and compute-intensive compared to conventional sensing methods.

While previous work provides reference designs of high-speed vision systems, the building blocks of these designs evolve over time, which calls for an evaluation of available technologies that are suitable for current and future generations of high-speed vision processing systems. The evaluation involves three building blocks of the vision systems, that is, image sensor, vision algorithm, and processing platform. These building blocks are illustrated in Figure 2.1, and discussed subsequently.

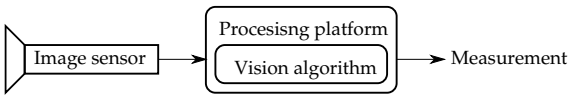


Figure 2.1: Three major components in high-speed vision systems, including image sensor, vision algorithm, and processing platform.

Industrial high-speed image sensors are able to continuously stream image frames at rates from 1KHz to 100KHz, with resolutions of a few hundred pixels in width and height. Table 2.1 lists three examples of image sensors that are capable of achieving frame rates at the scale of 1KHz, 10KHz, and 100KHz, respectively. To achieve such a high frame rate, these image sensors are configured to read out a part of a full image frame. During the period 2010 to 2015, these three image sensors are typical examples of high-speed image sensors on the market. Since 2015, there are newer models that supersede these examples, with higher speeds and lower costs, but they are incremental improvements that do not fundamentally change the performance and cost of these sensors.

① KAI-0340		② LUPA3000		③ Phantom v2010	
image size	frame rate	image size	frame rate	image size	frame rate
640x480	214	640x480	2653	640x480	62600
640x164	618	**256x256**	**10704**	**256x256**	**188900**
228x164	**1637**	128x128	26178	256x128	349500

Table 2.1: Configurations of three image sensors (the Phantom v2010 is a continuous recording system with an undisclosed image sensor). **Bold:** configurations that achieve frame rates at the scales of 1KHz, 10KHz, and 100KHz, respectively, with comparable image sizes.

There exists a wide range of generic solutions for vision-based object detection and tracking, from manually designed vision algorithms to deep neural networks.

Manually designed vision algorithms and simple neural networks for object detection and tracking are surveyed in (Yilmaz et al., 2006). Deep neural networks and their applications in computer vision are surveyed in (Pouyanfar et al., 2018), and a deep neural network such as AlexNet (Krizhevsky et al., 2012) can be trained for visual servoing applications (Bateux et al., 2018). The computational complexity of different solutions can be measured in terms of operations performed on a pixel, noted op/px. Manually designed vision algorithms for pre-defined patterns could have a computational complexity below 100 op/px; generic algorithms for natural feature detection and object recognition, such as scale-invariant-feature-transform (SIFT), typically have a computational complexity of 1000 op/px (Mizuno et al., 2010); a deep neural network such as AlexNet has a computational complexity of more than 10,000 op/px (Sze et al., 2017). Figure 2.2 illustrates the computational complexity of these solutions. In semi-structured environments, pre-defined patterns are often used, for which manually designed vision algorithms are sufficient in detecting and tracking these patterns. Focusing on semi-structured environments, this section evaluates the computational complexity of manually designed vision algorithms with computational complexities between 10 op/px and 100 op/px.

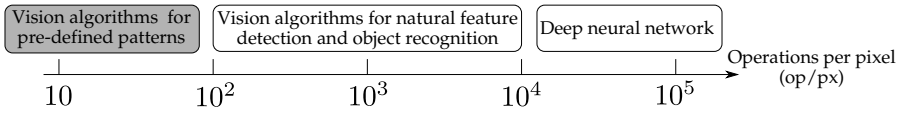


Figure 2.2: Computational complexity of different vision algorithms, measured in operations per pixel (op/px). The focus of this work is on algorithms for pre-defined patterns (box in gray). The computational complexity of a typical algorithm for natural feature detection and object recognition is analyzed in (Mizuno et al., 2010); a deep neural network in (Sze et al., 2017).

A wide range of processing platforms are used in high-speed vision systems, as survey in (Gu and Ishii, 2016; Watanabe et al., 2014). These processing platforms include central processing units (CPU), graphics processing units (GPU), field-programmable gate arrays (FPGA), and custom chips. Due to a high cost of implementation and manufacture, custom chips are not considered in this work. In addition to the aforementioned processing platforms in the surveys, digital signal processors (DSP) has also been used in visual servo systems (Asaad et al., 1996). The theoretical performance of DSP, CPU, GPU and FPGAs can be compared using the giga operations per seconds (gop/s). Representative devices of these platforms and their performances are listed in Table 2.2.

Since most high-speed vision systems are designed to meet performance requirements under cost constraints, the choice of image sensors, vision algorithms, and processing platforms is made leading to a balanced system, "*for which the primary applications are limited in performance by the most expensive component(s) of the system*" (McCalpin, 2004). The cost of image sensors dominates the cost of other building blocks of vision systems, as detailed in Appendix A. Therefore, a balanced vision system requires the processing platform to match the throughput

Label	Processing Platform	gop/s
(a)	DSP (TMS320C6657)	40
	FPGA (XC5VSX50T)	100
	CPU (i5-3570)	136
	CPU (E5-2690)	300
(b)	DSP (TMS320C6678)	320
	FPGA (XC7K325T)	1245
	GPU (GTX 780 Ti)	5046
(c)	FPGA (XC7VX690T)	5335

Table 2.2: Processing platforms and their computational capability. The three systems labeled with $\big((a)(b)(c)\big)$ are further displayed in Figure 2.4.

of the image sensors. Given an image sensor and its configuration, and a vision algorithm of certain computational complexity, designers can derive computational requirements on the processing platform, and accordingly choose a specific processing platform. Such a procedure is illustrated in Figure 2.3.

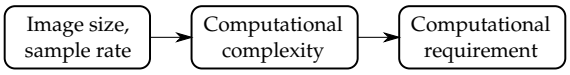


Figure 2.3: Procedure of choosing the processing platform for high-speed vision systems. First, an image sensor of specific image size and sample rate is chosen. After that, the computational complexity of the vision algorithm is estimated. Subsequently, the computational requirement of the processing platform is derived.

Based on the aforementioned procedure, different processing platforms can be evaluated with regard to their achievable sample rate at a certain image size and for a specific vision algorithm. As an example, three representative image sensors, two vision algorithms of different computational complexity, and three processing platforms are evaluated, as illustrated in Figure 2.4.

Figure 2.4 provides an estimation on the computational requirements at different scales of sample rates, and subsequently guides the choice of processing platform. For illustration purposes, certain simplifications are made in Figure 2.4. First, the computational complexity of the vision algorithms are considered between 10 op/px and 100 op/px. Such a computational complexity is representative, as shown in Chapter 5.3.2, but the method can be trivially applied to vision algorithms of higher or lower computational complexities. Second, this example illustrates the theoretical peak performance of processing platforms, which is hardly obtainable in practice. As detailed in Chapter 3.1.1, it is not uncommon that only 10% of the theoretical peak performance of a processing platform is utilized by an application.

To provides a generic and conservative estimation on the choice of processing

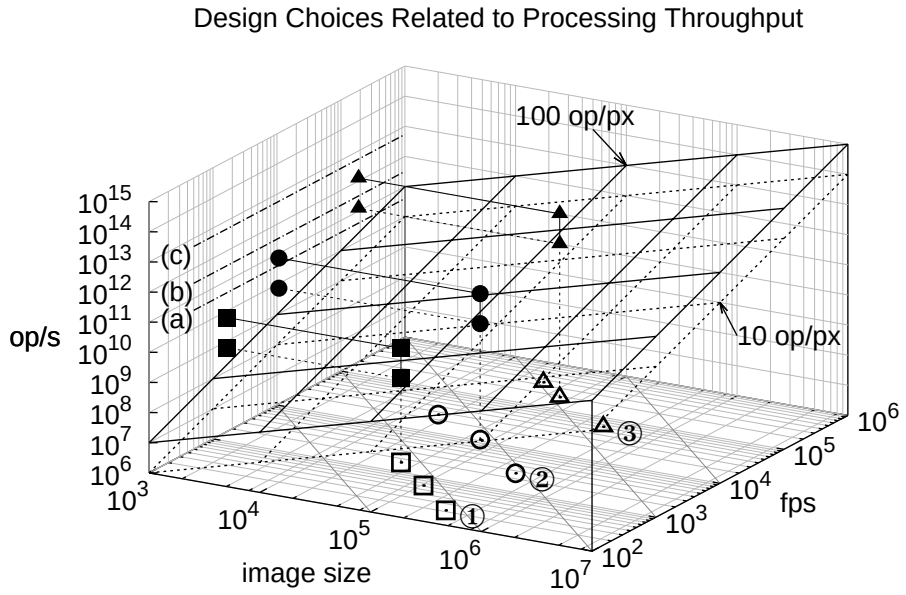


Figure 2.4: Deriving computational requirements from image sensors and vision algorithms, using the procedure described in Figure 2.3. First, three image sensors (① ② ③ , described in Table 2.1) of different image sizes and sample rates (fps) are chosen, illustrated as 9 unfilled symbols at the bottom plane; after that, two hypothetical vision algorithms with computational complexity of 10 op/px and 100 op/px are illustrated as two planes with dotted line and solid lines respectively; by choosing one configuration from each of the three image sensors, and evaluating both vision algorithms, six computational requirements are derived (shown as 6 filled symbols on the vertical axis); subsequently, three processing systems ((a)(b)(c), see Table 2.2) are chosen to support different level of computational requirements, and illustrated as three lines with different op/s in this figure.

	Sample Rate		
	1KHz	10KHz	100KHz
Processing Platform	CPUs/ DSPs	GPUs/ FPGAs	GPUs/ FPGAs

Table 2.3: Choices of processing platform, assuming an image size of 10^5 pixels, a vision algorithm of 100 op/px, and 10% of the theoretical peak performance of the processing platform can be obtained.

platform, this section assumes that the vision algorithm has a computational complexity of 100 op/px, and only 10% of the theoretical peak performance of a processing platform can be achieved by the applications, and the resulted qualitative guidelines are shown in Table 2.3.

		GPU-based systems		FPGA-based systems	
		Device	Delay	Device	Delay
Stage	①	Ethernet	$\approx 10\mu s$	CameraLink	$\approx 10\mu s$
	②	Ethernet Adapter[a]	$\approx 5\mu s$	Dedicated circuit	$\approx 0\mu s$
	③	PCI-e bus to GPU[b]	$\approx 10\mu s$	Dedicated circuit	$\approx 0\mu s$
	④	GPU	$\approx 10\mu s$	Dedicated circuit	$\approx 10\mu s$

Table 2.4: Delay estimations of two processing systems, for a target sample rate of 100KHz, and an image size of 10^5 pixels. The delay is broken down into four stages: ① image readout, ② camera interface, ③ transfer to device memory, ④ vision processing. The delay numbers are obtained from data sheets, listed in the footnote of this table.

[a] The delay of the first package of 64 bytes (MCorelab, 2012)
[b] The delay of copying 10^5 bytes over the PCI-e bus (Lustig and Martonosi, 2013)

Besides throughput, another aspects of the visual feedback is latency. In the domain of communication and computing technology, the improvement on latency lags that on throughput (Patterson, 2004). While GPUs and FPGAs both have high throughput, the latency of GPUs are often more than several sample periods, as shown in Table 2.4. As described in the table, a GPU-based vision system has a larger delay due to intermediate communication and buffering, which can be eliminated in FPGA-based systems. Among these common processing platforms, only FPGAs are capable of achieving a frame rate of 100KHz while keeping the latency to no more than two sample periods. Therefore, this work focuses on FPGAs as the processing platform, and proposes design methods for vision processing systems that are capable of achieving a throughput of 100,000 frames-per-second with a small latency.

There are previous works that present a specific design point in the design space of high-speed visual system. Different from previous works, this work provides a framework for designers to generate vision systems at different scales of performance and cost. This work provides a design template of vision processing system for high throughput and low latency. The design template can be scaled to future generations of image sensors, at the speed of 100,000 frames-per-second, while capable of achieving a latency of no more than 2 sample periods. A specific example of such vision processing systems is provided in Chapter 3, while the generic design template is described in Chapter 5.

2.2 Controller design for visual feedback

From a historical perspective, a major challenge of integrating visual feedback into control systems has been that visual feedback typically has "a relatively low sample rate, significant latency (one or more sample intervals) and coarse quan-

tization" (Corke and Good, 1996). Recent advances of high-speed vision systems have significantly improved the sample rate and reduced the delay of visual feedback (Gu and Ishii, 2016; Watanabe et al., 2014). However, recent improvements of sample rate and delay have not eliminated the need that controller designers have to take into account the trade-offs between sample rate, delay, and quantization errors in visual feedback, explained as follows.

At present, high-speed vision systems remain custom-built and application-specific, with their sample rate and delay tightly coupled to the choices of vision algorithms. In addition, designers can hardly implement the most accurate vision algorithms while maintaining a sufficiently high sample rate and low delay for the application, and the contrary is true as well. As a result, despite recent advances in technology, the trade-off between accuracy and speed of visual feedback remains a challenge for designers.

The focus of this work is therefore on controller design taking into account three major characteristics of visual feedback, that is, sample rate, delay, and quantization error. There are prior works on controller design taking into account certain characteristics of visual feedback, but not without limitation. They either omit certain characteristics of visual feedback, or make certain assumptions on delay and measurement error which limit their applicability to certain use cases. Table 2.5 provides a comparison of related works on controller design taking into account characteristics of visual feedback. These related works are discussed in remaining parts of this section.

Related work	Delay	Sample rate	Delay $\neq$ sample rate	Measurement error	Error has non-zero mean	Control structure	Controller gain
(Zhang et al., 2003)	✓					✓	✓
(Kawamura et al., 2012)	✓	✓	✓			✓	
(Fujimoto and Hori, 2001)	✓	✓	✓			✓	
(Khamesee et al., 2008)	✓					✓	✓
(Medina et al., 2017)	✓	✓	✓			✓	✓
(Mohamed et al., 2018)	✓	✓	✓			✓	✓
(Kyrki et al., 2006)				✓		✓	
(Chesi and Yung, 2009)				✓	✓	✓	
(Assa and Janabi-Sharifi, 2013)		✓		✓		✓	
(van Horssen et al., 2015)	✓	✓	✓	✓		✓	✓
This work	✓	✓	✓	✓	✓	✓	✓

Table 2.5: Comparison of related works on controller design taking into account the sample rate, delay, and measurement error of visual feedback.

There are prior works on methods of controller design taking into account the sample rate and time delay of visual feedback. In (Zhang et al., 2003), methods are proposed to derive stability regions based on the delay of visual feedback, while the sample rate of visual feedback is not explicitly analyzed. In (Kawamura et al., 2012), a control method is proposed which alleviates the effects of sample rate and delay in visual feedback, while the impact of sample rate and delay on control performance is not explicitly analyzed. In (Fujimoto and Hori, 2001), a multirate sampling controller is designed to allow the controller and actuator to operate at a higher rate than the sample rate of visual feedback, while the controller gain is empirically derived. In (Khamesee et al., 2008), the sample rate, delay, and quantization error of visual feedback is included in the model, under an assumption that the delay is equal to the sample period. In (Medina et al., 2017; Mohamed et al., 2018), the sample rate and delay of visual feedback, under the influence of processing systems, are taken into account in controller design, while the measurement error is left out.

There are prior works on methods of controller design taking into account the measurement error of visual feedback. In (Kyrki et al., 2006), the source of measurement error is modelled as zero-mean random variables with a certain covariance, and the propagation of measurement error into overall visual servo performance is analyzed. In (Chesi and Yung, 2009), the source of measurement error is considered to be bounded, without any assumption on its statistical property, and the bound of overall visual servo performance is subsequently derived. In (Assa and Janabi-Sharifi, 2013), the source of measurement error is modelled in a similar way as (Kyrki et al., 2006), but the visual servo system is modelled as discrete-time system, which is demonstrated to be more accurate than continuous-time models.

In (van Horssen et al., 2015), all three characteristics of visual feedback are taken into account in controller design. In addition, it provides an on-line control strategy for switching between different vision algorithms that offers speed-versus-accuracy trade-offs. It assumes the measurement error induced by visual feedback is zero-mean random variable with a certain covariance. While such an assumption holds for certain use cases, it does not hold for various vision algorithms used in the case study of this work. More specifically, as shown in Chapter 5.3, all three vision algorithms in the case study exhibit position-dependent measurement errors that can be better modelled as quantization error instead of zero-mean random variables. Such an observation motivates this work to model measurement error as quantization error, which is a major difference between (van Horssen et al., 2015) and this work.

Different from the aforementioned related work, as compared in Table 2.5, this work takes into account all three characteristics of visual feedback with realistic assumptions, and subsequently models a visual servo system as a quantized sampled-data system with time delay. Methods of controller design for such systems are available in prior work, especially in the domain of networked control systems. In (Cloosterman et al., 2009), methods are provided for controller de-

sign for systems with time-varying delays that can be larger than one or multiple sample periods, while quantization error is not considered. In (van Loon et al., 2013), quantization errors, time-varying sampling intervals, time-varying delays, and other network-induced communication imperfections are taken into account in controller design. This work models a sub-set of feedback imperfections from those described in (van Loon et al., 2013), and considers quantization errors, constant sampling intervals, and constant delays.

While the aforementioned work in the domain of networked control systems take into account quantization errors, sample periods, and delays, these methods have not been applied to the domain visual servo systems. As shown in Table 2.5, there is a gap between available methods and the adoption of these methods in the domain of visual servo systems.

The contribution of this work is therefore in bridging the gap between available methods and the adoption of these methods in visual servo systems. More specifically, based on representative case studies, this work identifies and takes into account realistic assumptions of visual feedback, which are not fully addressed in previous work. Subsequently, available methods of controller design, mainly from the domain of networked control systems, are adopted for the domain of visual servo systems, such that characteristics of visual feedback are fully addressed.

2.3 Exploring algorithmic and architectural choices

For a specific use case, designers of visual servo systems typically face a wide range of choices in vision algorithms and processing architectures. These design choices directly influence the accuracy and delay of visual feedback, which subsequently affects the overall performance of visual servo systems. On evaluating the accuracy of vision systems, there exist different methods described in prior work (Santo et al., 2000). In contrast, evaluating the delay of vision systems remains a research challenge, and therefore is the focus of this section.

Delays of high-speed vision systems are hard to estimate due to multiple reasons. First, high-speed vision systems are mostly custom-built and application-specific (Gu and Ishii, 2016; Watanabe et al., 2014), which makes it hard to estimate their delays prior to actual implementations. Second, co-design and co-optimization of vision algorithms and processing architectures are required for these custom-built platforms. Third, there is a wide range of vision algorithms and processing architectures, each having multiple variations, which creates a large design space for exploration.

To overcome the aforementioned challenges, this work performs high-level synthesis of vision systems based on the concepts of algorithmic patterns and architectural templates (Caarls, 2008). Algorithmic patterns, also called algorithmic skeletons, describe the communication patterns of algorithms. Architectural tem-

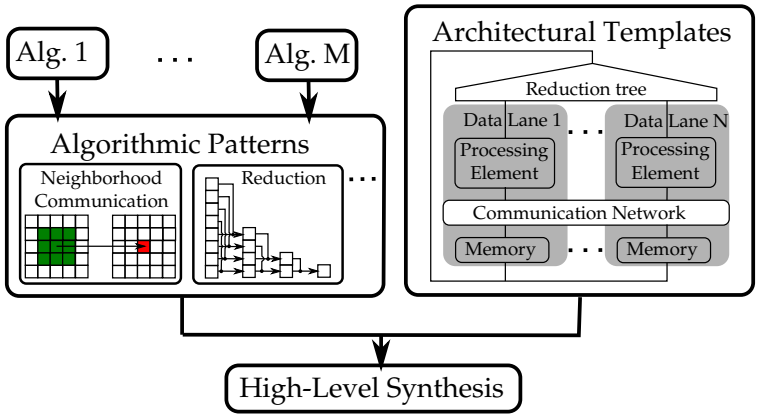


Figure 2.5: The design method using algorithmic patterns and architectural templates to perform high-level synthesis of customized vision systems.

plates are parameterized building blocks of processing systems. High-level synthesis is a method of generating circuit specifications from high-level languages, and is supported by a wide range of tools (Nane et al., 2016). Figure 2.5 provides examples of algorithmic skeletons and architectural templates that are used to generate vision processing system using high-level synthesis tools. Via the aforementioned method, vision processing systems can be rapidly generated on FPGA platforms. The delay of the vision processing system is subsequently obtained by simulation, using representative input images.

Common algorithmic patterns in the domain of image processing and computer vision are described in (Caarls, 2008; Nugteren et al., 2013). A computer vision application typically involves multiple algorithms, each with different algorithmic patterns. In addition, a computer vision application typically involves low-level, intermediate-level, and high-level operations. Low-level operations perform computations on image pixels; intermediate-level operations reduce image pixels into image features or a subset of pixels; high-level operations perform computations on image features and a subset of pixels. An example of vision processing pipeline is shown in Figure 2.6, illustrating different algorithmic patterns. Details of this vision processing pipeline are described in (He et al., 2011a).

Architectural templates are typically used in two ways in processing systems. First, architectural templates can be extracted from existing processing platforms, and are subsequently used to support efficient implementation of algorithmic patterns on these platforms. As demonstrated in (Caarls, 2008), architectural templates are extracted from existing microprocessors and massively-parallel vector processors, and are subsequently used by automation tools to generate code for algorithmic skeletons. Second, architectural templates can be designed and subsequently implemented on new processing platforms that can efficiently support algorithmic patterns. As demonstrated in (Benkrid and Crookes, 2004; Fernando

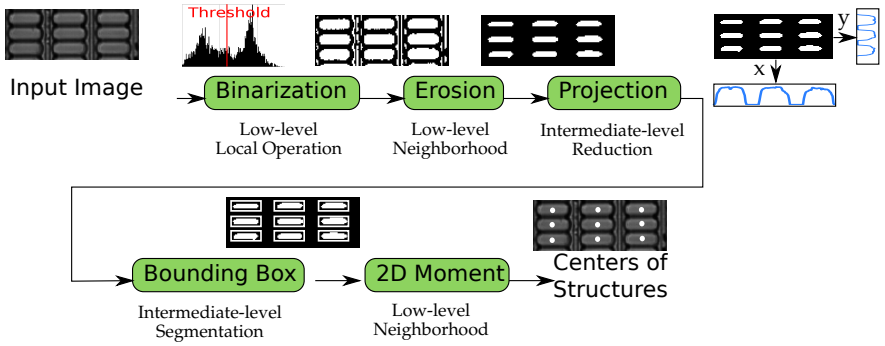


Figure 2.6: An example vision processing pipeline that detects the centers of repetitive structures in an image. The algorithmic patterns of each processing stage are annotated.

et al., 2015; Reiche et al., 2015; Schmid et al., 2014), a set of architectural templates that support various algorithmic patterns are designed and implemented on field-programmable gate arrays (FPGAs).

This work applies architectural template in the latter way, that is, designing and implementing architectural templates on new processing platforms. More specifically, architectural templates are designed to support algorithmic patterns on FPGAs. FPGAs are chosen as the target processing platform because it is most suitable for high-speed vision systems that demand a low latency and a high sample rate, as discussed in Chapter 2.1. The architectural template proposed in this work is illustrated in Figure 2.7, which is based on the parallel architectural presented in (He et al., 2011a; Pu et al., 2011). Details of the proposed architectural template are provided in Chapter 5.4.This work generalizes and parameterizes the parallel architecture of (He et al., 2011a; Pu et al., 2011) into an architectural template that can be rapidly instantiated by high-level synthesis tools.

High-level synthesis tools (Nane et al., 2016) are used in this work to rapidly instantiate the architectural template for the algorithmic patterns of a vision application. For each type of algorithmic pattern, the algorithm can be written in a predefined style to generate a specific type of architectural building blocks, either using a domain-specific language (Reiche et al., 2015) or a generic language such as C (Fernando et al., 2015; Schmid et al., 2014). The architectural template can be parameterized by explicit specifications from user input, as demonstrated in (Fernando et al., 2015), which allows designers to to rapidly explore a combination of algorithmic skeletons and architectural templates with different design parameters.

This work implements algorithmic patterns by writing each algorithmic pattern in a certain style using a generic language, similar to the approach of (Fernando et al., 2015; Schmid et al., 2014). In comparison, (Schmid et al., 2014) generates one specific instance for an algorithmic pattern, while (Fernando et al., 2015) and

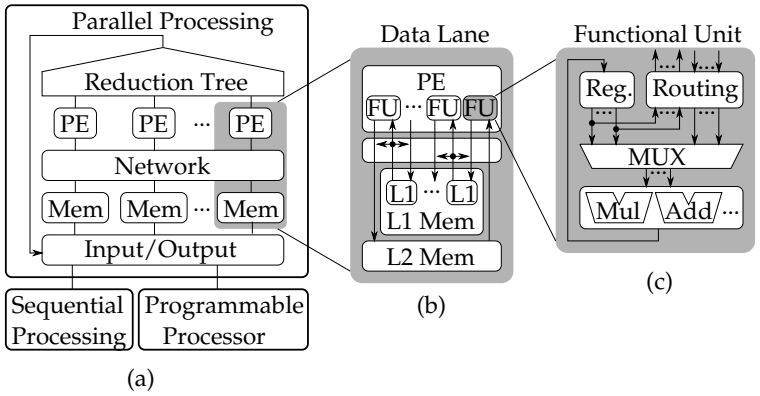


Figure 2.7: Architectural template of the processing system. (a) the overall architecture, with a programmable processor, and two types of dedicated circuits, parallel and sequential. (b) one data lane of the parallel processing system. (c) one functional unit (**FU**) of the processing element (**PE**). Other abbreviations: Memory (**Mem**), Level 1 (**L1**), Level 2 (**L2**), Register (**Reg.**), Multiplexer (**MUX**), Multiplier (**Mul**), Adder (**Add**).

Related work	Input language	Parameterized implementation	Parallel architectural template
(Schmid et al., 2014)	generic		
(Reiche et al., 2015)	domain-specific	✓	
(Fernando et al., 2015)	generic	✓	
This work	generic	✓	✓

Table 2.6: Comparison between this work and related works that perform high-level synthesis of image processing algorithms based on algorithmic patterns.

this work supports generating multiple designs based on parameterized template. This work extends (Fernando et al., 2015) by incorporating a parallel architectural template that supports high-speed vision processing.

Table 2.6 compares this work with related works that perform high-level synthesis of image processing algorithms based on algorithmic patterns. While this work applies methods that are similar to those used in related works, this work proposes an architectural template that is suitable for high-level synthesis and demonstrates the proposed template can achieve a high throughput and a low

latency at an application level. In addition, this work applies the methods to the exploration of algorithmic and architectural choices at an application level, which is not presented in previous works.

2.4 Cross-domain modeling and optimization

Given a design problem and a set of possible solutions, it is nontrivial to evaluate the performance of different solutions, especially when cross-domain trade-offs are involved. Accuracy and cost of different evaluation methods are surveyed in (Haveman and Bonnema, 2013), and illustrated in Figure 2.8.

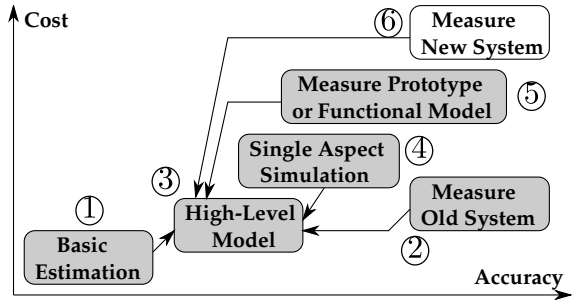


Figure 2.8: Accuracy and cost of different evaluation methods (Haveman and Bonnema, 2013). The gray boxes are covered in this thesis. The development of new systems is not performed due to cost reasons.

As shown in Figure 2.8, the high-level cross-domain models constructed in this thesis are obtained via different methods, explained as follows by examples. Basic estimations are used to derive coupling patterns of different domains. An old system is measured to obtain baseline parameters of a mechanical plant. Single aspect simulations are used to obtain accuracy of different vision algorithms. Prototypes of various processing architectures are generated and subsequently measured. The applications of these methods in modeling visual servo systems are described in details in Chapter 5.

Methods of cross-domain modeling have been actively studied in the field of systems engineering. The cross-domain modeling methods used in this thesis are based on prior concepts and methods in system engineering, and three of these concepts and methods are particular relevant. The concepts of abstraction and hierarchy in multi-domain modeling are well explained in (Heemels and Muller, 2006). The axiomatic design method for evaluating cross-domain coupling is presented in details by (Suh, 1990). The separation of concerns between Computation, Communication, Coordination, Configuration and Composition, also called the 5C's principle of separation of concerns, is described in (Bruyninckx et al., 2013). Subsequent parts of this section explain these concepts and methods in

details, and in the end review related work on cross-domain modeling of visual servo systems.

2.4.1 Hierarchical multi-domain model

Due to complexities of multi-domain systems, it is necessary to impose hierarchy via abstraction. For visual servo systems, domains of interest include vision systems, electronic systems, control systems, among others. Subsequently, the scope of this chapter can be described in an abstraction pyramid (Heemels and Muller, 2006), shown in Figure 2.9.

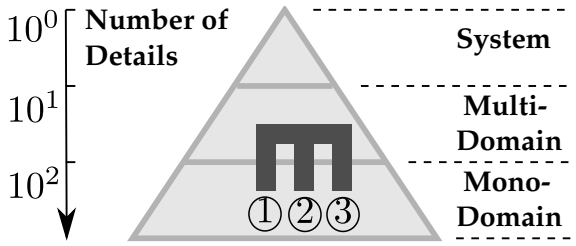


Figure 2.9: An abstraction pyramid, also called design pyramid or exponential pyramid, for visual servo systems, adapted from (Heemels and Muller, 2006). The 'm' shape in dark gray represents the domains relevant to the exploration of design trade-offs in this thesis, that is, ① vision system, ② electronic system, ③ control system.

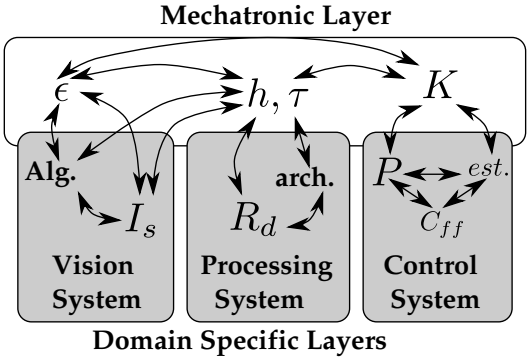


Figure 2.10: An example of hierarchical models, adapted from (Hehenberger et al., 2010). Abbreviations: measurement error (ϵ); sample period (h); delay (τ); feedback controller gain (K); vision algorithm ($Alg.$); image size (I_s); processing architecture ($arch.$); data transfer rate between camera and processor (R_d); plant (P); state estimator ($est.$); feedforward controller (C_{ff}).

A high-level model can be built hierarchically from multiple domains (Hehenberger et al., 2010), as shown in Figure 2.10. At the top-level mechatronic layer,

the overall performance of a visual servo system is affected by four cross-domain parameters, which depend on single-domain parameters at the domain-specific layer.

The focus of this thesis is on cross-domain coupling that involves design parameters across multiple domains. As illustrated in Figure 2.10, four cross-domain parameters are identified, that is, measurement error (ϵ), sample period (h), delay (τ), and feedback controller gain (K). Among numerous single-domain parameters, four of them are used in subsequent parts of this section as examples, that is, image size (I_s), vision algorithm ($alg.$), architecture ($arch.$), and the mechanical plant (P). Dependency between cross-domain parameters and single-domain parameters can be modeled using the axiomatic design method, as described next.

2.4.2 Axiomatic design method

To analyze high-level couplings of design parameters, the axiomatic design method (Suh, 1990; Suh et al., 2001) is considered. It is one of the fundamental methods in the exploration of design alternatives (Haveman and Bonnema, 2013). The relation between the requirement vector $[\epsilon, h, \tau, K]$ and design parameter vector $[I_s, alg., arch, P]$ can be represented in a *design equation*,

$$\begin{bmatrix} \epsilon \\ h \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \\ P \end{bmatrix}, \tag{2.1}$$

in which the matrix is called a *design matrix*, the '$\star$' symbol represents dependence, the '0' symbol represents independence. Note that the design matrix does not represent a linear matrix equation between the design parameter vector and the requirement vector, but rather represents dependencies, potentially nonlinear, between them. The elements in gray are candidates to be changed such that the system can become a quasi-coupled design, which will be elaborated next.

If the design matrix is a diagonal matrix or a triangular matrix, the system is called *uncoupled* and *quasi-coupled* respectively (Suh, 1990). Otherwise the system is a coupled design. According to this definition, (2.1) is a coupled design. However, it can be turned into a quasi-coupled design by imposing restrictions on the systems and the design parameters.

The sample period of visual feedback can be made independent of vision algorithms and processing architectures, by a design template described in Chapter 3. If such a design template is used, the sample period of a high-speed vision system is determined only by design parameters of image sensors, and as a result (2.1)

becomes

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \\ P \end{bmatrix}, \tag{2.2}$$

in which the elements in gray are changed from '$\star$' to '0'. The system becomes a quasi-coupled design, which simplifies design space exploration by imposing ordering in design parameter exploration.

The design space exploration can be further simplified by imposing hierarchy into the design matrices. In (2.2), the controller gain K unnecessarily depends on every element of the design vector. Designers with domain knowledge of control systems can apply methods that derive controller gain based on a few high-level design parameters, i.e.,

$$\begin{bmatrix} K \end{bmatrix} = \begin{bmatrix} \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} h \\ \epsilon \\ \tau \\ P \end{bmatrix}. \tag{2.3}$$

Combining (2.2) and (2.3) leads to

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \\ P \end{bmatrix}, \tag{2.4}$$

in which the symbol '1' in the design matrix means the output element is the same as the input element. The benefit of performing the modeling in separate steps is the reduced complexity of deriving certain high-level parameters. More specifically, the controller gain K can be derived from $[h, \epsilon, \tau, P]$ of (2.3) instead of from $[I_s, alg., arch., P]$ of (2.2). The latter contains significantly more parameters. The benefit of hierarchical design is more apparent for the example in Appendix B.

2.4.3 Generalization and limitation of axiomatic design

The previous section presents a quasi-coupled design, described in (2.2), as an example of visual servo systems. While instrumental and representative as an example, a quasi-coupled design does not cover the full range of possible designs. It is necessary to generalize the method to coupled and redundant designs due to reasons as follows.

First, there is no guarantee that a coupled design can be converted into an uncoupled or quasi-coupled design without ruling out optimal design choices. Therefore, if decoupling a system results in a severely sub-optimal or even infeasible design, the design needs to remain coupled.

Second, in mechatronics systems, the number of design parameters are often larger than the number of functional requirements, even after these functional requirements are broken down into the lowest level. If the redundant design parameters cannot be ruled out, additional analysis steps are needed, including grouping multiple design parameters into a lumped parameter, or imposing cost functions (Suh, 1990).

The axiomatic design method is applicable to coupled and redundant designs (Suh, 1990). Based on the axiomatic design method, the modeling method of this thesis can be similarly generalized beyond the example provided in (2.2). Appendix B provides examples of applying the modeling method to visual servo systems that are coupled and redundant designs.

While the axiomatic design method is effective for high-level coupling analysis in general, it is not without limitation. Based on the axiomatic design method, the modeling method of this thesis inherently has the same limitations, described as follows.

First, the design matrix is assumed to be known a priori. There are multiple methods to obtain the design matrix, which are similar to those used to obtain high-level models shown in Figure 2.8. However, for a radical modification of an existing design that changes how different domains interact, a correct design matrix requires an implementation and measurements of a new system. Therefore, the axiomatic design method is not meant to eliminate design iterations for radical design changes. In general, it is difficult to rigidly enforce a specific design process to radical innovation (Benner and Tushman, 2003).

Second, the decoupling of an inherently coupled design may lead to a sub-optimal or even infeasible design. Despite the benefits of uncoupled and quasi-coupled designs (Suh et al., 2001), certain designs should remain coupled during the design space exploration. For such cases, axiomatic design method can be used as a tool for analysis instead of synthesis. An example of design analysis that can be performed on a coupled design is to cluster redundant design parameters, as described in Appendix B. While the decoupling process should be selectively applied, axiomatic design can be used to analyze any design in general.

Due the limitations stated above, this thesis applies the axiomatic design method to incremental design choices and applies it to design analysis instead of design synthesis. Despite such a limitation, it is instrumental in cross-domain modeling of visual servo systems.

2.4.4 The 5C's principle of separation of concerns

The 5C's principle separates the concerns between Computation, Communication, Coordination, Configuration and Composition. As defined in (Bruyninckx et al., 2013), Computation concerns what functionality is computed; Communi-

Computation	
5C's principle	Computation in all domains.
This work	Computation in vision domain.
Communication	
5C's principle	Generic data flow, events, and service calls.
This work	Data flow in vision domain.
Coordination	
5C's principle	Multiple scenarios and states.
This work	Single scenario.
Configuration	
5C's principle	On-line configurable.
This work	Off-line configured.
Composition	
5C's principle	Generic hierarchical building blocks.
This work	Single layer of hierarchy at domain boundary.

Table 2.7: Comparison between the 5C's principle and the method used in this work on each of the 5Cs.

cation concerns how input and output of computation are being communicated; Coordination concerns when the components change their behaviors based on their states; Configuration concerns what parameters define the behavior of all components; Composition concerns how the prior four aspects are coupling and interacting.

As a comparison between the 5C's principle and the method used in this thesis, the former provides a comprehensive methodology addressing each of these five concerns, while the latter addresses a subset or a special case of these concerns. Table 2.7 describes such differences on each concern of the 5Cs. While the method used in this thesis is not as comprehensive and generic as the 5C's principle, this thesis shares two common goals with the 5C's principle and approaches these goals from different aspects.

There are two common goals between 5C's principle and this thesis. Among these 5Cs, the first four "C"s are motivated by the desire to decouple the design concerns as much as possible, while the last "C", Composition, aims to model the couplings that are inherently required between certain components. This thesis also aims to achieve these two goals, but approaches them from different aspects.

Regarding the goal of decoupling, the 5C's principle explicitly decouples four concerns (Computation, Communication, Coordination, Configuration). In com-

parison, the method used in this thesis implicitly decouples a subset of these four concerns, but instead explicitly decouples the design matrix at domain boundary, such as reducing the coupling patterns from (2.1) to (2.2).

Regarding the goal of modeling the couplings of components, the 5C-based composition pattern (Vanthienen et al., 2014) groups system entities together into composites that support hierarchy, and models the interactions between these system entities. Such a composition pattern provides an architectural building block that is reusable and maintainable. In comparison, the method used in this thesis explicitly composes a single layer of hierarchy over each domain, and optionally applies additional layers of hierarchy to reduce the number of parameters for design space exploration, which is exemplified in (2.3) and (2.4). It models the coupling in a hierarchical way to achieve the end result of reduced complexity in design space exploration, and does not investigate into the issue of re-usability and maintainability that is addressed by the 5C-based composition pattern. In addition to modeling, it investigates into exploration and optimization of the composed system, which is not explicitly addressed by the 5C-based composition pattern.

Therefore, the method used in this thesis is complementary to the 5C's principle in achieving these common goals. In addition, both the 5C's principle and the method of this thesis have been applied to visual servo systems. A comparison of them on the use case of visual servo systems is provided in the next section, along with other similar work.

2.4.5 Cross-domain optimization of visual servo systems

There exists prior works that address specific coupling problems in the design matrix (2.2) in order to optimize the overall system performance. As an example, in (Chen et al., 2017), couplings between illumination, optics, image sensors, and vision algorithms, as well as their impacts on measurement errors are explored. Another example is (Medina et al., 2017), which examines effects of processing resource on sample rate, delay, controller gain, and the resulted control performance.

The 5C's principle of separation of concerns has also been applied to visual servo systems, as described in (Zhang et al., 2014a) and (Zhang et al., 2014b). They focus on methodology and architecture that support the composition of components from different domains, and not on the issue of cross-domain optimization with regard to overall system performance. More specifically, in (Zhang et al., 2014b), it is demonstrated that optimizations on image size, computational precision, and mathematical functions can be performed within the proposed framework. However, it does explore the resulted accuracy and latency trade-off with regard to the overall performance of the system.

In summary, the aforementioned related works have not fully addressed the cou-

pling problems in the design matrix (2.2). A contribution of this thesis is in applying the axiomatic design method to the domains of visual servo systems, such that a holistic and quantitative framework is constructed for cross-domain modeling and optimization of visual servo systems. It bridges the gap between methods that address a specific set of coupling problems in visual servo systems and generic methods for cross-domain modeling.

2.5 Summary

Prior work is presented in four areas relevant to this thesis, that is, design of high-speed vision systems, controller design for visual feedback, methods of exploring algorithmic and architectural choices, and cross-domain modeling and optimization. Prior work is reviewed and compared with this thesis, through which open issues are identified and contributions of this thesis are described. Methods that address these challenges are elaborated in subsequent chapters.

It is revealed in this chapter that, despite recent advances in the four areas mentioned above, there are open issues in each area when combining the methods of these areas into the design and implementation of visual servo systems. These open issues are identified when putting related work of each area in the context of visual servo systems. The contributions of this thesis are a set of methods that help designers overcome these issues. The proposed methods are not only tools useful for analyzing existing systems or designing incrementally improved systems, but also for evaluating and setting up a road-map for future generations of systems.

Chapter 3

Design of high-speed precision vision systems

One bit of IO/sec per instruction/sec; One byte of memory per instruction/sec.

Amdahl's Laws

Abstract. *This chapter describes the design method of high-speed vision systems and applies it to a case study. A dedicated vision processing pipeline for the case study is designed and its accuracy is analyzed. The vision pipeline is implemented on an FPGA-based processing system, consisting of a dedicated vision accelerator and a programmable processor. The implementation is analyzed in terms of sample rate, delay, and resource utilization. The implemented vision system achieves micro-/nano-scale vision measurement at a sample rate higher than* $1\ KHz$*, which is limited by the camera readout speed.*

3.1 Introduction

A high-speed vision system consists of several major components, as illustrated in Figure 3.1. The performance of a vision system, typically specified by measure-

Parts of this chapter have appeared in (He et al., 2011b; Ye et al., 2011a,b).

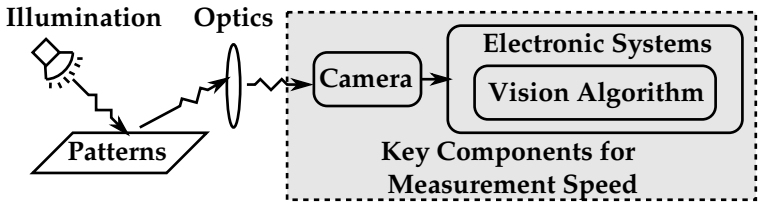


Figure 3.1: Major components of a high-speed vision system. This chapter focuses on the measurement speed, which largely depends on the choices of camera, vision algorithm, and electronic systems.

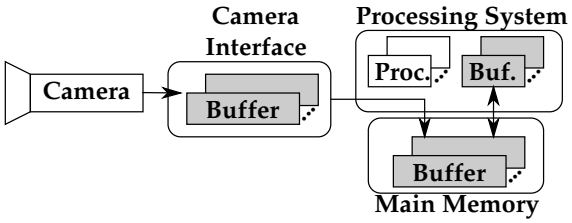


Figure 3.2: An example of vision system that consists of multiple buffers.

ment quality and speed, depends on the design choice of these components. The *measurement quality* depends on the choices of illumination, optics, camera, and vision algorithm. Assuming the illumination and optics are already properly chosen, the *measurement speed* depends on the choices of camera, vision algorithms and electronic systems. This chapter focuses on the design issues of measurement speed, while Chapter 6 will address the design issues of measurement quality.

With emphasis on measurement speed, the vision system should have a high *sample rate* and small *delay*, both of wich are critical to the stability of digital control systems (Franklin et al., 1997). The delay of a typical vision system is induced by the multi-stage buffering of images, from capturing to subsequent processing, as illustrated in Figure 3.2. The end-to-end delay of the vision system is often larger than the inverse of sample rate, i.e., the sample period, as illustrated in Figure 3.3. In a closed loop system, both sample rate and delay are critical to system performance. Therefore, both of them need to be addressed.

3.1.1 Sample rate

Most vision systems have synchronous pipelines, i.e., all stages are triggered at the same time. The achievable sample rate, also called throughput, of a synchronous pipelined vision system is limited by the pipeline stage of the largest delay. While the delay of image transfer stages can be estimated by the specification of hardware, the delay of processing stages depends on multiple design choices. Subsequently, the analysis of the sample rate will focus on the processing

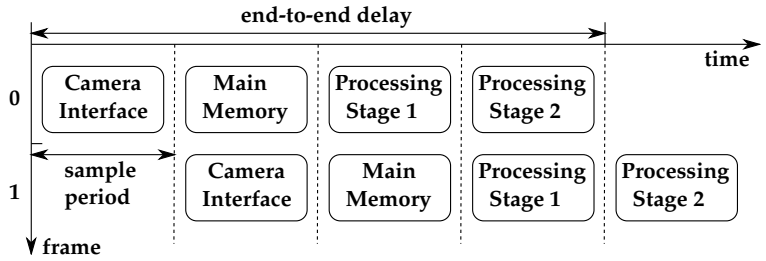


Figure 3.3: An example of vision system that consists of four pipeline stages, two for image transfer and two for processing. The sample period and delay are annotated in this example.

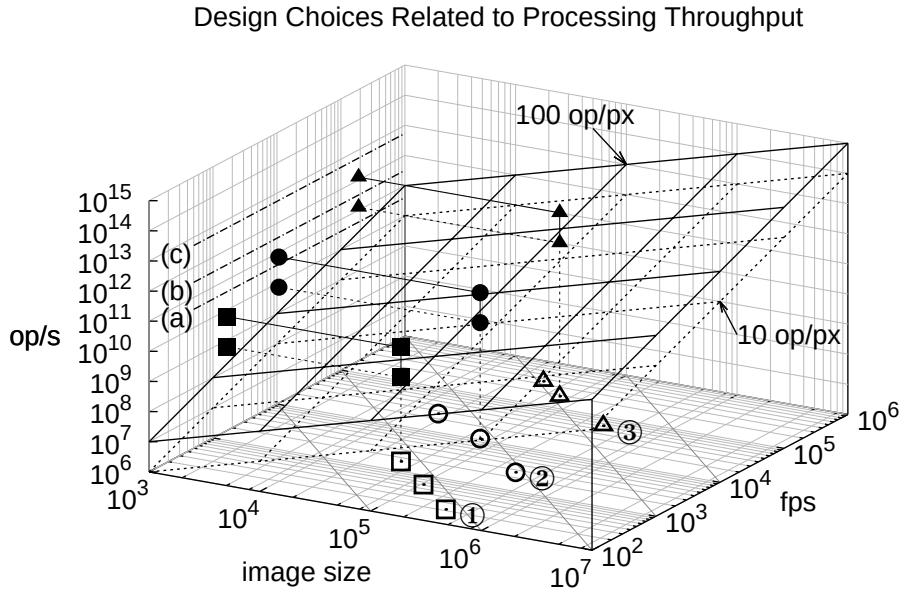


Figure 3.4: The design choices of image sensor, vision algorithm, and processing system. Three image sensors (① ② ③ , see Table 3.1), two vision algorithms of different computational intensity (10 op/px and 100 op/px), and three processing systems ((a)(b)(c), see Table 3.2) are illustrated.

system.

Figure 3.4 illustrates the design choices that affect the throughput of the processing systems. These design choices include the sample size and sample rate of the camera, computational complexity of the vision algorithm, and the processing capacity of the electronic systems. The figure consists of three choices of cameras, described in Table 3.1, two vision algorithms of different computational intensity, and three choices of processing platforms, among others, described in Table 3.2.

① KAI-0340		② LUPA3000		③ Phantom v2010	
image size	frame rate	image size	frame rate	image size	frame rate
640x480	214	640x480	2653	640x480	62600
640x164	618	**256x256**	**10704**	**256x256**	**188900**
228x164	**1637**	128x128	26178	256x128	349500

Table 3.1: Configurations of three image sensors (the Phantom v2010 is a continuous recording system with an undisclosed image sensor). **Bold:** configurations that achieve frame rates at the scales of 1KHz, 10KHz, and 100KHz, respectively, with comparable image sizes.

Label	Processing Platform	gop/s
(a)	DSP (TMS320C6657)	40
	FPGA (XC5VSX50T)	100
	CPU (i5-3570)	136
	CPU (E5-2690)	300
(b)	DSP (TMS320C6678)	320
	FPGA (XC7K325T)	1245
	GPU (GTX 780 Ti)	5046
(c)	FPGA (XC7VX690T)	5335

Table 3.2: Processing systems and their computational capability. The three systems labeled with $\big((a)(b)(c)\big)$ are illustrated in Figure 3.4.

The computational capability of a processing platform is often specified by its applicaiton-independent peak throughput (R_{peak}) and application-independent maximum throughput (R_{max}), both in the unit of (op/s). In practice, vision applications hardly reach the R_{peak} of a modern processing platform (Fowers et al., 2012; Pauwels et al., 2012). Therefore, an application-dependent maximum throughput (R_{max}) is used in subsequent discussions. Given a data input rate of R_d from the camera, in the unit of ($Bytes/second$), and a vision algorithm of computational complexity ρ, in the unit of ($op/Byte$), the output rate of the processing platform (R_c) can be described by

$$R_c = \begin{cases} \rho \cdot R_d & \text{if } (\rho \cdot R_d < R_{max}), \text{ i.e., bandwidth bound,} \\ R_{max} & \text{if } (\rho \cdot R_d \geq R_{max}), \text{ i.e., computation bound,} \end{cases} \tag{3.1}$$

which is equivalent to the performance estimation method of the Roofline model (Williams et al., 2009). Although the Roofline model assumes the performance is either memory bound (bandwidth bound) or computation bound, it can be extended to model other types of bottlenecks (Williams, 2008).

Since most high-speed vision systems are designed to meet performance requirements under cost constraints, the design should lead to a balanced system, "*for which the primary applications are limited in performance by the most expensive component(s) of the system*" (McCalpin, 2004). For high-speed vision systems,

	Sample Rate		
	1KHz	10KHz	100KHz
Processing Platform	CPUs/ DSPs	GPUs/ FPGAs	GPUs/ FPGAs

Table 3.3: Choices of processing platform according to (3.3), assuming an image size of 10^5 pixels and a vision algorithm of $100 op/px$.

the cost is dominated by the camera (see Appendix A for details). Therefore, the throughput of a balanced high-speed vision system should be limited by the throughput of the camera, i.e.,

$$\rho \cdot R_d < R_{max} \tag{3.2}$$

Although R_{max} is unknown before an algorithm is implemented on a processing platform, it can be estimated analytically and empirically. While analytical performance models for multicore processors (Cassidy et al., 2011), GPUs (Hong and Kim, 2009), and FPGAs (Holland et al., 2011) can be accurate, the modeling process is time consuming. An empirical estimation often suffices as a first attempt. According to prior studies, the overhead of supporting instructions (Cope et al., 2010) and the low instruction throughput (Pauwels et al., 2012) can make R_{max} an order of magnitude lower than R_{peak}. Therefore, as a rough estimation, the R_{peak} can be empirically chosen to be an order of magnitude higher than $\rho \cdot R_d$, i.e.,

$$(\rho \cdot R_d) \times 10 < R_{peak}. \tag{3.3}$$

The required R_{peak} estimated by (3.3) can be conservative, which results in choosing an oversized processing platform. However, as discussed earlier, a balanced high-speed vision system allows an oversized processing platform, by as much as an order of magnitude (see Appendix A for details). Therefore, (3.3) can be considered reasonable as an empirical estimation. A better estimation of the relation between R_{max} and R_{peak} will be provided later in this chapter.

Table 3.3 summarizes the choices of processing platform based on the estimation of throughput requirements. These choices are further refined when time delay is considered.

3.1.2 Time delay

Compared to sample rate, delay imposes an even larger challenge for the designers of high-speed vision systems. It has been observed that, in modern processing systems, latency lags bandwidth (Patterson, 2004), i.e., the improvement of delay lags that of sample rate.

		GPU-based systems		FPGA-based systems	
		Device	Delay	Device	Delay
Stage	①	Ethernet	$\approx 10\mu s$	CameraLink	$\approx 10\mu s$
	②	Ethernet Adapter[a]	$\approx 5\mu s$	Dedicated circuit	$\approx 0\mu s$
	③	PCI-e bus to GPU[b]	$\approx 10\mu s$	Dedicated circuit	$\approx 0\mu s$
	④	GPU	$\approx 10\mu s$	Dedicated circuit	$\approx 10\mu s$

Table 3.4: Delay estimations of two processing systems, for a target sample rate of 100KHz, and an image size of 10^5 pixels. The delay is broken down into four stages: ① image readout, ② camera interface, ③ transfer to device memory, ④ vision processing. The delay numbers are obtained from data sheets, listed in the footnote of this table.

[a] The delay of the first package of 64 bytes (MCorelab, 2012)
[b] The delay of copying 10^5 bytes over the PCI-e bus (Lustig and Martonosi, 2013)

Table 3.4 estimates the delay of two vision systems based on GPUs and FPGAs, with delay numbers obtained from datasheets (source provided in the footnote of Table 3.4). The delay analysis assumes the image readout stage and the vision processing stage of both GPU-based and FPGA-based systems can reach 100KHz, and focuses on the buffering stages in between. While both systems can achieve a sample rate of 100KHz, the delay of a GPU-based system can be significantly larger. Therefore, an FPGA-based system is preferable for applications that require minimal delay.

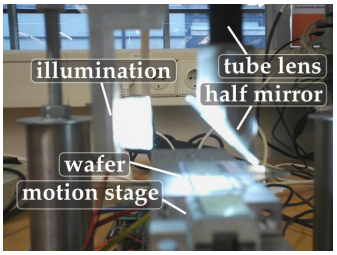
In summary, for minimal delay, an FPGA-based vision system is preferable. According to the delay analysis in Table 3.4, for an FPGA-based vision system, the end-to-end delay (τ) and sample period (h) can be designed to satisfy

$$h \leq \tau \leq 2h. \tag{3.4}$$

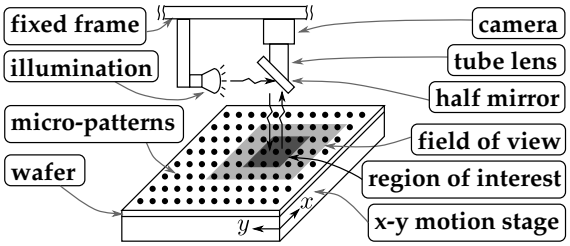
The remaining part of this chapter will apply the generic design principle on a case study. First, the case study will be described. After that, a vision algorithm is designed under computational constraints. Subsequently, the algorithm will be implemented on an FPGA. In the end, the implementation is evaluated.

3.2 Case study: detecting repetitive product patterns

As described in Chapter 1.1, repetitive product patterns are ubiquitous in the semiconductor manufacturing industry, and require a relative low computational workload compared to natural features. Therefore, a case study on repetitive patterns is both theoretically feasible and practically relevant.



(a) The vision system.



(b) The diagram of the vision system.

Figure 3.5: The vision system and its diagram. In the diagram, the micro-patterns, field of view, and region of interest are enlarged for illustration purposes. Their actual size is at micro-meter scale.

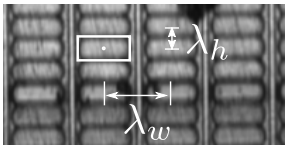


Figure 3.6: An ROI of 160×100 pixels.

Parameter	Description	Value	Unit
λ_w	pitch (width)	220	$[\mu m]$
λ_h	pitch (height)	80	$[\mu m]$
M_o	optical magnification	1.5	[-]
s_p	pixel size of sensor	7.4	$[\mu m]$

Table 3.5: Parameters of the repetitive product patterns, optics, and image sensor.

Figure 3.5 illustrates the vision system for the measurement, in which an OLED display wafer with prefabricated patterns is used. The light source, optics, and camera are mounted on a fixed frame, while the wafer is mounted on a motion stage with two degrees of freedom. The integration between the vision system and the mechanical system is elaborated in Chapter 4. This chapter focuses on the vision system only.

The image sensor is configured to capture a region of interest (ROI) of 160×100 pixels, as shown in Figure 3.6. The figure highlights an OLED cell with a white bounding box, and its center with a white dot. The pitches of the repetitive patterns are also annotated. Table 3.5 summarizes the configuration of the repetitive patterns, optical systems, and image sensors that are relevant to the measurement.

The goal of the vision algorithm is to locate the center of each OLED cell, as illustrated in Figure 3.6. The next section describes possible solutions of this problem.

3.3 Vision algorithms

This section investigates vision algorithms that can locate the centers of the OLED cells. First, several generic solutions will be discussed, which explains why they

are not sufficient for the goal of this case study. After that, a dedicated vision pipeline will be designed.

3.3.1 Generic vision algorithms

Several types of generic vision algorithms can potentially satisfy the requirements of the given task, i.e., detecting and tracking the centers of OLED cells. These algorithms are often used in the context of object tracking, summarized in (Yilmaz et al., 2006). However, due to the specific context of the application and the strict timing requirements, generic vision algorithms have limitations for this case study.

Three generic vision algorithms are particular relevant, but have limitations in the context of this case study. First, gray-scale template matching is commonly used to locate predefined patterns, but it is sensitive to illumination variations (Kim and Arajo, 2007). Second, optical flow can be used to segment objects and locate their centers when objects have a relative motion with regard to the background (Beauchemin and Barron, 1995), but this can only provide incremental motion if objects and background have the same motion. Third, frequency domain analysis and phase correlation can be used to detect local motion and global motion (Reddy and Chatterji, 1996), but for the case study it can only detect global motion, which results in the same limitation as optical flow. These algorithms are summarized in Table 3.6.

Algorithm	Output	Limitation
gray-scale template	location of patterns	sensitive to illumination
optical flow	incremental motion	no absolute position of objects
phase correlation	incremental motion	no absolute position of objects

Table 3.6: Output and limitations of generic vision algorithms *in the context of this case study.*

Beside the three generic algorithms discussed above, there are other generic algorithms that can potentially satisfy the accuracy requirement of this case study. However, these algorithms tend to have irregular memory access patterns, which makes them difficult to be implemented on data-parallel architectures. Therefore, an alternative approach will be pursued.

While each generic algorithm by itself is not sufficient to meet the requirement, a combination of them can potentially lead to feasible solutions. The choice of algorithms to compose a vision pipeline is application-dependent. Therefore, a vision pipeline will be designed specifically in the context of this case study, as described in the next section.

3.3.2 Dedicated vision pipeline

Due to the limitations of generic solutions, a dedicated vision pipeline is designed to detect the centers of the OLED cells. To detect the centers of objects, a typical vision pipeline is to first segment the objects from their background, followed by the localization of their centers, as illustrated in Figure 3.7.

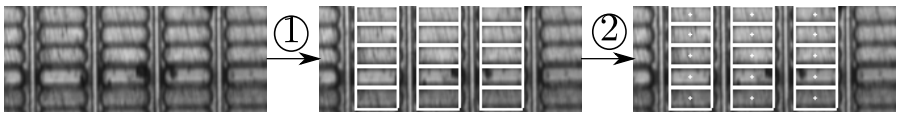


Figure 3.7: A typical vision pipeline for locating the centers of objects. It consists of two major steps: ① segmentation of objects, and ② locating the centers of objects.

For the segmentation algorithm, a common method is to perform thresholding based on a gray scale value. For the case study and the given experimental setup, a single threshold is not sufficient to segment all the objects in the region of interest, as illustrated in Figure 3.8. For the case study, a high threshold fails to separate some OLED cells from the background, while a low threshold fails to separate some OLED cells between each other. It is caused by a nonuniform illumination.

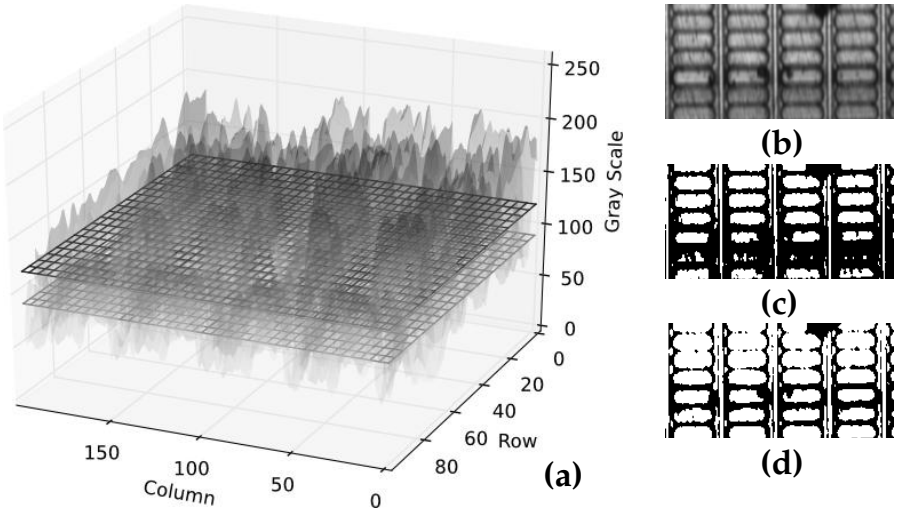


Figure 3.8: Segmentation of the gray scale image at two different threshold values. Due to a nonuniform illumination, one global threshold is not sufficient to segment every pattern. (a) two different thresholds applied to the gray scale input image. (b) input image. (c) a threshold that can segment the upper patterns, but not the low ones. (d) a threshold that can segment the lower patterns, but not the upper ones.

To overcome the issue of a nonuniform illumination, each pixel can be scaled by the local illumination before being segmented. However, such an operation is compute intensive when applied to a two dimensional image. Duo to the fact that repetitive patterns, when properly oriented, can be segmented by their horizontal and vertical projection, the segmentation can be performed on a projection vector instead. The vision algorithm that operates on the projection vector is illustrated in the step ① and ② of Figure 3.9.

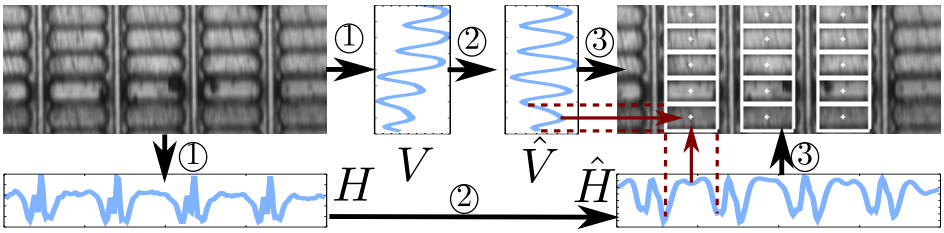


Figure 3.9: Overview of the vision algorithm, which consists of three steps: ① projection, ② filtering, and ③ segmentation and image moment.

As shown in Figure 3.9, the first step of the vision pipeline is projecting the image horizontally and vertically, which results in two vectors H and V. Let $I_{i,j}$ be the pixels of an image of width m and height n, with $1 \leq i \leq m$ and $1 \leq j \leq n$. The projection operation can be described as

$$H_i = \sum_{j=1}^{n} I_{i,j}, \quad V_j = \sum_{i=1}^{m} I_{i,j}. \tag{3.5}$$

The second step is applying filters that eliminate illumination non-uniformity and noise. For the horizontal vector H, a moving-average filter, with window size of $(2w+1)$, is applied to obtained the background illumination $\bar{H}$. By subtracting $\bar{H}$ from H, the illumination-invariant vector $\tilde{H}$ is obtained. To eliminate the high-frequency noise, a low-pass filter G is applied on $\tilde{H}$ to produce a smooth vector $\hat{H}$. The second step is summarized in (3.6). The vertical vector V is processed similarly to obtained $\hat{V}$.

$$\bar{H}_i = \frac{\sum_{j=i-w}^{i+w} H_j}{2w+1}, \quad \tilde{H} = H - \bar{H}, \quad \hat{H} = G * \tilde{H} \tag{3.6}$$

The third step is segmentation and image moment. The segmentation on $\hat{H}$ and $\hat{V}$ produces bounding boxes for OLED cells. Within each bounding box b, the center of the OLED cell (c_x, c_y) is obtained by image moment

$$c_x = \frac{\sum_{i \in b}(i \cdot \hat{H}_i)}{\sum_{i \in b} \hat{H}_i}, \quad c_y = \frac{\sum_{j \in b}(j \cdot \hat{V}_j)}{\sum_{j \in b} \hat{V}_j}. \tag{3.7}$$

3.3.3 Accuracy of the dedicated vision pipeline

After the three steps mentioned above, the center of each OLED cell is located with sub-pixel accuracy. Because the centers of the OLED cells are used for control purposes, the accuracy of the vision algorithm is evaluated by its measurement error. If an image at position x makes a displacement d in the next frame, the vision algorithm (lumped to a function g) should ideally detect the displacement d, but in practice it will induce a measurement error ϵ defined as

$$\epsilon(x, d) = g(x + d) - g(x) - d. \tag{3.8}$$

The measurement error ϵ can be obtained by simulating fine-grained displacements d of the input image at different base positions x, as shown in Figure 3.10. The figure only shows the range of one pixel, because ϵ is repetitive across pixels. According to simulation, the measurement error induced by the vision algorithm is within 0.05 pixel, which is equal to 250 nm.

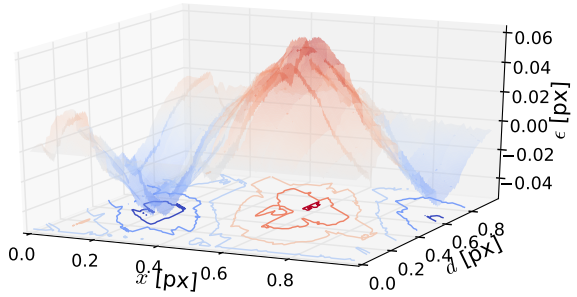


Figure 3.10: Measurement error, defined in (3.8), induced by the vision algorithm.

Therefore, the accuracy of the dedicated vision pipeline satisfies the accuracy requirement of the case study. The next step is to implemented the vision pipeline under the requirements of sample rate and delay.

3.4 Electronic systems

As discussed in Chapter 3.1, an FPGA is the only cost-effective processing platform that can scale to a sample rate above 10 KHz and with a delay less than two sample periods, given a vision algorithm of complexity above 10 op/px. Therefore, despite that the camera in the experimental system only has a throughput of 1.6 KHz, an FPGA is chosen for this case study.

Subsequently, an FPGA-based vision system is connected between the camera and the amplifier of the motor, as illustrated in Figure 3.11. The system consists of three major components. First, a dedicated vision accelerator is used for compute-intensive vision algorithms. Second, a programmable processor is

used to perform the light-weight and irregular high-level vision processing, and the control algorithm. Third, a shared memory is used to communicate the data between the vision accelerator and the programmable processor.

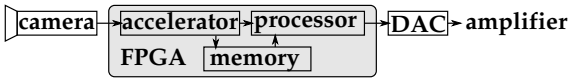


Figure 3.11: Overview of the system architecture

3.4.1 Hardware-software partitioning

To implement the vision pipeline on the architecture illustrated in Figure 3.11, the vision pipeline needs to be partitioned into two parts, one implemented on the dedicated accelerator, the other implemented on the programmable processor. This step is also called hardware-software partitioning. A typical method to perform hardware-software partition is to analyze the computational complexity, operations per data, and the regularity of memory access of the algorithms. Subsequently, the vision algorithm is analyzed from these three aspects, as described in Table 3.7. The steps of high computational complexity or high operations per data, and with regular memory access patterns, are mapped to the dedicated accelerator on the FPGA.

Table 3.7: Algorithm analysis and mapping. The three steps are the same as in Figure 3.9, that is, ① projection, ② filtering, ③ segmentation and image moment.

Step	Comp. complexity	Op. per data	Memory	Mapping
①	$O(m \times n)$	Low	Regular	Accelerator
②	$O(m + n)$	High	Regular	Accelerator
③	$O(m + n)$	Low	Irregular	Processor

3.4.2 Design of vision accelerator

According to the analysis in Table 3.7, the first two stages of the vision pipeline, i.e., projection and filtering, will be implemented on the dedicated vision accelerator. The dedicated circuit design for these two stages are illustrated in Figure 3.12, with details explained next.

The projection step, detailed in Figure 3.13, requires multiple memories to store horizontal vector H, and requires pipeline registers for the projection of vertical vector V. The asymmetric design of the horizontal projection and the vertical projection is due to three major reasons. First, pixels are streamed out from the image sensors row-wise, which requires the vision accelerator to process multiple

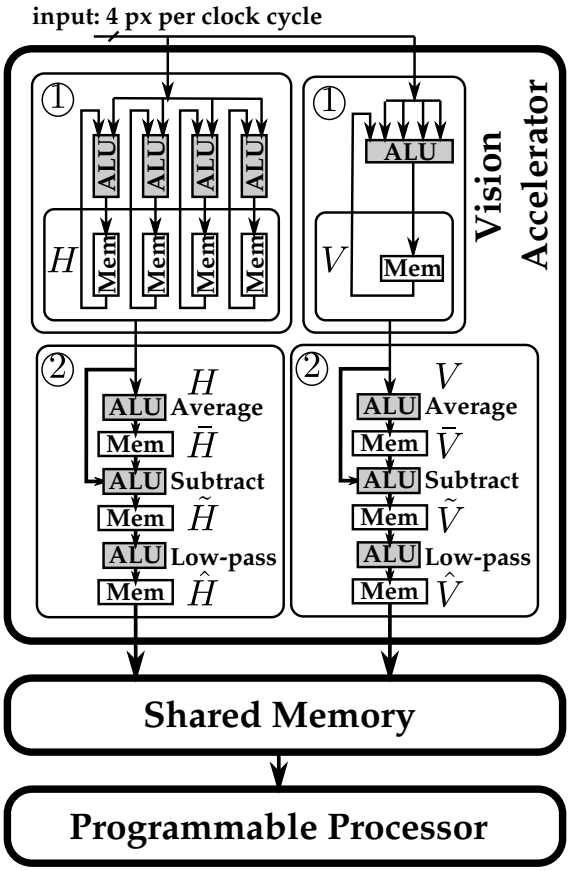


Figure 3.12: Architecture of vision accelerator. The two steps are the same as in Figure 3.9, that is, ① projection and ② filtering. **ALU**: arithmetic logic unit. **Mem**: memory.

pixels of the same image line at the same cycle. Second, for the projection of H, multiple elements of the vector H need to be accessed at the same clock cycle, but the dedicated memory of the FPGA has a limited number of read and write ports. The elements of vector H therefore need to be interleaved over multiple dedicated memories, which allows simultaneous access of multiple elements of H. Third, for the projection of V, pixels at consecutive clock cycles frequently access the same element of V, which can potentially cause a read-after-write hazard if the operation can not be completed in one clock cycle. Therefore, pipeline registers are used to allow multiple cycles of delay in the pipeline.

The projection stage is scalable to a higher throughput, although it is implemented in the case study to process four pixels per clock cycle, which is limited by the throughput of the camera. To scale to a higher throughput, the resources needed for the projection steps scales linearly. For the projection of H, the number of

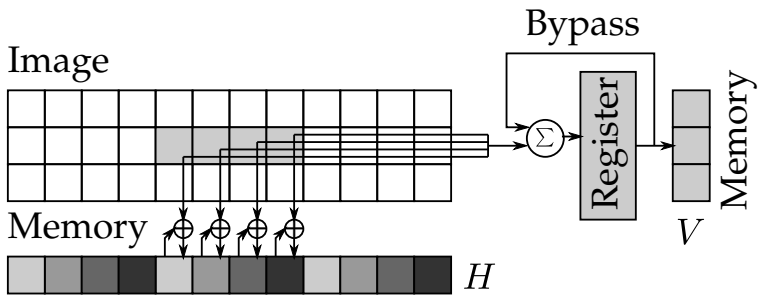


Figure 3.13: Details of the projection step with 4 pixels streamed in at the same clock cycle. The four different colors of the elements of H represent fours different memories to store the elements in an interleaved way.

arithmetic logic units (ALUs) and memories increase linearly with the throughput requirement. For the projection of V, a pipelined reduction tree will be needed for a higher throughput, in which the number of ALUs scales sub-linearly with the throughput requirement. For an extreme case of 100 KHz vision processing, the whole image line needs to be processed in one clock cycle. Such a scenario will be explored in Chapter 6.

Details of the filtering stage to obtain $\hat{H}$ is illustrated in Figure 3.14. The counterpart for $\hat{V}$ is similar, and therefore skipped in the discussion. The filtering stage have three processing elements that each dedicates to one operation, i.e., moving average, subtraction, and low-pass filter. The filtering stage is controlled to sequentially process each vector from head to tail, the processing time therefore is equal to the length of the vector plus the overhead of the window size. Despite the filtering stage uses a sequential, instead of parallel, implementation, its processing time is negligible, which will be elaborated next.

3.5 Evaluation

The processing system is implemented on an Xilinx Virtex 5 FPGA. The vision accelerator is manually implemented at register-transfer level. Two programmable processors (Microblaze) are included in the system, one executing the vision algorithms and control law, the other performing optional tasks, e.g., reading and processing data from other sensors.

The timing breakdown of the vision processing pipeline is shown in Figure 3.15, in which the processing is performed in a pipeline of two stages. The first pipeline stage has two overlapped time lines, representing the overlap of the camera readout and the projection operation. The second pipeline stage is performed by a programmable processor. The vision processing system achieves a sample rate of 1600 KHz, and an end-to-end delay of approximately 1 ms.

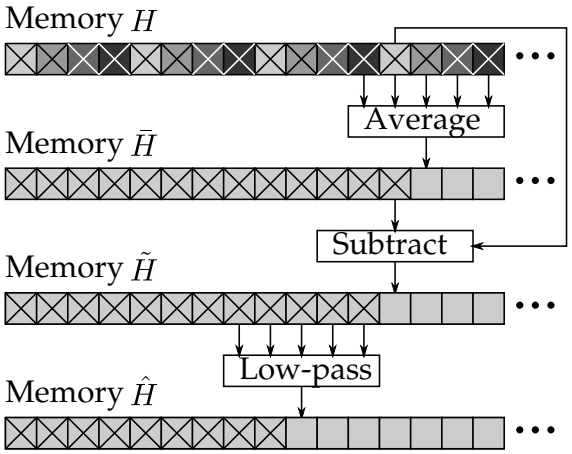


Figure 3.14: Three operations of the filtering stage, controlled to process the vectors from left to right. The memory elements with a cross store the data already processed. For illustration purposes, the window size of filters is set to 5. The actual implementation has a window size of 32.

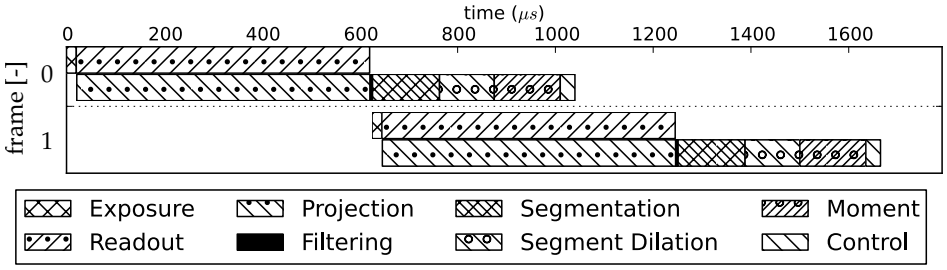


Figure 3.15: Timing breakdown of the vision processing pipeline. The time of the filtering operations is negligible, and therefore hardly visible on this figure.

According to the timing analysis, the image readout from the camera is the bottleneck to further shorten the sample period h. The second stage does not limit the sample rate, but increases the end-to-end delay τ. However, as analyzed in Chapter 3.1, even a properly implemented high-speed vision processing system is expected to have a delay of between one and two sample periods, i.e., $h < \tau < 2h$. Therefore, the delay of the implemented system is within the range of expectation.

The whole processing system utilizes less than half of the available resource available on a middle-range FPGA, as shown in Figure 3.16. The vision accelerator only utilizes approximately 10% of the FPGA. Therefore, if a camera of a higher throughput is available, there is room on the FPGA to scale the architecture accordingly.

In summary, this chapter presents the design method of high-speed vision process-

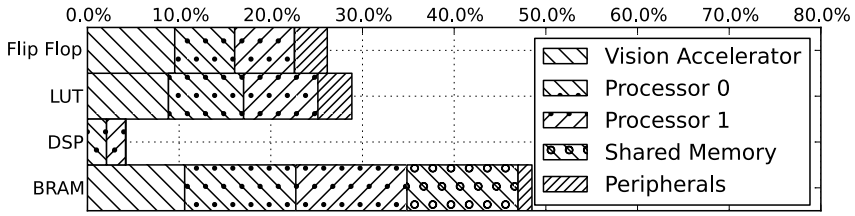


Figure 3.16: Percentage of hardware resource utilized on a Virtex 5 FPGA (5vsx50tff1136). Four types of resources are analyzed: flip flop, look-up table (LUT), digital signal processor (DSP), and block random-access memory (BRAM).

ing systems, and applies it on a case study. The dedicated vision pipeline satisfies the accuracy requirements of the application, while the implemented processing system satisfies the speed requirements of the application. As a proof of concept, visual feedback at micro-/nano-scale accuracy and at 1 KHz sample rate is feasible.

However, it remains challenging to integrate visual feedback into closed-loop control systems. The next chapter discusses the implementation method of visual servo systems, and applies it to the case study.

Chapter 4

Implementation of visual servo systems

Scientists build to learn; Engineers learn to build.

Fred Brooks

Abstract. *There are three major challenges of using visual feedback in direct visual servo systems, that is, a low sample rate, significant delay, and coarse quantization. The characterization of these three parameters requires prototyping. This chapter performs the implementation of a minimum viable prototype for such a purpose. A baseline controller and a trajectory generator are designed for the evaluation of the visual servoing prototype. While the visual servo prototype is subject to further optimizations, the experimental results obtained in this chapter make a case for the advantage of visual feedback for precision motion control.*

4.1 Introduction

Visual servo systems utilize visual feedback for control applications in various ways, which are summarized by Hutchinson et al. (1996). Visual servo systems

Parts of this chapter have appeared in (Pieters et al., 2014).

are commonly classified by whether the visual feedback is used directly in the inner loop or indirectly in the outer loop, and whether the control law utilizes the visual feedback based on image features in image space or based on positions in world coordinate. The difference between these classifications are illustrated in Figure 4.1 and Figure 4.2, and elaborated as follows.

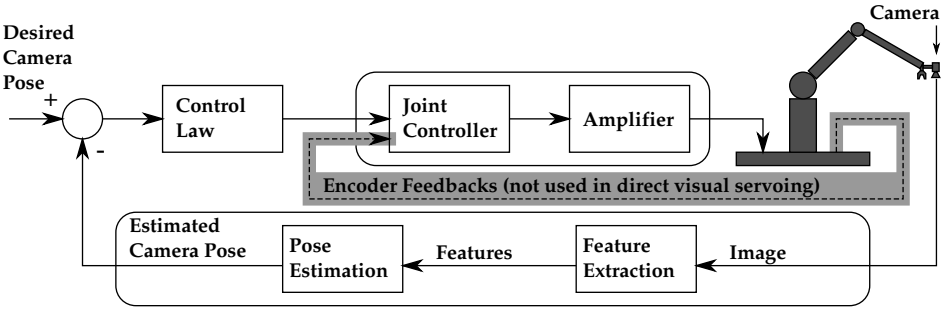


Figure 4.1: Direct and indirect position-based visual servoing, adapted from (Hutchinson et al., 1996). The encoder feedback (with gray background) is not used in direct visual servoing.

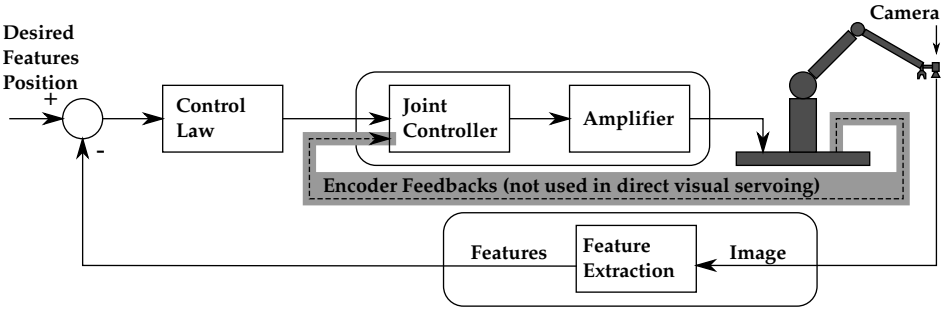


Figure 4.2: Direct and indirect image-based visual servoing, adapted from (Hutchinson et al., 1996). The encoder feedback (with gray background) is not used in direct visual servoing.

Direct and indirect visual servoing

Indirect visual servo systems, also called dynamic look-and-move visual servo systems, are typically configured as multi-loop systems that utilize visual feedback in the outer loop. Direct visual servo systems, on the other hand, utilize visual feedback in the inner loop. Indirect visual servoing on the one hand allows a low-speed visual feedback, but on the other hand is subjected to measurement errors of other sensors. For applications in which other sensors cannot provide an accuracy feedback, indirect visual servo systems are limited by the low-speed visual feedback. For such applications, direct visual servoing with a high-speed and accurate visual feedback is preferable.

Image-based and position-based visual servoing

Image based visual servoing (IBVS) uses image features, e.g., points and edges, in the image space to derive the control values. Position based visual servoing (PBVS), on the other hand, uses the pose of the camera or the object in world coordinate to derive the control values. As explained by Hutchinson et al. (1996), IBVS does not require an accurate kinematic model of the system, but the controller for IBVS is difficult to design if the plant is nonlinear and highly coupled. On the other hand, PBVS derives the Cartesian pose for the controller which simplifies the controller design, but it is sensitive to model errors and calibration errors.

The choice of the types of visual servoing configuration depends on the application and system, and this thesis focuses on image-based direct visual servoing, for reasons described as follows. First, this thesis investigates precision motion control applications in which only visual measurement can provide an accurate feedback for the control objectives. For such applications, direct visual servoing is preferable. Second, this research is driven by a case study that utilizes a linear motion stage, as described in the previous chapter, and therefore ideal for IBVS. For these reasons, image-based direct visual servoing is chosen to be the focus of this study.

While the research issues related to IBVS and PBVS are well addressed (Janabi-Sharifi et al., 2011), direct visual servoing, compared to indirect visual servoing, remains challenging in multiple ways. The remaining parts of this chapter focus on direct visual servoing.

4.2 Related work

Three major challenges are imposed by visual feedback for direct visual servo systems, which require specific treatments besides those for typical control systems. These three challenges are, as described by Corke and Good (1996), "*a relatively low sample rate, significant latency (one or more sample intervals) and coarse quantization*". Extensive researches have attempted to tackle each of these three challenges.

4.2.1 Sample rate and delay

The first two issues, sample rate and delay, are commonly tackled together. Four approaches, and a combination of them, have been widely used. First, controllers can be designed to alleviate the effects of a low sample rate (Fujimoto and Hori, 2001) and large delay (Khamesee et al., 2008), up to a certain limit (Zhang et al., 2003). Second, vision algorithms can be modified to increase the sample rate

and reduce the delay, sometimes at the cost of measurement accuracy (Liu et al., 1998). Third, image sensors can be tuned to reduce the size of an image in order to increase speed, for example, using region of interest (Dahmouche et al., 2012) or pixel binning (Nasibov et al., 2010). Fourth, vision processing systems can be designed and optimized to speedup a given vision algorithm. The last approach is most relevant to the work of this chapter, and therefore further investigated.

While all four approaches aforementioned are important, the last approach, design of high-speed vision processing systems, is indispensable to high-performance direct visual servo systems, and therefore applied in this chapter. Other approaches are addressed in subsequent chapters. Indeed, most previous high-performance direct visual servo systems utilize high-speed vision processing. Examples of these systems, and the system designed in Chapter 3 of this thesis, are compared in Figure 4.3.

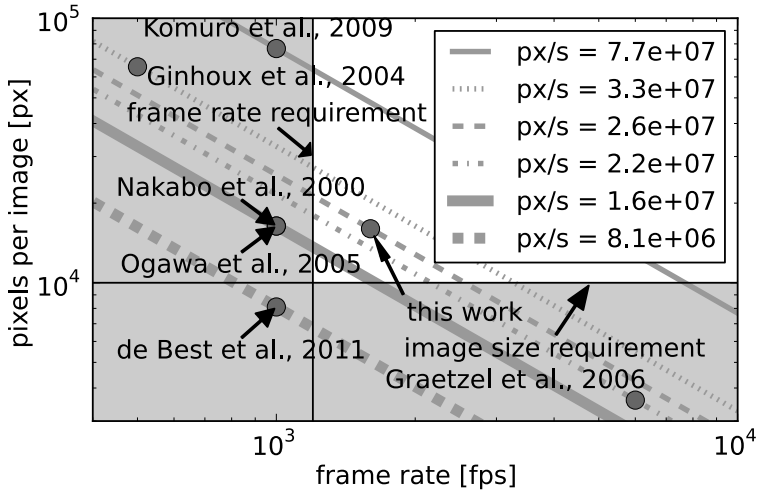


Figure 4.3: High-speed vision systems and corresponding equal-throughput lines. The references are (Komuro et al., 2009), (Ginhoux et al., 2004), (Nakabo et al., 2000), (Ogawa et al., 2005), (de Best, 2011) and (Graetzel et al., 2006).

Figure 4.3 annotates the image size and frame rate of several high-speed vision systems, and their corresponding equal-throughput lines. The vision system presented in (Komuro et al., 2009), which utilizes a customized chip that incorporates an image sensor and a dedicated processing system, has the highest throughput. An earlier system with a similar design is presented in (Ogawa et al., 2005) and (Nakabo et al., 2000), which is implemented using discrete components and therefore with a relatively lower throughput. Other systems are implemented using commercial off-the-shelf components, the throughput of which are mainly limited by the cameras.

At a given throughput, vision systems can be designed to trade image size with frame rate, as explained in Chapter 3. While the vision system implemented in

this work is not of the highest throughput, it is tuned for a balanced between image size and frame rate, such that it satisfies both the accuracy and speed requirements of the case study.

Besides throughput, the delay of a vision system is also critical for control performance. The high-speed vision systems described in Figure 4.3 have a delay between one and two sample periods, i.e., $h \leq \tau \leq 2h$, as shown in Table 4.1. The vision system implemented in this work is aligned with others in terms of delay.

Reference	Sample period (h)	Delay (τ)
(Graetzel et al., 2006)	0.16 ms	0.16 ms
this work	0.6 ms	1 ms
(Komuro et al., 2009)	1 ms	1 ms
(Nakabo et al., 2000; Ogawa et al., 2005)	1 ms	1 ms
(de Best, 2011)	1 ms	2 ms
(Ginhoux et al., 2004)	2 ms	2 ms

Table 4.1: The delays (τ) of high-speed vision systems described in Figure 4.3. Compared to the sample period (h), the delays of these systems are between one and two sample periods, i.e., $h \leq \tau \leq 2h$.

4.2.2 Quantization error

The challenge of a coarse quantization is commonly tackled in three ways, and a combination of them. First, controllers can be designed to alleviate the effects of quantization error, up to a certain limit (Delchamps, 1990). Second, vision algorithms can be designed to reduce the granularity of the quantization (Nobach et al., 2005). Third, optical systems and image sensors can be tuned to increase the resolution of the object, which reduces the quantization error (Hussain and Kabuka, 1990). This chapter applies the last two approaches to alleviate the effects of quantization in the implementation of visual servo systems. The subsequent chapters will further incorporate the first approach.

In summary, while there are multiple approaches to tackle the three challenges of direct visual servo systems, this chapter emphases on the approaches related to the implementation of electronic systems and vision systems. The approaches related to controller design will be addressed in subsequent chapters of this thesis.

Evident from related works aforementioned, prototyping is necessary to characterize the effects of sample rate, delay, and quantization of high-speed visual feedback. To isolate these three issues from the interference of other factors and to implement a cost-effective prototype, a minimum viable prototype[1] that al-

[1] A minimum prototype, adapted from the term "minimum viable product" (Junk, 2000), is a prototype which allows the collection of the maximum amount of validated learning with the least effort.

lows the characterization of these three factors is desired. Subsequently, such a minimum viable prototype is implemented for a case study, and described in the remaining parts of this chapter.

4.3 Prototyping a visual servo system

As described in the related works section, a minimum viable prototype is needed to characterize the sample rate, delay, and quantization of high-speed visual feedback, which are the three major challenges of direct visual servoing.

Requirements and design choices

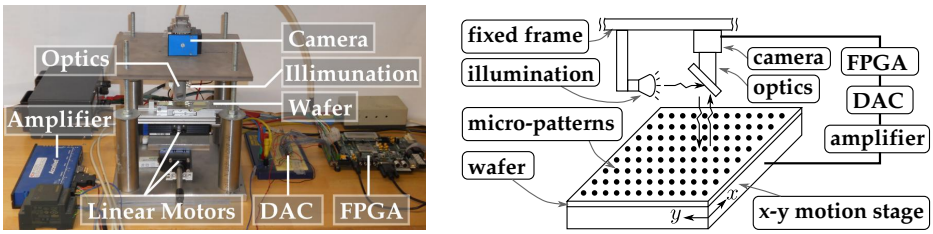
There are several requirements for such a prototype, which lead to specific design choices. First, to characterize the visual feedback at the sample rate of 10 KHz and beyond, only an FPGA-based processing system can cost-effectively scale to such a throughput, as discussed in Chapter 3. Second, the prototype is intended for precision visual servo control, which requires patterns at micrometer or nanometer scale as visual features. Third, precision visual servo systems are mostly configured as two dimensional motion systems (Banerjee and Gupta, 2013), where a simple motion stage with one or two degrees of freedom is preferable.

Based on the requirements aforementioned, this work implements a visual servo prototype that uses an FPGA-based system for vision processing, a wafer with repetitive micro-patterns as visual features, and a linear motion stage as a wafer carrier. This work is based on a previous prototype presented by de Best (2011), but replaces the PC-based processing system with an FPGA-based processing system, and redesigns the vision algorithm for an efficient implementation on FPGAs.

Implementation of the prototype

The visual servoing prototype and its diagram are shown in Figure 4.4, with detailed specifications listed in Table 4.2. The illumination, optics, and camera are mounted on a fixed frame. The wafer with repetitive micro-patterns is mounted on the motion stage. For the case study, the continuous motion is in the x direction, which is most critical to the performance of the application. Therefore, the focus of this work is visual servoing with one degree of freedom.

This work makes a case for the necessity of prototyping for the characterization of visual feedback. Among all the parameters in Table 4.2, only the sample rate, delay, and measurement error of visual feedback are obtained after the prototype is implemented, while other parameters can be obtained before the prototype is



(a) The visual servoing prototype.

(b) The diagram of the prototype.

Figure 4.4: The visual servoing prototype and its diagram. In the diagram, the micro-patterns on the wafer are enlarged for illustration purposes. Their actual size is at micro-meter scale.

implemented. There are two reasons behind it. First, the design of the vision algorithm depends on the characteristics of the illumination, optics, visual features, and image sensor of the prototype, which are only practically obtainable by in-situ measurements. Second, the sample rate, delay, and measurement error of visual feedback depend on the design of the vision algorithm, among others. Therefore, it reconfirms the necessity of prototyping for the characterization of visual feedback, as discussed in the previous parts of this thesis.

4.4 Dynamics and control

After the prototype is implemented, and visual feedback characterized, a dynamic model of the system can be constructed, followed by controller design. The dynamic model described in this chapter is meant for control purposes, while elaborated models for design space exploration are investigated in Chapter 5.

4.4.1 Dynamic modeling and a baseline controller

The visual servo system of the case study is modeled as a sampled-data system with time delay and quantized feedback, as shown in Figure 4.5. The model of measurement system incorporates the most important dynamics effects of visual feedback, that is, sample rate, delay, and quantization. The plant is modeled as a mass with friction, which is highly nonlinear while operating at a low velocity. Subsequently, feedforward is needed to compensate for the friction, while a feedback controller based on a state estimator is used for tracking the reference signal.

While system identification is instrumental in the design of high-performance controllers, a baseline controller is required at the first place before performing system identification, because the parameters of nonlinear friction are practically

Linear motion stage[a]	
Peak force	53 N
Maximum acceleration	359 m/s^2
Maximum velocity	5.6 m/s
Motor encoder resolution	8 μm
Position repeatability	12 μm
Absolute accuracy	350 μm
Micro-patterns[b]	
Pitch at x direction (width)	220 μm
Pitch at y direction (height)	80 μm
Optics[c] and camera[d]	
Pixel resolution	5 $\mu m/px$
Image size	160 $\times$ 100 px
Region of interest	800 μm $\times$ 500 μm
Vision algorithm	
Measurement error[e]	0.05 px (250 nm)
Processing System[f]	
Sample period	0.6 ms
Delay	1 ms

Table 4.2: Specification of the setup.

[a]Linear motor: Dunkermotoren STA1108-116-S-R03C; Slider: FESTO SLT-10-80-P-A-CC.
[b]Pre-fabricated OLED cells on a display wafer, provided by Roth & Rau BV.
[c]Opto Engineering MC Series: Zero Distortion Macro Lenses MC1.50X, S/N: 03202.
[d]SVS-VISTEK 340XUCP camera with an ON Semiconductor KAI-0340 CCD sensor.
[e]The measurement error is obtained by simulation, described in Chapter 3.
[f]Xilinx Virtex-5 ML506 development board with an XC5VSX50TFFG1136 FPGA.

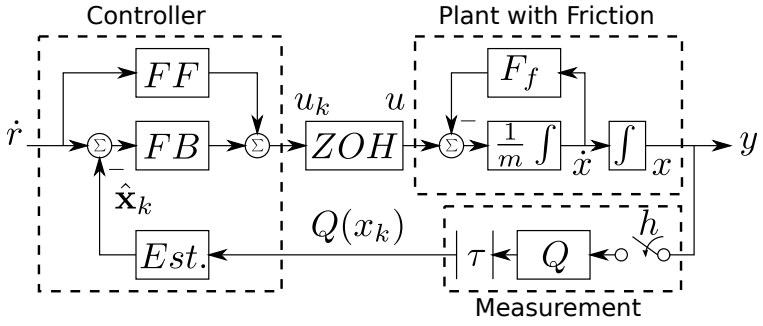


Figure 4.5: System diagram.

difficult to identify in open loop (Altpeter, 1999). Subsequently, a baseline controller is introduced in this chapter. The system identification and the design of high-performance controllers are performed in Chapter 5.

The dynamic model of the system and a baseline controller are described in five major parts as follows. First, the plant model with friction is described as

$$\begin{bmatrix}\dot{x}(t)\\ \ddot{x}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x(t)\\ \dot{x}(t)\end{bmatrix} + \begin{bmatrix}0\\ 1/m\end{bmatrix}\left(u(t) - F_f(t)\right), \tag{4.1}$$

with m representing the mass, x the position, $\dot{x}$ the velocity, u the controller force, and F_f the friction force. The friction force is modeled as

$$F_f = \left(F_C + (F_S - F_C)\, e^{-(\dot{x}/v_s)^2}\right) \cdot sign(\dot{x}) + F_V \dot{x}, \tag{4.2}$$

with F_S representing the stick (static) friction, F_C the Coulomb friction, F_V the viscous friction, v_s the Stribeck velocity.

Second, the sampling, delay, and zero-order hold are modeled as

$$\begin{cases} x_k = x(kh), \quad k \in \mathbb{N} \\ u(t) = u_k, \quad for\ t \in [kh + \tau,\ (k+1)h + \tau) \end{cases} \tag{4.3}$$

with h the sample period, τ the time delay.

Third, the measurement error of visual feedback is described as a quantization function

$$Q(x) = \Delta \left\lfloor \frac{x}{\Delta} + \frac{1}{2} \right\rfloor, \tag{4.4}$$

in which Δ is the granularity of the quantization. The value of Δ during the operation of the plant is yet to be identified.

Fourth, a state estimator is used to obtain the velocity of the plant from position measurements. An $\alpha - \beta$ filter is used as a state estimator for the baseline controller, with time update described by (4.5) and measurement update described by (4.6),

$$\begin{cases} \hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + h \cdot \hat{\dot{x}}_{k-1|k-1} \\ \hat{\dot{x}}_{k|k-1} = \hat{\dot{x}}_{k-1|k-1} \end{cases} \tag{4.5}$$

$$\begin{cases} \hat{x}_{k|k} = \hat{x}_{k|k-1} + \alpha \cdot \left(Q(x_k) - \hat{x}_{k|k-1}\right) \\ \hat{\dot{x}}_{k|k} = \hat{\dot{x}}_{k|k-1} + \frac{\beta}{h} \cdot \left(Q(x_k) - \hat{x}_{k|k-1}\right) \end{cases} \tag{4.6}$$

in which the values of α and β are empirically determined.

Fifth, the feedforward and the feedback controller are described as

$$\begin{cases} u_k = u_k^{ff} + u_k^{fb} \\ u_k^{ff} = \hat{F}_s \\ u_k^{fb} = K_P\left(e_k\right) + K_I\left(\sum_{i=1}^{k} e_i \cdot h\right) + K_D\left(\frac{e_k - e_{k-1}}{h}\right) \\ e_k = \dot{r}_k - \hat{\dot{x}}_k \end{cases} \tag{4.7}$$

in which the u^{ff} and u^{fb} are the feedforward and feedback force respectively. The feedforward only compensates for the static friction, which can be identified in open loop, while a better feedforward can be designed after system identification. The feedback controller is a proportional-integral-derivative (PID) controller for velocity. The gains of the PID controller, K_P, K_I, and K_D, are manually tuned.

4.4.2 Vision based online trajectory generation

For most precision mechatronics systems, a reference trajectory is designed to ensure smooth motion, eliminate vibration, and improve the overall speed of operations, while taking into account kinematic and dynamic constraints. Specific to the application of the case study, trajectory generation is used to improve the overall speed of inkjet printing of an OLED display wafer, while maintaining a high printing quality. On the one hand, to ensure a high printing quality, when the inkjet printhead is firing a droplet at the point of interest, that is, the center of each OLED cell, the printhead is required to have a low velocity relative to the OLED wafer. On the other hand, to improve the overall speed of the printing process, the printhead is required to move from one OLED cell to anther at a high velocity. Therefore, trajectory generation is investigated for the visual servoing prototype system from three aspects, including the type of trajectory, the method to incorporate visual feedback for trajectory generation, and the optimization specific to the case study.

First, the type of trajectory for the case study is a polynomial of degree n, described as

$$q(t) = \sum_{j=0}^{n} a_j t^j, \tag{4.8}$$

which consists of $n+1$ coefficients a_j. A polynomial of degree five is required to ensure smooth acceleration at two points, an initial point x_i at time t_i and a final point x_f at time t_f respectively, which can be described by

$$\begin{bmatrix} q(t_i) \\ \dot{q}(t_i) \\ \ddot{q}(t_i) \\ q(t_f) \\ \dot{q}(t_f) \\ \ddot{q}(t_f) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & t_i & t_i^2 & t_i^3 & t_i^4 & t_i^5 \\ 0 & 1 & 2t_i & 3t_i^2 & 4t_i^3 & 5t_i^4 \\ 0 & 0 & 2 & 6t_i & 12t_i^2 & 20t_i^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} x_i \\ \dot{x}_i \\ \ddot{x}_i \\ x_f \\ \dot{x}_f \\ \ddot{x}_f \end{bmatrix}}_{\mathbf{q}}. \tag{4.9}$$

If the elements of $\mathbf{q}$ and the time t_i and t_f are given, the coefficients of the trajectory can be determined by inverting the matrix $\mathbf{T}$, and therefore,

$$\mathbf{a} = \mathbf{T}^{-1}\mathbf{q}. \tag{4.10}$$

Second, using the visual feedback, two consecutive OLED cells can be detected and set as the initial and final points for the trajectory, which provides the constraints $\mathbf{q}$. Each time the printhead passes an OLED cell, a new trajectory is generated toward the next OLED cell. If the time at the initial point is reset to 0, that is, $t_i = 0$, and the time required to reach the final point is larger than zero, that is, $t_f > 0$, the matrix $\mathbf{T}$ is invertible. In practice, the final time is always set to one or multiple sample periods, which makes $\mathbf{T}$ always invertible and numerically stable.

Third, for the case study, the final time t_f can be set as a constant between two OLED cells, which allows $\mathbf{T}^{-1}$ to be computed offline. Further simplication can be made on the constraints $\mathbf{q}$ by setting acceleration to zero and velcoty to a constant at the points of interest, that is, the center of each OLED cell where the printhead fires the droplet. Given the average pitch of the OLED cells ($\bar{\lambda}$), the desired average velocity of the continuous motion ($\bar{v}$), and the required velocity at the point of interest (v_p), it is possible to set the final time of each point-to-point trajectory as

$$t_f = \bar{\lambda}/\bar{v}, \tag{4.11}$$

and to simplify (4.9) to

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} 0 \\ v_p \\ 0 \\ \lambda \\ v_p \\ 0 \end{bmatrix}}_{\mathbf{q}}, \tag{4.12}$$

in which λ denotes the distance between the centers of two consecutive OLED cells measured online by visual feedback. Despite that the average pitch $\bar{\lambda}$ and all the individual pitches between the OLED cells can be obtained offline, visual feedback can provide online measurements of the pitch λ for each pair of OLED cells, which compensates for the online deformation and misalignment of the OLED wafer.

In summary, the vision based online trajectory generation enables the compensation of online deformation and misalignment of the OLED wafer, and in the meanwhile allows a higher average velocity of the continuous motion than the velocity at the point of interest, that is, $\bar{v} > v_p$. Therefore, it is able to improve the overall speed of the printing process while maintaining a high printing quality. The performance of trajectory generation and tracking is quantified in experiments.

4.5 Experimental validation

The performance metric of the visual servo prototype is the tracking error of velocity references, described as

$$\dot{e}_k = \dot{r}_k - \hat{\dot{x}}_k, \tag{4.13}$$

which can be evaluated in terms of the root mean square error over N samples, that is,

$$RMS(\dot{e}) = \sqrt{(\sum_{k=1}^{N} \dot{e}_k^2)/N}. \tag{4.14}$$

Two types of references are used. The first type of reference is a step function, for which the $RMS(\dot{e})$ includes the samples in the steady state. The step velocity reference is commonly used in printing processes and numerous other applications, for which the steady state performance is most important. The second type of reference signal is online generated trajectory, for which the $RMS(\dot{e})$ includes all the samples during the tracking. The trajectory generation method is described in the previous section, which is commonly used in applications that perform continuous point-to-point motion, for example, high-speed pick-and-place tasks. These two reference signals are summarized in Table 4.3.

Table 4.3: Details of step reference and trajectory reference. The trajectory is generated online based on points of interest with varying pitches (±20%).

Reference	$\bar{v}$ $[mm/s]$	v_p $[mm/s]$	Varying λ $[mm]$
Step	32	32	-
Trajectory	40	32	$\bar{\lambda} = 3.52$, $3.08 \leq \lambda \leq 3.96$

Two different sensors, vision and motor encoder, are used for tracking, while the trajectory references are always generated based on vision. When vision is used for tracking, the motor encoder is not used by the controller, and vice versa. Both sensors are sampled at 1.6 KHz. The motor encoder induces a large amount of measurement noise, which is processed by a low-pass filter with a cutoff frequency empirically tuned at 50Hz. Setting the cutoff frequency either higher or lower results in a larger tracking error.

The results of the experiments are illustrated in Figure 4.6 and summarized in Table 4.4. The motor encoder of the prototype is not designed for precision measurement, therefore it is mainly used as an auxiliary check. Indeed the measurements obtained from vision and motor encoder are aligned with each other and have comparable tracking errors. While the vision measurement leads to smaller tracking errors than those of the motor encoder, the main advantage of visual feedback shown in the experiment is that it can perform online compensation for varying pitches between points of interest.

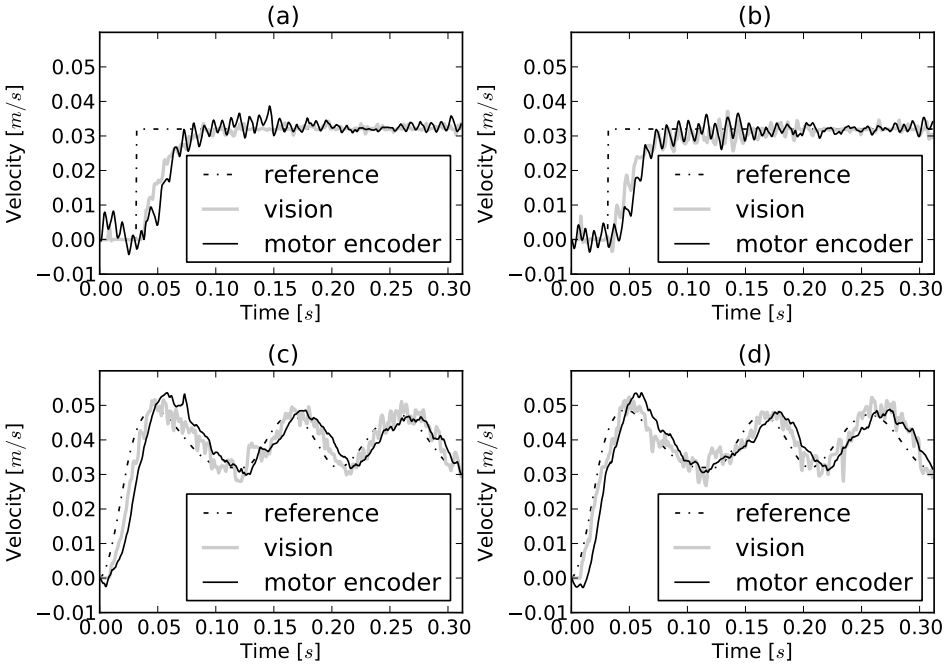


Figure 4.6: Experiments using two types of reference and two different sensors for tracking. (a) step reference tracked by vision. (b) step reference tracked by encoder. (c) trajectory tracked by vision. (d) trajectory tracked by motor encoder. When vision is used for tracking, the motor encoder measurement shown in the figure is not used by the controller, and vice versa.

Table 4.4: Results of tracking two types of references with two different sensors.

Reference	Sensor	Figure	$RMS(\dot{e})$ $[mm/s]$
Step	Vision	Figure 4.6 (a)	0.6
Step	Motor encoder	Figure 4.6 (b)	1.1
Trajectory	Vision	Figure 4.6 (c)	4.0
Trajectory	Motor encoder	Figure 4.6 (d)	5.8

In summary, the vision measurement outperforms a baseline motor encoder for the tracking of typical reference signals, and, more importantly, it is able to perform online measurement and compensation for varying pitches between points of interest, which would otherwise be impossible by other sensors. Therefore, the visual servo prototype does not only proves the feasibility of using visual feedback for precision motion control, but also makes a case that it can improve the performance of precision motion control by measuring and compensating for certain online disturbance.

The performance of the visual servo system can be further improved. System

identification would be required for the design of high-performance controllers. The design choices made for the implementation of the prototype are sub-optimal, which requires further exploration. These issues are addressed in subsequent chapters.

Chapter 5

Modeling of visual servo systems

All models are wrong, but some are useful.

George E. P. Box

Abstract. *This chapter constructs a cross-domain model of visual servo systems based on the prototype implemented in the previous chapter. The model links design parameters from different domains to the overall performance of the visual servo system. As an example, this chapter chooses four cross-domain parameters that are tightly coupled, that is, image size, vision algorithm, processing system, and the plant. The constructed model derives the sample rate, measurement error, delay, and controller gain of the system, which are subsequently used to obtained the overall performance of the system.*

5.1 Introduction

The multi-domain view of a visual servo system is shown in Figure 5.1. As discussed in Chapter 2.4.2, while there is a large number of parameters from

Parts of this chapter have appeared in (Ye et al., 2018).

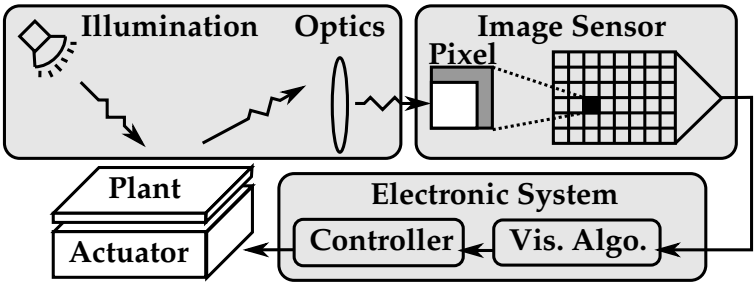


Figure 5.1: Multi-domain view of a visual servo system.

multiple domains, the coupling between them can be analyzed using the axiomatic design method (Suh, 1990; Suh et al., 2001), and the fundamental coupling can be typically represented by the design equation

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \\ P \end{bmatrix}, \tag{5.1}$$

in which the matrix is called a *design matrix*, and the '$\star$' symbol represents dependency. The two vectors on the left and right of the equation, $[\epsilon, h, \tau, K]^T$ and $[I_s, alg., arch, P]^T$, are called requirement vector and design parameter vector respectively, as explained in Table 5.1.

Requirement vector		Design parameter vector	
Symbol	Description	Symbol	Description
h	Sample period	I_s	Image size
ϵ	Measurement error	$alg.$	Vision algorithm
τ	Delay	$arch.$	Processing architecture
K	Controller gain	P	Plant

Table 5.1: Explanations of the requirement vector and design parameter vector in (5.1).

The remaining parts of this chapter model the four elements of the requirement vectors in order, that is, sample period (h), measurement error (ϵ), delay (τ), and controller gain (K), and in the end relate the requirement vector to the performance of the visual servo system.

5.2 Sample period

This section focuses on vision systems consisting of multiple pipeline stages, as shown in Figure 5.2 and discussed in Chapter 3.1, in which the sample period is

decoupled from the vision algorithm and the processing architecture. While such a design is not universally used by every vision system, it is necessary for the implementation of a balanced high-speed vision systems, as discussed in Chapter 3.1.

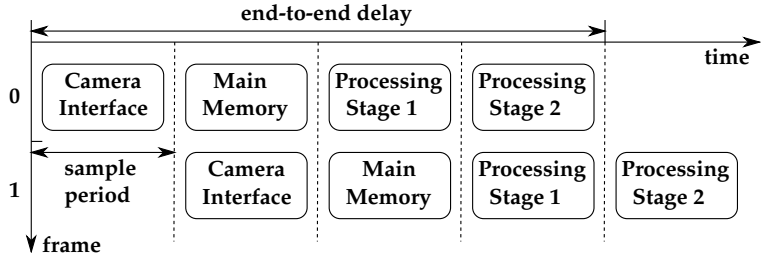


Figure 5.2: An example of a vision system that consists of four pipeline stages, two for image transfer and two for processing. The sample period and delay are annotated in this example.

By adopting a design that decouples the sample period from the vision algorithm and the processing architecture, the sample period mainly consists of the exposure time and the image transfer time, which are modeled as

$$h = t_{exp} + \frac{I_s}{R_d}, \tag{5.2}$$

with t_{exp} representing the exposure time, I_s the image size, and R_d the data rate of the image transfer.

The sample period modeled in (5.2) can be adapted for a specific camera system, by either eliminating redundant parameters or incorporating additional parameters. As an example for the former case, there are cameras that allow an overlap between the image transfer of the current frame and the exposure of the next frame, which therefore eliminates either the exposure time or the image transfer time from h; As an example for the latter case, different cameras may induce different overheads for each image line, image frame, or data package during the image transfer, which requires additional parameters to model the image transfer time. Nevertheless, (5.2) is a representative model that relates the design parameter I_s to the requirement parameter h for a large group of high-speed vision systems.

5.3 Measurement error

The measurement error of visual feedback is defined as the difference between the measured displace of an object compared to its true displacement. For example, if an image at position x makes a displacement d in between two image frames, the

vision algorithm (lumped to a function g) should ideally detect the displacement d, but in practice it will induce a measurement error ϵ defined as

$$\epsilon(x, d) = g(x + d) - g(x) - d. \tag{5.3}$$

It defines the relative error between two measurements. The absolute measurement error of visual sensing at the point of interest is practically difficult to obtain, because the ground truth is often not obtainable from other sensors.

This section focuses on the effects of the image size and vision algorithm on the measurement error. While the measurement error is typically induced by multiple components, including illumination, optics, and image sensor, they can be lumped onto one design parameter, as shown in Appendix B. Therefore, the fundamental coupling behavior for the measurement error can still be represented and investigated using two design parameters, image size and vision algorithm, which are discussed subsequently.

5.3.1 Image size

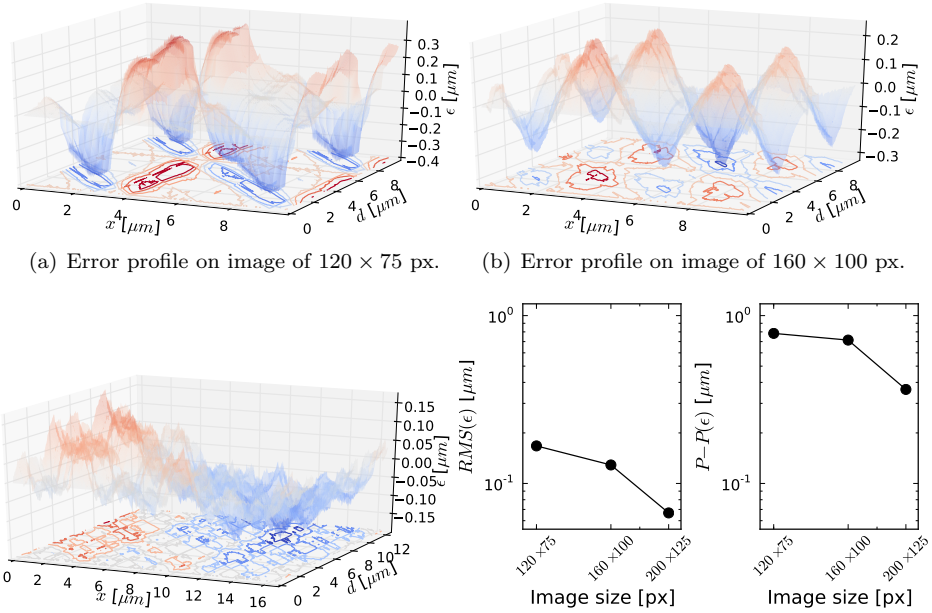
The measurement error of a vision algorithm is characterized in three major steps, using a combination of in-situ measured images and their synthetic versions. First, an image with a region-of-interest (ROI) of 160×100 pixels is obtained by in-situ measurement. Second, the in-situ measured image is downsampled by 25% and upsampled by 25% to obtain images of different sizes. Third, for each size of the image, the corresponding ROI is resampled with sub-pixel shifts to simulate different base position x and displacement d. The properties of these images are summarized in Table 5.2.

Image size	Image Type	Pixel resolution
120×75	synthetic by downsampling	6.67 $[\mu m/px]$
160×100	in-situ measurement	5 $[\mu m/px]$
200×125	synthetic by upsampling	4 $[\mu m/px]$

Table 5.2: Properties of images used for the characterization of measurement error. The input of the vision algorithm is the ROI image, which is resampled with sub-pixels shifts to simulate base position s and displacement d.

The measurement error of the vision algorithm is profiled in Figure 5.3 (a)-(c), given different sizes of input images described in Table 5.2, and the vision algorithm described in Chapter 3.3.2. The measurement error can be characterized in terms of root-mean-square error, denoted RMS(ϵ), and peak-to-peak error, denoted P-P(ϵ), as illustrated in Figure 5.3 (d).

The result indicates that, for the vision algorithm under investigation, the measurement error decreases when the image size increases. Yet two limitations are



(a) Error profile on image of 120 × 75 px.

(b) Error profile on image of 160 × 100 px.

(c) Error profile on image of 200 × 125 px.

(d) Corresponding RMS(ϵ) and P-P(ϵ).

Figure 5.3: Measurement error profile for different image sizes, shown in (a)-(c), and the corresponding root-mean-square error, $RMS(\epsilon)$, and peak-to-peak error, P-P(ϵ), shown in (d).

worth noting. First, the result is based on images with a small variation in size ($\pm 25\%$), and therefore the vision algorithm remains accurate without much modification, except changing the expected feature pitches for different image sizes. If the image size is significantly different, the vision algorithm is required to be redesigned accordingly. Second, the synthetic images can induce artifacts that are not present in the actual measurement, and vice versa. On the other hand, synthetic images are effective for relative comparisons of different vision algorithms (Shimizu and Okutomi, 2005).

Despite these limitations, the methods and results presented in this section are sufficient to investigate the relation between image size and measurement error for the design problems at hand. Nevertheless, if a more accurate and generic model is needed, the current methods and results can be incrementally improved by taking additional measurements and further modifications of the vision algorithm. Therefore, the remaining parts of this chapter proceed with the current methods and results, and extend them to study the effects of vision algorithms on measurement error.

5.3.2 Vision algorithm

It is a common case that multiple vision algorithms can perform the same measurement task, but lead to different measurement errors and measurement delays. This section investigates the effects of different vision algorithms on measurement error.

The measurement error does not only depend on the vision algorithm, but also several other factors, as analyzed by (Alexander and Ng, 1991), including the configuration of the object, the optics, and the image sensor. However, among these factors, the vision algorithm is the design parameter of interest in this chapter. Therefore, this section focuses on the measurement error induced by the vision algorithm, assuming all other factors are equal. In case other factors need to be explicitly modeled as well, they can be combinatorially investigated together with the vision algorithm, as discussed in Appendix B.

The measurement task under investigation is presented as a case study in Chapter 3.2, that is, detecting the centers of repetitive micro-patterns. Three vision algorithms, including one previously introduced in Chapter 3.3.2, are considered. The major steps of these algorithms are summarized in Table 5.3 and illustrated in Figure 5.4 and Figure 5.5. The details of these three algorithms are described subsequently.

Stages	Algorithm 1	Algorithm 2	Algorithm 3
①	projection	binarization	binarization
②	1D filter	2D filter	2D filter
③	segmentation	projection	projection
④	1D moment	segmentation (bounding box)	segmentation (bounding box)
⑤	-	2D moment (bounding box)	segmentation (contour)
⑥	-	-	2D moment (contour)

Table 5.3: Major steps of three vision algorithms.

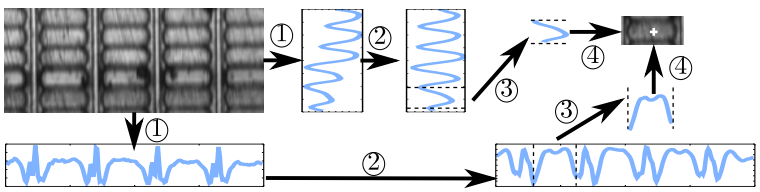


Figure 5.4: Vision algorithm 1, with multiple stages of operations described in Table 5.3.

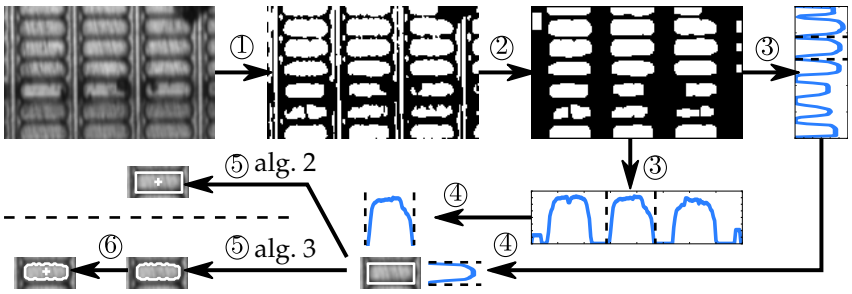


Figure 5.5: Vision algorithm 2 and 3, with multiple stages of operations described in Table 5.3. Algorithm 2 and 3 share the same stages from ① to ④, but differ since stage ⑤. Algorithm 2 performs image moment within a bounding box, while algorithm 3 performs image moment within a mask area, the contour of which is illustrated in this figure.

5.3.2.1 Algorithm 1

Algorithm 1 is introduced in Chapter 3.3.2. For comparison purposes, it is briefly described in this section. The stage ① is projection algorithm, which projects an image horizontally and vertically. It results in two vectors H and V. Let $I_{i,j}$ denote the pixels of an image of width m and height n, with $1 \leq i \leq m$ and $1 \leq j \leq n$. The projection operation can be described as

$$H_i = \sum_{j=1}^{n} I_{i,j}, \quad V_j = \sum_{i=1}^{m} I_{i,j}. \tag{5.4}$$

The stage ② performs a series of filters on two vectors H and V to eliminate illumination non-uniformity and noise. For the horizontal vector H, a moving-average filter, with window size of $(2w+1)$, is applied to obtained the background illumination $\bar{H}$. By subtracting $\bar{H}$ from H, the illumination-invariant vector $\tilde{H}$ is obtained. To eliminate the high-frequency noise, a low-pass filter G is applied on $\tilde{H}$ to produce a smooth vector $\hat{H}$. The algorithm is to process H and V are described in (5.5) and (5.6) respectively.

$$\bar{H}_i = \frac{\sum_{j=i-w}^{i+w} H_j}{2w+1}, \quad \tilde{H} = H - \bar{H}, \quad \hat{H} = G * \tilde{H} \tag{5.5}$$

$$\bar{V}_i = \frac{\sum_{j=i-w}^{i+w} V_j}{2w+1}, \quad \tilde{V} = V - \bar{V}, \quad \hat{V} = G * \tilde{V} \tag{5.6}$$

The stage ③ is segmentation based on two vectors $\hat{H}$ and $\hat{V}$. The segmentation on $\hat{H}$ and $\hat{V}$ produces bounding boxes for OLED cells. The segmentation

algorithm scans the vectors $\hat{H}$ and $\hat{V}$ to find continuous segments with regard to a gray scale threshold and a length threshold. The algorithm that performs the segmentation on $\hat{H}$ is described in Pseudocode 1. The segmentation on $\hat{V}$ is similar, and therefore left out.

Pseudocode 1: Segmentation at stage ③ of algorithm 1.

input : $\hat{H}$ of m elements, gray scale threshold θ_g, length threshold θ_l
output: K segments with start and end coordinates S_i and E_i, $0 \leq i < K$.

```
1  SegmentTracked ← false ;    /* Is a segment currently tracked?  */
2  i ← 0 ;                     /* Index of the segment currently tracked.  */
3  for j ← 0 to m − 1 do
4      if SegmentTracked == false then
5          if Ĥ_j > θ_g then
6              S_i ← j ;              /* Indexing the start of a segment */
7              SegmentTracked ← true ; /* Start tracking the segment */
8      else
9          if Ĥ_j ≤ θ_g then
10             E_i ← j ;              /* Indexing the end of a segment */
11             SegmentTracked ← false ;  /* Stop tracking the segment */
12             if E_i − S_i > θ_l then
13                 i ← i + 1 ;  /* Count as valid; Otherwise overwrite */
14 K ← i;
```

The stage ④ is image moment. Within each segment starting with S_i and ending with E_i, the center of the OLED cell (c_x, c_y) is obtained by performing one dimensional image moment

$$c_x = \frac{\sum_{j\in[S_i,E_i]}(j \cdot H_j)}{\sum_{j\in[S_i,E_i]} H_j}, \quad c_y = \frac{\sum_{j\in[S_i,E_i]}(j \cdot V_j)}{\sum_{j\in[S_i,E_i]} V_j}. \tag{5.7}$$

5.3.2.2 Algorithm 2

The major difference between algorithm 2 and algorithm 1 is that the former performs filtering and image moment on two dimensional images while the latter on one dimensional vectors, as summarized in Table 5.3. Algorithm 2 consists of five stages, illustrated in Figure 5.5, which are explained subsequently.

Stage ① is binarization. It takes as input a gray scale image I, and applies a gray scale threshold θ_g to convert it to a binarized image B.

$$B_{i,j} = \begin{cases} 1 & \text{if } I_{i,j} \geq \theta_g, \\ 0 & \text{if } I_{i,j} < \theta_g. \end{cases} \tag{5.8}$$

Stage ② performs filtering on the binarized image to remove noises. Morphological filters, consisting of erosion and dilation, are applied. The erosion and dilation operations are respectively described as,

$$B_{i,j} = \begin{cases} 1 & \text{if } \forall k,l \in W_{i,j}\text{: } B_{k,l} == 1, \\ 0 & \text{otherwise,} \end{cases} \tag{5.9}$$

$$B_{i,j} = \begin{cases} 1 & \text{if } \exists k,l \in W_{i,j}\text{: } B_{k,l} == 1, \\ 0 & \text{otherwise,} \end{cases} \tag{5.10}$$

in which $W_{i,j}$ denotes coordinates within a 3×3 window which centers at coordinate (i, j).

Stages ③ and ④ perform projection and segmentation in the same way as algorithm 1. The projection and segmentation are described in (5.4) and Pseudocode 1 respectively. Algorithm 2 and 3 create two-dimensional bounding boxes, denoted b_i, from the one-dimensional horizontal and vertical segments, for subsequent processing. The formation of bounding box is trivial and therefore left out.

Stage ⑤ of algorithm 2 performs two-dimensional image moment on the original image I. The image moment operation is performed for each bounding box b, described as

$$c_x = \frac{\sum_{i,j \in b}(i \cdot I_{i,j})}{\sum_{i,j \in b} I_{i,j}}, \quad c_y = \frac{\sum_{i,j \in b}(j \cdot I_{i,j})}{\sum_{i,j \in b} I_{i,j}}, \tag{5.11}$$

in which c_x and c_y are the x and y coordinates of the center of the bounding box, weighted by gray scale values of each pixel.

5.3.2.3 Algorithm 3

As illustrated in Figure 5.5, stages ① to ④ of algorithm 3 are the same as those of algorithm 2. At stage ⑤ , algorithm 3 performs segmentation within the bounding box b, which creates a mask M that separates the OLED structure from the bounding box. The creation of the mask is described as,

$$M_{i,j} = \begin{cases} 1 & \text{if } I_{i,j} \geq \theta_g, \\ 0 & \text{if } I_{i,j} < \theta_g, \end{cases} \tag{5.12}$$

for $i, j \in b$, with θ_g representing a gray scale threshold, which is determined empirically. To simplify the computation, the threshold of the mask can be set to the same value as the threshold of binarization in (5.8). Therefore, the mask can be approximated by the binarized and filtered image B, that is,

$$M = B. \tag{5.13}$$

At stage ⑥ of algorithm 3, for each bounding box b, two-dimensional image moment is applied to the pixels considered valid by the mask M. The algorithm is described as

$$c_x = \frac{\sum_{i,j\in b}(i \cdot I_{i,j} \cdot M_{i,j})}{\sum_{i,j\in b}(I_{i,j} \cdot M_{i,j})}, \quad c_y = \frac{\sum_{i,j\in b}(j \cdot I_{i,j} \cdot M_{i,j})}{\sum_{i,j\in b}(I_{i,j} \cdot M_{i,j})}, \tag{5.14}$$

in which c_x and c_y are the x and y coordinates of the center of the mask, weighted by gray scale values of each pixel. If the aforementioned approximation of $M = B$ is applied, (5.14) can be written as,

$$c_x = \frac{\sum_{i,j\in b}(i \cdot I_{i,j} \cdot B_{i,j})}{\sum_{i,j\in b}(I_{i,j} \cdot B_{i,j})}, \quad c_y = \frac{\sum_{i,j\in b}(j \cdot I_{i,j} \cdot B_{i,j})}{\sum_{i,j\in b}(I_{i,j} \cdot B_{i,j})}. \tag{5.15}$$

5.3.2.4 Measurement error

For the aforementioned algorithms, their measurement errors are defined and profiled in the same way as in Chapter 5.3.1. For an input image size of 160×100 pixels, the measurement errors of the three algorithms are profiled in Figure 5.6 (a)-(c). If different image sizes are considered as well, as discussed in Chapter 5.3.1, the corresponding error characteristics of these three algorithms are illustrated in Figure 5.6 (d).

5.3.3 Modeling measurement errors

According to the profiles of measurement errors of different image sizes and different vision algorithms, illustrated in Figure 5.3 and Figure 5.6, quantization functions are used as an approximation of the measurement error of visual feedback, described as

$$\epsilon(x) = \Delta \cdot (\left\lfloor \frac{x}{\lambda_\Delta} + \frac{1}{2} \right\rfloor - \frac{x}{\lambda_\Delta}), \tag{5.16}$$

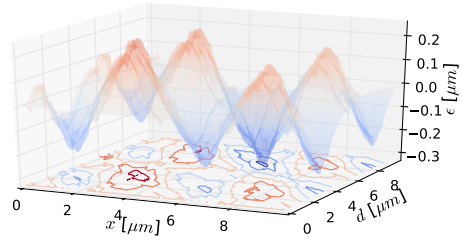
in which Δ is the granularity of the quantization, and λ_Δ is the period of the quantization. The measurement error can be bounded by $\Delta/2$, that is,

$$|\epsilon(x)| \leq \Delta/2. \tag{5.17}$$

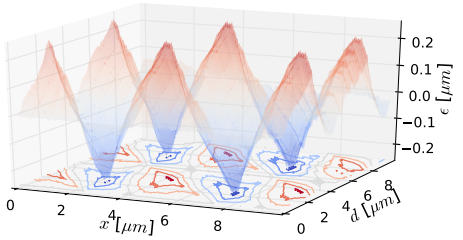
The visual measurement can subsequently be described as a function $Q(x)$, that is,

$$Q(x) = x + \epsilon(x) = (1 - \frac{\Delta}{\lambda_\Delta}) \cdot x + \Delta \cdot (\left\lfloor \frac{x}{\lambda_\Delta} + \frac{1}{2} \right\rfloor). \tag{5.18}$$

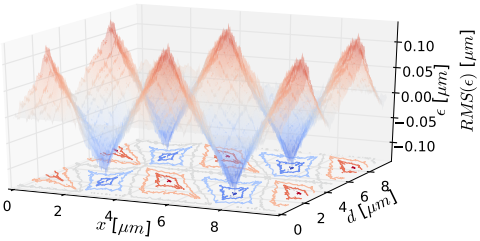
As an example, the $Q(x)$ and $\epsilon(x)$ of algorithm 1 with input size of 160×100 pixels are illustrated in Figure 5.7. The value of Δ is set to the peak-to-peak error, P-P(ϵ). The value of λ_Δ is set to the spatial distance between the peaks.



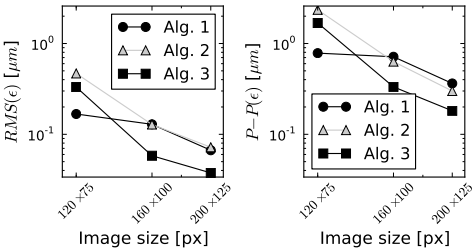
(a) Error profile of algorithm 1.



(b) Error profile of algorithm 2.



(c) Error profile of algorithm 3.



(d) RMS(ϵ) and P-P(ϵ) of different image sizes and algorithms.

Figure 5.6: Measurement error profile of different algorithms on image of 160×100 px., shown in (a)-(c), and the corresponding RMS(ϵ) and P-P(ϵ) when different image sizes are also considered, shown in (d).

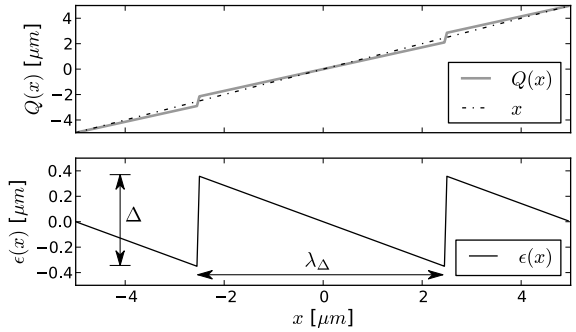


Figure 5.7: The quantization function $Q(x)$ and the corresponding error function $\epsilon(x)$ for algorithm 1 with input size of 160×100 pixels.

5.4 Delay

As described in the design matrix (5.1), the delay (τ) of the vision system depends on three design parameters of interest, that is, image size (I_s), vision algorithm ($alg.$), and processing architecture ($arch.$). The range of image sizes and the options of vision algorithms are described in the previous sections. This section

introduces the design choices of the processing architectures, and their effects on the delay of the vision systems.

5.4.1 Processing architecture

As discussed in Chapter 3, the FPGA is the choice of processing platform for high-speed vision processing. Yet there are multiple options of processing architectures that can be implemented on FPGAs. For low level image processing, data-level parallelism can be exploited effectively by single-instruction-multiple-data (SIMD) architectures. SIMD architectures and their variants are shown to be effective for a wide range of vision processing applications.

Therefore, this work uses an SIMD-like massively parallel architecture template, and adapts it to different image sizes and vision algorithms. To quickly explore different design choices, high-level-synthesis (HLS) is used to implement and adapt the architecture template. The overview of the architecture template is illustrated in Figure 5.8, and explained subsequently.

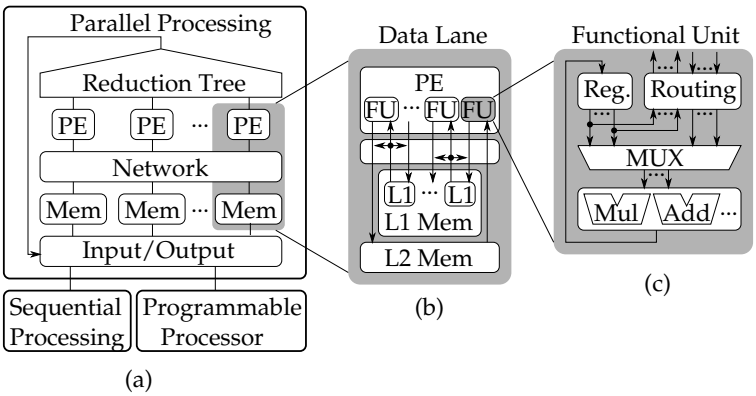


Figure 5.8: Architecture template of the processing system. (a) the overall architecture, with a programmable processor, and two types of dedicated circuits, parallel and sequential. (b) one data lane of the parallel processing system. (c) one functional unit (**FU**) of the processing element (**PE**). Other abbreviations: Memory (**Mem**), Level 1 (**L1**), Level 2 (**L2**), Register (**Reg.**), Multiplexer (**MUX**), Multiplier (**Mul**), Adder (**Add**).

As shown in Figure 5.8, the overall architecture consists of three major components. First, a programmable processor is used to perform light-weight computations or auxiliary tasks. It can be either a built-in hard core or a synthesized soft core on the FPGA. Second, an application-specific parallel processing system is used to accelerate low-level image processing. It is synthesized from behavioral description of the algorithm, written to explicitly reflect the architecture template. Third, an application-specific sequential processing system is used to

perform medium-level or high-level image processing, for example, segmentation and bounding box. It is also synthesized from behavioral description of the algorithm. The three architecture components are summarized in Table 5.4.

Component	Function	Implementation
programmable processor	light-weight computation, and auxiliary tasks	hard core, or soft core
parallel processing system	low-level image processing	high-level synthesis
sequential processing system	medium-/high-level image processing	high-level synthesis

Table 5.4: Major components of the architecture template.

The three vision algorithms described in Chapter 5.3.2 are subsequently implemented using the architecture template. After that, the timing analysis is performed for different vision algorithms and image sizes.

5.4.1.1 Architecture for algorithm 1

Different steps of algorithm 1 are implemented on different components of the architecture template, according to the computational complexity, operations per pixel data, and memory access patterns of each step, as described in Table 5.5. The decisions of the mapping are elaborated next.

Table 5.5: Analysis and mapping of algorithm 1, for the input size of $m \times n$. The four steps are ① projection, ② 1D filter, ③ segmentation, and ④ 1D moment, as described in Chapter 5.3.2. The decisions of mapping are elaborated in the following text.

Step	Comp. complexity	Op. per data	Memory	Mapping
①	$O(m \times n)$	Low	Regular	Parallel proc.
②	$O(m + n)$	High	Regular	Sequential proc.
③	$O(m + n)$	Low	Irregular	Sequential proc.
④	$O(m + n)$	High	Semi-regular	Parallel proc.

Based on the analysis described in Table 5.5, mapping decisions are made. *First*, due to the computational complexity, step ① is mapped to the parallel processing unit. *Second*, due to the irregular memory access, step ③ is mapped to the sequential processing unit. *Third*, step ② can stream results to step ③, therefore it can be mapped to the sequential processing unit together with step ③. *Fourth*, step ④ cannot be overlapped with step ③ due to dependencies between them, and therefore mapped to the parallel processing unit to reduce the delay. The overlap between step ② and ③, and the dependence between step ③ and ④, are illustrated in Figure 5.10 in the timing analysis.

To implement step ④ on the parallel processing unit, its algorithm described by (5.7) are splitted into twp steps. First, parts of the computation can be performed in parallel, and stored in temporary accumulators (ACC), described as

$$ACC_j^H = j \cdot H_j, \quad ACC_j^V = j \cdot V_j. \tag{5.19}$$

Second, the reduction tree derives $\sum ACC_j^H$, $\sum H_j$, $\sum ACC_j^V$, $\sum V_j$ for $j \in [S_i, E_i]$, which are used to compute the 1D image moment,

$$c_x = \frac{\sum_{j\in[S_i,E_i]} ACC_j^H}{\sum_{j\in[S_i,E_i]} H_j}, \quad c_y = \frac{\sum_{j\in[S_i,E_i]} ACC_j^V}{\sum_{j\in[S_i,E_i]} V_j}. \tag{5.20}$$

After the adaptation of step ④ , algorithm 1 is implemented on the architecture template, as illustrated in Figure 5.9. The corresponding timing analysis is illustrated in Figure 5.10. The analysis assumes the image readout rate is one image line per clock cycle, that is, 100 pixels every $10ns$. The timing analysis with realistic readout rate is provided in Chapter 5.4.1.4.

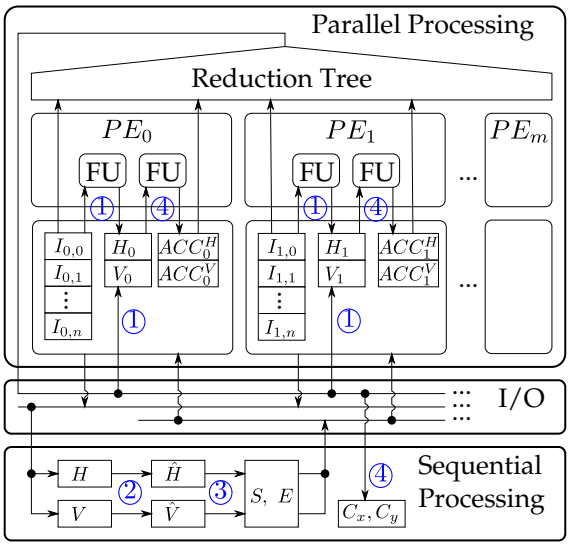


Figure 5.9: Mapping of algorithm 1 on the architecture template.

5.4.1.2 Architecture for algorithm 2

Algorithm 2 is analyzed in a similar way as algorithm 1, described in Table 5.6. The major difference between the two algorithms is that algorithm 2 performs more computation based on 2D data, while algorithm 1 on 1D data. As a consequence, most steps of algorithm 2, except step ④ , have a computational complexity of $m \times n$. Therefore, step ④ is mapped on sequential processing unit, while other steps on parallel processing unit.

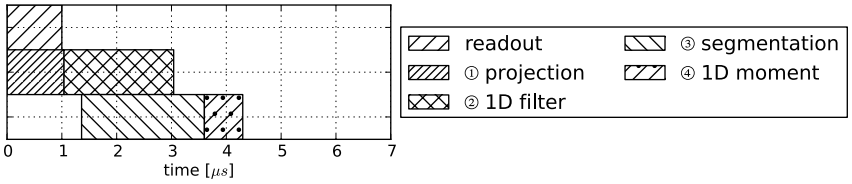


Figure 5.10: Delay of algorithm 1, assuming an image readout rate of one image line per clock cycle, that is, 100 pixels every $10ns$.

Table 5.6: Analysis and mapping of algorithm 2, for the input size of $m \times n$. The four steps are ① binarization, ② 2D filter, ③ projection, and ④ segmentation, ⑤ 2D moment, as described in Chapter 5.3.2. The decisions of mapping are elaborated in the following text.

Step	Comp. complexity	Op. per data	Memory	Mapping
①	$O(m \times n)$	Low	Regular	Parallel proc.
②	$O(m \times n)$	High	Regular	Parallel proc.
③	$O(m \times n)$	Low	Regular	Parallel proc.
④	$O(m + n)$	Low	Irregular	Sequential proc.
⑤	$O(m \times n)$	High	Semi-regular	Parallel proc.

Similar to the 1D moment of algorithm 1, the 2D moment of algorithm 2, described in (5.11), is also adapted to be implemented on the architecture template. First, the multiply-accumulate operations are performed column-wise in parallel, and store in temporary accumulators (ACC),

$$ACC_k^H = \sum_{\substack{i=k,\\ j\in b}} i \cdot I_{i,j}, \quad ACC_k^V = \sum_{\substack{i=k,\\ j\in b}} j \cdot I_{i,j}, \quad ACC_k^I = \sum_{\substack{i=k,\\ j\in b}} I_{i,j}. \tag{5.21}$$

Second, the reduction tree derives $\sum_{i\in b} ACC_i^H$, $\sum_{i\in b} ACC_i^V$, and $\sum_{i\in b} ACC_i^I$, which are used to compute the 2D image moment,

$$c_x = \frac{\sum_{i\in b} ACC_i^H}{\sum_{i\in b} ACC_i^I}, \quad c_y = \frac{\sum_{i\in b} ACC_i^V}{\sum_{i\in b} ACC_i^I}. \tag{5.22}$$

After the adaptation of step ⑤ , algorithm 2 is implemented on the architecture template, as illustrated in Figure 5.11. The corresponding timing analysis is illustrated in Figure 5.12. The analysis assumes the image readout rate is one image line per clock cycle, that is, 100 pixels every $10ns$. The timing analysis with realistic readout rate is provided in Chapter 5.4.1.4.

5.4.1.3 Architecture for algorithm 3

Algorithm 3 has similar steps as algorithm 2, and therefore leads to a similar mapping method. The difference is at step ⑤ and ⑥ . Algorithm 3 performs

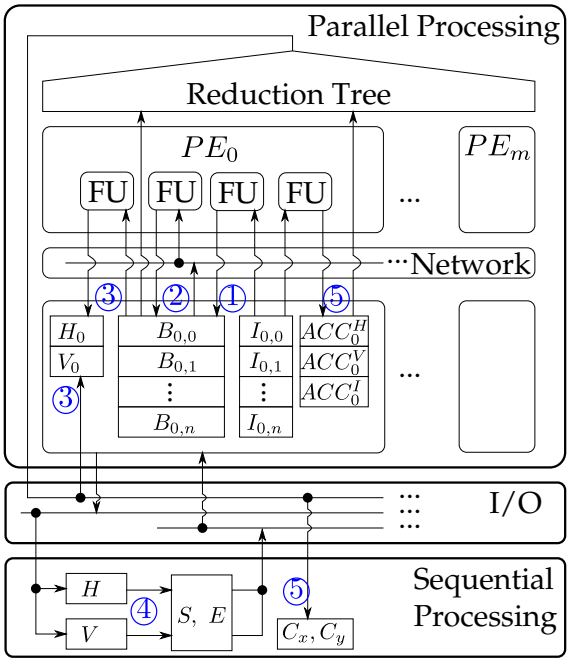


Figure 5.11: Mapping of algorithm 2 on the architecture template.

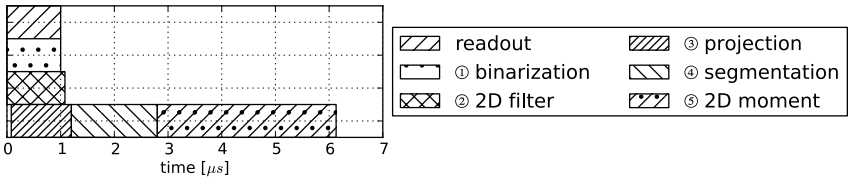


Figure 5.12: Delay of algorithm 2, assuming an image readout rate of one image line per clock cycle, that is, 100 pixels every $10ns$.

image moment based on a contour region, which is defined by a mask M. The mask is approximated by the binary image B, that is $M = B$. Therefore, in the actual implementation, step ⑤ and step ② are performed at the same time, which is illustrated in Figure 5.14 of the timing analysis.

Similar to algorithm 2, the 2D image moment, described by (5.15), is adapted to be mapped on the architecture template. First, the column-wise accumulation is performed in parallel,

$$ACC_k^H = \sum_{\substack{i=k,\\ j\in b}} i \cdot I_{i,j} \cdot B_{i,j}, \quad ACC_k^V = \sum_{\substack{i=k,\\ j\in b}} j \cdot I_{i,j} \cdot B_{i,j}, \quad ACC_k^I = \sum_{\substack{i=k,\\ j\in b}} I_{i,j} \cdot B_{i,j}. \tag{5.23}$$

Table 5.7: Analysis and mapping of algorithm 3, for the input size of $m \times n$. The five steps are ① binarization, ② 2D filter, ③ projection, and ④ segmentation (bounding box), ⑤ segmentation (contour), ⑥ 2D moment, as described in Chapter 5.3.2. The decisions of mapping are elaborated in the following text.

Step	Comp. complexity	Op. per data	Memory	Mapping
①	$O(m \times n)$	Low	Regular	Parallel proc.
②	$O(m \times n)$	High	Regular	Parallel proc.
③	$O(m \times n)$	Low	Regular	Parallel proc.
④	$O(m + n)$	Low	Irregular	Sequential proc.
⑤	$O(m \times n)$	Low	Regular	Parallel proc.
⑥	$O(m \times n)$	High	Semi-regular	Parallel proc.

Second, the reduction tree derives the 2D image moment the same way as that of algorithm 2, described in (5.22).

After the adaptation of step ⑥ , algorithm 3 is implemented on the architecture template, as illustrated in Figure 5.13. The corresponding timing analysis is illustrated in Figure 5.14. The analysis assumes the image readout rate is one image line per clock cycle, that is, 100 pixels every $10ns$. The timing analysis with realistic readout rate is provided next.

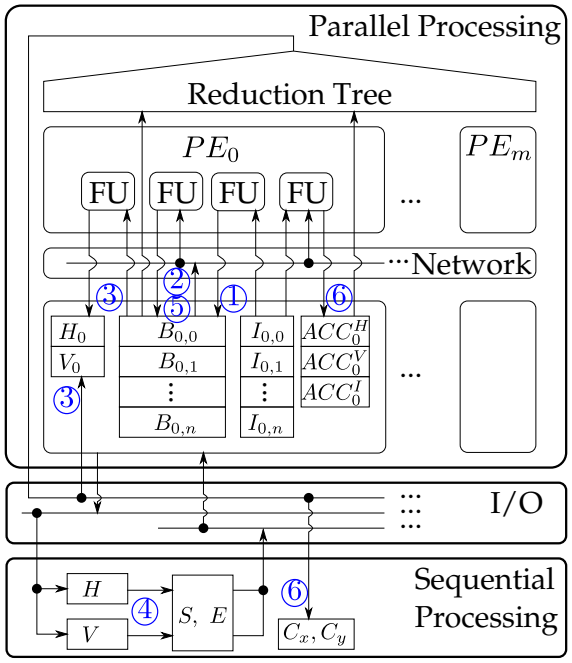


Figure 5.13: Mapping of algorithm 3 on the architecture template.

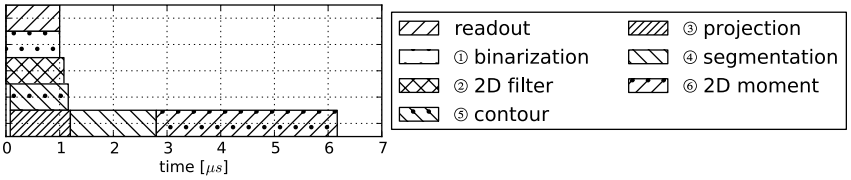
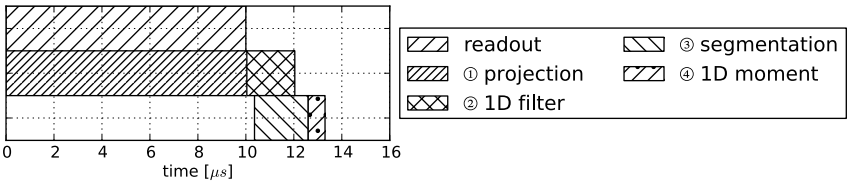


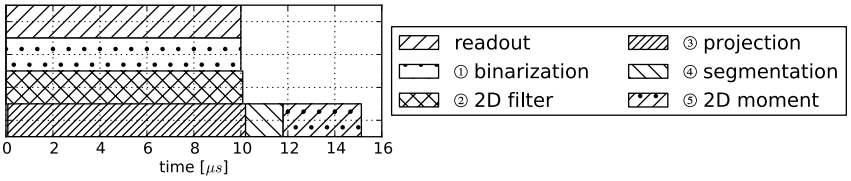
Figure 5.14: Delay of algorithm 3, assuming an image readout rate of one image line per clock cycle, that is, 100 pixels every $10ns$.

5.4.1.4 Evaluation

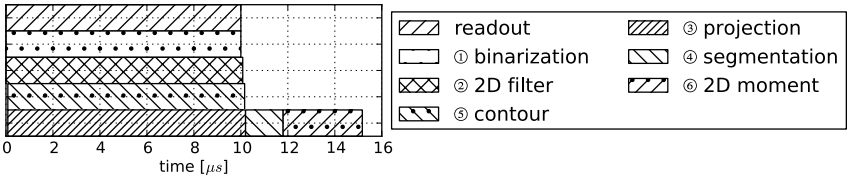
This section derives the delay of visual feedback based on realistic image readout time. As discussed in the Chapter 3, the state-of-the-art sensor can stream images at $100K\ fps$ with a resolution of 100×100 pixels. For such scenarios, the delay of the vision processing are illustrated in Figure 5.15.



(a) Algorithm 1.



(b) Algorithm 2.



(c) Algorithm 3.

Figure 5.15: Delay of three algorithms, assuming an image readout rate of $100K\ fps$ for an image size of 160×100.

For different image sizes, the algorithm can be implemented on the same architecture template by adapting the number of data lanes accordingly. In this section, a simple adaptation of the architecture is performed, that is, setting the number of data lanes equal to the number of columns of the image. The image readout

time is also scaled with the image size, assuming the readout time is proportional to the number of rows in the image. The configurations and the corresponding results are summarized in Table 5.8.

Table 5.8: Configurations and results of different image sizes.

Configurations and results	Image size		
	120×75	160×100	200×125
data lanes [-]	120	160	200
readout time [μs]	7.5	10.0	12.5
sample period [μs]	7.5	10.0	12.5
delay of alg. 1 [μs]	10.4	13.3	16.2
delay of alg. 2 [μs]	11.9	15.1	18.2
delay of alg. 3 [μs]	11.9	15.1	18.2

The delay and measurement errors of three different vision algorithms and three different image sizes are evaluated and illustrated in Figure 5.16. The processing architecture is also adapted accordingly, as described in Table 5.8.

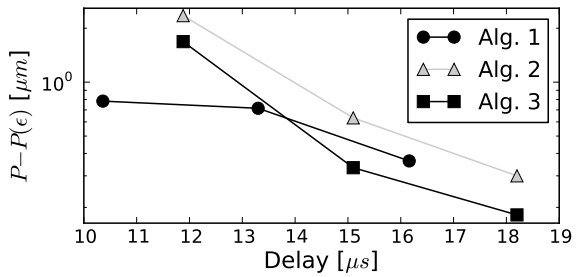


Figure 5.16: The delay and measurement error of three different vision algorithms and three different image sizes, with architectures adapted accordingly, as described in Table 5.8.

According to Figure 5.16, there is a trade-off between measurement error and measurement speed, including sample period and delay. To relate such a trade-off to the overall system performance, the modeling of the plant and the design of the controller are performed in the next section.

5.5 Controller design

As described by (5.1), the controller gain K depends on all design parameters, that is, $[I_s, alg., arch., P]^T$. However, after $[h, \epsilon, \tau]^T$ are obtained from the design parameters $[I_s, alg., arch.]^T$, as discussed in previous sections, K can be derived

from $[h, \epsilon, \tau, P]^T$, that is,

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} h \\ \epsilon \\ \tau \\ P \end{bmatrix} \tag{5.24}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \\ P \end{bmatrix}. \tag{5.25}$$

This section derives the controller gain K from $[h, \epsilon, \tau, P]^T$, assuming h, ϵ, and τ are already available from previous sections. First, a model of the plant is proposed. Second, system identification is performed to obtain parameters of the plant model. Third, the controller gain is designed with regard to the parameters in (5.24).

5.5.1 Plant model

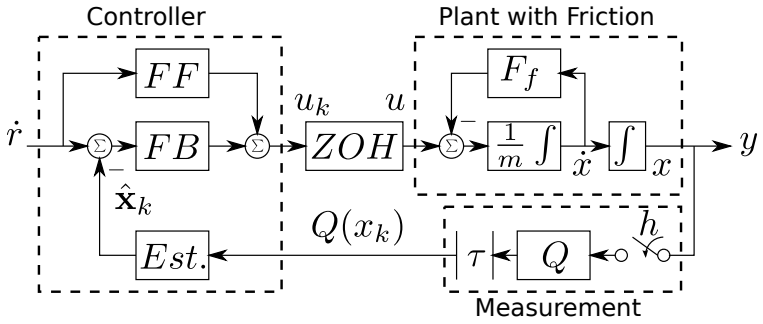


Figure 5.17: Model of the visual servo system.

The system diagram of the visual servo system is illustrated in Figure 5.17, which includes all the parameters in (5.24). The plant is modeled as a mass with friction, described as

$$m\dot{v}(t) + F_f(v(t), u(t)) = u(t), \tag{5.26}$$

with m representing the mass, v the velocity, u the controller force, and F_f the friction force. In the visual servo system, u and v are measurable[1], while m and F_f are to be identified. This work assumes m is constant, and F_f is stick-slip friction. This work uses a stick-slip friction model with Stribeck effect, which is illustrated in Figure 5.18 for the positive velocity region ($v > 0$). Details of the friction model are described as follows.

[1]The velocity v is obtained from a state estimator.

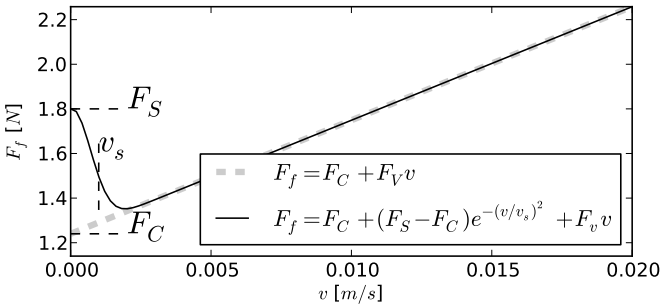


Figure 5.18: The stick-slip friction F_f model in the positive velocity region, without Stribeck effect and with Stribeck effect. The parameters of the friction model are summarized in Table 5.9.

When the plant is stationary, the friction force is equal to the actuation force if the latter is smaller than the static friction F_s. Otherwise the friction force is equal to F_s. That is,

$$F_f = \begin{cases} -u & \text{if } v = 0 \text{ and } |u| < F_s, \\ -sign(u) \cdot F_s & \text{if } v = 0 \text{ and } |u| \geq F_s. \end{cases} \tag{5.27}$$

When the plant is not stationary, the stick-slip friction force F_f can be described as

$$F_f = \left(F_C + (F_S - F_C)\, e^{-(v/v_s)^2}\right) \cdot sign(v) + F_V v \tag{5.28}$$

with F_S representing the stick (static) friction, F_C the Coulomb friction, F_v the viscous friction, v_s the Stribeck velocity. In the experiments, only positive force and velocity are used for identification. Therefore, the friction model is simplified to

$$F_f = F_C + (F_S - F_C)\, e^{-(v/v_s)^2} + F_V v. \tag{5.29}$$

For the plant model described by (5.26) and (5.29), the parameters to be identified are summarized in Table 5.9. The procedure to identify these parameters are described next.

5.5.2 System identification

Accurate measurements of F_f at different velocities cannot be directly obtained, because the measurement of velocity is noisy and the mass is unknown. Instead, steady-state velocity-friction force characteristics can be used for the identification of friction parameters. As discussed in (Altpeter, 1999), the steady-state

Parameter	Description
m	Mass
F_V	Viscous friction
F_C	Coulomb friction
F_S	Static friction
v_s	Stribeck velocity

Table 5.9: Parameters of the plant to be identified.

velocity-friction force characteristics can provide good results for the identification of F_c, F_v and reasonable results for the identification of F_s, while an accurate identification of v_s remains challenging.

For a constant velocity reference $\dot{r}$, the system reaches steady-state when the velocity of the plant v settles at $\dot{r}$ with a small margin, before which the system is at transient period. An example of transient period (T_{tr}) and steady-state period (T_{ss}) that are used for system identification is shown in Figure 5.19. The friction parameters are identified in the steady-state period (T_{ss}), while the mass is identified in the transient period (T_{tr}).

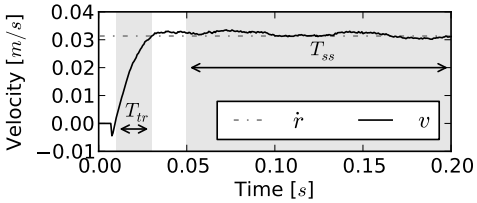


Figure 5.19: An example of transient period (T_{tr}) and steady-state period (T_{ss}) used for system identification, for the system response (v) to a constant velocity reference ($\dot{r}$). The friction parameters are identified in T_{ss}, while the mass is identified in T_{tr}.

5.5.2.1 Identifying friction parameters in steady-state period

To obtain the steady-state velocity-friction force characteristics, this work iteratively measures the step response of the plant under velocity control. In each iteration, the velocity reference is changed but kept constant. Each iteration contributes to a measurement point used to fit the velocity-friction force curve, as shown in Figure 5.20.

To simplify numerical optimization, this work applies the method used in (Garcia et al., 2002), which identifies different friction parameters in separate velocity regions. Using this method, F_v and F_c are identified at a high velocity region, while F_s and v_s are identified at a low velocity region.

The step response at the high velocity region simplifies the identification in two ways. First, at steady state, the acceleration is relatively small, which eliminates the unknown mass from (5.26) and simplifies it to

$$F_C + (F_S - F_C)\, e^{-(v/v_s)^2} + F_V v = u. \tag{5.30}$$

Second, at high velocity, the Stribeck effects can be safely ignored, which further simplifies (5.30) to

$$F_C + F_V v(t) = u(t). \tag{5.31}$$

Due to the noise in velocity measurement and output force, the velocity and control force are averaged over the steady-state period T_{ss}. The average velocity $\bar{v}$ and average output force $\bar{u}$ are computed from

$$\bar{v} = \frac{1}{|T_{ss}|} \int_{T_{ss}} v(t)\, dt, \tag{5.32}$$

$$\bar{u} = \frac{1}{|T_{ss}|} \int_{T_{ss}} u(t)\, dt. \tag{5.33}$$

The aforementioned procedure is applied to a wide range of reference velocities, described in Algorithm 2, which results in a wide range of velocity and force points $(\bar{v}, \bar{u})$.

Algorithm 2: Identification Procedure

input : n velocity values $\{v_1, v_2, ..., v_n\}$
output: n pairs of $(\bar{v}, \bar{u})$ steady-state velocity and force points

1 **for** *each velocity* $v_i \in \{v_1, v_2, ..., v_n\}$ **do**
2 Stabilize the plant at v_i using velocity control;
3 Obtains the trace of u_i and v_i over steady state period $T_{ss,i}$;
4 Compute $\bar{v}_i$ and $\bar{u}_i$ over $T_{ss,i}$ using (5.32) (5.33);

With a wide range of velocity and force points $(\bar{v}, \bar{u})$, the identification of the friction parameters can be performed in two steps, as proposed by (Garcia et al., 2002). First, fit F_V and F_C at high velocity region based on (5.31). Second, fit F_S and v_s over all velocity measurements based on (5.30).

Using the least-squares fitting method based on Levenberg-Marquardt algorithm (More, 1978), the steady-state velocity-friction force curve is obtained and plotted in Figure 5.20.

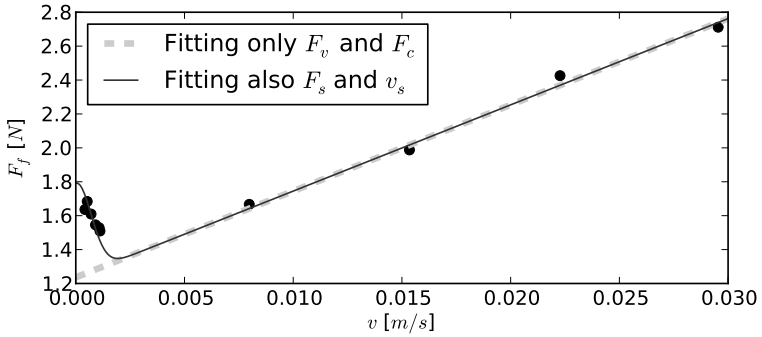


Figure 5.20: Two-step fitting of friction parameters. The measurement points used for fitting are illustrated as dots. First, F_v and F_c are fitted at high velocity region. Second, F_S and v_s are fitted over all velocity measurements.

5.5.2.2 Identifying mass in transient period

After the friction parameters in F_f are identified, the mass m can be identified based on the integral over the transient period T_{tr},

$$m \int_{T_{tr}} \dot{v}\, dt + \int_{T_{tr}} F_f\, dt = \int_{T_{tr}} u(t)\, dt. \tag{5.34}$$

Using the methods described above, all the parameters of the plant model are identified, and listed in Table 5.10.

F_C $[N]$	F_V $[Ns/m]$	F_S $[N]$	v_s $[m/s]$	m $[kg]$
1.24	50.9	1.8	0.001	5.0

Table 5.10: Identified parameters of the plant model.

The plant model with the identified parameters are compared with the experimental results, as shown in Figure 5.21. In the simulation using the plant model, the real velocity of the plant ($\dot{x}$) can be directly obtained, while in experiment the velocity can only be estimated ($\hat{\dot{x}}_k$).

As shown in Figure 5.21, the plant model with the identified parameters can approximate the results obtained from experiments. Due to unmodeled disturbances, the experiment exhibits worse performance than the simulation based on the plant model. Despite that the plant model can be quantitatively compared with experiments and can be further improved, this work focuses on the coupling of multi-domain models. Therefore, the aforementioned plant model is considered sufficient for such a purpose, and used for the controller design in the next section.

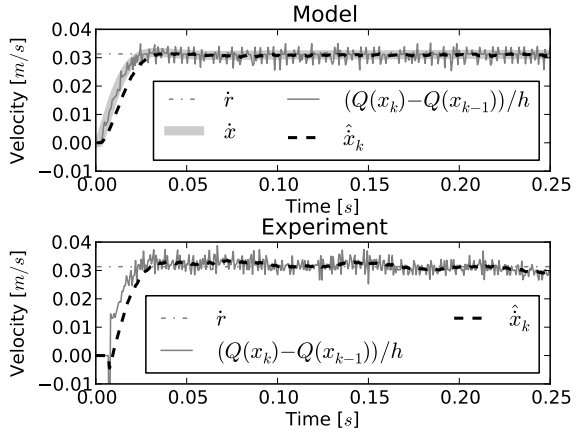


Figure 5.21: Comparison between plant model (top figure) and experiment (bottom figure) on the response of a constant velocity reference ($\dot{r}$). The velocity obtained from a direct difference of two consecutive position measurements divided by the sample period is denoted $(Q(x_k)-Q(x_{k-1}))/h$. The velocity obtained from an estimator is denoted $\hat{\dot{x}}_k$. The real velocity of the plant ($\dot{x}$) is only available in the simulation using the plant model.

5.5.3 Controller design

For a visual servo system illustrated in Figure 5.17, the gain of the feedback controller (K) depends on sample period (h), measurement error (ϵ), and delay (τ), as described by the design equation (5.24). To design a controller based on these parameters, a combination of analytical techniques and simulation is used in this work.

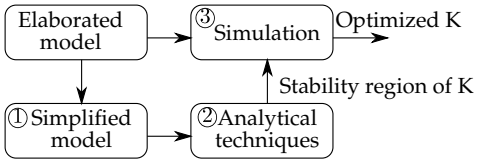


Figure 5.22: The controller is designed using a combination of analytical techniques and simulation, performed in three steps (① ② ③).

The controller design consists of three steps, as illustrated in Figure 5.22. First, the system model is simplified such that analytical techniques can be applied, and all design parameters of interest are incorporated. Second, the simplified model is used to derive a stability region of controller gain K. Third, based on the stability region of K provided by the analytical techniques, simulations are perform using the elaborated system model, which optimizes controller gain K for performance.

This work applies the design approach described in Figure 5.22 for several rea-

sons. First, analytical techniques are necessary to quickly explore the design space, such that simulations are performed subsequently only within the feasible design space. Performing simulation is significantly slower than finding analytical solutions, which makes simulation not practical for an exhaustive search of a large design space. Second, simulation is still needed for performance optimization of controllers, because analytical methods are typically based on simplified models of the system and the reference, therefore less accurate than simulation. Third, a simplified model is instrumental in the application of analytical techniques. Otherwise the system becomes difficult, if not impossible, to analyze. Based on the aforementioned reasons, a simplified model of the system, and a combination of analytical techniques and simulation, are used. Details of these three steps are described subsequently.

5.5.3.1 Simplified model of the system

As explained in the design approach, a simplified model of the system is instrumental in the applicaiton of analytical techniques for controller design. Subsequently, three requirements are imposed on the simplified model. First, is needs to incorporate the design parameters of interest, that is, measurement error (ϵ), sample period (h), and time delay (τ). Second, analytical techniques of controller design can be applied to the simplified model. Third, it is a good approximation of the elaboarated system model.

Guided by the aforementioned requirements, the simulation model in Figure 5.17 is approximated by a quantized sampled-data system with time delay, as shown in Figure 5.23. The simplified model incorporates all the design parameters of interest, can be used by existing controller analysis techniques, and is considered a good approximation of the elaborated model, for three reasons. First, the feedforward can be considered sufficient to compensate for the friction, especially for high-speed continuous motions that are most relevant to this work. Therefore, both the feedforward and friction can be safely omitted in the simplified model. Second, a proportional controller is modeled as the feedback controller, because the proportional gain is often the first parameter to be tuned for various types of controllers. Third, the measurement error of visual feedback can be approximated by a quantization function, according to the discussion in Chapter 5.3.3.

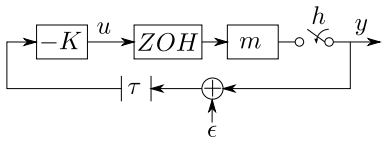


Figure 5.23: A quantized sampled-data system with time delay.

The simplified model of the system can be described in several parts. First, the plant is modelled in continuous time. Second, the effects of sampling, time delay,

and zero-order hold are added. Third, the measurement error is induced and the controller is modelled in discrete time. Last, by incorporating the aforementioned three parts, the closed-loop system is modelled as a quantized sampled-data system with time delay. The details of each part are elaborated as follows.

Plant model

The continuous time model of the plant is

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu^*(t), \\ y(t) &= Cx(t). \end{aligned} \tag{5.35}$$

For the velocity control of a mass, (5.35) is a scalar system, with

$$A = 0, \quad B = \frac{1}{m}, \quad C = 1, \tag{5.36}$$

and $x(t)$ representing the velocity of the plant.

Sampling and time delay

The sampling period h and time delay τ can be lumped into the zero-order hold of the actuator, that is,

$$u^*(t) = u_k, \quad for\ t \in [kh + \tau, kh + h + \tau). \tag{5.37}$$

To reflect the actual experimental setup, the delay is assumed to be larger than one sample period but smaller than two sample period, i.e.,

$$h \leq \tau \leq 2h. \tag{5.38}$$

As discussed in Chapter 4.2.1, most high-speed visual servo systems have the timing characteristics described by (5.38). Therefore, the assumption made by (5.38) is representative. Without lost of generality, shorter, longer, and time-varying delays can also be modelled similarly, as described in (Cloosterman et al., 2009). For the sake of simplicity, (5.38) is assumed for the remaining parts of this chapter.

Controller and measurement error

The measurement error is induced and the controller is described in discrete time, that is,

$$u_k = -K(x_k + \epsilon_k), \tag{5.39}$$

where ϵ is the quantization error induced by measurement, defined as

$$\epsilon_k = Q(x_k) - x_k = \Delta \left\lfloor \frac{x_k}{\Delta} + \frac{1}{2} \right\rfloor - x_k, \tag{5.40}$$

with $\Delta \in \mathbb{R}_{>0}$ representing the quantization granularity. By definition,

$$|\epsilon| \leq \Delta. \tag{5.41}$$

Closed-loop model

Combining the models of different parts mentioned above, the closed-loop system can be modelled as a quantized sample-data system with time delay, that is,

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma_1(-K)(x_{k-2} + \epsilon_{k-2}) + \Gamma_0(-K)(x_{k-1} + \epsilon_{k-1}), \\ y_k &= H x_k \end{aligned} \tag{5.42}$$

The values of Φ, Γ_1, Γ_0, and H are

$$\begin{aligned} &\Phi = e^{Ah} = 1, \\ &\Gamma_1 = \textstyle\int_0^{\tau-h} e^{As} B ds = \frac{\tau-h}{m}, \\ &\Gamma_0 = \textstyle\int_0^{2h-\tau} e^{As} B ds = \frac{2h-\tau}{m}, \\ &H = 1. \end{aligned} \tag{5.43}$$

The simplified model of the system, described in (5.42), includes all the parameters of interest, that is, sample period (h), delay (τ), measurement error (ϵ), and controller gain (K), as stated in (5.24). In the next section, this simplified model is used to analytically derive the controller gain with regard to other parameters.

5.5.3.2 Controller design under stability constraints

Given the simplified model of the system described in (5.42), this section applies stability criteria to analytically derive the constraints of controller gain. Two steps are performed. First, for the case that the measurement error ϵ is zero, the asymptotic stability of the system is analyzed. Second, for the case that the measurement error ϵ is considered as an input, the bounded-input-bounded-output stability of the system is analyzed. The two steps are described as follows.

Asymptotic stability without measurement errors

For the case that measurement error is zero, (5.42) can be described using an extended state $\xi_k = [x_k, x_{k-1}, x_{k-2}]^T$, and becomes

$$\begin{aligned} \xi_{k+1} &= \tilde{A}\xi_k, \\ y_k &= \tilde{C}\xi_k, \end{aligned} \tag{5.44}$$

with

$$\xi_k = \begin{bmatrix} x_k \\ x_{k-1} \\ x_{k-2} \end{bmatrix}, \quad \tilde{A} = \begin{bmatrix} \Phi & -\Gamma_0 K & -\Gamma_1 K \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \tilde{C} = [1,\ 0,\ 0]. \tag{5.45}$$

The system is asymptotically stable if all the eigenvalues of $\tilde{A}$ are within the unit disk. Let $\lambda_i(\tilde{A})$, $i = 1, 2, 3$, denote the three eigenvalues of $\tilde{A}$. The asymptotic stability criteria can be written as

$$|\lambda_i(\tilde{A})| < 1, \quad \forall i = 1, 2, 3. \tag{5.46}$$

Equivalently, the above criteria can be written using the maximum absolute value of the eigenvalues, denoted by $|\lambda(\tilde{A})|_{max}$, that is,

$$|\lambda(\tilde{A})|_{max} < 1. \tag{5.47}$$

In practice, to allow a stability margin and to avoid numerical issues, it is set to be less than 0.95, that is,

$$|\lambda(\tilde{A})|_{max} < 0.95. \tag{5.48}$$

With condition (5.48), stability tests can be performed to determine the range of K, given τ and h. Figure 5.24 shows an example of the stability region with regard to K and τ, assuming a fixed sample period of 0.7 ms.

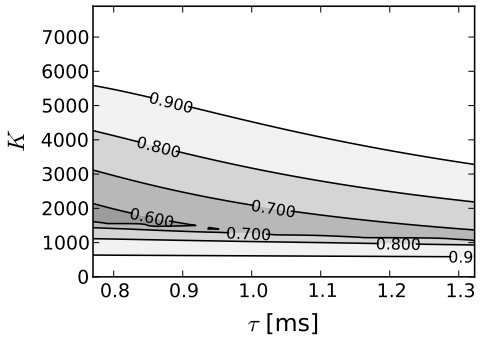


Figure 5.24: The range of controller gain K for different delay τ, under the asymptotic stability constraint (5.48). The curves in the figure represent systems of equal $|\lambda(\tilde{A})|_{max}$, with the values of $|\lambda(\tilde{A})|_{max}$ annotated on the curves.

Bounded-input-bounded-output stability with measurement errors

For the case that the measurement error is not zero, the bounded-input-bounded-output (BIBO) stability is analyzed, in which the input is measurement error and the output is the state error. To derive the BIBO stability, (5.42) is written for two cases, one without quantization error ϵ, that is,

$$\begin{cases} x_{k+1} = \Phi x_k + \Gamma_1(-K)x_{k-2} + \Gamma_0(-K)x_{k-1}, \\ y_k = Hx_k, \end{cases} \tag{5.49}$$

and the other with quantization error ϵ, that is,

$$\begin{cases} \hat{x}_{k+1} = \Phi \hat{x}_k + \Gamma_1(-K)(\hat{x}_{k-2} + \epsilon_{k-2}) + \Gamma_0(-K)(\hat{x}_{k-1} + \epsilon_{k-1}), \\ \hat{y}_k = H\hat{x}_k. \end{cases} \tag{5.50}$$

Define the state error $\tilde{x}$ and output error $\tilde{y}$ as

$$\begin{aligned} \tilde{x} &= x - \hat{x}, \\ \tilde{y} &= y - \hat{y}. \end{aligned} \tag{5.51}$$

By subtracting (5.49) from (5.50), the error dynamics can be described as

$$\tilde{x}_{k+1} = \Phi\tilde{x}_k + \Gamma_1(-K)\tilde{x}_{k-2} + \Gamma_0(-K)\tilde{x}_{k-1} + \Gamma_1 K\epsilon_{k-2} + \Gamma_0 K\epsilon_{k-1}. \tag{5.52}$$

Using the extended states $\tilde{\xi}_k = [\tilde{x}_k, \tilde{x}_{k-1}, \tilde{x}_{k-2}]^T$ and $\tilde{\epsilon}_k = [\epsilon_k, \epsilon_{k-1}]^T$, (5.52) can be written as

$$\begin{aligned} \tilde{\xi}_{k+1} &= \tilde{A}\tilde{\xi}_k + \tilde{B}\tilde{\epsilon}_{k-1}, \\ \tilde{y}_k &= \tilde{C}\tilde{\xi}_k, \end{aligned} \tag{5.53}$$

with

$$\tilde{A} = \begin{bmatrix} \Phi & -\Gamma_0 K & -\Gamma_1 K \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} \Gamma_0 K & \Gamma_1 K \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \tilde{C} = [1,\ 0,\ 0]. \tag{5.54}$$

If the system without quantization error is asymptotically stable, that is, $|\lambda(\tilde{A})|_{max} < 1$, the output error can be bounded with regard to input error,

$$|\tilde{y}| \leq ||\tilde{x}|| \leq \frac{c||\tilde{B}||}{1 - |\lambda(\tilde{A})|_{max}}|\epsilon| \leq \frac{c||\tilde{B}||}{1 - |\lambda(\tilde{A})|_{max}}\Delta, \tag{5.55}$$

with some $c \in \mathbb{R}_{>0}$.

The bound of the output error can be empirically set. For example, if the bound of the output error is 0.1, (5.55) can be extended to

$$|\tilde{y}| \leq \frac{c||\tilde{B}||}{1 - |\lambda(\tilde{A})|_{max}}\Delta < 0.1. \tag{5.56}$$

With condition (5.56), the range of K in Figure 5.24 is further limited by Δ, as shown in Figure 5.25.

The simplified system model and stability analysis provide an approximation of the stability region for gain K. The next section further optimizes the controller with regard to certain performance criteria.

5.5.3.3 Controller optimization for performance

Stability and performance are both important objectives for controller design. This section optimizes the controller gain for performance, based on the stability

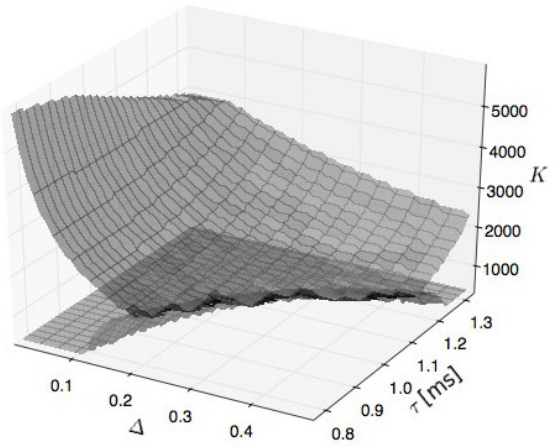


Figure 5.25: The range of controller gain K for different delay τ and quantization error Δ, under the asymptotic stability constraint (5.48) and the bounded-input-bounded-output stability constraint (5.56). The surfaces represent the upper bound and lower bound of K respectively.

region obtained from the previous section. While there are various definitions of control performance, tracking errors of reference signals are particularly relevant for precision motion control, which is the focus of this thesis.

This section uses simulation based on the elaborated system model illustrated in Figure 5.17. Simulation, instead of analytical methods, is used for controller optimization due to two reasons. First, simulation allows the inclusion of complicated models, which are hard to analyze or lead to a conservative design. Second, simulation of control performance for a limited number of design parameters described in (5.24), and within the stability region derived by analytical methods, is computationally effective.

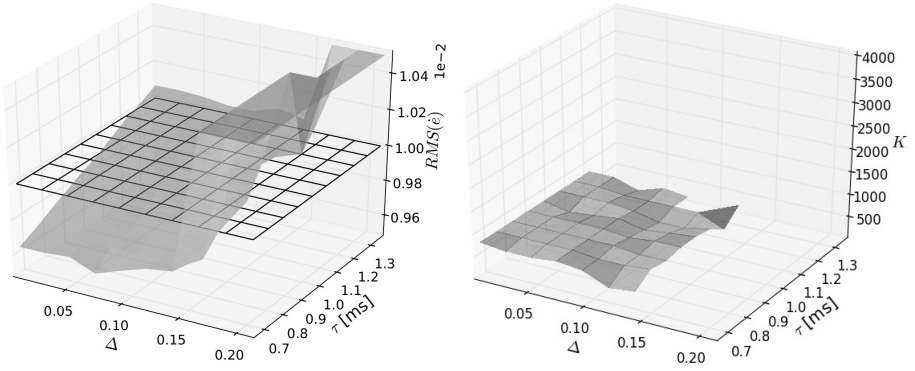
This section investigates the control performance defined as the steady-state tracking error of a constant velocity reference. First, define the tracking error at time k as

$$\dot{e}_k = \dot{r}_k - \dot{x}_k. \tag{5.57}$$

Second, the tracking error is evaluated over N samples at steady state, using the root-mean-square error metric defined as

$$RMS(\dot{e}) = \sqrt{(\sum_{k=1}^{N} \dot{e}_k^2)/N}. \tag{5.58}$$

Subsequently, the controller gain K is tuned to minimize $RMS(\dot{e})$, for different values of quantization error Δ and delay τ. First, a grid of parameters Δ and



(a) The achievable $RMS(\dot{e})$ (surface) and requirement of $RMS(\dot{e})$ (mesh).

(b) The corresponding K that achieves (a) and within the requirement range.

Figure 5.26: The achievable $RMS(\dot{e})$ for different values of quantization error Δ and delay τ, by tuning controller gain K. (a) The achievable $RMS(\dot{e})$ and the requirement of $RMS(\dot{e})$. (b) The corresponding K that achieves the minimal $RMS(\dot{e})$ and within the requirement range.

τ is made for the range of interest. For each Δ and τ value in the grid, the K value that leads to minimal $RMS(\dot{e})$ is considered optimal for the given Δ and τ value. Second, an initial search range of K is made from the stability region, derived by analytical methods as shown in Figure 5.25, at Δ equal to zero and τ equal to h. Third, the search range of K is expended if it leads to a smaller $RMS(\dot{e})$. By applying the aforementioned method on the grid of Δ and τ, the minimal $RMS(\dot{e})$ that can be achieved and the corresponding K are obtained in Figure 5.26. The requirement of the $RMS(\dot{e})$, set to 1×10^{-2}, is plotted along with the minimal $RMS(\dot{e})$ surface in Figure 5.26. The simulation assumes a fixed sample period of 0.7 ms, the same as previous analysis.

The result derived from simulation, shown in Figure 5.26, represents a similar trend as the result derived from analytical methods, shown in Figure 5.25. An increase of the quantization error Δ deteriorates the control performance and leads to instability. On the other hand, an increase in time delay τ deteriorates the control performance, but has a stabilization effect when Δ is large. At the region where Δ and τ are small, the optimal controller gain K derived from simulation is within the stability bound derived from the analytical methods, which indicates that analytical methods are effective to provide an initial search range of controller gain for performance optimization.

By the aforementioned methods, the controller gain K is obtained from $[h, \epsilon, \tau, P]^T$, and therefore from the design parameters $[I_s, alg., arch., P]^T$, as shown in (5.25). The requirement vector $[h, \epsilon, \tau, K]^T$ is also linked to the control performance of the system, measured by tracking error, $RMS(\dot{e})$.

5.6 Summary

As explained in the introduction section of this chapter, the cross-domain coupling of a typical high-speed visual servo systems can be represented by the design equation

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \\ P \end{bmatrix}. \tag{5.59}$$

This chapter proposes models that link each element of the requirement vector, $[h, \epsilon, \tau, K]^T$, to the design parameter vector, $[I_s, alg., arch., P]^T$. In the end, the tracking error, $RMS(\dot{e})$, of a reference signal is used as an example of overall requirement, and methods are proposed to link the requirement vector to the overall requirement of the system, that is,

$$RMS(\dot{e}) = f(\epsilon, h, \tau, K), \tag{5.60}$$

in which f is a function of the requirement vector, and implicitly other external parameters, for example, reference signal (r), as explained in Section 2.4.

This chapter leaves out two relevant issues that are to be further explored in the next chapter. First, this chapter separates the link from design parameters to overall requirements into two parts, described by (5.59) and (5.60) respectively, such that the axiomatic design method can be applied. Yet the designers of visual servo systems need a direct link from design parameters to overall requirements, which is the subject of study in the next chapter. Second, this chapter measures overall requirements using only one criteria, that is, the tracking error of a reference signal. In practice, multiple requirements are present, which are further explored in the next chapter.

Chapter 6

Exploration of design trade-offs

It is not unscientific to make a guess, although many people who are not in science think it is.

Richard Feynman

Abstract. *This chapter explores the design trade-offs across multiple domains in visual servo systems. The exploration of design choices is guided by the axiomatic design method, and applied to the model of a prototype visual servo system. Representative requirements of precision visual servo system, including accuracy, bandwidth, and cost, are used as optimization objectives of design parameters. The result confirms the necessity of cross-domain modeling and exploration, and the effectiveness of the proposed method.*

6.1 Introduction

Visual servo systems involve design choices across multiple domains, as illustrated in Figure 6.1. In the previous chapter, a cross-domain model of visual servo

Parts of this chapter have appeared in (Ye et al., 2018).

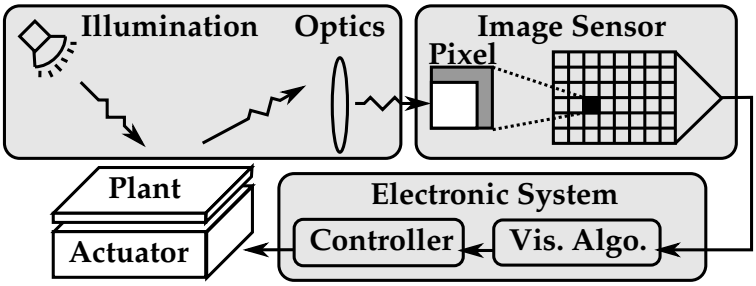


Figure 6.1: Multi-domain view of a visual servo system.

systems is introduced, and the relation between the design parameters and the overall requirement, using the tracking error as an example, is demonstrated. However, the previous chapter only deals with design parameters of specific values, which represent a single instance of design point in the design space.

This chapter explores the design space by investigating different choices of design parameters with regard to the overall functional requirements of the system. First, the choices of functional requirements and design parameters for visual servo systems are discussed. Second, various choices of design parameters are investigated, using tracking error as the functional requirement. After that, multiple functional requirements of visual servo systems are investigated, which imposes further trade-offs in the design parameters. In the end, the implications of the results are discussed.

6.2 Functional requirements and design parameters

The functional requirements of a mechatronic system can be defined using different metrics. A specific set of functional requirements (FR) can be described as a function (f) of design parameters (DP), that is,

$$FR = f(DP). \tag{6.1}$$

The various options of functional requirements (FR) and design parameters (DP), and their application to a prototype visual servo system, are discussed in the following parts of this section.

6.2.1 Functional requirements (FR)

A mechatronic system is designed to achieved a set of performance metrics under certain constraints. The functional requirements therefore consist of performance metrics and constraints. For precision motion control systems, the performance

metrics often includes the precision and bandwidth of the system. The constraints can be the cost of the system, the type of the actuator, and the time delay induced by the off-the-shelf motion controller, to name a few. These two aspects of functional requirements are elaborated as follows.

6.2.1.1 Performance metrics

The choice of a performance metric is often domain specific and application dependent. In the domain of precision motion control, typical performance metrics include precision, bandwidth, and range, which are often interdependent as illustrated in Figure 6.2 (Devasia et al., 2007).

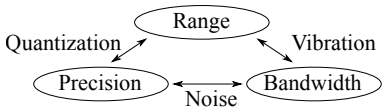


Figure 6.2: Typical performance metrics (precision, bandwidth, and range) and their tradeoffs (due to quantization, noise, and vibration) in the domain of precision motion control. This figure is adapted from (Devasia et al., 2007).

The aforementioned performance metrics and trade-offs also apply to precision visual servo systems, but there exist other performance metrics that are also important and are specific to visual servo systems. First, precision and bandwidth are commonly used as performance metrics in visual servo systems (Gans et al., 2002; Hashimoto and Noritsugu, 1998; Janabi-Sharifi et al., 2011; Okumura et al., 2011), and range is also investigated in applications like visual servo control of scanning probe microscopes (Clayton and McManus, 2011). In addition, there is a wide range of visual servo applications that use other performance metrics, including the magnification and trackability of a moving target (Ogawa et al., 2005), the speed of scanning an area (Tabata et al., 2010), and the time of completing a control task (Zhang et al., 2009), etc.

As a result of the diverse performance metrics for visual servo systems, there are different options of choosing the functional requirement vector. In the discussion above, precision is often associated with tracking error (e). In addition, the magnification and trackability (Ogawa et al., 2005) are directly associated with image size (I_s) and sample period (h). If common performance metrics like bandwidth (BW) is also included, for example, the functional requirement vector becomes

$$FR = [e, I_s, h, BW]^T. \tag{6.2}$$

The performance metric specified by (6.2) is an example. As mentioned earlier, the choice of performance metrics is application dependent. For the prototype visual servo system described in previous chapters of this thesis, the most relevant performance metrics are the root-mean-square error of tracking a velocity reference, denoted $RMS(\dot{e})$, and the bandwidth of the system, denoted BW.

6.2.1.2 Constraints

Besides performance metrics, constraints can be specified as functional requirements similarly. In the terminology of axiomatic design, constraints differ from functional requirements in that they do not have to be independent (Suh, 1998). If a constraint is independent of other functional requirements, the constraint can be included as a functional requirement.

Cost is an important constraint in most engineering designs. In practice, if the performance of a system is good enough, cost becomes an optimization objective. A typical scenario in the design of precision mechatronic system is to explore accuracy and bandwidth at different cost levels. In such a case, the functional requirement vector is

$$FR = [e, BW, cost]^T. \tag{6.3}$$

Besides cost, sensor and actuator impose various constraints on precision motion control systems (Smith and Seugling, 2006). For precision visual servo systems, additional constraints exist. For example, the choice of an image sensor is often fixed due to technical and non-technical reasons, which induces constraints on image size (I_s), sample rate (h), and delay (τ). Under such a constraint, the corresponding requirement vector is

$$FR = [I_s, h, \tau]^T. \tag{6.4}$$

To satisfy the functional requirements described by (6.4), design parameters such as image algorithm ($alg.$) and processing architecture ($arch.$) need to be chosen accordingly, such that the given image size (I_s) can be processed within the given sample period (h) and the given delay (τ).

As an another example, if customized processing systems are used, the processing architecture ($arch.$) and the image processing algorithm ($alg.$) that runs on it are often fixed, which results in a certain measurement error (ϵ) depending on the image size. The corresponding functional requirement vector is

$$FR = [\epsilon, alg., arch.]^T. \tag{6.5}$$

To satisfy the functional requirements described by (6.5), design parameters such as image size (I_s), sample period (h), and delay (τ) need to be chosen accordingly, such that the chosen image size (I_s) leads to a measurement error less than that of the requirement (ϵ), and the chosen sample period (h) and delay (τ) can be achieved by the given algorithm ($alg.$) running on the given processing architecture ($arch.$).

6.2.2 Design parameters (DP)

Design parameters are parameters that designers have freedom to choose and tune, which results in a design space that designers can explore. For visual servo

systems, design parameters from different domains are involved. For control engineers, typical design parameters are gains of the controller (K), sample period (h), delay (τ), and often the parameters of the plant (P) and measurement error (ϵ) as well. For imaging scientists, typical design parameters are image size (I_s) and vision processing algorithm ($alg.$). For electronic engineers, a typical design parameter is the processing architecture ($arch.$). If the choice of one design parameter does not impose constraints on the choice of other parameters, they are independent and can be included in the design parameter vector, that is,

$$DP = [h, \epsilon, \tau, K, I_s, alg., arch., P]^T. \tag{6.6}$$

If the visual feedback can be made sufficiently fast and the measurement error negligibly small, the design parameters of (6.6) can be reduced to

$$DP = [h, \tau, K, P]^T, \tag{6.7}$$

in which the sample period (h) and delay (τ) can be tuned for functional requirements (FR), for example, control performance and energy consumption. In such a case, the design of visual servo system is reduced to the design of a typical sample-data system with time delay.

However, the assumption behind (6.7) does not hold for a wide range visual servo systems. First, the sample period (h) and the delay (τ) of the vision systems are often limited by image size (I_s), vision algorithm ($alg.$), and the processing architecture ($arch.$), such that the sample period and delay are not sufficient for the required control performance. Second, it is known that the measurement error (ϵ) of visual feedback cannot be neglected, which depends on image size (I_s) and vision algorithm ($alg.$).

Therefore, for most visual servo systems, simplifying the design parameters from (6.6) to (6.7) leads to sub-optimal or unfeasible design. There are situations in which the image size (I_s), vision algorithm ($alg.$), and processing architecture ($arch.$) are fixed, and designers are therefore only able to tune the design parameters of (6.7). Otherwise, all the design parameters in (6.6) should be consider to achieve an optimal design.

6.2.3 FR and DP of a prototype visual servo system

As discussed in previous sections, the functional requirements (FR) of visual servo systems are application dependent, and the design parameters (DP) of visual servo systems are across multiple domains. This thesis simplifies the functional requirements, and focuses on cross-domain coupling of design parameters. The method derived in this chapter can be applied similarly to the analysis of other functional requirements.

As an example, root-mean-square error of tracking a velocity reference, denoted $RMS(\dot{e})$, the bandwidth of the system, denoted BW, and the *cost* of the sys-

tem, are chosen as functional requirements. If the design parameters of (6.6) are independent, the design equation can be described as

$$\begin{cases} RMS(\dot{e}) = f_1(h, \epsilon, \tau, K, I_s, alg., arch., P) \\ BW = f_2(h, \epsilon, \tau, K, I_s, alg., arch., P) \\ cost = f_3(h, \epsilon, \tau, K, I_s, alg., arch., P) \end{cases} \tag{6.8}$$

However, as explained in Chapter 6.2.2, the parameters of (6.8) are not independent. The sample period (h) and delay (τ) depend on the image size (I_s), vision algorithm ($alg.$), and processing architecture ($arch.$). The measurement error (ϵ) depends on image size (I_s) and vision algorithm ($alg.$). Therefore, $[I_s, alg., arch., P]^T$ are chosen as design parameters, and $[h, \epsilon, \tau, K]^T$ are chosen as functional requirements. For the prototype visual servo system implemented in Chapter 4 and modeled in Chapter 5, the resulted design equations are

$$\begin{cases} RMS(\dot{e}) = f_1(\epsilon, h, \tau, K) \\ BW = f_2(\epsilon, h, \tau, K) \\ cost = f_3(\epsilon, h, \tau, K) \end{cases} \tag{6.9}$$

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \\ P \end{bmatrix}. \tag{6.10}$$

Despite that the design equations (6.9) and (6.10) are specific for a prototype visual servo system, they incorporate the fundamental coupling of design parameters across multiple domains, and the method presented in this chapter can be applied to other design equations of different functional requirements and different coupling of design parameters as well. The remaining parts of this chapter explore the design space of visual servo system based on the aforementioned design equations, first for a single requirement, and then for multiple requirements.

6.3 Design trade-offs for a single requirement

This section focuses on the requirement of tracking error ($RMS(\dot{e})$) and combines (6.9) and (6.10) to explore the choices of design parameters with regard to $RMS(\dot{e})$. For different choices of the design parameters, the corresponding performance metric can be obtained using the model introduced in Chapter 5. The exploration of design trade-offs is performed in four steps, as illustrated in Figure 6.3, and explained subsequently.

First, several combinations of design parameters are chosen, including three image sizes (I_s), three vision algorithms ($alg.$), and three processing architectures

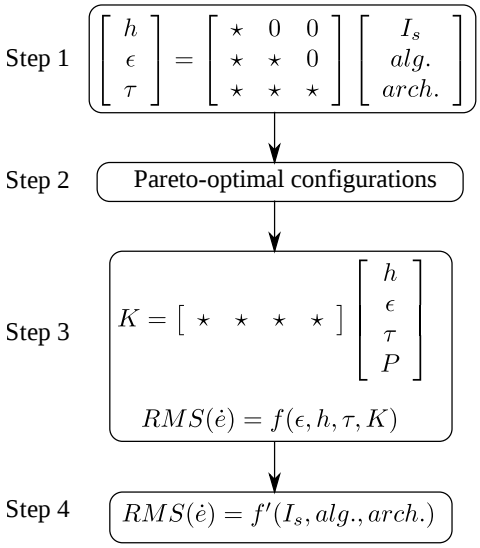


Figure 6.3: Four major steps in the exploration of design trade-offs. First, intermediate functional requirements of sample period (h), measurement error (ϵ), and delay (τ) are derived from design parameters of image size (I_s), vision algorithm ($alg.$) and processing architecture ($arch.$). Second, configurations of design parameters that lead to Pareto-optimal functional requirements are chosen for further exploration. Third, the optimal controller gain (K) and the corresponding optimal performance metric ($RMS(\dot{e})$) are derived for each Pareto-optimal configuration. Fourth, at each scale of sample rate, the corresponding optimal configurations of design parameters are identified.

($arch.$) that are designed for different scales of sample rate, as described in Table 6.1. From these design parameters, the intermediate functional requirements of sample period (h), measurement error (ϵ), and delay (τ) can be derived by the corresponding design equation (6.11), which is described in Chapter 5.

$$\begin{bmatrix} h \\ \epsilon \\ \tau \end{bmatrix} = \begin{bmatrix} \star & 0 & 0 \\ \star & \star & 0 \\ \star & \star & \star \end{bmatrix} \begin{bmatrix} I_s \\ alg. \\ arch. \end{bmatrix}. \tag{6.11}$$

The resulted sample periods of each configuration are listed in Table 6.1. The resulted measurement error and delay are plotted in Figure 6.4, from which the Pareto-optimal configurations of design parameters are labeled (Ⓐ - Ⓛ), as explained next.

Second, for each configuration of design parameters listed in Table 6.1, the corresponding delay (τ) and measurement error (peak-to-peak error, $P - P(\epsilon)$) are obtained from the model of Chapter 5, and are plotted in Figure 6.4. Some of the configurations are Pareto-optimal in the $h - \epsilon - \tau$ trade-offs, that is, these configurations have a higher sample rate, or a smaller error, or a smaller delay

Algorithms	Image size	Architecture for different sample rate					
		arch. for fps≈10^3		arch. for fps≈10^4		arch. for fps≈10^5	
		h $[ms]$	optimal?	h $[ms]$	optimal?	h $[ms]$	optimal?
Alg. 1	120 × 75	0.45	Yes - Ⓐ	0.075	Yes - Ⓔ	0.0075	Yes - Ⓘ
	160 × 100	0.6	Yes - Ⓑ	0.1	Yes - Ⓕ	0.01	Yes - Ⓙ
	200 × 125	0.75	No	0.125	No	0.0125	No
Alg. 2	120 × 75	0.45	No	0.075	No	0.0075	No
	160 × 100	0.6	No	0.1	No	0.01	No
	200 × 125	0.75	No	0.125	No	0.0125	No
Alg. 3	120 × 75	0.45	No	0.075	No	0.0075	No
	160 × 100	0.6	Yes - Ⓒ	0.1	Yes - Ⓖ	0.01	Yes - Ⓚ
	200 × 125	0.75	Yes - Ⓓ	0.125	Yes - Ⓗ	0.0125	Yes - Ⓛ

Table 6.1: Configurations of algorithms, image sizes, and architectures for different scales of samples rate. The resulted sample periods (h) of the configurations are listed in this table. The resulted measurement errors (ϵ) and delays (τ) are illustrated in Figure 6.4. At each scale of sample rate, the configurations that are Pareto-optimal in the $h - \epsilon - \tau$ trade-offs are labeled Ⓐ - Ⓛ in this table.

than all the other configurations. The Pareto-optimal configurations are labeled with Ⓐ - Ⓛ in Figure 6.4 and in Table 6.1. The design space is subsequently reduced to the Pareto-optimal configurations.

Third, for each Pareto-optimal configuration derived in the previous step, which is labeled (Ⓐ - Ⓛ) in Table 6.1, the optimal controller gain (K) and the corresponding optimal performance metric ($RMS(\dot{e})$) can be obtained by (6.12) and (6.13), which are described in Chapter 5.

$$K = \begin{bmatrix} \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} h \\ \epsilon \\ \tau \\ P \end{bmatrix}, \tag{6.12}$$

$$RMS(\dot{e}) = f(\epsilon, h, \tau, K). \tag{6.13}$$

The Pareto-optimal configurations have different sample periods (h), as described in Table 6.1. At each sample period (h), the control performance ($RMS(\dot{e})$) at different measurement errors (ϵ) and delay (τ) can be derived using the method described in Chapter 5, which are plotted in Figure 6.5. In the figure, the control performance at different sample period, measurement error, and delay are illustrated by equal-$RMS(\dot{e})$ lines. Along the equal-$RMS(\dot{e})$ lines, the control performance of Pareto-optimal configurations are annotated.

Fourth, the resulting $RMS(\dot{e})$ of the Pareto-optimal configurations are compared in Figure 6.6. For three processing architectures that lead to three different scales of sample rates (fps), the optimal choice of design parameters, that is, vision algorithms and image sizes, are identified.

With the aforementioned steps, the trade-offs in design parameters with regard to overall functional requirement, described by (6.14), are obtained and illustrated

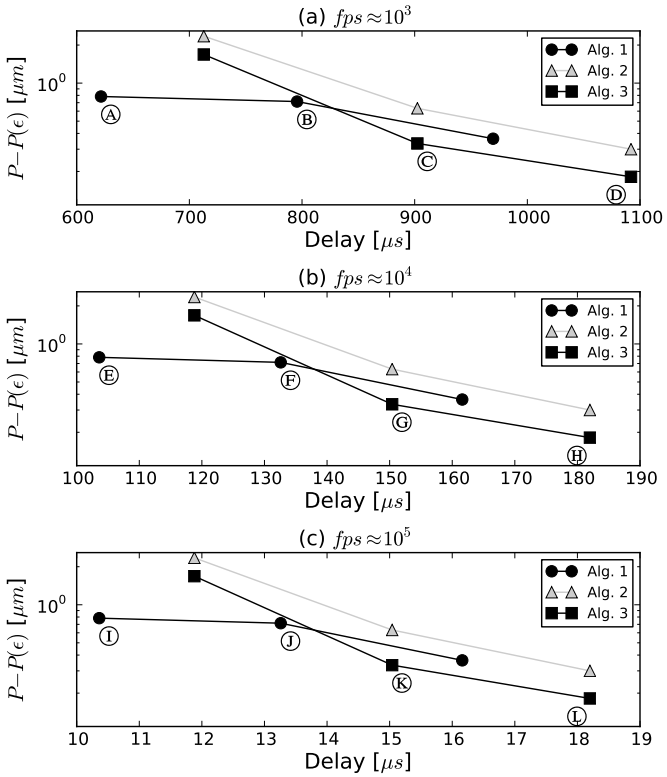


Figure 6.4: $h-\epsilon-\tau$ trade-offs of three vision algorithms (alg. 1, 2, 3) at different image sizes and implemented on processing architectures that can achieve different sample rates: (a) $fps \approx 10^3$; (b) $fps \approx 10^4$; (c) $fps \approx 10^5$. At each sample rate, different vision algorithms and image sizes are evaluated, as described in Table 6.1. The peak-to-peak error, P-P(ϵ), is used.

in Figure 6.6.

$$RMS(\dot{e}) = f'(I_s, alg., arch.). \tag{6.14}$$

Despite that only three design parameters are explored in this section, several non-trivial design trade-offs can be observed. First, the difference in performance between optimal configurations and sub-optimal configurations is significant, which is approximately 20% at low sample rate and larger than 30% at high sample rate, as shown in Figure 6.7. It makes a case for the potential benefit of performing design space exploration. Second, at each scale of sample rate, the optimal configuration is not necessarily the configuration of the least delay or the configuration of the least measurement error, as shown by mapping the optimal configurations Ⓒ Ⓗ Ⓙ in Figure 6.4. It confirms that there are significant trade-offs between sample period, delay, and measurement error. Third, the optimal image size and optimal vision algorithm are different at each scale of sample rate, as shown in

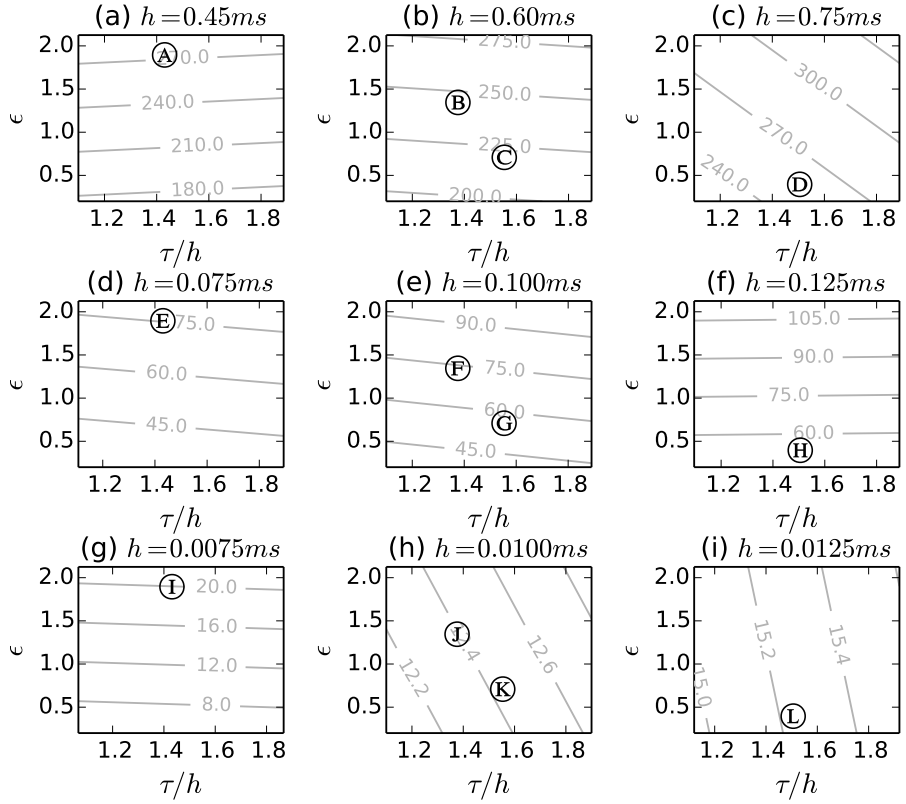


Figure 6.5: Equal-$RMS(\dot{e})$ lines, with the actual $RMS(\dot{e})$ number annotated on the line, at different sample periods (a)-(i). For each sample period, the equal-$RMS(\dot{e})$ lines resulted from different delay (τ/h) and measurement error (ϵ) are illustrated. The Pareto-optimal configurations Ⓐ - Ⓛ , described in Table 6.1 and illustrated in Figure 6.4, are annotated along the equal-$RMS(\dot{e})$ lines.

Figure 6.6. It implies that design parameters across multiple domains are tightly coupled, and need to be explored using a holistic approach.

In summary, this section provides the method to explore design parameters across multiple domains and applies it to the model of a prototype visual servo system. The result indicates that, by exploring the design trade-offs, a significant improvement of overall system performance can be obtained. It also confirms that cross-domain modeling and exploration is necessary for achieving the potential improvement of overall system performance. Yet this section focuses on one single performance metric only, that is, the tracking error of the system. The next section extends the analysis by exploring design trade-offs for multiple requirements, which is a scenario frequently encountered in practice.

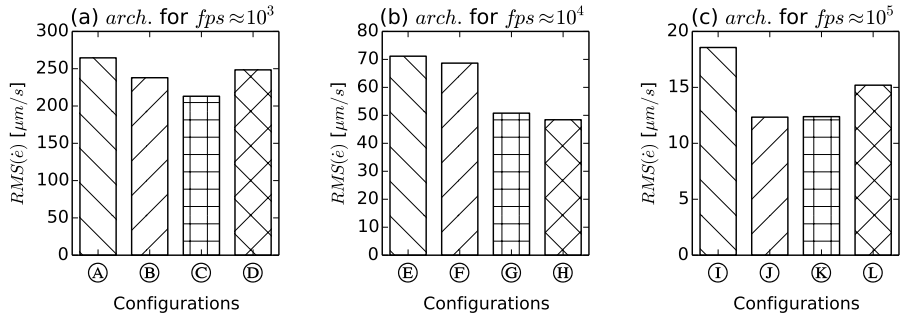


Figure 6.6: Comparison of control performance, in $RMS(\dot{e})$, for processing architectures that lead to different scales of sample rates: (a) $fps \approx 10^3$; (b) $fps \approx 10^4$; (c) $fps \approx 10^5$. The corresponding optimal configurations are: (a) config Ⓒ , $alg.$ 3 at image size of 160×100; (b) config Ⓗ , $alg.$ 3 at image size of 200×125; (c) config Ⓙ , $alg.$ 1 at image size of 160×100.

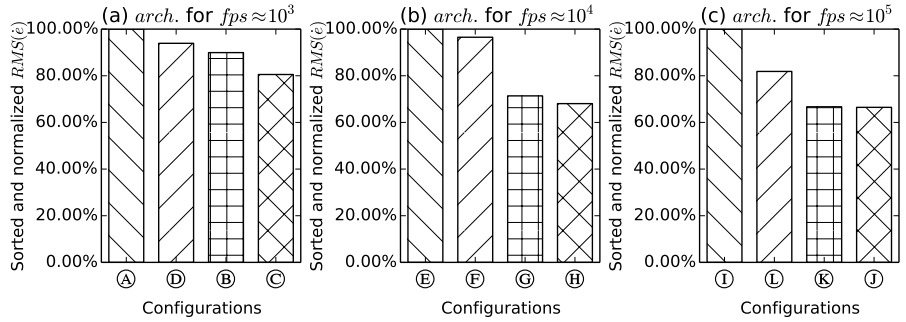


Figure 6.7: Similar to Figure 6.6, but control performance is sorted by tracking error ($RMS(\dot{e})$) and normalized at each scale of sample rate.

6.4 Design trade-offs for multiple requirements

This section explores the design trade-offs when multiple requirements are presented. For precision visual servo systems, bandwidth and tracking errors are two common requirements, as discussed in Chapter 6.2. Besides bandwidth and tracking error, the cost of the system is also important in practice, and therefore included as the third requirement. In the remaining parts of this section, the method of obtaining the bandwidth of the system is described, followed by the exploration of design trade-offs between tracking error, bandwidth, and cost of the system.

The bandwidth of a visual servo system is typically specified by the frequency of a reference signal that it can track within a certain error margin. To measure the bandwidth of the system, sinusoidal signals of different frequencies are used as references, with an example shown in Figure 6.8. The reference signal can be

described by

$$\dot{r} = A \cdot sin(2\pi ft) + b, \tag{6.15}$$

in which f is the frequency, A the amplitude, and b the bias. For the prototype visual servo system, parameters of the reference signal are chosen to represent its typical applications. The parameters of the reference signal are summarized in Table 6.2.

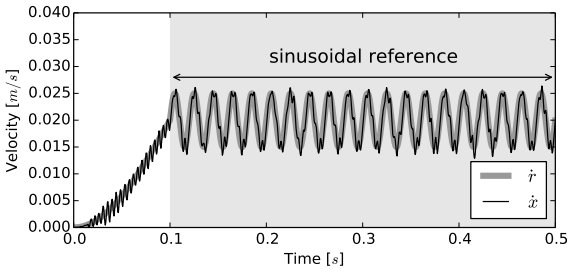


Figure 6.8: Example of a velocity reference ($\dot{r}$) that consists of sinusoidal waves and the corresponding velocity of the plant ($\dot{x}$).

A $[m/s]$	f $[Hz]$	b $[m/s]$
5×10^{-3}	10, 20, 50	20×10^{-3}

Table 6.2: Parameters of the reference signal (6.15).

Using sinusoidal reference signals of different frequencies, the corresponding tracking errors of different configurations can be obtained using the procedure described in Figure 6.9. The procedure is similar to that of Chapter 6.3, except that Step 3 and 4 are repeated for each reference frequency, because the optimal controller gain (K) is different for each reference frequency.

Following the steps described in Figure 6.9, the tracking errors at different reference frequencies are obtained, and illustrated in Figure 6.10. The result indicates that, at each scale of sample rate, the optimal configuration depends on the accuracy-bandwidth preference of the application, that is, applications emphasizing on accuracy may have a different optimal configuration than those emphasizing on bandwidth. It can also be observed that, for the accuracy-bandwidth (or $RMS(\dot{e}) - f$) performance metric, increasing the sample rate by an order of magnitude results in an improvement of $RMS(\dot{e}) - f$ by approximately an order of magnitude as well, as illustrated by the improvement of $RMS(\dot{e})$ from (a) to (c) in Figure 6.10. Without considering cost, using image sensors and processing architectures for higher sample rate leads to improved accuracy-bandwidth performance. However, cost is an important requirement in most engineering designs, and therefore cannot be ignored. Subsequently, the cost of the system is discussed next.

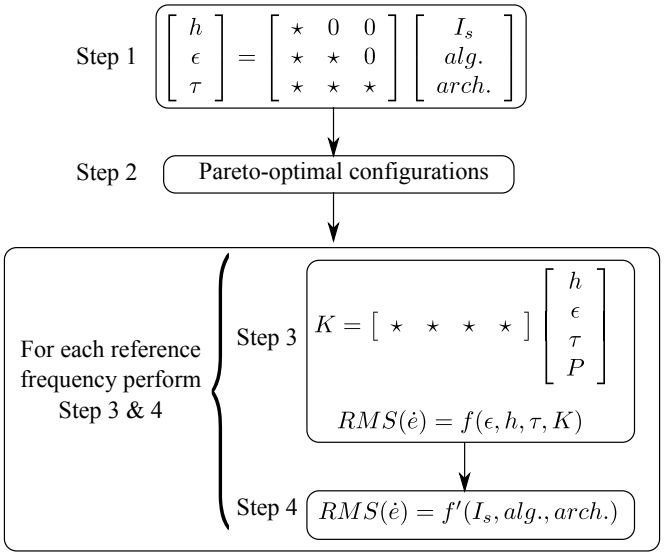


Figure 6.9: The procedure to obtain tracking error at different reference frequency. The four steps are the same as those described in Figure 6.3, except that Step 3 and 4 are repeated for each reference signal, because the optimal controller gain (K) is different for each reference frequency.

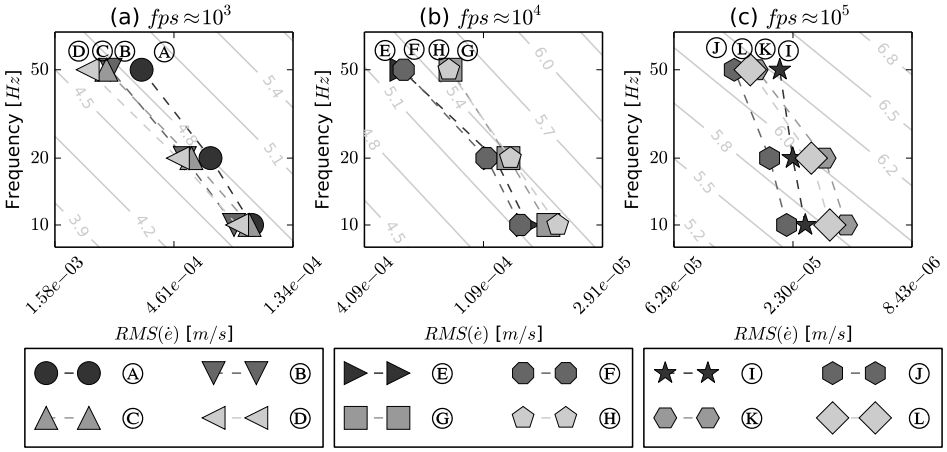


Figure 6.10: Tracking errors ($RMS(\dot{e})$) at different reference frequencies (f), in log-log scale, for configurations (Ⓐ - Ⓛ) at different scales of sample rates: (a) $fps \approx 10^3$; (b) $fps \approx 10^4$; (c) $fps \approx 10^5$. Along with the configuration points, equal-($f/RMS(\dot{e})$) lines are plotted, with the values of $log(f/RMS(\dot{e}))$ annotated at the lines.

This chapter performs cost analysis based on the configurations described in Chapter 3 for cameras (Table 3.1) and processing systems (Table 3.2) at different scales

Scale of sample rate (Hz)	Camera	Cost ($Euro$)
10^3	KAI-0340	1000
10^4	LUPA3000	5000
10^5	Phantom v2010	30000

Table 6.3: Estimated costs of cameras. The source of the costs is from private communication, and therefore the costs are coarsely rounded off. An elaborated version of this table is shown in Appendix A.

of sample rates. The cost of the cameras at each scale of sample rate is listed in Table 6.3. As discussed in Appendix A, the cost of the camera dominates that of the vision system. Therefore, for the sake of simplicity, this section uses the cost of camera system to approximate the cost of the total vision system, which is sufficient for relative comparisons.

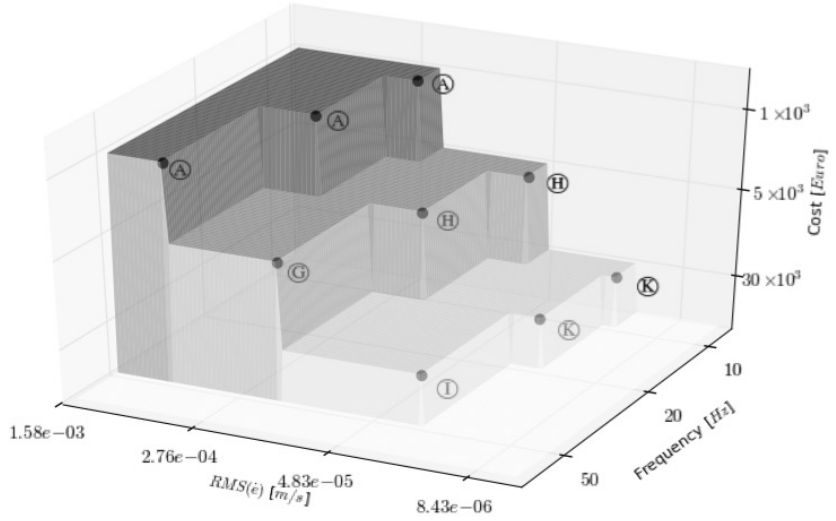


Figure 6.11: Pareto-optimal configurations for accuracy-bandwidth-cost trade-offs, among all configurations (Ⓐ - Ⓛ) listed in Table 6.1.

With cost added to the requirement, the Pareto-optimal configurations in the accuracy-bandwidth-cost (or $RMS(\dot{e}) - f - cost$) trade-offs are illustrated in Figure 6.11. It indicates the trend that, with a higher cost and therefore a higher sample rate and a lower delay, the accuracy-bandwidth performance metrics of the system improves accordingly. To further quantify the $RMS(\dot{e}) - f - cost$ trade-offs, the value of $f/RMS(\dot{e})/cost$ can be used as a metric to measure bandwidth and accuracy over cost, which is shown in Figure 6.12.

Two observations can be made from Figure 6.12 regarding the accuracy-bandwidth-cost trade-offs, measured in $f/RMS(\dot{e})/cost$. First, for applications that do not require a high bandwidth, increasing the sample rate of the system is not a cost-

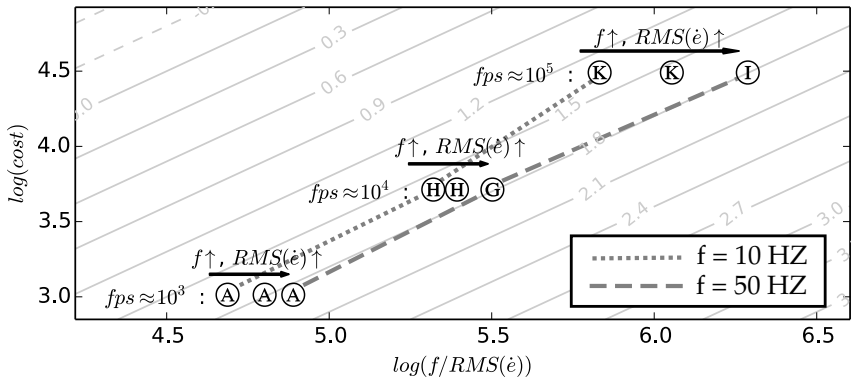


Figure 6.12: The Pareto-optimal configurations of Figure 6.11, along with equal-$(f/RMS(\dot{e})/cost)$ lines (solid lines). The values of $log(f/RMS(\dot{e})/cost)$ are annotated at the lines. For each scale of sample rate (fps), the trend of increasing bandwidth ($f \uparrow$) and increasing tracking error ($RMS(\dot{e}) \uparrow$) are annotated. The effects of increasing cost for tracking a 10 Hz reference and a 50 Hz reference are annotated by a dotted line and a dashed line, respectively.

effective way of improving the accuracy of the system. For example, if the requirement is to track a reference frequency of 10 Hz, increasing the sample rate decreases the value of $log(f/RMS(\dot{e})/cost)$ from 1.6 to 1.3 (illustrated by the dotted line). Second, for applications that require a high bandwidth, increasing the sample rate of the system is a cost-effective way of improving the accuracy of the system. For example, if the requirement is to track a reference frequency of 50 Hz, increasing the sample rate has little impact on the value of $log(f/RMS(\dot{e})/cost)$ (illustrated by the dashed line).

While the aforementioned observations are made from a specific example of visual servo system, several generic conclusions can be drawn from the observations. First, even with a limited number of design parameters and requirements, there are non-trivial design trade-offs, which requires explorations of design space to obtain optimal configurations. Second, depending on the requirements of the applications, increasing sample rate and processing capacity can be cost-effective to improve performance in one case, but not cost-effective in another case. In both cases, a more cost-effective way of scaling performance can be further explored, by including additional design parameters and therefore additional design alternatives. The method proposed in this chapter can be applied similarly to a larger set of design parameters, requirements, and systems.

6.5 Summary

This chapter investigates the design trade-offs of precision visual servo systems. One of the major challenges is the cross-domain coupling of visual servo systems. Besides that, multiple, and often conflicting, requirements are present in such systems. A common scenario in practice is to explore achievable accuracy and bandwidth under certain cost constraints, and to derive the corresponding design parameters.

This chapter presents a tool to overcome the aforementioned challenges, and applies it to a prototype visual servo system as an example. Despite that only a subset of possible design parameters and requirements are included in the example, they reflect the fundamental cross-domain coupling of precision visual servo systems and, as shown in this chapter, lead to non-trivial design trade-offs. The same method can be applied to a larger set of design parameters, requirements, and systems similarly.

The result of the design space exploration confirms a tight coupling of design parameters across multiple domains, including vision algorithms, processing architecture, and controller design. It indicates that, without cross-domain modeling and exploration, sub-optimal design will be resulted, which can be significantly inferior to optimal designs. It makes a case for the necessity of cross-domain modeling and exploration for precision visual servo systems.

The proposed method can be applied to multiple requirements, and derive Pareto-optimal configurations accordingly. This chapter uses accuracy, bandwidth, and cost as examples, which are typical requirements of precision mechatronic systems. It is shown that, depending on the requirements of a certain application, the corresponding optimal configuration changes, and a configuration optimized for one set of requirements could be far from optimal for another set of requirements. The result indicates the importance of design space exploration for multiple requirements, and the effectiveness of the proposed method for such a purpose.

Chapter 7

Conclusions and recommendations

It is better to have an approximate answer to the right question than an exact answer to the wrong one.

John Tukey

Abstract. *This chapter revisits the research problems and the corresponding solutions provided by each part of this thesis. Key findings, implications, and contributions are summarized. Limitations of this thesis are subsequently discussed, followed by recommendations for future work.*

7.1 Conclusions

As visual feedback provides unique benefits for mechatronics and robotics systems, major challenges of integrating visual feedback into these systems are identified and tackled in this thesis. These challenges include, first, implementing high-speed vision systems for control purposes; second, designing and optimizing control laws for visual feedback; third, having an early evaluation on delays of different vision

algorithms on customized processing systems; fourth, modeling cross-domain couplings within visual servo systems and exploring cross-domain design trade-offs. Tackling these research challenges leads to a number of contributions and findings, detailed as follows.

Identification of research challenges and open issues

In Chapter 1, four areas are identified that are major challenges of integrating visual feedback into mechatronics and robotics systems. Subsequently, in Chapter 2, backgrounds on each of these four areas are provided. Despite recent advances, there are open issues in each area when combining the methods of these areas into the design and implementation of visual servo systems. Open issues are identified by reviewing related work in each area and putting them in the context of visual servo systems. By comparing related work to this work, contributions of this thesis are clarified.

Design method of high-speed vision systems for control purposes

Chapter 3 tackles the challenge of designing vision systems for control purposes, that is, "*a relatively low sample rate, significant latency (one or more sample intervals) and coarse quantization*" (Corke and Good, 1996) of a typical vision system. Using a high-level method to evaluate available technologies, guidelines are provided on the choice of appropriate processing platforms for achieving different scales of sample rate, ranging from the scale of 100 frames-per-second to the scale of 10,000 frames-per-second. Design methods of restricting delays to no more than two sample periods are proposed. The quantization effect of vision systems is often application-dependent, which is generally hard to analyze. However, a method of simulating and modeling the quantization effects of vision systems is proposed, and used as a guideline for the design of algorithms in a case study. The case study uses the proposed methods to implemented a 1000 frames-per-second vision system with a delay of less than two sample periods, and with a sub-micrometer scale accuracy. The case study demonstrates the feasibility and effectiveness of the proposed design methods.

Controller design and optimization for visual feedback

In Chapter 4, a vision-based trajectory generator is proposed to utilize a major benefit of visual feedback, that is, online and direct measurements of objects of interest. The method is applied to a case study which uses high-speed visual feedback to position an actuator on non-uniform micro-structures. It is experimentally demonstrated that visual feedback leads to a better control performance compared to traditional motor encoders, and more importantly, enables online compensation for geometric non-uniformity in the objects of interest, which is not possible using conventional sensing techniques.

In Chapter 5, a method of controller optimization for visual feedback is proposed that takes into account three major limitations of visual feedback, that is, sample rate, delay, and quantization error. The method first applies analytical techniques on a simplified model of the system to constrain a search space of the controller gain, followed by simulations that are based on an elaborated model of the system for optimization of the control gain. The method is demonstrated in a case study and is able to perform controller optimization over a wide range of design parameters with regard to different performance requirements.

Method of evaluating algorithmic and architectural choices

In Chapter 5, algorithmic patterns and architectural templates that are suitable for high-level synthesis are proposed, which enable rapid generation and early evaluation of customized vision processing systems. The proposed method is applied to a case study by synthesizing several vision algorithms and processing architectures that have different accuracy and delay trade-offs. Subsequently, in Chapter 6, different vision systems are synthesized over a wide range of sample rates, depending on image sensors of choice, scaling from 1000 frames-per-second to 100,000 frames-per-second. It is demonstrated that the optimal choices of vision algorithms and processing architectures vary for different requirements. Without evaluating different algorithmic and architectural choices, a sub-optimal system will be realized as a result. The result confirms the necessity and effectiveness of the proposed method.

Modeling cross-domain couplings and exploring design trade-offs

A method of modeling cross-domain coupling is proposed in Chapter 5, which is subsequently used to explore design trade-offs across multiple domains in Chapter 6. The axiomatic design method is used to explicitly model the cross-domain coupling in visual servo systems and to eliminate unnecessary coupling such that the complexity of cross-domain exploration can be reduced.

In Chapter 5, a method of constructing a holistic model of visual servo systems guided by the axiomatic design principle is proposed. The proposed method takes as inputs design parameters across multiple domains including, but not limited to, image sensors, vision algorithms, processing architectures, and plant dynamics. In combination with other design methods of this thesis, it derives the sample rate, delay, and quantization error of visual feedback, which are subsequently used to obtain an optimized control gain and its associated control performance of the system.

In Chapter 6, a method is proposed to explore a wide range of design parameters across multiple domains of visual servo systems with regard to multiple conflicting requirements. The method is applied to a case study that explores the achievable accuracy and system bandwidth of a visual servo system under certain cost con-

straints, and to derive corresponding design parameters. The case study reveals that, without cross-domain exploration, significantly inferior performance will be derived. In addition, the optimal design parameters for one set of requirements could be far from optimal for another set of requirements. Therefore, it is practically necessary to perform cross-domain exploration for multiple requirements, and the proposed method provides an effective tool for such a purpose.

7.2 Limitations and Recommendations

Despite that multiple contributions are made, this work is not without limitations. The limitations and the reasons behind them are discussed next. In addition, new research challenges arise as a result, which are recommended for future research.

7.2.1 Limitations

The proposed methods are intended to be generic for a wide range of use cases and to be applicable for practitioners. While these aspects are addressed to a large extent, there are nevertheless limitations. There are three major limitations worth discussing in details. The first two limitations relate to how generic the proposed methods are, while the third limitation regards how much effort is required to apply the proposed methods in practice.

First, the design methods proposed in this thesis have been applied to and validated on a single, albeit representative, case study. Additional use cases are intentionally left out from the scope of this thesis. They are intentionally left out because, while additional use cases provide added values, the case study of this thesis is self-contained and sufficient in demonstrating the effectiveness of the proposed methods. Moreover, the proposed methods are applicable to a wide range of precision visual servo systems, because the proposed methods are based on combinations and extensions of existing single-domain methods that are applicable to a wide range of use cases. That is, the proposed methods are as generic as the underlying single-domain methods of choice. Therefore, the limited number of case study presented in this thesis shall be considered as a limitation on the scope of this thesis, not as a limitation on the applicability of the proposed methods.

Second, the cross-domain modeling method proposed in this thesis is based on standard and basic methods used in each domain. The vision algorithms, processing architectures, and control laws used in this thesis are baseline examples in each of these domains, where more advanced techniques exist. The focus of this thesis is on cross-domain coupling, which has been demonstrated to be nontrivial even if baseline examples from each domain are considered. In addition, the proposed cross-domain modeling method does not prevent designers from us-

ing more advanced methods from each domain. With more advanced techniques and additional design parameters, the design matrix increases in dimensions but the fundamental coupling patterns in the design matrix can be shaped similarly, as discussed in Chapter 5 and in the Appendix. The design effort does increase with advanced methods, yet the proposed cross-domain modeling method remains applicable and effective.

Third, applying the proposed methods of cross-domain modeling and exploration to a use case demands considerable efforts. For each domain in which design choices need to be explored, a corresponding model, either analytical or simulation-based, needs to be constructed. In addition, if the number of design parameters to be explored within a domain increases, the complexity of the corresponding single-domain model increases accordingly. On the other hand, the aforementioned limitations are inherent in model-based design methods. The focus of the proposed methods is not on alleviating the design effort inherent in constructing single-domain models, but on an effective way to couple single-domain models into a holistic model that supports exploration of cross-domain trade-offs. For such a purpose, if single-domain models are available, the additional effort required by the proposed methods is insignificant. In comparison to the additional effort which is insignificant, the achievable gain in overall performance is demonstrated to be substantial.

7.2.2 Recommendations

Since the aforementioned limitations are not addressed by this thesis, they give rise to research problems worth investigating in future research. In addition, the proposed methods can be further improved on multiple aspects. These potential improvements are subsequently discussed, while the solutions of them are left for future research.

Generalization of the proposed methods

As discussed in the limitations of this thesis, the proposed methods are yet to be demonstrated on a wider range of visual servo systems and on use cases that involve more advanced techniques in each domain. Despite that the proposed methods are based on previous researches which are known to be generic, and, in addition, the combinations and extensions of previous researches are described in generic terms, it cannot be excluded that there are use cases which require a significant modification of the proposed methods in order to make them applicable. For example, in Chapter 5, the delay of the system is deterministic by design, which can be effectively enforced on customized visual servo systems in semi-structural environments of today. However, as technologies evolve, enforcing deterministic delay in vision algorithms, processing systems, and communication methods will come at an increased cost of performance over their counterparts

with varying delay. It remains a challenge in itself to optimize control systems with varying delay. It is an even greater challenge to model the variation of delay based on multiple cross-domain design parameters. Besides the example of deterministic delays, other assumptions of the proposed methods could be potentially challenged by new use cases. These challenges open new doors of opportunity for future research.

Impacts on performance from decoupling of design parameters

Design methodologies often impose constraints on design freedom, which potentially lead to sub-optimal designs. The design methods proposed in this thesis are no exception. More specifically, the proposed methods describe cross-domain coupling using the axiomatic design method and attempt to shape the design matrix into a triangular matrix thus resulting in a decoupled system. By eliminating cross-domain coupling, the complexity of evaluating design parameters is reduced. However, in this process, certain design choices are eliminated, which could otherwise lead to a superior performance. For example, in Chapter 3, a design method is proposed to decouple the sample rate from vision algorithms and processing architectures. The overhead of imposing this design rule, such as additional delays induced by data buffering, may lead to a slightly worse overall performance. The proposed method of eliminating cross-domain decoupling, albeit effective for high-speed visual servo systems, is not necessarily optimal. It is left for future research to quantify the impacts on the achievable system performance from decoupling of design parameters, and systematically balance both aspects.

Methods for selecting design parameters

In a complex system that involves multiple tightly coupled domains, such as visual servo systems, the number of design parameters that might impact a functional requirement reaches the scale of hundreds and beyond. The proposed methods are able to take into account a large number of design parameters, but the corresponding efforts to construct a holistic model, either analytical or simulation-based, increase as a result. Despite that there exist methods of constructing detailed single-domain models, and the proposed methods attempt to decouple these domains when possible, there are functional requirements that require a combinatorial optimization of design parameters across multiple domains, otherwise designs that are infeasible or significantly sub-optimal will be resulted. In this thesis, design parameters that are empirically known to have a large impact on the functional requirements are chosen. For example, in Chapter 5, it is shown that the delay of the system is tightly coupled with the design parameters of the image sensors, vision algorithms, and processing architectures. There are multiple parameters of image sensors that can impact the delay, yet only the image size is chosen as a design parameter because it is empirically known to have a large

impact on the delay. It remains a research challenge to have a generic method of selecting a set of design parameters that are most relevant to the requirements, and applying it to a cross-domain scenario.

7.3 Summary

Visual feedback in mechatronic and robotic systems provides unique benefits, but is a major challenge. It is a major challenge not only due to fast developments in the domains of high speed vision systems, digital hardware, and feedback control laws, but mostly due to cross-domain couplings and possible trade-offs in the design of visual servo systems. In conclusion, one can say that despite the fact that a formal analysis of either a design of an existing system or a system on the drawing board is not an easy task, it provides way more insights in the possibilities and performance of the system and its variants than an ad-hoc design. That is a benefit that eventually will pay itself back. For such a purpose, this work provides a tool to analyze and optimize existing systems, to set up a product family of systems with various trade-offs, and to explore a road map for future generations of systems.

Bibliography

B. F. Alexander and K. C. Ng. Elimination of systematic error in subpixel accuracy centroid estimation [also letter 34(11)3347-3348(nov1995)]. *Optical Engineering*, 30(9):1320–1331, 1991.

F. Altpeter. *Friction modeling, identification and compensation*. PhD thesis, STI, Lausanne, 1999.

S. Asaad, M. Bishay, D. M. Wilkes, and K. Kawamura. A low-cost, dsp-based, intelligent vision system for robotic applications. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1656–1661 vol.2, Apr 1996.

A. Assa and F. Janabi-Sharifi. Closed-loop uncertainty modeling for visual servoing. In *2013 IEEE International Conference on Robotics and Automation*, pages 3089–3094, May 2013.

A. Banerjee and S. Gupta. Research in automated planning and control for micromanipulation. *Automation Science and Engineering, IEEE Transactions on*, 10(3):485–495, July 2013. ISSN 1545-5955.

Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke. Training deep neural networks for visual servoing. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2018.

S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433–466, Sept. 1995. ISSN 0360-0300.

K. Benkrid and D. Crookes. From application descriptions to hardware in seconds: a logic-based approach to bridging the gap. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(4):420–436, April 2004. ISSN 1063-8210.

M. J. Benner and M. L. Tushman. Exploitation, exploration, and process management: The productivity dilemma revisited. *The Academy of Management Review*, 28(2):pp. 238–256, 2003. ISSN 03637425.

H. Bruyninckx, M. Klotzbücher, N. Hochgeschwender, G. Kraetzschmar, L. Gherardi, and D. Brugali. The brics component model: A model-based development paradigm for complex robotics software systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC 13, page 17581764, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450316569. doi: 10.1145/2480362.2480693. URL `https://doi.org/10.1145/2480362.2480693`.

W. Caarls. *Automated design of application-specific smart camera architectures*. PhD thesis, Delft University of Technology, 2008.

A. Cassidy, K. Yu, H. Zhou, and A. Andreou. A high-level analytical model for application specific cmp design exploration. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, 2011.

F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *Robotics Automation Magazine, IEEE*, 13(4):82–90, 2006.

Y. Chen, T. Ogata, T. Ueyama, T. Takada, and J. Ota. Automated design of the field-of-view, illumination, and image pre-processing parameters of an image recognition system. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1079–1084, Aug 2017.

G. Chesi and H. L. Yung. Performance limitation analysis in visual servo systems: Bounding the location error introduced by image points matching. In *2009 IEEE International Conference on Robotics and Automation*, pages 695–700, May 2009.

G. Clayton and B. McManus. Two axis image-based measurement and control for scanning probe microscopes. In *Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on*, pages 640–645, July 2011.

M. B. G. Cloosterman, N. van de Wouw, W. P. M. H. Heemels, and H. Nijmeijer. Stability of networked control systems with uncertain time-varying delays. *Automatic Control, IEEE Transactions on*, 54(7):1575–1580, July 2009. ISSN 0018-9286.

B. Cope, P. Y. K. Cheung, W. Luk, and L. Howes. Performance comparison of graphics processors to reconfigurable logic: A case study. *Computers, IEEE Transactions on*, 59(4):433–448, 2010. ISSN 0018-9340.

P. Corke and M. Good. Dynamic effects in visual closed-loop systems. *Robotics and Automation, IEEE Transactions on*, 12(5):671–683, Oct 1996. ISSN 1042-296X.

R. Dahmouche, N. Andreff, Y. Mezouar, O. Ait-Aider, and P. Martinet. Dynamic visual servoing from sequential regions of interest acquisition. *The International Journal of Robotics Research*, 31(4):520–537, 2012.

J. de Best. *Feature-Based Motion Control for Near-Repetitive Structures*. PhD thesis, Eindhoven University of Technology, 2011.

J. de Best, R. van de Molengraft, and M. Steinbuch. High speed visual motion control applied to products with repetitive structures. *IEEE Trans. on Control Systems Technology*, 20(6):1450–1460, 2012.

D. F. Delchamps. Stabilizing a linear system with quantized state feedback. *Automatic Control, IEEE Transactions on*, 35(8):916–924, Aug 1990. ISSN 0018-9286.

S. Devasia, E. Eleftheriou, and S. Moheimani. A survey of control issues in nanopositioning. *Control Systems Technology, IEEE Transactions on*, 15(5): 802–823, Sept 2007. ISSN 1063-6536.

S. Fernando, M. Wijtvliet, C. Nugteren, A. Kumar, and H. Corporaal. (as)2: Accelerator synthesis using algorithmic skeletons for rapid design space exploration. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 305–308, March 2015.

J. Fowers, G. Brown, P. Cooke, and G. Stitt. A performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '12, pages 47–56, New York, NY, USA, 2012. ACM.

G. F. Franklin, M. L. Workman, and D. Powell. *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997. ISBN 0201820544.

H. Fujimoto and Y. Hori. Visual servoing based on intersample disturbance rejection by multirate sampling control-time delay compensation and experimental verification. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 334–339 vol.1, 2001.

N. Gans, P. Corke, and S. Hutchinson. Performance tests of partitioned approaches to visual servo control. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1616–1623 vol.2, 2002.

W. Gao, S. Kim, H. Bosse, H. Haitjema, Y. Chen, X. Lu, W. Knapp, A. Weckenmann, W. Estler, and H. Kunzmann. Measurement technologies for precision positioning. {*CIRP*} *Annals - Manufacturing Technology*, 64(2):773 – 796, 2015. ISSN 0007-8506.

E. Garcia, P. Gonzalez de Santos, and C. Canudas de Wit. Velocity dependence in the cyclic friction arising with gears. *The International Journal of Robotics Research*, 21(9):761–771, 2002.

R. Ginhoux, J. Gangloff, M. de Mathelin, L. Soler, M. Sanchez, and J. Marescaux. Beating heart tracking in robotic surgery using 500 hz visual servoing, model predictive control and an adaptive observer. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 274–279, 2004.

K. Goldberg. What is automation? *Automation Science and Engineering, IEEE Transactions on*, 9(1):1–2, 2012.

C. Graetzel, S. Fry, and B. Nelson. A 6000 hz computer vision system for real-time wing beat analysis of drosophila. In *Proc. of IEEE/RAS-EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob)*, pages 278–283, 2006.

Q.-Y. Gu and I. Ishii. Review of some advances and applications in real-time high-speed vision: Our views and experiences. *International Journal of Automation and Computing*, 13(4):305–318, Aug 2016. ISSN 1751-8520.

K. Hashimoto and T. Noritsugu. Performance and sensitivity in visual servoing. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 3, pages 2321–2326 vol.3, May 1998.

S. P. Haveman and G. M. Bonnema. Requirements for high level models supporting design space exploration in model-based systems engineering. *Procedia Computer Science*, 16(0):293 – 302, 2013. ISSN 1877-0509.

Y. He, Z. Ye, D. She, B. Mesman, and H. Corporaal. Feasibility analysis of ultra high frame rate visual servoing on fpga and simd processor. In J. Blanc-Talon, R. Kleihorst, W. Philips, D. Popescu, and P. Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, pages 623–634, Berlin, Heidelberg, 2011a. Springer Berlin Heidelberg. ISBN 978-3-642-23687-7.

Y. He, Z. Ye, D. She, B. Mesman, and H. Corporaal. Feasibility analysis of ultra high frame rate visual servoing on fpga and simd processor. In *Advances Concepts for Intelligent Vision Systems*, pages 623–634. Springer, 2011b.

M. Heemels and G. Muller. Boderc: Model-based design of high-tech systems. *Embedded Systems Institute, Eindhoven*, 2006.

P. Hehenberger, F. Poltschak, K. Zeman, and W. Amrhein. Hierarchical design models in the mechatronic product development process of synchronous machines. *Mechatronics*, 20(8):864 – 875, 2010. ISSN 0957-4158. ¡ce:title¿Special Issue on Theories and Methodologies for Mechatronics Design¡/ce:title¿.

B. Holland, A. D. George, H. Lam, and M. C. Smith. An analytical model for multilevel performance prediction of multi-fpga systems. *ACM Trans. Reconfigurable Technol. Syst.*, 4(3):27:1–27:28, Aug. 2011.

S. Hong and H. Kim. An analytical model for a gpu architecture with memory-level and thread-level parallelism awareness. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 152–163, New York, NY, USA, 2009. ACM.

B. Hussain and M. Kabuka. Real-time system for accurate three-dimensional position determination and verification. *Robotics and Automation, IEEE Transactions on*, 6(1):31–43, Feb 1990. ISSN 1042-296X.

S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on*, 12(5):651–670, 1996.

F. Janabi-Sharifi, L. Deng, and W. J. Wilson. Comparison of basic visual servoing methods. *Mechatronics, IEEE/ASME Transactions on*, 16(5):967–983, Oct 2011. ISSN 1083-4435.

W. S. Junk. The dynamic balance between cost, schedule, features, and quality in software development projects. *Computer Science Department, University of Idaho, SEPM-001*, 2000.

A. Kawamura, K. Tahara, R. Kurazume, and T. Hasegawa. Robust visual servoing for object manipulation with large time-delays of visual information. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4797–4803, Oct 2012.

M. B. Khamesee, M. Rapolti, and A. Amirfazli. Controller design and optimization for large-delays image processing in visual closed-loop systems. *Mechatronics*, 18(5):251 – 261, 2008. ISSN 0957-4158. Special Section on Optimized System Performances Through Balanced Control Strategies.

H. Kim and S. Arajo. Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast. In D. Mery and L. Rueda, editors, *Advances in Image and Video Technology*, volume 4872 of *Lecture Notes in Computer Science*, pages 100–113. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77128-9.

T. Komuro, A. Iwashita, and M. Ishikawa. A QVGA-size pixel-parallel image processor for 1000-fps vision. *IEEE Micro*, 29(6):58–67, 2009. ISSN 0272-1732.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

V. Kyrki, D. Kragic, and H. Christensen. Measurement errors in visual servoing. *Robotics and Autonomous Systems*, 54(10):815 – 827, 2006. ISSN 0921-8890.

H. Liu, T.-H. Hong, M. Herman, T. Camus, and R. Chellappa. Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding*, 72(3):271 – 286, 1998. ISSN 1077-3142.

C. Lu, M. J. G. van de Molengraft, and G. Dubbelman. Monocular semantic occupancy grid mapping with convolutional variational encoderdecoder networks. *IEEE Robotics and Automation Letters*, 4(2):445–452, April 2019. ISSN 2377-3766.

D. Lustig and M. Martonosi. Reducing gpu offload latency via fine-grained cpu-gpu synchronization. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pages 354–365, 2013.

J. McCalpin. Optimum system balance for systems of finite price. In *Supercomputing, Panel on HPC Challenge Benchmarks*, 2004.

MCorelab. Standards-based solutions for ultra-low latency, high-throughput computing, 2012.

R. Medina, S. Stuijk, D. Goswami, and T. Basten. Exploring the trade-off between processing resources and settling time in image-based control through lqr tuning. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 1456–1459, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4486-9.

K. Mizuno, H. Noguchi, G. He, Y. Terachi, T. Kamino, H. Kawaguchi, and M. Yoshimoto. Fast and low-memory-bandwidth architecture of sift descriptor generation with scalability on speed and accuracy for vga video. In *2010 International Conference on Field Programmable Logic and Applications*, pages 608–611, Aug 2010.

S. Mohamed, D. Zhu, D. Goswami, and T. Basten. Optimising quality-of-control for data-intensive multiprocessor image-based control systems considering workload variations. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 320–327, Aug 2018.

J. More. The levenberg-marquardt algorithm: Implementation and theory. In G. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 105–116. Springer Berlin Heidelberg, 1978.

Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno. 1 ms column parallel vision system and its application of high speed target tracking. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 650–655, 2000.

R. Nane, V. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels. A survey and evaluation of fpga high-level synthesis tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(10):1591–1604, Oct 2016. ISSN 0278-0070.

H. Nasibov, A. Kholmatov, B. Akselli, A. Nasibov, and S. Baytaroglu. Performance analysis of the ccd pixel binning option in particle-image velocimetry measurements. *Mechatronics, IEEE/ASME Transactions on*, 15(4):527–540, Aug 2010. ISSN 1083-4435.

H. Nobach, N. Damaschke, and C. Tropea. High-precision sub-pixel interpolation in particle image velocimetry image processing. *Experiments in Fluids*, 39(2): 299–304, 2005. ISSN 0723-4864.

C. Nugteren, P. Custers, and H. Corporaal. Algorithmic species: A classification of affine loop nests for parallel programming. *ACM Trans. Archit. Code Optim.*, 9(4):40:1–40:25, Jan. 2013. ISSN 1544-3566.

N. Ogawa, H. Oku, K. Hashimoto, and M. Ishikawa. Microrobotic visual control of motile cells using high-speed tracking system. *IEEE Trans. on Robotics*, 21 (4):704–712, 2005. ISSN 1552-3098.

K. Okumura, H. Oku, and M. Ishikawa. High-speed gaze controller for millisecond-order pan/tilt camera. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 6186–6191, May 2011.

D. A. Patterson. Latency lags bandwith. *Commun. ACM*, 47(10):71–75, Oct. 2004. ISSN 0001-0782.

K. Pauwels, M. Tomasi, J. Diaz Alonso, E. Ros, and M. Van Hulle. A comparison of fpga and gpu for real-time phase-based optical flow, stereo, and local image features. *Computers, IEEE Transactions on*, 61(7):999–1012, 2012. ISSN 0018-9340.

R. Pieters, Z. Ye, P. Jonker, and H. Nijmeijer. Direct motion planning for vision-based control. *IEEE Transactions on Automation Science and Engineering*, 11 (4):1282–1288, Oct 2014. ISSN 1545-5955.

S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.*, 51(5):92:1–92:36, Sept. 2018. ISSN 0360-0300. doi: 10.1145/3234150.

Y. Pu, Y. He, Z. Ye, S. M. Londono, A. A. Abbo, R. Kleihorst, and H. Corporaal. From xetal-ii to xetal-pro: On the road toward an ultralow-energy and high-throughput simd processor. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(4):472–484, April 2011. ISSN 1051-8215.

R. Ramesh, M. Mannan, and A. Poo. Error compensation in machine tools a review: Part i: geometric, cutting-force induced and fixture-dependent errors. *International Journal of Machine Tools and Manufacture*, 40(9):1235 – 1256, 2000a. ISSN 0890-6955.

R. Ramesh, M. Mannan, and A. Poo. Error compensation in machine tools a review: Part ii: thermal errors. *International Journal of Machine Tools and Manufacture*, 40(9):1257 – 1284, 2000b. ISSN 0890-6955.

B. Reddy and B. N. Chatterji. An fft-based technique for translation, rotation, and scale-invariant image registration. *Image Processing, IEEE Transactions on*, 5(8):1266–1271, Aug 1996. ISSN 1057-7149.

O. Reiche, K. Hublein, M. Reichenbach, M. Schmid, F. Hannig, J. Teich, and D. Fey. Synthesis and optimization of image processing accelerators using domain knowledge. *Journal of Systems Architecture*, 61(10):646 – 658, 2015. ISSN 1383-7621.

M. D. Santo, C. Liguori, and A. Pietrosanto. Uncertainty characterization in image-based measurements: a preliminary discussion. *IEEE Transactions on*

Instrumentation and Measurement, 49(5):1101–1107, Oct 2000. ISSN 0018-9456.

M. Schmid, N. Apelt, F. Hannig, and J. Teich. An image processing library for c-based high-level synthesis. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, Sep. 2014.

M. Shimizu and M. Okutomi. Sub-pixel estimation error cancellation on area-based matching. *International Journal of Computer Vision*, 63(3):207–224, 2005. ISSN 0920-5691.

S. T. Smith and R. M. Seugling. Sensor and actuator considerations for precision, small machines. *Precision Engineering*, 30(3):245 – 264, 2006. ISSN 0141-6359.

N. P. Suh. *The principles of design*, volume 990. Oxford University Press New York, 1990.

N. P. Suh. Axiomatic design theory for systems. *Research in Engineering Design*, 10(4):189–209, 1998. ISSN 0934-9839.

N. P. Suh et al. *Axiomatic design: advances and applications*, volume 4. Oxford university press New York, 2001.

V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, Dec 2017. ISSN 0018-9219. doi: 10.1109/JPROC.2017.2761740.

T. Tabata, T. Komuro, and M. Ishikawa. Surface image synthesis of moving spinning cans using a 1,000-fps area scan camera. *Machine Vision and Applications*, 21(5):643–652, 2010. ISSN 0932-8092.

R. van Herpen, T. Oomen, E. Kikken, M. van de Wal, W. Aangenent, and M. Steinbuch. Exploiting additional actuators and sensors for nano-positioning robust motion control. *Mechatronics*, 24(6):619 – 631, 2014. ISSN 0957-4158. Control of High-Precision Motion Systems.

E. P. van Horssen, D. Antunes, and W. P. M. H. Heemels. Switching data-processing methods for feedback control: Breaking the speed versus accuracy trade-off. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2313–2318, Dec 2015.

S. van Loon, M. Donkers, N. van de Wouw, and W. Heemels. Stability analysis of networked and quantized linear control systems. *Nonlinear Analysis: Hybrid Systems*, 10:111 – 125, 2013. ISSN 1751-570X. Special Issue related to IFAC Conference on Analysis and Design of Hybrid Systems (ADHS 12).

D. Vanthienen, M. Klotzbcher, and H. Bruyninckx. The 5c-based architectural composition pattern: lessons learned from re-developing the itasc framework for constraint-based robot programming. *JOSER: Journal of Software Engineering for Robotics*, 5(1):17–35, 2014. ISSN 2035-3928. URL `$$Uhttps://lirias.kuleuven.be/retrieve/274224$$DOAarticle[freelyavailable]`.

Y. Watanabe, H. Oku, and M. Ishikawa. Architectures and applications of high-speed vision. *Optical Review*, 21(6):875–882, 2014. ISSN 1349-9432.

S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, Apr. 2009. ISSN 0001-0782.

S. W. Williams. *Auto-tuning Performance on Multicore Computers*. PhD thesis, EECS Department, University of California, Berkeley, Dec 2008.

Z. Ye, Y. He, R. Pieters, B. Mesman, H. Corporaal, and P. Jonker. Demo: An embedded vision system for high frame rate visual servoing. In *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–2, Aug 2011a.

Z. Ye, Y. He, R. Pieters, B. Mesman, H. Corporaal, and P. Jonker. Bottlenecks and tradeoffs in high frame rate visual servoing: A case study. In *IAPR Conference on Machine Vision Applications*, Nara, Japan, 2011b.

Z. Ye, H. Corporaal, P. Jonker, and H. Nijmeijer. Cross-domain modeling and optimization of high-speed visual servo systems. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1791–1798, Nov 2018.

A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), Dec. 2006. ISSN 0360-0300.

J. Zhang, R. Lumia, J. Wood, and G. Starr. Delay dependent stability limits in high performance real-time visual servoing systems. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 485–491 vol.1, Oct 2003.

L. Zhang, P. Slaets, and H. Bruyninckx. An open embedded industrial robot hardware and software architecture applied to position control and visual servoing application. *International Journal of Mechatronics and Automation*, 4 (1):63–72, 2014a. doi: 10.1504/IJMA.2014.059775. PMID: 59775.

L. Zhang, P. Slaets, and H. Bruyninckx. An fpga based architecture for concurrent system design applied to human-robot interaction applications. In *Proceedings of the 21st ISPE International Conference on Concurrent Engineering*, volume 1, pages 555–563. IOS PRESS, 2014b.

Y. Zhang, K. K. Tan, and S. Huang. Vision-servo system for automated cell injection. *Industrial Electronics, IEEE Transactions on*, 56(1):231–238, Jan 2009. ISSN 0278-0046.

Appendix A

Cost estimation of vision systems

The cost of several cameras and processing systems are listed in Table A.1 and Table A.2 respectively.

Label	Sensor	Image size	Sample rate	Camera Cost
①	KAI-0340	228x164	1637	1000
②	LUPA3000	256x256	10704	5000
③	Phantom v2010[a]	256x256	188900	30000

Table A.1: Estimated costs of cameras. The source of the cost is from private communication. The price listed in the table is coarsely rounded off, because the exact price cannot be disclosed to the public. The Phantom v2010 has an undisclosed sensor, thus the cost of the sensor is estimated from another sensor of comparable performance.

[a]Phantom v2010 is a recording system with an undisclosed image sensor.

For a given vision algorithm, Equation (3.3) can be used to match processing systems with cameras approximately. For a vision algorithm with computational complexity of 100 op/px, the cost of cameras dominates that of processing systems, as shown in Figure A.1.

Label	System	gop/s	Cost
(a)	DSP (TMS320C6657)	40	50
	FPGA (XC5VSX50T)	100	1000
	CPU (i5-3570)	136	200
	CPU (E5-2690)	300	2000
(b)	DSP (TMS320C6678)	320	400
(b+)	FPGA (XC7K325T)	1245	1500
	GPU (GTX 780 Ti)	5046	700
(c)	FPGA (XC7VX690T)	5335	5000
(d)	FPGA (XCKU115)[a]	8180	8000

Table A.2: Estimated costs of processing systems.

[a]The price of the Kintex UltraScale FPGA (XCKU115) is not yet available. Therefore it is estimated from comparable devices.

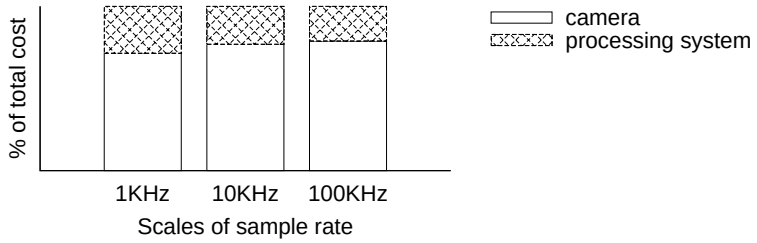


Figure A.1: Cost comparison of cameras and processing systems at different scales of sample rate. As a relative comparison, the total cost includes only cameras and processing systems. The choices of cameras and processing systems are: ① and (b) for 1KHz; ② and (b+) for 10KHz; ③ and (d) for 100KHz.

Appendix B

Redundant and coupled design

This section provides an example of redundant and coupled design, and methods to simplify them. First, design parameters are clustered into lumped parameters. After that, design hierarchy is imposed.

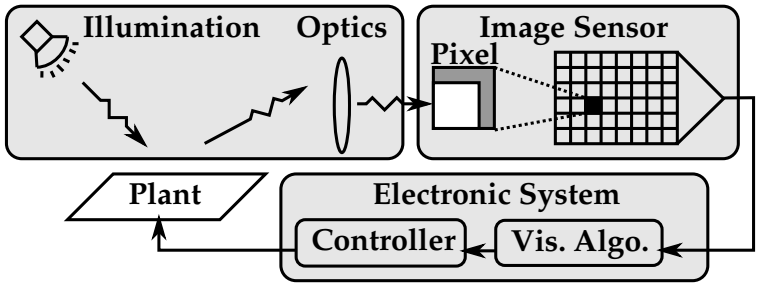


Figure B.1: Multi-domain view of a visual servo system.

For a visual servo system illustrated in Figure B.1, multiple design parameters from different domains are involved. By way of example, a design equation can

Symbol	Description
$ill.$	Illumination
$opt.$	Optics
n_s	Noise induced by image sensor
I_s	Image size
P_o	Number of pixels for object of interest
N_o	Number of objects in the field of view
$alg.$	Vision algorithm
$arch.$	Processing architecture
R_d	Data throughput rate between camera and processor
P	Plant
$est.$	State estimator
C_{ff}	Feedforward controller structure
C_{fb}	Feedback controller structure
u_{max}	Maximum actuator force

Table B.1: Explanations of the design parameter vector in (B.1).

be as complicated as

$$\begin{bmatrix} \epsilon \\ h \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 \\ \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} ill. \\ opt. \\ n_s \\ I_s \\ P_o \\ N_o \\ alg. \\ arch. \\ R_d \\ P \\ est. \\ C_{ff} \\ C_{fb} \\ u_{max} \end{bmatrix}, \quad \text{(B.1)}$$

in which the functional parameters on the left is the same as defined in Chapter 2.4 and Chapter 5.1, and the design parameters on the right are described in Table B.1.

The design parameters in (B.1) can be clustered according to the similarity of their coupling with functional requirements. Subsequently, if two columns of the design matrix are the same, the two design parameters applied on them can be grouped into a lumped parameter. By grouping multiple design parameters into

a lumped parameter, the design equation becomes

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} \star & 0 & 0 & 0 & 0 \\ \star & \star & \star & 0 & 0 \\ \star & \star & 0 & \star & 0 \\ \star & \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} I_s \\ DP_{obj.alg.} \\ DP_{i.o.s.} \\ DP_{arch.d.} \\ DP_{p.ctrl.u.} \end{bmatrix}, \tag{B.2}$$

in which

$$DP_{obj.alg.} = \begin{bmatrix} P_o \\ N_o \\ alg. \end{bmatrix}, DP_{i.o.s.} = \begin{bmatrix} ill. \\ opt. \\ n_s \end{bmatrix},$$

$$DP_{arch.d.} = \begin{bmatrix} arch. \\ R_d \end{bmatrix}, DP_{p.ctrl.u.} = \begin{bmatrix} P \\ est. \\ C_{ff} \\ C_{fb} \\ u_{max} \end{bmatrix}. \tag{B.3}$$

The four groups of design parameters in (B.3) are explained in Table B.2.

Symbol	Description
$DP_{i.o.s}$	Design parameters of illumination, optics, and image sensor
$DP_{obj.alg.}$	Design parameters of object of interest and vision algorithm
$DP_{arch.d.}$	Design parameters of processing architecture and image data rate
$DP_{p.ctrl.u.}$	Design parameters of plant, controller and actuator

Table B.2: Explanations of design parameter groups in (B.3).

After grouping the design parameters, the design hierarchy can be imposed explicitly on the design equation, i.e.,

$$\begin{bmatrix} h \\ \epsilon \\ \tau \\ K \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \star & \star & \star & \star \end{bmatrix} \begin{bmatrix} \star & 0 & 0 & 0 & 0 \\ \star & \star & \star & 0 & 0 \\ \star & \star & 0 & \star & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_s \\ DP_{obj.alg.} \\ DP_{i.o.s.} \\ DP_{arch.d} \\ DP_{p.ctrl.u} \end{bmatrix}. \tag{B.4}$$

The design is still redundant and coupled, but the design space exploration process is simplified to a similar case as (2.4). The complexity induced by the redundant and coupled design is contained within the cluster of each parameter group, which only involves additional combinatorial optimization within each parameter group. Therefore, the modeling and exploration of redundant and coupled designs, such as (B.1), can be approached in the same way as (2.4), which are detailed in the main chapters of this thesis.

Samenvatting

Visuele waarnemingen wordt veel gebruikt in mechatronische en robotsystemen om de on-line dynamiek van een systeem in zijn omgeving waar te nemen. Dat is moeilijk en soms onmogelijk te verkrijgen van andere sensoren. Er is tientallen jaren onderzoek gedaan naar dit onderwerp. Het blijft echter een uitdaging om visuele detectie te integreren in nauwkeurige mechatronische en robotsystemen.

Er zijn meerdere uitdagingen, en veel daarvan blijven open problemen. Ten eerste is visuele waarnemingen reken intensief en gegevens intensief. Het vereist een afweging tussen samplefrequentie, vertraging en nauwkeurigheid. Ten tweede moeten regelaar worden ontworpen om aanvullende informatie te gebruiken die wordt geboden door visuele feedback en om de nadelen die hierdoor worden veroorzaakt te verlichten. Ten derde wordt high-speed visuele waarnemingen vaak bereikt door co-ontwerp van software en hardware. Het is moeilijk om een vroege inschatting te maken van de vertraging voordat deze wordt geimplementeerd. Ten slotte is een cross-domeinmodel vereist dat uitgebreid en te analyseren is, om ontwerp afwegingen over meerdere domeinen in visuele servosystemen te verkennen.

In dit proefschrift wordt een reeks methoden voorgesteld om elk van deze uitdagingen te overwinnen. Eerst worden ontwerpmethoden voorgesteld voor zeer snelle en zeer nauwkeurige vision-systemen en vervolgens toegepast op een implementatie van een prototype. Ten tweede worden regelaars voorgesteld om online metingen van visuele feedback te gebruiken en zijn ze geoptimaliseerd voor samplefrequentie, vertraging en kwantisatie fout van visuele feedback. Ten derde worden algoritmische patronen en architecturale sjablonen gebruikt om een vroege schatting van vertraging in aangepaste visiesystemen te verkrijgen. Ten vierde wordt een axiomatische ontwerpmethode toegepast op visuele servosystemen. Het modelleert expliciet cross-domein koppelingen en maakt verkenning mogelijk van cross-domein trade-offs met betrekking tot de algehele systeemprestaties.

Het aanpakken van de bovengenoemde uitdagingen leidt tot meerdere bijdragen en bevindingen in dit proefschrift. Deze bijdragen omvatten methoden om high-speed vision-systemen te ontwerpen met het oog op controle, methoden om regelaars voor visuele feedback te ontwerpen en te optimaliseren, methoden om

algoritmische en architecturale keuzes te evalueren en methoden om cross-domein trade-offs te modelleren en te verkennen. Deze methoden worden toegepast op implementaties van prototypes, wat leidt tot meerdere bevindingen. Het blijkt dat er, zelfs voor een basis voorbeeld en voor een kleine set ontwerpparameters, niet-triviale cross-domein afwegingen zijn. Wat nog belangrijker is, is dat er significante verbeteringen zijn in de systeemprestaties die alleen kunnen worden bereikt door domeinoverschrijdende verkenning.

Samenvattend biedt dit proefschrift een hulpmiddel voor het begrijpen en evalueren van fundamentele ontwerpkeuzes in visuele servosystemen met hoge precisie en hoge snelheid. Deze tool kan worden gebruikt om bestaande systemen te analyseren en te optimaliseren, een productfamilie van systemen met verschillende afwegingen op te zetten en een routekaart voor toekomstige generaties van systemen te verkennen.

Acknowledgements

This thesis is a long journey. I am grateful to many people who accompanied me and supported me along the way.

I would like to thank my promoters Henk Nijmeijer and Pieter Jonker, who offered me an opportunity to work on the Embedded Vision Architecture project, on which this thesis is based, and gave me invaluable guidance throughout my PhD project and afterwards. Henk, thank you for giving me a guided tour into the domain of dynamics and control, being very patient with my progress, and being open-minded and supportive to the research direction that I decided to pursue. Pieter, thank you for introducing me to a wide range of domains including machine vision, vision processing systems, software code generators, augmented reality, mechatronics, robotics, machine learning, autonomous vehicles, luckily without getting me lost along the way. Your broad and practical experiences provided a fertile breeding ground to the cross-domain research being pursued in this thesis.

I would like to thank my adviser Henk Corporaal, who inspired me to pursue a PhD degree when I was a student under his supervision, and later provided great support when we worked together in the Embedded Vision Architecture project. Henk, thank you for mentoring me since the beginning of my academic career, welcoming me to be part of your team, providing me guidance on how to teach and how to perform research. It is a great pleasure to work with you, and be part of your team.

I would like to thank members of my promotion committee, prof.dr. Bradley Nelson, prod.dr. Robert Babuska, and prof.dr.ir. Herman Bruyninckx, for their efforts to review this thesis and their valuable feedback on it.

I would like to thank my colleagues and friends from the Mechanical Engineering department of Eindhoven University of Technology. First of all, thank you Roel. It is a great pleasure to work with you in the same project. You helped me enter into new fields and integrate into new teams. Most importantly, you made the journey fun. I would like to thank Jonatan, Nandra, Alejandro, Alper, Mark, Carlos, and Haitao from my lunch group for their companion and support at work and outside work. I would like to thank many other colleagues from the Dynamics and

Control, Control Systems Technology, and Networked Control Systems groups, who together created a great working place. And a special thanks to Geertje for providing day-to-day support when I was working in the group, and helping with various arrangements for my PhD defense.

I would like to thank my colleagues and friends from the Biorobotics group of Delft University of Technology. I would like to thank Wouter, Oytun, Maja, Erik, Xin, and Berk for providing advices and sharing thoughts during my PhD project. I would like to thank Boris and Eelko for being very supportive when we were working together in the same project during 2014. I would like to thank many others in the group with whom I share a good time.

I would like to thank my colleagues and friends from the Electronic Systems group of Eindhoven University of Technology. I would like to thank Yifan for being a supportive colleague and friend since we were students in the same group, and later when we were working together in the Embedded Vision Architecture project, the low-power image processor project, and other small projects. I would like to thank Ahsan, Mark, and Jan who supported me in the Embedded Vision Architecture project. I would like to thank members of the Parallel Architecture ReSearch Eindhoven (PARSE) team (Cedric, Gert-jan, Dongrui, Shakith, Maurice, Luc) for their countless presentations, discussions, and feedback that have inspired me. I would like to thank Yu and Sebastian who worked together in the low-power image processor project which leaded to a paper we co-authored. I would like to thank Bart Mesman, who guided me in my master project, and was involved at the starting phase of the Embedded Vision Architecture project. Bart, I still remembered the question you kept asking me, "Which algorithm is better? Is it the fast-but-inaccurate one or the accurate-but-slow one?" It inspired me to pursue the research direction of this thesis.

I would like to thank my colleagues at Intel, Connecterra BV, and Caspar AI, where I worked after I left university. At Intel, I would like to thank Mark for offering striking lessons on system architecture, via his inspirational speeches and his special sense of humor. I would also like to thank my manager Lars and my colleagues at Intel with whom I shared a great time with. At Connecterra, I would like to thank Yasir, Saad, Jorge, Sicco, Timo, and Niels for inviting me to join the small team back in the early days, and many others who joined afterwards for being together into the adventure. A special thanks to Nela, who sit next to me for a short time and yet inspired me with countless brave, authentic, and thought provoking "small" talks. At Caspar, I would like to thank Maja for inviting me to be an initial member of the team at Rotterdam, and my colleagues there who helped create a great working environment.

I would like to thank my parents for their unconditional support that enabled me to pursue the career I dreamed about since I was a child. And, finally, I would like to thank my wife, Nan. Thank you, Nan, for being together with me through all the good times and difficult times. It is the best journey in my life, because of you.

List of publications

List of publications by the author of this thesis.

In preparation

- **Z. Ye**, H. Corporaal, P. Jonker, and H. Nijmeijer. Cross-domain design space exploration of high-speed visual servo systems. In preparation for journal publication.

Published peer-reviewed journal articles

- R. Pieters, **Z. Ye**, P. Jonker, and H. Nijmeijer. Direct motion planning for vision-based control. IEEE Transactions on Automation Science and Engineering, 11 (4):12821288, Oct 2014. ISSN 1545-5955.

- Y. Pu, Y. He, **Z. Ye**, S. M. Londono, A. A. Abbo, R. Kleihorst, and H. Corporaal. From xetal-ii to xetal-pro: On the road toward an ultralow-energy and high-throughput simd processor. IEEE Transactions on Circuits and Systems for Video Technology, 21(4):472484, April 2011. ISSN 1051-8215.

Published peer-reviewed articles in conference proceedings

- **Z. Ye**, H. Corporaal, P. Jonker, and H. Nijmeijer. Cross-domain modeling and optimization of high-speed visual servo systems. In 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pages 1791-1798, Nov 2018.

- **Z. Ye**, Y. He, R. Pieters, B. Mesman, H. Corporaal, and P. Jonker. Demo: An embedded vision system for high frame rate visual servoing. In 2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras, pages 1-2, Aug 2011.

- Y. He, **Z. Ye**, D. She, B. Mesman, and H. Corporaal. Feasibility analysis of ultra high frame rate visual servoing on fpga and simd processor. In Advances Concepts for Intelligent Vision Systems, pages 623-634. Springer, 2011.

- **Z. Ye**, Y. He, R. Pieters, B. Mesman, H. Corporaal, and P. Jonker. Bottlenecks and tradeoffs in high frame rate visual servoing: A case study. In IAPR Conference on Machine Vision Applications, Nara, Japan, 2011.

Curriculum Vitae

Zhenyu Ye was born on October 16th, 1983 in Huizhou, China. In 2006, he graduated with Bachelor of Science degree in Electrical Engineering from Harbin Institute of Technology in Harbin, China. In 2009, he graduated with Master of Science (ir.) degree in Embedded Systems from Eindhoven University of Technology in Eindhoven, the Netherlands. He performed his master thesis in Electronic Systems group, Department of Electrical Engineering. Between 2009 and 2014, he worked full-time on his PhD project, Embedded Vision Architecture, in Dynamics and Control group, Department of Mechanical Engineering, Eindhoven University of Technology. Results of his PhD project are presented in this dissertation.

Since 2014, he worked part-time on his PhD dissertation and full-time on different jobs. Between April 2014 and July 2014, he worked on vision processing systems at Delft University of Technology. Between August 2014 and June 2016, he worked on image signal processors at Intel in Eindhoven. Between July 2016 and August 2019, he worked on wireless sensor systems at Connecterra BV in Amsterdam. Since September 2019, he worked on intelligent operating systems for smart homes at Caspar AI in Rotterdam.

In 2006, he was awarded TU/e scholarship for his study at Eindhoven University of Technology. In 2010, a paper he co-authored, titled "Xetal-Pro: an ultra-low energy and high throughput SIMD processor," was given the HiPEAC Paper Award.