A New Optimal Trajectory Planner for Position Based Visual Servoing

YUQUAN WANG

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2010
A New Optimal Trajectory Planner
for Position Based Visual Servoing

Y U Q U A N W A N G

Master's Thesis in Computer Science (30 ECTS credits)
at the Systems, Control and Robotics Master’s Program
Royal Institute of Technology year 2010
Supervisor at CSC was Patric Jensfelt
Examiner was Danica Kragic

TRITA-CSC-E 2010:116
ISRN-KTH/CSC/E--10/116--SE
ISSN-1653-5715
A New Optimal Trajectory Planner for Position Based Visual Servoing

Abstract
A new optimal trajectory planner for position-based 6 degrees of freedom (6DOF) visual servoing is presented in this Thesis. Basically it is inspired from spline optimization method. A comparative study with the new method is conducted in section 5, where the new method is compared to a classical position based visual servoing method (PBVS), and to three image based visual servoing methods (IBVS). Compared with the other visual servoing methods, its advantages are shown in the following aspects: First, it will 100% keep the image features within the field of view due to the hard constraints in the optimization procedure; Secondly, the camera motion will be optimized; Thirdly, the flexibility of the optimization framework allows us to choose different combinations of objectives and constraints.

En ny optimal banplaneringsalgoritm för positionsbaserad visuell styrning

Sammanfattning
En ny optimal banplaneringsalgoritm för positionsbaserad visuell styrning i sex frihetsgrader (6DOF) presenteras i denna avhandling. Metoden bygger på spline-optimering. En jämförande studie av den nya metoden relativt fyra andra styrlagar har utförts, där styrlagen i tre av dem bygger på information i bildplanet och den fjärde styrlagen bygger på tredimensionell information. I studien framgår att den nya metoden har följande fördelar jämfört med de andra metoderna: För det första så garanterar metoden att punkterna som observeras hela tiden kommer att finnas i bildplanet. För det andra kommer kamerarörelser optimeras. För det tredje ger flexibiliteten i denna formulering oss möjligheten att konstruera olika kombinationer av målfunktioner och bivillkor.
Preface

This report is the result out of my seven months’ thesis work supervised by Johan Thunberg and Xiaoming Hu at the department of System Theory and Optimization at Royal Institute of Technology (Kungliga Tekniska Högskolan).

My supervisors lead me into the world of visual feedback control. We started from basic concepts and equations. As a beginner, I always had plenty of questions. I was keen on asking, Johan was also keen on answering. Xiaoming is a Professor with a lot of tasks, but I was always allowed to go to his office and search for help. We experienced busy but happy spring and summer together. Thanks to their patient and frequent help, we got the result step by step.

Before I came to Sweden, I only have limited background in Mechanical Engineering. Patric Jensfelt is the director of my Master’s Program. Under his guide, I came to know the beauty of our interdisciplinary program, which is a combination of robotics and system theory. Carlo Fischione, Ulf Jönsson, Danica Kragic, Örjan Ekeberg, Oscar Flärhd and all my tutors gave me great knowledge and support to explore Control Theory and Computer Science.

I would like to express my sincere gratitude to all the teachers and classmates who are always giving me help. Unfortunately, I will leave Sweden next month. I deeply appreciate my two years’ experiences here. Wish everyone the best of luck!

Stockholm, September 2010
Wang, Yuquan
# Contents

1 Introduction .................................................................................................................. 1
   1.1 Problem description and assumptions .................................................................... 1
   1.2 Motivation and approach ....................................................................................... 2
   1.3 Report outline ....................................................................................................... 3

2 Related work .................................................................................................................. 4

3 Rigid body dynamics and control ............................................................................... 6
   3.1 Rigid body dynamics ............................................................................................ 6
      3.1.1 Angular momentum and the inertia matrix .................................................... 6
      3.1.2 Dynamics equations ....................................................................................... 7
   3.2 Hierarchical control structure .............................................................................. 8
   3.3 Low level actuator control ................................................................................... 9
      3.3.1 Model of one reaction wheel ......................................................................... 9
      3.3.2 Torque control of reaction wheel ................................................................. 10
      3.3.3 Attitude control system based on reaction wheels ....................................... 11
   3.4 Middle level control design ............................................................................... 13

4 Theoretical analysis of the new optimal trajectory planner .................................. 16
   4.1 Introduction of the classical visual servoing methods ........................................ 16
      4.1.1 General introduction of PBVS ..................................................................... 16
      4.1.2 Embedded problems of PBVS ................................................................. 19
      4.1.3 General introduction of IBVS .................................................................... 20
   4.2 Linearization of the rigid body motion ............................................................... 21
   4.3 3D pose estimation Using EPnP ......................................................................... 23
      4.3.1 Points parameterization ............................................................................. 24
      4.3.2 Solution as weighted sum of Eigenvectors ............................................... 24
      4.3.3 Choosing the right linear combination ...................................................... 25
      4.3.4 Simulation result ....................................................................................... 25
   4.4 Motion optimization ............................................................................................ 26
   4.5 Pyramid constraint for keeping features within field of view ........................................ 28
   4.6 Summary of the new optimal trajectory planner .............................................. 31

5 Simulation environment and results ......................................................................... 33
   5.1 Simulation toolbox setup ................................................................................... 33
      5.1.1 Camera visualization tool ......................................................................... 33
      5.1.2 Optimization tool and visualization work .................................................. 34
   5.2 Demonstration of basic functionalities ............................................................... 34
      5.2.1 Performances of PBVS ............................................................................. 34
      5.2.2 Performances of the new optimal trajectory planner ................................ 37
   5.3 Comparison with PBVS .................................................................................... 40
      5.3.1 Performance on keeping image features within field of view .................. 40
      5.3.2 Camera motion performance ................................................................. 42
   5.4 Comparison with IBVS ..................................................................................... 44

6 Analysis of the new optimal trajectory planner .................................................. 50
   6.1 Analysis of objectives/constraints .................................................................... 50
   6.2 Analysis on different initial conditions ............................................................ 55
6.3 Multi-stage trajectory planning ................................................................. 56
7 Conclusions ................................................................................................. 57
Bibliography ................................................................................................. 58
1 Introduction

The purpose of this Master’s thesis is to investigate, develop and simulate Visual Servoing algorithms for the rigid body with six degrees of freedom (6DOF). Visual servoing refers to the control of a certain plant by using visual input (CHAUMETTE, o.a., Jan 2009). In this thesis, we will consider the problem of controlling the pose of the rigid body relative to a reference frame, e.g. a desired position or another rigid body, while using visual information in a closed loop. We will assume that 3D coordinates of the feature points are known in advance and the 2D images of these feature points are available to feed into the control part. There are many solutions to this problem according to a newly published visual servoing tutorial (CHAUMETTE, o.a., Jan 2009), however in this thesis a new optimal trajectory planner inspired from spline optimization (Demeulenaere, o.a., July 11-13, 2007) will be presented.

1.1 Problem description and assumptions

In this thesis, the rigid body model on which the visual servoing algorithm is going to be implemented is assumed to have 6DOF and the affection of gravity will be neglected, this means it is more or less like a spacecraft floating in the universe. Under these assumptions the rigid body model is free to make any Euclidean motion. All Euclidean motions are denoted by SE(3) and the details about the rigid body dynamics can be found in section 3.

Basically the rigid body model needs to fulfill a certain servoing task. We assume there is a fixed goal and the rigid body model needs to adjust the relative pose between itself and the fixed goal based on visual inputs.

Like other visual servoing methods mentioned in (CHAUMETTE, o.a., Jan 2009) or (Malis, o.a., APRIL 1999), the visual input will be the vision data from a camera directly mounted on the rigid body model and this is the so-called eye-in-hand case, which means the camera motion is identical to the rigid body motion.

Besides the assumptions of the rigid body model, we also assume that the 3D pose estimation algorithm directly has access to the 3D coordinates of the feature points in the desired camera frame, 2D projections of the feature points in the current camera frame and the correspondences between them. This means the image processing will not be included in this thesis.

The last assumption is about the camera model which will be used through this thesis, according to (Ma, o.a., 2003), a perspective camera model could be described by equation (1.1.1)

\[
m = K \pi x_w. \tag{1.1.1}
\]

Where \( K \in R^{3\times3} \) is called the camera intrinsic parameters matrix given by:

\[
K = \begin{bmatrix}
k_u f & \gamma & u_0 \\
0 & k_v f & v_0 \\
0 & 0 & 1
\end{bmatrix}. \tag{1.1.2}
\]

Here \((u_0, v_0)\) represents the pixel coordinate in the image frame, \(k_u\) and \(k_v\) are the number of pixels per unit distance in image coordinate, \(f\) is the focal length and \(\gamma\) is the orthogonality
factor of the CCD image axes (skew factor). However, for simplicity, in simulation the camera intrinsic parameters matrix will be assumed to be known and set as identity, which means \((u_o, v_o) = (0,0), \gamma = 0, f = 1\) and \((k_u,k_v) = (1,1)\).

In equation (1.1.1), \(\pi = [R|t] \in \mathbb{R}^{3 \times 4}\) is called camera external parameters matrix, which contains the rotation matrix \(R\) and the translation vector \(t\). According to (MARIOTTINI, o.a.), the optical axis \(Z_c\) of the perspective camera should be chosen parallel to the \(Y\) axis of the world frame. So usually, we need to define the rotation of the camera in the very beginning of every setting like the following:

\[
R = (R_{x,-\frac{\pi}{2}}R)^T \text{ and } t = -(R_{x,-\frac{\pi}{2}}R)^T t, \tag{1.1.3}
\]

which means the camera frame will be rotated about its \(X\) axis by \(-\frac{\pi}{2}\). This relationship could be shown in Figure 1.1.1.

In all the simulations in this thesis, the camera will first rotates as described in equation (1.1.3) and shown in Figure 1.1.1. One visualization example can be found later in section 5.1.1.

### 1.2 Motivation and approach

The classical visual servoing methods have their own advantages, but on the other hand they also have their own problems. For example, the classical position-based visual servoing method (PBVS) doesn’t concern the field of view problem. Classical image-based visual servoing method (IBVS) lacks long step optimality. In order to overcome the short comings and achieve better optimization result with long prediction steps, we are going to develop a new optimal trajectory planner inspired from spline optimization (Demeulemaere, o.a., July 11-13, 2007). Basically constraints concerning the short comings of the classical visual servoing methods will
be taken into account. In addition, since it’s an optimization method, objectives/constraints are flexible, which gives us the flexibility to adapt it to different scenarios.

An very important step in visual servoing is the pose estimation, unlike other methods, the 3D pose estimation method used in this thesis consists of EPnP (Lepetit, o.a., 2008) and Horn’s absolute orientation algorithm (Horn, o.a., March 25, 1988). The procedure is first to use EPnP to solve the perspective-n-point problem in a non-iterative way. Then use Horn’s algorithm to solve the orientation. Notice that the EPnP method is only $O(n)$ and applicable for all $n \geq 4$ and handles properly both planar and non-planar situations.

The overall control structure will be presented in section 3.2. Basically, we are going to use feedback linearization to control the rigid body motion.

### 1.3 Report outline

The rest of this thesis will be organized as follows. In section 2, related work will be introduced to offer a background about the existing visual servoing methods. Then in section 3 the rigid body motion dynamics as well as the hierarchical control structure will be introduced. Theoretical analysis of the new optimal trajectory planner introduced in this thesis will be explained in detail in section 4. Simulation results, the corresponding analysis and comparisons will be shown in section 5. Some other possibilities of further developments will be discussed in section 6. Conclusions will be drawn in section 7.
2 Related work

In this section, we are going to review different visual servoing methods. First the two classical visual servoing methods IBVS and PBVS will be introduced. Then hybrid methods like 2-½ D visual servoing and switching schemes will be briefly discussed. Modifications in pose estimation, path planning and simulation tools will also be talked about.

Two classical visual servoing approaches are known as Image-based visual servoing (IBVS) and Position-based visual servoing (PBVS). Indicated from their names the first one is based on the 2D image and the second one relies on 3D estimation. According to Wiess’ paper (Wiess, o.a., August 26 1985), image-based control schemes use the image-plane coordinates of a set of points to define the control variables, whereas the position-based control schemes (Wiess, o.a., August 26 1985) use the pose of the camera relative to a reference coordinate frame to define the control variables. Detailed reviews of different visual servoing methods can be found from visual servoing tutorials (CHAUMETTE, o.a., Jan 2009) and (CHAUMETTE, o.a., Jan 2009). In addition, different from PBVS and IBVS other interesting features other than points are also possible choices. For example, slope or line is invariant to translations. Zeng presented one line-based Homography estimation method in (Zeng, o.a., 2008).

Basically the position-based visual servoing has its own problems according to Malis’ paper (Malis, o.a., APRIL 1999). For example, there’s no control in the image, so position-based visual servoing needs to run the risk that the image features may get out of the field of view. If that happened, the control loop is broken down and the servo task will fail.

In order to overcome this problem, different solutions have been presented. A path planning approach based on potential field is presented in (Mezouar, o.a., Jan 2009). In (Thuilot, o.a., May 2002) an online closed loop path planning approach is introduced. Both these two methods have been tested by 6DOF robotic platform.

Another problem of PBVS is that the velocity is proportional to the error term so if the error is big, saturation of the actuators may happen. In order to maintain the smooth motion, optimal control theory has been implemented. In (Hashimoto, o.a., 1993) an approach based on linear quadratic Gaussian control has been proposed to choose gains that minimize a linear combination of the states and the control.

Besides the above mentioned problems, 3D pose estimation is a crucial topic in position-base visual servoing. In 2-½ D visual servoing, planar feature points are used to estimate pose with Homography. It is also possible to estimate the numerical value of the Homography matrix directly (Irani, o.a., 2000).

Considering both the pros and the cons of IBVS and PBVS, 2-½ D visual servoing introduced in (Malis, o.a., APRIL 1999) combines IBVS and PBVS together. This method relies on Homography estimation/decomposition. Basically 2-½ D visual servoing is able to decouple the translational and rotational motions and do not need 3D pose estimation in every step. However the accuracy of this method highly depends on Homography estimation/decomposition. A recent review of Homography decomposition could be found in (Malis, o.a., Septembre 2007). Nevertheless another intuitive idea is to use IBVS and PBVS together, when we detect IBVS performs badly we switch to PBVS, in the other way round, when we detect PBVS performs
badly we switch back to IBVS, a switching Lyapunov function scheme is introduced in (Gans, o.a., Oct. 2003).

Regarding simulation tools, there are many good platforms. VISP (VIS) is a modular software that allows fast development of visual servoing applications. It is implemented in C++ and is available under Linux, Windows and OSX. Epipolar geometry toolbox (MARIOTTINI, o.a.) is a toolbox designed for Matlab. Omi-directional camera model and choice of multiple views make it different. Visual Servoing Toolbox (Cervera) aims to provide a set of functions and blocks for simulation of vision-controlled systems. Classical IBVS, PBVS and a simple 2-½ D visual servoing model in simulink could be found inside. However, Robotics toolbox for Matlab (Corke, 2008) is a more general platform including kinematics, dynamics and trajectory generation. The toolbox is useful for simulation as well as analyzing results from experiments with real robots.
3 Rigid body dynamics and control

In this section, the dynamics of a rigid body with six degrees of freedom (6DOF) will be discussed in detail. Notice that in this thesis the gravity will be neglected. Based on the nonlinear dynamic equations, nonlinear control method based on feedback linearization will be used.

3.1 rigid body dynamics

3.1.1 Angular momentum and the inertia matrix

Here we first introduce the well known operator equation acting on a given vector $A$ from (Sidi, 1997):

$$\frac{d}{dt} A|_I = \frac{d}{dt} A|_B + \omega \times A.$$  \hspace{1cm} (3.1.1)

We use this equation to illustrate the relationship between the rate of change of the vector $A$ observed in the inertial frame (fixed frame) and the rate of change of the vector $A$ observed in the body frame (rotating frame with rotating rate $\omega$).

As indicated from Figure 3.1.1.1, particle $m$ is defined in the rigid body with a fixed triad axis frame located at the center of mass. The coordinate of particle $m$ in the inertial frame is defined as:

$$R_I = R_O + r_i.$$  \hspace{1cm} (3.1.2)

According to (3.1.1), the derivative of $R_I$ is:
\[ \dot{\mathbf{R}}_i = \dot{\mathbf{R}}_0 + \dot{\mathbf{r}}_i + \omega \times \mathbf{r}_i = \mathbf{V}_0 + \mathbf{v}_i + \omega \times \mathbf{r}_i. \]  

(3.1.3)

Where \( \mathbf{V}_0 \) denotes the translational velocity of the object in the inertial frame, \( \mathbf{v}_i \) denotes the translational velocity of particle \( m \) in the body frame and \( \omega \) defined in the inertial frame denotes the angular velocity between the body frame and the inertial frame.

Then we see that the moment of momentum of a body particle \( m \) is

\[ h_i = \mathbf{r}_i \times m_i \dot{\mathbf{R}}_i = \mathbf{r}_i \times m_i (\dot{\mathbf{R}}_0 + \dot{\mathbf{r}}_i + \omega \times \mathbf{r}_i). \]  

(3.1.4)

Since we are discussing rigid body, so \( \dot{\mathbf{r}}_i \) is equal to zero. Then (3.1.4) could be simplified as:

\[ h_i = \mathbf{r}_i \times m_i (\dot{\mathbf{R}}_0 + \omega \times \mathbf{r}_i) = -\mathbf{V}_0 \times m_i \mathbf{r}_i + \mathbf{r}_i \times m_i (\omega \times \mathbf{r}_i). \]  

(3.1.5)

We sum up all the momentum components of the rigid body.

\[ h = \sum_{m_i} -\mathbf{V}_0 \times m_i \mathbf{r}_i + \sum_{m_i} \mathbf{r}_i \times (\omega \times \mathbf{r}_i) m_i \]

\[ = -\mathbf{V}_0 \times \sum_{m_i} m_i \mathbf{r}_i + \sum_{m_i} \mathbf{r}_i \times (\omega \times \mathbf{r}_i) m_i \]  

(3.1.6)

Since the angular motion about the center of mass \( \sum_{m_i} m_i \mathbf{r}_i \) should be zero. Then (3.1.6) is equal to:

\[ h = \sum_{m_i} \mathbf{r}_i \times (\omega \times \mathbf{r}_i) m_i. \]  

(3.1.7)

After performing the vector triple product, we get the following:

\[ h = i[\omega_x l_{xx} - \omega_y l_{xy} - \omega_z l_{xz}] + j[\omega_y l_{yy} - \omega_x l_{yx} - \omega_z l_{yz}] + k[\omega_z l_{zz} - \omega_x l_{zx} - \omega_y l_{zy}] = ih_x + jh_y + kh_z. \]  

(3.1.8)

Where the \( l_i \) components are defined as the inertia of the rigid body about its axes. If we define the angular velocity vector and the inertia matrix (inertia tensor) as:

\[ \omega = [\omega_x \omega_y \omega_z]^T; \quad J = \begin{bmatrix} l_{xx} & -l_{xy} & -l_{xz} \\ -l_{yx} & l_{yy} & -l_{yz} \\ -l_{zx} & -l_{zy} & l_{zz} \end{bmatrix}. \]  

(3.1.9)

Equation (3.1.8) could be rewritten as:

\[ h = J \omega, \]  

(3.1.10)

And the time derivative of \( h \) is given below:

\[ \dot{h} = J \dot{\omega}. \]  

(3.1.11)

### 3.1.2 Dynamics equations
We know that a moment, acting on a rigid body about its center of mass, equals to the time derivative of its angular momentum. In our case, the moment is the torque applied by the momentum exchange device.

According to (3.1.1) and (3.1.11), the relationship could be sum up as:

\[ T_e = \dot{h}_I = \dot{h}_B + \omega \times h_B. \]  

(3.1.12)

Where \( T \) denotes the external torque input and momentum of the entire system \( h_B \) will be divided into two parts. One part is the momentum of the rigid body \( h_{BY} = [h_x, h_y, h_z]^T \) and the other part is the momentum of the moment exchange device \( h_w = [h_{wx}, h_{wy}, h_{wz}]^T \). In this thesis we assume that there’s no external torque input, so \( h_w \) will provide internal torque \( T_i \) for attitude control.

According to equation (3.1.11) \( \dot{h} = J \omega \), equation (3.1.12) could be rewritten as:

\[ T_i = -\dot{h}_w - \omega \times h_w = \dot{h}_{BY} + \omega \times h_{BY} = J \omega + \omega \times J \omega. \]  

(3.1.13)

In general equation (3.1.13) summarizes the full attitude dynamics of a rigid body with 6DOF under no gravity influence.

Considering the influence of the force in the similar way, according to equation (3.1.1),

\[ \dot{V}_I = V_B + w \times V_B \]  

(3.1.14)

Where \( V_I \) is the translational velocity of the object in the inertial frame, \( V_B \) is the translational velocity of the object in the body frame and \( w \) defined in the inertial frame is the relative rotational velocity between the inertial frame and the body frame. By using \( \dot{V}_I = a_I = \frac{F_I}{m} \) and \( \dot{V}_B = a_B = \frac{F_B}{m} \), equation (3.1.14) could be rewrite into the following:

\[ F_I = m(a_B + \omega \times v_B) \]

\[ T_i = J \omega + \omega \times J \omega. \]  

(3.1.15)

Equation (3.1.15) describes the whole dynamics of rigid body motion. In order to introduce the velocity controller, equation (3.1.15) could be written into state space form as:

\[ \dot{w} = - J^{-1} w \times J \dot{w} + J^{-1} u_w = H_w + k_w u_w \]

\[ \dot{v} = - w \times v + m^{-1} u_v = H_v + k_v u_v. \]  

(3.1.16)

Where \( u_w \) and \( u_v \) replaced \( T_i \) and \( F_I \) in equation (3.1.15) respectively.

### 3.2 Hierarchical control structure

In this thesis, the control structure could be divided into three levels. The high level control is the visual servoing control scheme, which is going to generate desired spatial velocity based on vision input.

The middle level control will take care of the output from the high level control. Basically the middle level control will perform a tracking task. It will generate desired torques and forces to feed into the low level control such that the rigid body spatial velocity will track the desired spatial velocity generated from the high level control.
Lastly, the low level controller will ensure that the actuators will generate the desired torques and forces according to the output from the middle level control. The overview of the whole control structure is shown in Figure 3.2.1.

![Figure 3.2.1 Overview of the whole hierarchical control structure.](image)

Actually, image processing part will not be included in this thesis, the visual servoing part have access to the image features directly. In section 3.3 and section 3.4 the low level control and the middle level control will be introduced respectively. The high level control, which is the main part of this thesis, will be presented in section 4.

### 3.3 Low level actuator control

Here, the actuator control framework will be briefly introduced. Because the attitude control system is usually of more interests, so torque control system will be focused on.

The two basic kinds of momentum exchange devices are momentum wheels and reaction wheels, which are distinguished by their mode of operation. The reaction wheel is primarily used to provide with sufficient torque for various attitude-maneuvering tasks. In this thesis, we assume that we also use reaction wheels to provide torque.

#### 3.3.1 Model of one reaction wheel

According to (Sidi, 1997), the conventional model of a reaction wheel incorporates a DC motor, where the parameters of the reaction wheel are similar to a normal DC motor. Below is a typical DC motor model borrowed from course material of MF 2007 ‘Dynamics and motion control’ at
KTH. We use this model to mimic the reaction wheel’s behavior. The difference is we increased the inertia of the rotor as the reaction wheel needs to work on rigid body with big inertia.

![Figure 3.3.3.1 Model of the reaction wheel.](image)

Parameters of the reaction wheel are listed in table 3.3.3.1:

<table>
<thead>
<tr>
<th>Back EMF constant (EMF)</th>
<th>Torque constant ($k_t$)</th>
<th>Stator resistance ($R_M$):</th>
<th>Brush friction (d)</th>
<th>Rotor Static friction ($F_c$)</th>
<th>Rotor inertia ($J_w$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$8.6 \times 10^{-3}$</td>
<td>$8.64 \times 10^{-3}$</td>
<td>$24 \Omega$</td>
<td>$9 \times 10^{-7}$</td>
<td>$9.8 \times 10^{-6}$</td>
<td>$0.1$</td>
</tr>
</tbody>
</table>

Then the systems equations from the torque balance and Kirchhoff’s law are shown in equation (3.3.1) and equation (3.3.2) respectively:

\[
J_w \ddot{\phi} = k_t i - d \dot{\phi}, \quad (3.3.1)
\]

\[
U = Ri + K_{emf} \dot{\phi}. \quad (3.3.2)
\]

Where $\phi$ is just the rotation angle. Notice that in practice, the DC motor model could be a third order model, but we will use a second order approximation by neglecting the fast pole. That means the inductance of DC motor disappears in this approximation model.

### 3.3.2 Torque control of reaction wheel

Basically we need the reaction wheel to react fast enough to demanding torque command, such that the velocity of the rigid body will follows the desired velocity out of the visual servoing control scheme.
From Figure 3.3.3.2, we can see that there is a feedback from the motor current, which is proportional to the torque provided by the electrical motor. The transfer function between the torque command (In Figure 3.3.3.2, that’s the step input) and the achieved angular torque $h_w$ is:

$$\frac{h_w}{T_c} = \frac{K_{RM}}{1 + \frac{K_{RM}}{s}}$$

In our case, $\frac{EMF_k}{I} \approx \frac{(0.0086) \times 8.64 - 3}{1}$ is much smaller than $K_{RM} = 240$. So equation (3.3.3) could be rewritten as:

$$\frac{h_w}{T_c} = \frac{1}{1 + s\left(\frac{RM}{KRM}\right)} = \frac{1}{1 + 0.15}$$

This equation describes the relationship between output angular torque and the input torque command. Indicated from the time constant in equation (3.1.19) the rise time is around 0.1 sec, so the step response will be fast enough.

### 3.3.3 Attitude control system based on reaction wheels

The reaction wheel can be mounted in the rigid body with its rotation axis in any direction relative to the rigid body axis frame. That means three reaction wheels with each one’s rotational axis parallel to one of the body axes is the simplest configuration of the attitude control system. However this configuration is not safe. If one of the wheels is broken, the whole system can’t work. Therefore we introduce the four wheels attitude control system to increase the reliability according to method mentioned in (Sidi, 1997). The figure below is the geometrical set up of the four wheels system.
We can read from the Figure 3.3.3.3 that because all the four wheels are inclined to the XB - YB plane by an angle $\beta$, each wheel can apply torque in the ZB direction. To make it clear that we want torques out of the reaction wheel system, we denote the desired input to the actuators control system as $\bar{T}$. Thus we have the following equations:

$$
\bar{T} = \begin{bmatrix}
\frac{1}{\cos(\beta)}(1 & 0 & -1 & 0) & T_1 \\
\frac{1}{\cos(\beta)}(0 & 1 & 0 & -1) & T_2 \\
\frac{1}{\sin(\beta)}(1 & 1 & 1 & 1) & T_3 \\
\end{bmatrix} = [A_w] \begin{bmatrix} T_1 \\
T_2 \\
T_3 \\
T_4 \\
\end{bmatrix} = [A_w]T. \tag{3.3.5}
$$

Where $\bar{T}_i$ denotes the desired torque of each direction, matrix $A_w$ denotes the geometrical relationship between the desired torque $\bar{T} \in \mathbb{R}^3$ and actual torque $T \in \mathbb{R}^4$ of each reaction wheel shown in Figure 3.3.3.3.

Because $A_w$ is a non-square matrix, there are infinite many different combinations of $T_i$ to achieve the desired $\bar{T}_{ci}$. According to (Sidi, 1997) an optimizing criterion will be introduced to get the right combination. For example we define the Hamiltonian as $H = \sum_{i=1}^{4} \alpha_i T_i^2$, which means we want to minimize the sum of the torques generated from each reaction wheel. Then From equation (3.3.5):

$$
\bar{T} = \begin{bmatrix}
\frac{1}{\cos(\beta)} & \frac{1}{\cos(\beta)} & \frac{1}{\sin(\beta)} \\
1 & 0 & -1 & 0 & T_1 \\
T_2 \\
T_3 \\
T_4 \\
\end{bmatrix}.$$

Then we define the constraints:
The lagrangian will be:

\[ L = H + \lambda_1 g_1 + \lambda_2 g_2 + \lambda_3 g_3 + \lambda_4 g_4. \]  

(3.3.7)

In order to minimizing H we can derive the following conditions:

\[
\begin{align*}
\frac{\partial L}{\partial T_1} &= 2 \alpha_1 T_1 + \lambda_1 + \lambda_3 = 0 \\
\frac{\partial L}{\partial T_2} &= 2 \alpha_2 T_2 + \lambda_2 + \lambda_3 = 0 \\
\frac{\partial L}{\partial T_3} &= 2 \alpha_3 T_3 - \lambda_1 + \lambda_3 = 0 \\
\frac{\partial L}{\partial T_4} &= 2 \alpha_4 T_4 - \lambda_2 + \lambda_3 = 0
\end{align*}
\]

(3.3.8)

If we choose \( \alpha_i = 1, i = 1, \ldots, 4 \), we can derive the forth condition on \( T_i \):

\[ \Delta T = T_1 - T_2 + T_3 - T_4 = 0. \]  

(3.3.9)

This square matrix \( \tilde{A}_w \) has an inverse to calculate the desired torque on each reaction wheel.

### 3.4 Middle level control design

In this section, the middle level control, that is the velocity control, will be introduced. The velocity controller will ensure the rigid body velocity to track the desired velocity generated by the high level visual servoing.

The state space model (3.1.16) describing the rigid body dynamics could be rewritten as:

\[ \dot{x} = H + Ku, \]  

(3.4.1)

where \( x = [v^T w^T]^T, H = \begin{bmatrix} H_v \\ H_w \end{bmatrix} = \begin{bmatrix} -w \times v \\ -I^{-1} w \times I w \end{bmatrix} \) and \( K = \begin{bmatrix} k_v & 0_{3 \times 3} \\ 0_{3 \times 3} & k_w \end{bmatrix} = \begin{bmatrix} m^{-1} & 0_{3 \times 3} \\ 0_{3 \times 3} & I^{-1} \end{bmatrix} \). In order to make states \( x \) stable, controller in equation (3.4.2), which is based on feedback linearization, is a possible solution.

\[ u = K^{-1}(-H - A(x - x_d) + \dot{x_d}). \]  

(3.4.2)
Where $x_d$ is the desired state and $A$ is a certain matrix with positive Eigen values. In order to prove the validity of this controller, the error dynamics will be checked, which means if the error could converge to zero under the specified controller then the tracking problem could be solved.

First the error term could be defined as $\bar{x} = x - x_d$, then combining equation (3.4.1), the derivative of the error term is straight forward:

$$\dot{\bar{x}} = \dot{x} - \dot{x}_d = (H + Ku) - \dot{x}_d. \quad (3.4.3)$$

If we plug in equation (3.4.2), then:

$$\dot{\bar{x}} = \dot{x} - \dot{x}_d = -A(x - x_d) + \dot{x}_d - \dot{x}_d = -A\bar{x}. \quad (3.4.4)$$

Equation (3.4.4) clearly shows that the error dynamics is stable and the simulation in Figure 3.4.1 also supports this result. But the downside of this method is that controller in equation (3.4.2) involves $\dot{x}_d$, which is the derivative of $x_d$. Notice that $x_d$ is the result out of the high level visual servoing, so it probably contains noise. If we calculate the derivative of the noise, the result won’t be good.

![Figure 3.4.1 Tracking performance of the velocity controller.](image)

In Figure 3.4.1 the reference speed are shown in the following table.

<table>
<thead>
<tr>
<th>Rotational in x</th>
<th>Rotational in y</th>
<th>Rotational in z</th>
<th>Translational in x</th>
<th>Translational in y</th>
<th>Translational in z</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2\sin t$</td>
<td>0.8</td>
<td>$\cos t$</td>
<td>$\sin t$</td>
<td>5</td>
<td>$10\cos t$</td>
</tr>
</tbody>
</table>

Notice that the $A$ matrix in (3.4.2), which determines the convergence speed, is free to choose in theory, so we can have really fast responding system. But usually the $A$ matrix is subjected to the saturation limits of the actuators.
Because the middle level control will perform a tracking task so its sampling time will be much smaller compared with the sampling time of the high level visual servoing which will be introduced in the next section. However, details about visual servoing control architecture can be found in (Namiki, o.a., 2003).
4 Theoretical analysis of the new optimal trajectory planner

In this section, one new optimal trajectory planner will be introduced. The objectives of the trajectory planner are twofold. First, image features of the object should always be within the camera field of view. Second, the camera motion should be smooth. Basically theoretical analysis of the new optimal trajectory planner will be presented in this section and the simulation results will be shown in section 5.

In this thesis optimization methods developed in (Demeulenaere, o.a., July 11-13, 2007) will be focused on. But the difference is that instead of constructing optimal trajectory, we are going to construct optimal velocity inputs for the middle level velocity controller. The trajectory planner can predict \( n \) steps where \( n \) is a predefined number depends on the sampling time and servoing time period. In addition different motion constraints will be taken into account as well.

Besides these motion constraints, by using a so called ‘pyramid constraint’, the trajectory planner can also take care of the field of view problem, which is embedded in PBVS.

In order to let the trajectory planner work on linear dynamics, we will choose a certain kind of rotational velocity. That means we will optimize the linear velocity such that a certain serious of requirements will be fulfilled. The ‘linearization’ of the rigid body motion will be explained in section 4.2. Let’s first start with the classical visual servoing methods below.

4.1 Introduction of the classical visual servoing methods

4.1.1 General introduction of PBVS

Position-based control schemes (PBVS) use the pose of the camera relative to some reference to define the control variables \( s \), according to the visual servoing tutorial (CHAUMETTE, o.a., Jan 2009). Many solutions have been presented but the key ideas are similar. That is, first compute the pose of the object in the current coordinate frame. Second select a suitable \( s \) to compute the error term.

\[
e(t) = s - s^*.
\]  

(4.1.1)

Notice that different choices of \( s \) will determine the visual servoing schemes. The reason is \( s^* \) is directly related to the spatial velocity \( v = (v^T, w^T)^T \) and the relationship between these two is the following:
where $L_s \in \mathbb{R}^{6 \times 6}$ is well known as the interaction matrix related to $s$. Combining (4.1.1) and (4.1.2) the relationship between the spatial velocity and the error derivative is

$$\dot{e} = L_e \nu, \quad \text{(4.1.3)}$$

where $L_e = L_s$. Because $\nu$ is the input to the middle level control, one can choose

$$\nu = -\lambda L_e^{-1} e, \quad \text{(4.1.4)}$$

in this case the derivative of the error will be

$$\dot{e} = -\lambda e. \quad \text{(4.1.5)}$$

So by this choice the exponential decrease of the error will be ensured.

The final step of the PBVS is to construct the control scheme. The most popular approach is perhaps the velocity controller. Among different design methods, Malis (Malis, o.a., APRIL 1999) has come up with a good choice of the rotational velocity:

$$\omega = -\lambda \theta \mathbf{u}. \quad \text{(4.1.6)}$$

In this equation, $\theta \mathbf{u}$ means the rotation can be represented by a unit vector and an angle of revolution about that vector (Shown in the figure below). This axis-angle representation is very intuitive and easily understood. The biggest advantage of this representation comes from its derivative in equation (4.1.7) according to (Malis, o.a., APRIL 1999).

$$\frac{d(\theta \mathbf{u})}{dt} = \left[ I_3 - \frac{\theta}{2} [\mathbf{u}]_\times + \left( 1 - \frac{\sin\theta}{\sin^2\frac{\theta}{2}} \right) [\mathbf{u}]_\times^2 \right] \mathbf{w}. \quad \text{(4.1.7)}$$

Figure 4.1.0 Axis-angle representation of rotation.
If the rotational velocity is chosen as (4.1.6) the derivative of the rotational velocity will be $\theta \mathbf{u}$ itself up to a scalar. Therefore linear system theory could be applied to analyze the system dynamics.

To sum up, the involved in steps of the PBVS method are the following:

1. Calculating pose of the feature points given current image features and the 3D coordinates of the features in the desired camera frame.
   
   1.1) By using the current image features and the 3D feature coordinates in the desired camera frame, EPhP (Lepetit, o.a., 2008) method will be implemented to construct the 3D feature coordinates in the current camera frame. This part will be further explained in section 4.3.
   
   1.2) Given the 3D feature coordinates in both the desired and current camera frame, Horn’s method (Horn, o.a., March 25, 1988) could be easily implemented to compute the relative pose between the two sets of 3D feature coordinates.

2. Choose $\mathbf{s} = (\mathbf{t}_{cc,2dc}^{cc}, \mathbf{u} \theta)$, where $\mathbf{t}_{cc,2dc}^{cc}$ defined in the current camera frame stands for the translation of the origin of the current camera frame relative to the desired camera frame and $\mathbf{u} \theta$ represents the relative rotation.

Both the rotation axis $\mathbf{u}$ and the rotation angle $\theta$ in equation (4.1.6) come from the decomposition of the rotation matrix (Ma, o.a., 2003):

$$ \theta = \cos^{-1}\left(\frac{\text{trace}(\mathbf{R})-1}{2}\right) $$

$$ \mathbf{u} = \frac{1}{2 \sin \theta} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} . $$

(4.1.8)

3. According to this particular choice of control variable $\mathbf{s}$ and the rotational velocity $\mathbf{\omega} = -\lambda \mathbf{u} \theta$, we are controlling the relative pose directly. So the interaction matrix will be:

$$ L_e = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{I} \end{bmatrix} . $$

(4.1.9)

Finally, the simple control scheme will be:

$$ \mathbf{v} = -\lambda \mathbf{R}^T \mathbf{t}_{cc,2dc}^{cc} . $$

(4.1.10)

$$ \mathbf{\omega} = -\lambda \mathbf{u} \theta . $$

(4.1.11)

The stability of PBVS will be discussed shortly below. According to the visual servoing tutorial presented in (CHAUMETTE, o.a., Jan 2009) or 2-½ D method analyzed in (Malis, o.a., APRIL 1999) we should use Lyapunov analysis to access the stability of the closed loop system.

In particular, consider the candidate Lyapunov function defined by the square error form $L = \frac{1}{2} \| \mathbf{e}(t) \|^2$, whose derivative is given by:

$$ \dot{L} = \dot{\mathbf{e}}^T \mathbf{e} = \mathbf{e}^T L_e \dot{\mathbf{e}} = \mathbf{e}^T L_e \left(-\lambda \mathbf{L}_e^{-1} \mathbf{e}\right) = -\lambda \mathbf{e}^T L_e \mathbf{L}_e^{-1} \mathbf{e} . $$

(4.1.12)
According to equation (4.1.8) $L_e$ is nonsingular when $\theta \neq 2k\pi$, so in the ideal case $L_eL_e^{-1} = I_6$. If the estimation $\hat{L}_e^{-1}$ is not too coarse (4.1.12) will be smaller than zero, then global asymptotic stability of system could be obtained.

### 4.1.2 Embedded problems of PBVS

 Basically, PBVS mentioned in the previous section works well in certain scenarios. But the potential problems are the following:

1. Since there’s no control on the image plane, there is the possibility for image feature trajectories to get out of the image plane. If that happens, the control loop will break down. In this thesis, this problem is referred to as field of view problem. One example can be found in Figure 4.1.1. On the left hand of the figure, the image feature trajectories are about to cross the edge of the image plane.

If we have a look at Figure 4.1.2, we can easily see another problem, that is, the spatial velocity is absolutely proportional to the error term. This leads to very big velocity which may cause the saturation problem when the error is big enough. On the other hand, when the error is small, the convergence speed is really small. Of cause the long settling time is not wanted.
Concerning both the problems and the advantages of PBVS, optimal trajectory planner could be a possible solution. Because both the smooth motion constraints and the field of view constraint can be added to an optimization procedure as linear constraints, as long as we have a linear system with all the transition states defined as optimization variables. So the question is that how such a linear system can be obtained? A possible solution will be introduced in section 4.2.

4.1.3 General introduction of IBVS

Image-based visual servoing (IBVS) was first presented in (Wiess, o.a., August 26 1985) and recently reviewed by this tutorial (CHAUMETTE, o.a., Jan 2009). Basically it uses the image-plane coordinates to define the error term (4.1.1), so the visual input to the high level control scheme will be the image points directly. In this way, it has control in the image plane. But sometimes as we will see in section 5, certain relative rotation and translation could make the image feature trajectories out of the field of view even if we use IBVS.

Different from the interaction matrix (4.1.8) of PBVS, the interaction matrix of IBVS consists of corresponding part from each image feature point. Equation (4.1.13) describes how the derivative of the image point relates to the camera spatial velocity.

$$\dot{x}_i = \lambda L_x V_c = \begin{bmatrix} \frac{-1}{z_i} & 0 & \frac{x_i}{z_i} & x_i y_i & -(1 + x_i^2) & y_i \\ 0 & \frac{-1}{z_i} & \frac{y_i}{z_i} & 1 + y_i^2 & -x_i y_i & -x_i \end{bmatrix} V_c. \hspace{1cm} (4.1.13)$$

To form the complete interaction matrix for all the image feature points, we just need to pile up equation (4.1.13) of each image feature point in the following way:

$$L_x = \begin{bmatrix} L_{x_1} \\ L_{x_2} \\ \vdots \\ L_{x_N} \end{bmatrix}. \hspace{1cm} (4.1.14)$$
To control the six velocity outputs, at least three image points are needed to form the complete interaction matrix. According to the tutorial paper (CHAUMETTE, o.a., Jan 2009) considering the singularity of equation (4.1.14) and global stability, usually more than three image points will be used.

As long as the interaction matrix $L_x$ is obtained, the control scheme can be formed as equation (4.1.14), but the difference is now we need the Moore-Penrose Pseudo inverse of $L_x$, which is

$$L_x^+ = (L_x^T L_x)^{-1} L_x^T.$$  

In equation (4.1.13), $Z_i$ is the depth information of the corresponding image feature point, so either we do online pose estimation as in section 4.3 or we can just use $Z^*$, which is the desired depth or the depth of the feature points in the desired position according to Espiau’s work (Espiau, o.a., Jun 1992) and end in $L_x = L_x^*$. Notice that if $Z$ is chosen to be $Z^*$, we don’t need to update the depth information for each step. But as we can see in section 5, this choice leads to a very curly 3D camera trajectory. There is another compromise method mentioned in the tutorial (CHAUMETTE, o.a., Jan 2009), that is, $L_x^* = \frac{1}{2} (L_x + L_x^*)$. All the performances of these IBVS methods will be shown in section 5.4.

### 4.2 Linearization of the rigid body motion

One way of representing the rigid body motion is $g(t) \in SE(3)$, where $g(t)$ is given by (Ma, o.a., 2003):

$$g(t): R^3 \rightarrow R^3; \ X \rightarrow g(X), \quad (4.2.1)$$

in which $g(t)$ is equal to $(R, T)$ and its homogeneous representation is:

$$g(t) = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (4.2.2)$$

In this way, the rigid body motion can be represented as matrix multiplication. A chain of motion amounts to a chain of left multiplications of matrix $g(t)$. In equation (4.2.2), $x_1 x_2 x_3 x_5 x_6 x_7 x_9 x_{10} x_{11}$ together represent the rotation matrix $R$ and $x_4 x_8 x_{12}$ together represent the translation vector $T$.

Another point should also be clarified, that is, the derivative of the rotation matrix in equation (4.2.2) is given by (Ma, o.a., 2003):

$$\dot{R}(t) = \dot{\omega}(t) R(t_0) = -\lambda \theta \ddot{\theta} \hat{R}(t_0) = -\lambda \theta \begin{bmatrix} 0 & -u_3 & u_2 \\ -u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} R(t_0), \quad (4.2.3)$$

where $\omega = -\lambda u \theta$ is selected in equation (4.1.10). Notice form equation (4.2.3), if $R(t_0) = I$ at $t = t_0$, we obtain $\dot{R}(t) = \dot{\omega}(t_0)$. Hence around the identity matrix, a skew-symmetric matrix gives a first order approximation to a rotation matrix. So $\dot{R}(t)$ is equal to $e^{\omega t}$ under the condition that $\omega$ is fixed.

Finally, as we defined all the rotation and translation variables in equation (4.2.2), the derivative of equation (4.2.2) will be:
If we expand equation (4.2.4) according to
\[ R(t) = \mathbf{w}(t) R(t_0) \quad \text{and} \quad \mathbf{T}(t) = \mathbf{w}(t) \mathbf{T}(t_0) + \mathbf{v}(t), \]
equation (4.2.5) will be obtained:
\[ \dot{x}(t) = \begin{bmatrix} \ddot{R} & \dot{T} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{w} & \mathbf{v} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} R(t_0) \\ \mathbf{T}(t_0) \end{bmatrix}. \quad (4.2.5) \]

It’s a nonlinear system with 12 state variables \( x_i = 1, \ldots, 12 \) and 12 input variables \( \mathbf{w} \) and \( \mathbf{v} \). It is the 9 rotation input variables that make the system nonlinear, so the mission now is to somehow linearize equation (4.2.5).

One very interesting phenomenon could be easily observed if we have a look at the right hand side of Figure 4.1.1, which is a trajectory of the moving camera controlled by PBVS. We can find the trajectory is nearly an arc of a big 3D circle, which means the axis \( \mathbf{u} \) is almost fixed during the whole procedure. The reason is that in equation (4.1.10) the rotational velocity is chosen as \( \mathbf{w} = -\lambda \mathbf{u} \mathbf{\theta} \).

This interesting advantage will greatly simplify the problem if we always choose the rotational velocity as \( -\lambda \mathbf{u} \mathbf{\theta} \). This certain choice means the skew symmetric matrix of \( \mathbf{u} \) will be constant, so equation (4.2.5) will be almost a linear system except one scale variable, which is \( -\lambda \mathbf{\theta} \). Now equation (4.2.5) can evolve into:
\[ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = -\lambda \mathbf{\theta} \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_4 \\ -u_2 & u_4 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix}. \quad (4.2.6) \]

Where \( x_i = 1, 2, 3, 5, 6, 7, 9, 10, 11, 12 \) are defined by \( R(t_0) \) and \( x_i = 4, 8, 12 \) are defined by \( \mathbf{T}(t_0) \). If we rewrite equation (4.2.6) into the state space model, it will be:
\[ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \mathbf{A} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}. \quad (4.2.7) \]

According to the book (Ma, o.a., 2003), \( \xi \) is called a twist. The set of all twists is denoted by
\[ \text{se}(3) = \{ \xi = \begin{bmatrix} \mathbf{w} \\ v \end{bmatrix} \mid \mathbf{w} \in \text{so}(3), v \in \mathbb{R}^3 \} \subset \mathbb{R}^{4 \times 4}. \]
Actually a predefined $-\lambda \theta$ could be used to absolutely linearize equation (4.2.7). The fixed $-\lambda \theta$ and the fixed axis $u$ means during the visual servoing procedure the rotational speed will be constant such that the camera will rotate similarly as using PBVS, but with a constant rotational speed.

In this way, only the translational velocity $v$ will be controlled and optimized, so using the fixed $-\lambda \theta$ gives both the linearity and the inflexibility.

What also needs to be pointed out is that optimization always has an error residual. So actually the visual servoing task will be divided into two stages when using the new optimal trajectory planner. In the first stage, the middle level control will follow all the outputs from the trajectory planner. In the second stage, the high level visual servoing control scheme will switch to PBVS. According to simulation results in Figure 5.2.2.3, when it is switched to PBVS, because the error residual is very small, the remaining motion will be really tiny, which can’t bring in any trouble.

Let’s go back to the predefined value of $-\lambda \theta$. According to my simulation usually $-0.3 \theta$, where $\theta$ is the angle comes from the rotation matrix decomposition at the very beginning, will always be a good start.

4.3 3D pose estimation Using EPnP

Another crucial step in PBVS is the 3D pose estimation. In this thesis what needs to be done is to solve the Perspective–n-Point problem. The aim of PnP problem is to determine the relative rotation and translation given the camera intrinsic parameters and a set of n correspondences between 3D points and their 2D projections (Lepetit, o.a., 2008).

In this thesis the 3D pose estimation is done by using a non-iterative solution called EPnP method (Lepetit, o.a., 2008) with better accuracy and lower computational complexity than the non-iterative state of art methoded. Particularly, EPnP is $O(n)$ for $n \geq 4$, which is in contrast to the state-of-the-art methods that are $O(n^5)$ or even $O(n^8)$.

Assume that there is a set of $n$ feature points whose 3D coordinates are known in the world coordinate system and whose 2D image projections are also known. Most solutions to the PnP problem attempt to solve for the depths of the feature points in the camera coordinates frame.
directly, however, in EPnP method, the coordinates of the feature points are expressed as a weighted sum of virtual control points. In this thesis, general non-coplanar feature points are considered to be the case, then 4 control points are needed according to (Lepetit, o.a., 2008). In this way, compared with other methods with big \( n \), there are only 4 unknown depths. This is the key to the efficient implementation.

### 4.3.1 Points parameterization

Denote the \( n \) feature points whose 3D coordinates are known in the world coordinate frame as \( p_i \in \mathbb{R}^3, i = 1, \ldots, n \). Similarly, the 4 virtual control points can be denoted as \( c_j \in \mathbb{R}^3, j = 1, \ldots, 4 \). Each feature point can be expressed as a weighted sum of the virtual control points:

\[
p_i = \sum_{j=1}^{4} \alpha_{ij} c_j \quad \text{with} \quad \sum_{j=1}^{4} \alpha_{ij} = 1. \tag{4.3.1}
\]

Where \( \alpha_{ij} \) is homogeneous barycentric coordinate, which is uniquely defined and can easily be estimated. Notice that equation (4.3.1) holds for the 3D feature coordinates in both the world coordinate frame and the camera coordinate frame.

Although a good choice of the virtual control points \( c_j \) proposed by (Lepetit, o.a., 2008) is the principal directions of the data, but in theory they could also be chosen arbitrarily. In this thesis, \( c_j \) will be chosen as:

\[
\begin{align*}
C_1 &= \text{centroid}, \\
C_2 &= C_1 + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \\
C_3 &= C_1 + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \\
C_4 &= C_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.
\end{align*} \tag{4.3.2}
\]

Combining with equation (4.3.1), we can use least square method to calculate the barycentric coordinate for each 3D feature:\

\[
A = P^T(C_h^T)^+ = P^T(C_h C_h^T)^{-1} C_h. \tag{4.3.3}
\]

Where \( A \in \mathbb{R}^{n \times 4} \) is the matrix consists of barycentric coordinates of each feature point, \( P \in \mathbb{R}^{n \times 4} \) is the homogeneous coordinates matrix of the feature points and \( C_h \in \mathbb{R}^{4 \times 4} \) is the virtual control points matrix in the world frame expressed in homogeneous coordinates.

### 4.3.2 Solution as weighted sum of Eigenvectors

We now derive the matrix \( M \) in whose kernel the solution must lie given the 2D projections of the feature points. Since in this thesis the camera intrinsic parameters matrix is assumed to be identity matrix, we have:

\[
w_i \begin{bmatrix} u_i \\ v_i \end{bmatrix} = p_i = \sum_{j=1}^{4} \alpha_{ij} c_j^f = \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^f \\ y_j^f \\ z_j^f \end{bmatrix}. \tag{4.3.4}
\]

Where \( w_i \) is the depth of each feature point in the camera frame, \( u_i \) and \( v_i \) are the 2D projection of the feature point and \( c_j^f \in \mathbb{R}^3 \) is the virtual control point in the camera frame. Because \( u_i \) and \( v_i \) are given, \( \alpha_{ij} \) has been calculated in equation (4.3.3), the unknown parameters of equation (4.3.4) are the 3 virtual control point coordinates \( c_j^f, j = 1, \ldots, 4 \) in the camera frame and the unknown depth \( w_i \).
Notice that the last row of equation (4.3.4) implies that $w_i = \sum_{j=1}^{4} a_{ij} z^c_j$. Substituting this expression in the first two rows yields two linear equations for each reference point:

$$
\begin{align*}
\sum_{j=1}^{4} a_{ij} x^c_j - a_{ij} u_i z^c_j &= 0 \\
\sum_{j=1}^{4} a_{ij} y^c_j - a_{ij} v_i z^c_j &= 0 .
\end{align*}
$$

(4.3.5)

Notice that in equation (4.3.5), the unknown depth $w_i$ disappears. Hence, by concatenating it for all $n$ feature points, a linear system of the form $Mx = 0$ could be generated, where $x = \left[ c_1^T \ c_2^T \ c_3^T \ c_4^T \right]^T$ is a 12-vector made of the unknowns, and $M$ is a $2n \times 12$ matrix generated by arranging the coefficients in equation (4.3.5).

Then the solution lies in the null space of the matrix $M$ and can be expressed as:

$$
x = \sum_{i=1}^{N} \beta_i v_i .
$$

(4.3.6)

Where the set $v_i$ are the columns of the right-singular vectors of $M$ corresponding to the $N$th null singular values of $M$. They can be found efficiently as the null eigenvectors of matrix $M^T M \in R^{12 \times 12}$.

### 4.3.3 Choosing the right linear combination

In theory, due to different camera models, $N$ in equation (4.3.6), which is the dimension of the null space of $M^T M$, could be from 1, 2, 3 or 4. But in this thesis we have perspective camera model so $N$ is equal to one, which means equation (4.3.6) could be simplified as:

$$
x = \beta v .
$$

(4.3.7)

Since the distances between the virtual control points retrieved in the camera coordinate frame should be equal to the ones computed in the world coordinate frame, $\beta$ could be computed in equation (4.3.8):

$$
\beta = \frac{\sum_{(i,j) \in [1,4]} \| c_i^w - c_j^w \| \| c_i^c - c_j^c \|}{\sum_{(i,j) \in [1,4]} \| c_i^c - c_j^c \|} .
$$

(4.3.8)

Where $c_i^w$ are virtual control point calculated in (4.3.2), $c_j^c = v_i$ is the sub-vector of $v$ in equation (4.3.7) corresponding to the virtual control point coordinate in the camera frame.

Then we can calculate the rotation and translation by using $c^c \in R^{3 \times 4}$ and $c^w \in R^{3 \times 4}$ as inputs to Horn’s method (Horn, o.a., March 25, 1988).

### 4.3.4 Simulation result

The simulation result of the EPNP method is shown in Figure 4.3.4.1. One can see that the estimation is robust to Guassian noise.
Later in this thesis, we will use this pose estimation method for both PBVS and the new optimal trajectory planner.

### 4.4 Motion optimization

The idea of the new optimal trajectory planner for visual servoing is inspired from the work done by researchers at KUL (Demeulenaere, o.a., July 11-13, 2007). What they have done is to optimize the input $u$, which is the input to the position controller, such that the output position appears to be a dynamically optimal spline. According to the same paper (Demeulenaere, o.a., July 11-13, 2007), splines constitute an attractive parameterization because they allow the designer:

1. Locally control the curve
2. Directly impose its smoothness.

Local control implies the possibility to concentrate the effects of a change within a specific region of the curve. Smoothness is quantified here through the number $m$ of continuous derivatives: a curve belonging to $C^m_{[0,T]}$ has derivatives that are continuous on the interval $[0,T]$ up to the $m$th-order derivative. It is a common rule of thumb that motion trajectories should be at least $C^2$ (acceleration must be continuous).

With the same idea, the design procedure is the following:

1. Choose a suitable $\theta$ such that system (4.2.7) becomes a linear state space model. Then find a proper sampling time and discretize the model.
(2) Since visual servoing task must be fulfilled so the first objective should be that the relative rotation tends to identity, which is equivalent to say states $x_1$, $x_6$ and $x_{11}$ converges to 1 respectively.

States $x_1$, $x_2$, $x_3$, $x_5$, $x_6$, $x_7$, $x_9$, $x_{10}$ and $x_{11}$ will always form a rotation matrix together, because the properties of the rigid body motion have been embedded in A and B matrix in equation (4.2.7) and equation (4.2.7) will be added in the optimization procedure as hard constraints.

(3) The second objective is to minimize the relative translation between the current and desired position, which means states $x_4$, $x_8$ and $x_{12}$ converges to 0 respectively.

With two objectives in hand, weights are needed to specify their priorities, so finally the objective equation looks like:

$$\min_x \left[ \sum_{i=1,6,11} |x_i - 1| + \lambda \sum_{i=8,12} |x_i| \right].$$

First, if the relative rotation matrix is not identity matrix, the relative position will be quite big even if the relative translation is close to zero, so always weight $\lambda$ in equation (4.4.1) is smaller than one, which means the minimization of the rotation has higher priority. Second in equation (4.4.1), we are working on one-norm regularization, which could be transformed to linear optimization. According to spline optimization (Demeulenaere, o.a., July 11-13, 2007), due to the use of large number of prediction steps, there may exist many optimization solutions, one-norm regularization can select better solution.

(4) Now the underlying linear system has 12 state variables, 3 control inputs and one predefined $\theta$. Similar with (Demeulenaere, o.a., July 11-13, 2007), we are going to optimize the state variables in each time step by choosing the control inputs during the whole procedure. So in total if the time step is $n$, then we need to define $12n$ optimization variables for the states and $3(n-1)$ optimization variables for the inputs.

Notice that in the last control step the states should reach the desired states already so no more control is needed, we only need $3(n-1)$ optimization variables for the inputs instead of $3n$.

(5) The system transition equation (4.2.7) should be added as hard constraint such that properties of the rigid body motion will be embedded into the result.

According to equation (4.2.5), $\ddot{R} = \dot{\omega}R$, which means the translational velocity will not affect the rotation! So the objective function (4.4.1) could be simplified into equation (4.4.2). Basically, we need to define the optimization procedure in the following way:

**state variables**: $x_{\text{states}}(12)$ and **input variables**: $u_{\text{input}}(3)$

$$\min_x \left[ \sum_{i=4,8,12} |x_i| \right]$$

$$\tag{4.4.2}$$

\(^2\) We need to define the optimization variables first.
Subject to:

\[ x_{\text{states}} = A \cdot \theta \cdot x_{\text{initial}} + B \cdot u_{\text{input}} \]

Where A and B are picked up from equation (4.2.7), \( \theta \) is the predefined \( \theta \) in step one and \( x_{\text{initial}} \) is the pose estimated by EPnP (Lepetit, o.a., 2008) and Horn’s algorithm (Horn, o.a., March 25, 1988) as explained in section 4.3.

Notice that in this case \( x_{\text{states}} \) and \( u_{\text{input}} \) are one dimensional, which means (4.4.2) is only one step optimization. If we want n steps we just need to define 12n variables for \( x_{\text{states}} \) and 3(n − 1) variables for \( u_{\text{input}} \) as shown in an example in Figure 4.6.1.

(6) In order to achieve the smooth motion, several optimization constraints are needed. Since the rotational speed is chosen as fixed, so we don’t need to worry about the rotational motion. The translational motion will be the key point.

Because the input is the translational speed and the underlying system is in discrete time, so we will define the acceleration and the derivative of the acceleration by differencing method. Then we can add extra objectives like minimizing the one norm of the sum of acceleration or something similar to obtain the smooth motion.

For example, we can define the acceleration as:

\[ a = \frac{u_{\text{input}}(k+1) - u_{\text{input}}(k)}{T_s}. \]

Where \( T_s \) is the sampling time. Then we can add more constraints or objectives into (4.4.2). Motion performances could be found in Figure 5.2.2.1, Figure 5.3.2.2 and Figure 5.3.2.5 in section 5.

(7) Lastly, the optimization constraint for keeping the features within the field of view will be introduced in section 4.5 and together with this constraint, an optimization example could be found in Figure 4.6.1.

4.5 Pyramid constraint for keeping features within field of view

As we mentioned in Section two, there are many methods to keep the object features within the camera field of view. But in this thesis a new approach will be employed to meet the requirements. All needs to be done is to add constraints in the optimization procedure (4.4.2), which makes this approach very straight forward and flexible.

Basically, there is another point of view to interpret the conditions that we need to keep the image features within the field of view. On the left hand side of Figure 4.5.1 one can observe that the focus and image plane construct a pyramid or a corn in 3D space, which can be easily defined in Y-Z and X-Z plane shown on the right hand side of Figure 4.5.1.
The next step is to specify the constraint in the optimization procedure such that the 3D coordinates in the current camera frame of the feature points are inside this ‘Pyramid’.

Since in each time step, we defined all the transition positions as states variables vector $\mathbf{x}_t \in \mathbb{R}^{12}$ as indicated in equation (4.4.2) and we know the 3D coordinates $P_{dc} \in \mathbb{R}^{3 \times n}$ of the feature points in the desired frame, so the 3D coordinates $P_{cc} \in \mathbb{R}^{3 \times n}$ of the feature points in the current frame can be expressed as

$$P_{cc} = g_{dc,2cc}^{dc}(t)P_{dc}.$$  \hfill (4.5.1)  

Then what needs to be done is very simple, we can just take out the $x, y$ and $z$ coordinates vectors out of $P_{cc} \in \mathbb{R}^{3 \times n}$ as a linear combination of states variables vector $\mathbf{x}_t \in \mathbb{R}^{12}$ and specify their geometrical relation found in Figure 4.5.1, which is:

$$z \geq \alpha x;$$  
$$z \geq -\alpha x;$$  
$$z \geq \alpha y;$$  
$$z \geq -\alpha y;$$  \hfill (4.5.2)  

Notice that due to our specific choice, parameter $\alpha$ in equation (4.5.2) will be 2. In this thesis, we name this linear constraint as pyramid constraint, and it can be added to the optimization problem (4.4.2) as linear constraints.

Notice that in equation (4.5.1), $g_{dc,2cc}^{dc}(t)$ is a $4 \times 4$ matrix with the first three rows formed by reshaping vector $\mathbf{x}_t$ and the last row is equal to vector $[0\ 0\ 0\ 1]$. Indicated by the subscript, $g_{dc,2cc}^{dc}(t)$ is the transition from the desired camera frame to the current camera frame. Similarly indicated by the super script, $g_{dc,2cc}^{dc}(t)$ is defined in the desired camera frame.
So the last feature of the trajectory planner appears, that is, the states vector \( x_t \) represent the elements in the transition matrix \( g_{dc,2cc}^{de} \). In this way, the optimization results, from system (4.4.2) are the optimal translational velocities. Together with the selected rotational velocities, they are actually defined as \( v_{dc,2cc}^{de} \) because of \( g_{dc,2cc}^{de} \).

Then the question is that these velocities, defined in the desired camera frame, are velocities moving from the desired camera frame to the current camera frame. But actually what we need are the velocities defined in the current camera frame moving from the current camera frame to the desired camera frame.

One may argue that if we make the optimization variables \( x_t \) to represent transition matrix \( g_{cc,2dc}^{de} \), the optimization result will be \( v_{cc,2dc}^{de} \) directly. This is absolutely right, but in this case, if we want to add the pyramid constraint we need to calculate \( g_{dc,2cc}^{de}(t) \) by equation (4.5.3).

\[
g_{dc,2cc}^{de}(t) = g_{cc,2dc}^{de}(t)^{-1} = \begin{bmatrix} R_{cc,2dc}^{de} & T_{cc,2dc}^{de} \\ 0 & 0 & 1 \end{bmatrix}.
\]  (4.5.3)

Because we use optimization variables to represent \( g_{cc,2dc}^{de} \), obviously equation (4.5.3) will involves in nonlinearities. So we have to do it in an inverse way.

In order to calculate \( v_{cc,2dc}^{de} \) given \( v_{dc,2cc}^{de} \) and \( g_{dc,2cc}^{de} \), we need to use the following method according to page 37 in Ma’s book (Ma, o.a., 2003).

First let’s sort out the relationship between \( g(t) \in SE(3) \) and it’s derivative \( \dot{g} \), which is defined in equation (4.2.5) . We know the coordinates \( X(t) \) of a point \( p \in E^3 \) relative to a moving camera is a function of time \( t \):\n
\[
X(t) = g(t)X(t).
\]  (4.5.4)

Then the velocity of this point \( p \) relative to the (instantaneous) current camera frame is:

\[
\dot{X}(t) = \dot{g}(t)X(t).
\]  (4.5.5)

If we substitute \( X(t) \) by \( X(t)g(t)^{-1} \) and use the notation of twist, we get:

\[
\dot{\xi} = \dot{g}(t)g(t)^{-1} \in se(3).
\]  (4.5.6)

So equation (4.5.5) could be expressed as: \( \dot{X}(t) = \dot{\xi}(t)X(t) \).

Then, let’s go back to the coordinates of a point in the current camera frame relative to the desired camera frame. \( Y(t) = g_{dc,2cc}^{de}(t)X(t) \). We can compute the velocity in the current camera frame \( v_{cc,2dc}^{de} \) by using the following:

\[
g_{dc,2cc}^{de}(t) = \frac{1}{g_{cc,2dc}^{de}(t)}.
\]  (4.5.7)

\[
\dot{g}_{dc,2cc}^{de}(t) = -\frac{g_{cc,2dc}^{de}(t)}{g_{cc,2dc}^{de}(t)}.
\]  (4.5.8)
\[ Y(t) = \mathbf{v}_{cc,2dc}^c Y(t) = \mathbf{\dot{v}}_{cc,2dc}^c Y(t) \]  
(4.5.9)
\[ \mathbf{\dot{\xi}}_{cc,2dc}^c = \mathbf{g}_{cc,2dc}^{dc} (-\mathbf{\dot{\xi}}_{dc,2cc}^c) \mathbf{g}_{cc,2dc}^{dc}^{-1} = \mathbf{g}_{2dc,2dc}^{cc} (-\mathbf{v}_{dc,2cc}^c) \mathbf{g}_{cc,2dc}^{cc}^{-1} \]  
(4.5.10)

So it is clear that the optimal velocities \( \mathbf{v}_{dc,2cc}^{dc} \) are easy to be converted to \( \mathbf{v}_{cc,2dc}^c \), which are going to be feed into the middle level control directly.

### 4.6 Summary of the new optimal trajectory planner

Below is an example which consists of Pseudo Matlab code and cvx toolbox code to show the optimization procedure with all the constraints and optimization variables. In section 5, the new optimal trajectory planner is constructed by using exactly the same procedure.

```matlab
variable xstates(12,n)\(^4\)
variable uinput(3,n-1)\(^5\)

state_final = xstates(:,2:end);
state_initial = xstates(:,1:end-1);

motion = [xstates(4,:); xstates(8,:); xstates(12,:)]\(^6\);

velocity = motion(:,2:end) - motion(:,1:end-1) / sampling time;
acceleration = velocity(:,2:end) - velocity(:,1:end-1) / sampling time;
jerk = acceleration(:,2:end) - acceleration(:,1:end-1) / sampling time;

min_x [\lambda_1 \sum_{j=4,8,12} |x(j, end)| + \lambda_2 \sum_{jj=1}^{n-3} |jerk_{jj}|]
```

---

3. We use \( \mathbf{\dot{\xi}} \) instead of \( \mathbf{v} \) to keep the consistency with the concept of twist.
4. Each column of \( x_{states} \) represents the elements of the transition matrix \( g_{dc,2cc}^{dc}(k) \) at time step \( k \). All the \( n \) columns of \( x_{states} \) starts from left to the right represent \( n \) consecutive transition matrices \( g_{dc,2cc}^{dc}(k), k = 1, \ldots, n \).
5. As explained in section 4.3, the rotational velocity \( \mathbf{w} = -\lambda u \mathbf{\theta} \) is predefined, so only the translational velocity \( \mathbf{v} \) is going to be optimized. Here \( \mathbf{v} \) is the translational part of \( \mathbf{v}_{dc,2cc}^{dc} \) and \( u_{input} \) are defined for \( n \)-steps exclude the last step.
6. Here we pick out the states which represent the translation to define the actual translational velocity and etc by differencing method.
subject to:

\[ \text{state}_{\text{final}} - AA^* \text{state}_{\text{initial}} - BB^* u_{\text{input}} = 0; \]

\[ x_{\text{states}}(:,1) = [R(1,:), T(1), R(2,:), T(2), R(3,:), T(3)]^T; \]

\[
Z \geq 2 * X^8; \\
Z \geq -2 * X; \\
Z \geq 2 * Y; \\
Z \geq -2 * Y; \\
[ u_{\text{input}} \leq 0.1 * \max(T); ]_{10} \\
[ u_{\text{input}} \geq -0.1 * \max(T); ]_{10} \\
\lfloor \text{Jerk} \leq 1 \rfloor ; \\
\lfloor \text{Jerk} \geq -1 \rfloor ;
\]

Figure 4.6.1 Overview of the optimization algorithm of the trajectory planner.

Notice that the new optimal trajectory planner is using an optimization procedure, so there is always a residual of the error, although the error is trivial according to the simulation result in section 5. In order to eliminate the error residual, the visual servoing procedure will be divided into two stages when using the new optimal trajectory planner as we talked before. In the first stage the velocity controller obtained in section 3.4 will track the planned velocities out of the new optimal trajectory planner in each sampling time, in the second stage PBVS will take care of the error residual.

\[ X = P_{cc}(1,:), Y = P_{cc}(2,:) \text{ and } Z = P_{cc}(3,:). \ P_{cc} \text{ is specified in each time step as:} \]

\[ P_{cc} = g_{dc,2cc}(t) P_{dc} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\
x_5 & x_6 & x_7 & x_8 \\
x_9 & x_{10} & x_{11} & x_{12} \\
0 & 0 & 0 & 1 \end{bmatrix} P_{dc} \]

7 Actually, we need to discretize system (4.2.7) by pre-wrapping method or Tustin’s method. \( AA \text{ and } BB \) are \( A \text{ and } B \) matrices of the discretized system. By posing this hard constraint, the properties of the rigid body motion will be embedded into \( x_{\text{states}} \).

8 The rotation matrix \( R \) and the translation vector \( T \) are defined in the desired camera frame from desired camera frame to the current camera frame.

9 \( X = P_{cc}(1,:), Y = P_{cc}(2,:) \) and \( Z = P_{cc}(3,:) \). \( P_{cc} \) is specified in each time step as:

10 This constraint aims to limit the control input. Details about different constrains could be found in section 6.
5 Simulation environment and results

In this section, simulation results of PBVS and the new optimal trajectory planner will be presented. Based on these comparisons, analysis will be provided after each result. In the next section other possibilities of objective functions/constraints will be drawn.

5.1 Simulation toolbox setup

Basically all the simulations are written in a MatLab M file. Most of the code is provided by the authors, although we referred to some other people’s work.

5.1.1 Camera visualization tool

The visualization code is developed from the Epipolar Geometry Toolbox from (MARIOTTINI, o.a.). For example the visualization of the camera, image plane and coordinate frames. All the assumptions of the camera could be found in section 1.1.1. In order to show the performance, one visualization example is shown in Figure 5.1.1.

Figure 5.1.1 Example of the Visualization Code.

Where the green camera represents the current position and the red camera represents the desired position. The red dots are assumed to be the feature points in world frame. On the right hand side are the desired/current image planes respectively.
5.1.2 Optimization tool and visualization work

First, the optimization tool used in the simulation is called *cvx* toolbox (Grant, o.a., 2010), which is a Matlab-based modeling system for convex optimization. *cvx* toolbox turns Matlab into a modeling language, allowing constraints and objectives to be specified using Matlab expression syntax. Here is an example from (Grant, o.a., 2010):

```matlab
m = 20; n = 10; p = 4;
A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1);

cvx_begin
variable x(n)
minimize (norm(A*x - b,2))
subject to
    C*x == d;
    norm(x,inf) <= 0.4;

cvx_end
```

![Figure 5.1.2 example of cvx toolbox.](image)

Basically the above code segment randomly generates a constrained two norm minimization problem and solves it.

5.2 Demonstration of basic functionalities

In this section, we will see the performances of both PBVS and the new optimal trajectory planner. Three different sets of feature points will be used to test the two methods. In order to show their basic performances, all the sets of feature points in this section are selected such that the PBVS method will not get in trouble with keeping features within field of view.

5.2.1 Performances of PBVS

In this section, we are going to test PBVS developed from section 4.1.1. Gain $\lambda$ in equation 4.1.10 and 4.1.11 will be chosen as $\lambda = 5$ such that desired velocities out of PBVS are fast enough to position the camera within 1.5 sec. Sampling time will be $T_{zp} = 0.1 \text{sec}$, because we want PBVS to have better performance.

Below in table 5.2.1.1 there are three sets of feature points on the image plane. Relative translation and rotation can also be found on the right hand side. Notice that the relative rotation should not be too large, otherwise the features will be out of the field of view of the camera.
Table 5.2.1.1: Different sets of feature points.

<table>
<thead>
<tr>
<th>Points number</th>
<th>Relative rotation</th>
<th>Relative Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set one</td>
<td>$x: \frac{\pi}{5}; y: -\frac{\pi}{6}; z: \frac{\pi}{8}$</td>
<td>[10 10 10]</td>
</tr>
<tr>
<td>Set two</td>
<td>$x: \frac{\pi}{4}; y: -\frac{\pi}{3}; z: \frac{\pi}{8}$</td>
<td>[18 10 10]</td>
</tr>
<tr>
<td>Set five</td>
<td>$x: \frac{\pi}{5}; y: -\frac{\pi}{9}; z: \frac{\pi}{7}$</td>
<td>[18 10 25]</td>
</tr>
</tbody>
</table>

First we use feature points in set one as input to PBVS, in Figure 5.2.1.1, we can see both the translational velocities and the rotational velocities converge. Besides the camera motion information, in Figure 5.2.1.2 both of the image feature trajectories and the 3D camera trajectory are presented.
Performance of PBVS under feature points set two could be found in Figure 5.2.1.3 and Figure 5.2.1.4 respectively. Basically the 3D trajectories of the camera show similar behavior in these two cases as in Figure 5.2.1.2.

Due to the specific rotational velocity $\omega = -\lambda u \theta$, we can see that all the 3D camera trajectories shown in Figure 5.2.1.2 and Figure 5.2.1.3 are arcs belong to big virtual circles. This is kind of good, because the actual distances that the camera travelled are nearly the shortest trajectories, which is the Euclidean distance between the initial camera position and the desired camera position.
5.2.2 Performances of the new optimal trajectory planner

In this section we are going to show the performance of the new optimal trajectory planner developed from Figure 4.6.1. The computation time of the optimization procedure is around 0.45 sec depends on different scenarios. We can find some examples in Table 5.2.2.1.

<table>
<thead>
<tr>
<th>Example</th>
<th>Computation time</th>
<th>Relative rotation</th>
<th>Relative Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
<td>0.483603100000096</td>
<td>(x: \frac{\pi}{6}; y: \frac{\pi}{8}; z: \frac{\pi}{10})</td>
<td>[18 25 18]</td>
</tr>
<tr>
<td>two</td>
<td>0.452402900000038</td>
<td>(x: \frac{\pi}{5}; y: \frac{\pi}{6}; z: \frac{\pi}{8})</td>
<td>[18 20 15]</td>
</tr>
<tr>
<td>three</td>
<td>0.468003000000067</td>
<td>(x: \frac{\pi}{4}; y: \frac{\pi}{3}; z: \frac{\pi}{8})</td>
<td>[18 10 10]</td>
</tr>
<tr>
<td>four</td>
<td>0.421202699999981</td>
<td>(x: \frac{\pi}{5}; y: \frac{\pi}{9}; z: \frac{\pi}{7})</td>
<td>[18 10 25]</td>
</tr>
</tbody>
</table>

We chose the sampling time as \(T_{sp} = 0.2 \text{ sec}\), the time period as \(T = 3 \text{ sec}\) and the rotational velocity as \(\omega = -0.3\theta\). Since the translational velocity is left to the trajectory planner to optimize, we will not set anything for it.

In order to keep the consistency, simulation of the new optimal trajectory planner also starts with feature points in set one. In Figure 5.2.2.2 we can see that the trajectories are kind of similar. But the camera motion shown in Figure 5.2.2.1 is much smoother.

Figure 5.2.2.1 Camera motion information in the first stage using the trajectory planning method under feature points set one.
Section five

Figure 5.2.2.2 2D image feature trajectories and 3D camera trajectory in the first stage using the trajectory planning method under feature points set one.

As we mentioned many times, because trajectory planner is obtained by optimization method (section 4.5), so the remaining error term need to be handled in the second stage after the actuator have executed the planned velocities. For feature points in set one, camera motion information and trajectory information in the second stage could be founded in Figure 5.2.2.3 and Figure 5.2.2.4 respectively. Notice that although the image feature trajectories on the image plane in Figure 5.2.2.4 still count like one half or one third of the feature trajectories on image plane in Figure 5.2.2.2, the velocity in the second stage will be really trivial according to the values in Figure 5.2.2.3.

Figure 5.2.2.3 Camera motion information in the second stage using the trajectory planning method under feature points set one.
Section five

Figure 5.2.2.4 2D image feature trajectories and 3D camera trajectory in the second stage using the trajectory planning method under feature points set one.

Extra tests of the trajectory planner method with the same sets of feature points for PBVS could be found in Figure 5.2.2.5.

In Figure 5.2.2.5, one can see that the image feature trajectories made smaller turns compared with Figure 5.2.1.3. The reason could be found in the camera motion information comparison between Figure 5.2.2.1 and Figure 5.2.1.1. Basically by using PBVS the velocity is proportional to the error term, so the curves of the feature points will make curly turns as the decrease of the error term. On the other hand in Figure 5.2.2.1, we can see that both the translational velocity and the rotational velocity are quite smooth. That’s the reason why the image feature trajectories ended in a relative smooth curve in Figure 5.2.2.5.
5.3 Comparison with PBVS

Due to the limit of space, we are going to only present the comparison result of one set of feature points. However, this set of feature points will pose hard conditions on the visual servoing algorithm.

<table>
<thead>
<tr>
<th>Points number</th>
<th>Relative rotation</th>
<th>Relative Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set six</td>
<td>$x: \pi/4, y: -\pi/5, z: \pi/7$</td>
<td>$[30 \ 30 \ 30]$</td>
</tr>
</tbody>
</table>

For the new optimal trajectory planner, we will keep the settings in section 5.2, that is, the sampling time $T_{sp} = 0.2$ sec, the time period $T = 3$ sec and the rotational velocity $w = -0.3 \theta u$. But for PBVS we will use $\lambda = 1.5$ instead of $\lambda = 10$ as the convergence gain in equation (4.1.10) and (4.1.11) and $T_{sp} = 0.2$ sec instead of $T_{sp} = 0.02$ sec. Because we want the velocities out of PBVS and the trajectory planner to be on the same level such that we can have a better comparison. Meanwhile the gain $\lambda$ should not be too small, otherwise the translation and rotation angle can’t converge to zero within 3 sec. Based on the advantages of the new optimal trajectory planner the performance comparison will be twofold.

First, as mentioned many times before, if the feature points could not be kept in the field of view, the control loop will be broken down. So in section 5.3.1 comparisons about field of view constraint between PBVS and the new optimal trajectory planner will be presented.

Second, the smooth camera motion is also important. On one hand steep changes in the acceleration will lead to hard constraints in implementation on hard ware, on the other hand smooth motion also makes the image feature trajectories smoother. In section 5.3.2, comparison about camera motion between PBVS and the new optimal trajectory planner will be presented.

After the comparison with PBVS, we will also compare the new optimal trajectory planner with IBVS in section 5.4.

5.3.1 Performance on keeping image features within field of view

On the left hand side of Figure 5.3.1.1, one can see that by using PBVS the image feature trajectories will be out of field of view in between the initial position and the desired position.
On the contrary, in Figure 5.3.1.2, one can see that on the left hand side, the image feature trajectories will be kept in the field of view and the trajectories are more like straight lines which are better than the U shape trajectories on the left hand side in Figure 5.3.1.1.

Now we will continue the comparison with PBVS about the camera motion in the next section. Basically one can see that by using the trajectory planning method, we can obtain a much smoother camera motion.
5.3.2 Camera motion performance

In Figure 5.3.2.1, the portrait of the camera motion using PBVS is self-explaining. The magnitude of the translational acceleration and the magnitude of the translational acceleration (Jerk) are really big.

![Figure 5.3.2.1 Camera motion information using PBVS under feature points set six.](image)

However, in Figure 5.3.2.2, first one can see that the rotational velocity is fixed, so we don’t need to worry about its acceleration, second, although the maximum velocity is similar with the maximum velocity in Figure 5.3.2.1, but both the translational acceleration and the translational Jerk are quite smaller and smoother compared with the result in Figure 5.3.2.1.

![Figure 5.3.2.2 Camera motion information in the first stage using the trajectory planning method under feature points set six.](image)
We can also have a look at the convergence of translation and rotation angle by using PBVS and the new optimal trajectory planner.

Figure 5.3.2.3 Convergence of translation and rotation angle using PBVS under feature points set six.

Figure 5.3.2.4 Convergence of translation and rotation angle using the trajectory planning method under feature points set six.

In order to further show the difference in camera motion, we are going to use feature points set five in Table 5.2.1.1. Due to the challenging relative translation condition, the requirement on camera motion performance will be high.
Based on the comparison results we can see the advantages of the trajectory planning method obviously. In order to further address these advantages, comparison with IBVS will be shown in the following section.

### 5.4 Comparison with IBVS

In this section, the trajectory planning method will be compared with different image-based visual servoing (IBVS) methods. Basically IBVS has control in the image plane directly, so it
will not get image features out of the field of view. But the trajectory planning method will show its advantages in both the 3D camera trajectory and the image feature trajectories.

Since feature points set six in section 5.3 posed the harshest conditions on PBVS as most of the image features will get out of the image plane. So we use feature points six to test IBVS. Basically according to three choices of the interaction matrices, that is, $L_X$, $L_X^+$, and $\bar{L}_X$, we will test them one by one. Similarly, we will also choose the gain $\lambda = 1$ in equation (4.1.13) and $T_{sp} = 0.2$ sec as the sampling time.

This is the performance of the IBVS when $L = L_X$, we can see that on the right hand side of Figure 5.4.1, the 3D trajectory of the camera is similar with the performances in Figure 5.3.1.2. But on the right hand side of Figure 5.4.1, the image feature trajectories are all within the image plane although some of them made a small turn compared with Figure 5.3.1.2.

Figure 5.4.1 2D image feature trajectories and 3D camera trajectory using IBVS ($L = L_X$) under feature points set six.
Section five

Figure 5.4.2 2D image feature trajectories and 3D camera trajectory using IBVS ($L = L_x$) under feature points set six.

This is the performance of the IBVS when $L = L_x$. It is clear that because of the fixed depth information, the IBVS control scheme will make big turn in both 3D camera trajectory and in the image feature trajectories. Notice that the image feature trajectories almost get out of the image plane.

Figure 5.4.3 2D image feature trajectories and 3D camera trajectory using IBVS ($L = \tilde{L}_x$) under feature points set six.

When the compromise IBVS when $L = \tilde{L}_x$ is used, we can see that the 3D camera trajectory is similar with Figure 5.4.1 and Figure 5.3.1.2, the image feature trajectories perform similarly with IBVS when $L = L_x$. But either IBVS when $L = L_x$ or IBVS when $L = \tilde{L}_x$ have bad image feature trajectories compared with the trajectory planning method.

From Figure 5.4.4 to Figure 5.4.6, the camera motion performances when using the three IBVS methods are presented. They showed similar performances with PBVS.
Figure 5.4.4 Camera motion information using IBVS ($L = L_x$) under feature points set six.

Figure 5.4.5 Camera motion information using IBVS ($L = L_x^*$) under feature points set six.
Section five

Figure 5.4.6 Camera motion information using IBVS ($L = \mathcal{L}_x$) under feature points set six.

From Figure 5.4.7 to Figure 5.4.9, the convergence of translation and rotation angles by using different IBVS methods are also presented as a reference.

Figure 5.4.7 Convergence of translation and rotation angle using IBVS ($L = \mathcal{L}_x$) under feature points set six.
To sum up, by using IBVS, although all the image feature trajectories are within the image plane, they made more or less U turn, which is bad compared with the straight image feature trajectories by using the trajectory planning method. In addition, all the camera motion performances have the same problem as PBVS.
6 Analysis of the new optimal trajectory planner

Since the optimal trajectory planner is using an optimization procedure, so based on the performance and comparison results in section 5, analysis on optimization objectives, constraints, different initial conditions and other possible procedure of the optimization will be presented in this section.

6.1 Analysis of objectives/constraints

In section 5, the objective function being used is:

\[
\min_x \left[ \lambda_1 \sum_{j=4,8,12} |x(j, \text{end})| + \lambda_2 \sum_{jj=1}^{n-3} |\text{Jerk}_{jj}| \right] \tag{6.1.1}
\]

Where \( \text{Jerk}_{jj} \) is a function of variable \( x \). As explained in section 4.5, this objective function aims to make the rotation converges to identity, make the translation converges to zero and minimize the sum of the jerk, which is the derivative of the acceleration. Another intuitive objective function is, instead of minimizing the translation, we can minimize the difference between the current coordinates of the feature points and the desired coordinates of the feature points. Now the objective function turns into equation (6.1.2).

\[
\min_x \left[ \sum_{i=1}^{n} |X - X^*| + \sum_{i=1}^{n} |Y - Y^*| + \sum_{i=1}^{n} |Z - Z^*| + \lambda_2 \sum_{jj=1}^{n-3} |\text{Jerk}_{jj}| \right] \tag{6.1.2}
\]

Using the same definition as in Figure 4.6.1, \( X = P_{cc}(1,:) \), \( Y = P_{cc}(2,:) \) and \( Z = P_{cc}(3,:) \). \( P_{cc} \) is specified in each time step as:

\[
P_{cc} = g(t)P_{dc} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{dc}.
\]

\( P_{dc} \) is the desired coordinates of the feature points known in advance. By using this new objective function the performance of the trajectory planner method is shown from Figure 6.1.1 to Figure 6.1.4.

On the left hand side of Figure 6.1.1, we can see that the image feature trajectories made a ‘V’ turn, while on the right hand side of Figure 6.1.1 the 3D camera trajectory is similar with the performance of the original trajectory planning method.

Although, in Figure 6.1.2 we can see that the 3D camera motion in the first stage is good and in Figure 6.1.3 the 2D image feature trajectories and 3D camera trajectory in the second stage are trivial, but in Figure 6.1.4 the camera motion in the second stage is bad in terms of the magnitude of the acceleration and its derivative, the jerk.
Section six

Figure 6.1.1 2D image feature trajectories and 3D camera trajectory in the first stage using the trajectory planning method with objective function (6.1.2) under feature points set six.

Figure 6.1.2 Camera motion information in the first stage using the trajectory planning method with objective function (6.1.2) under feature points set six.
Figure 6.1.3 2D image feature trajectories and 3D camera trajectory in the second stage using the trajectory planning method with objective function (6.1.2) under feature points set six.

Figure 6.1.4 Camera motion information in the second stage using the trajectory planning method with objective function (6.1.2) under feature points set six.

Another possibility of the objective function is instead of minimize the sum of the jerk, we can minimize the sum of the derivative of the jerk such that the output will be smooth for $C^3$. Now the objective function will be equation (6.1.3).

$$
\min_x \left[ \lambda_1 \sum_{j=4,8,12} |x(j, \text{end})| + \lambda_2 \sum_{jj=1}^{n-3} |\text{jerk}_{jj}| \right] 
$$

(6.1.3)

Where the derivative of the \textit{jerk} is defined in equation (6.1.4) similarly as the definition of \textit{jerk} in Figure 4.6.1.
\[
Jerk = \frac{\text{Jerk (:2:end) } - \text{Jerk (:1:end - 1)}}{\text{sampling time}}
\] (6.1.4)

By using objective function (6.1.3), the camera motion information when using the trajectory planning method is shown in Figure 6.1.5 and Figure 6.1.6.

In Figure 6.1.5, we can see that compared with Figure 6.1.2 the translational velocity on the left up corner is even smoother, which is the result of the minimization of the derivative of the jerk. But in Figure 6.1.6, we can see that in the second stage the camera motion is bad indicated by the magnitude of the acceleration and its derivative.

Figure 6.1.5 Camera motion information in the first stage using the trajectory planning method with objective function (6.1.3) under feature points set six.

Figure 6.1.6 Camera motion information in the second stage using the trajectory planning method with objective function (6.1.3) under feature points set six.
Besides the objective function, other modifications could be done in the constraint parts, for example if the derivative of the acceleration, the jerk, is directly related to the input voltage of the actuator, we can add upper and lower limit on the jerk: \( \text{jerk} \leq \text{limit}_1, \text{jerk} \geq \text{limit}_2 \) and put into the constraints in Figure 4.6.1.

Constraint (6.1.5) aims to limit the translational velocity. This is constraint could be easily fulfilled sometimes, for example, when \( T \), the relative translation, is very big, this constraint means nothing.

\[
\begin{align*}
\mathbf{u}_{\text{input}} &\leq 0.1 \times \max(T) ; \\
\mathbf{u}_{\text{input}} &\geq -0.1 \times \max(T) ;
\end{align*}
\]

(6.1.5)

\[
\begin{align*}
x_{\text{states}}(4,:) &\leq T(1) \\
x_{\text{states}}(4,:) &\geq -T(1) \\
x_{\text{states}}(8,:) &\leq T(2) \\
x_{\text{states}}(8,:) &\geq -T(2) \\
x_{\text{states}}(12,:) &\leq T(3) \\
x_{\text{states}}(12,:) &\geq -T(3)
\end{align*}
\]

(6.1.6)

By using constraint in equation (6.1.6), we can make sure that the relative translation is always smaller than the initial translation. As stated in section 4.5 \( T \) is the relative translation estimated in advance.

If we want the relative translation to be always decreasing, we can simply use constraint in equation (6.1.7) instead of equation (6.1.6):

\[
\begin{align*}
x_{\text{states}}(4,1:end) &\leq x_{\text{states}}(4,1:end - 1) \\
x_{\text{states}}(4,:) &\geq x_{\text{states}}(4,1:end - 1) \\
x_{\text{states}}(8,:) &\leq x_{\text{states}}(8,1:end - 1) \\
x_{\text{states}}(8,:) &\geq x_{\text{states}}(8,1:end - 1) \\
x_{\text{states}}(12,:) &\leq x_{\text{states}}(12,1:end - 1) \\
x_{\text{states}}(12,:) &\geq x_{\text{states}}(12,1:end - 1)
\end{align*}
\]

(6.1.7)

But when this constraint is applied, we can see the performance in Figure 6.1.7. Basically although the relative translation is always decreasing, the 3D camera trajectory is too curly and the image feature trajectories are swinging left and right during the first stage.
Section six

Figure 6.1.7 2D image feature trajectories and 3D camera trajectory in the first stage using the trajectory planning method with extra constraint in equation (6.1.7) under feature points set six.

Constraints like in equation (6.1.8) and (6.1.9) can also be used to limit the translational velocity. Due to different requirements in practice, we can always use different values to constraint the optimization result.

\[
\begin{align*}
\begin{bmatrix}
velocity \\ velocity
\end{bmatrix} &\leq 
\begin{bmatrix}
50 \\ -50
\end{bmatrix}, \\
\begin{bmatrix}
Jerk \\ Jerk
\end{bmatrix} &\leq 
\begin{bmatrix}
1 \\ -1
\end{bmatrix}.
\end{align*}
\]

(6.1.8) (6.1.9)

6.2 Analysis on different initial conditions

For an optimization procedure, the initial conditions matter a lot. In this thesis the involved initial conditions are the following:

1. The sampling time of the visual servoing control scheme \( T_{sp} \).
2. The optimization time period \( T \).
3. The rotational velocity in equation (4.1.11), \( w = -\lambda \theta u \).

Let’s first talk about the sampling time \( T_{sp} \), which is crucial for application. Basically the sampling time \( T_{sp} \) can’t be too small. Because according to section 3.4, we need to use the middle level velocity controller to track the desired velocity from either the high level PBVS control scheme or the optimization result from the new optimal trajectory planner, if the sampling time is too small maybe the velocity controller can’t track the desired velocity within the sampling time. On the other hand, the sampling time should not be big, otherwise the control will be too coarse. In this thesis, the sampling time is chosen as \( T_{sp} = 0.2 \text{sec} \).

Second, let’s talk about the optimization time period \( T \). Basically in order to get the time optimal servoing result \( T \) should be as small as possible.

Lastly, a good guess of the rotational speed is also important or maybe crucial, it can neither too be bog, nor too small.

In a word, all the three initial conditions are sensitive to initial guess, in this thesis we choose \( T_{sp} = 0.2 \text{ sec} \), \( T = 3 \text{ sec} \) and \( w = -0.3 \theta u \). But if we want to get better result, bisection search is a possible solution for all these three values. For example, one possible bisection framework is shown in Figure 6.2.1.

\[
\begin{align*}
T_{now} &= T_{initial}; \\
T_{test} &= T_{now}; \\
\text{While } (abs(T_{now} - T_{test}) \leq \text{limit}) \\
\quad &\quad \text{[status] = Trajectory_planner}(T_{test}) \\
\quad &\quad \text{if } ((\text{strcmp}(	ext{status},'Solved')||\text{strcmp}(	ext{status},'Inaccurate/Solved'))) \\
\end{align*}
\]
Many other search algorithms can be implemented to find the optimal initial conditions, but we will not discuss them in this thesis.

### 6.3 Multi-stage trajectory planning

Basically the entire visual servoing process could be divided into several stages. Which means the new optimal trajectory planner could be used several times for one visual servoing task. For example, decrease one fourth of the initial error term could be the objective in each stage. Then combining the results of all the stages will give the optimal trajectory for the whole process.

In this thesis, the new optimal trajectory planner will only be used once. In the second stage PBVS will take care of the remaining part of the servoing task. Because the rotational velocity is fixed during the first stage, so basically according to the Figures in section 5.2 the 3D camera trajectory is an arc in the first stage. The fixed radius of the arc lacks flexibility. But division of the visual servoing task means the whole arc will be divided into several smaller arcs, and then the actual trajectory that the camera needs to travel will be better.

One potential problem when using the new optimal trajectory planner is the transition between the first stage and the second stage. According to section 5.2.2, the camera motion is good in either the first stage or the second stage, but the new optimal trajectory planner can’t help much with the transition between these two stages, basically we need to run the risk that both the translational velocity and the rotational velocity will run a steep decrease.
7 Conclusions

In this thesis, the rigid body dynamics, the overall control structure and the new optimal trajectory planner inspired by spline optimization (Demeulenaere, o.a., July 11-13, 2007) are discussed and simulated. Performances of the new optimal trajectory planner are fully shown in section 5.

Compared with other visual servoing methods, the advantages are, first, 100% keeping the image features within the field of view due to the hard constraints in the optimization procedure; secondly, relatively straight 3D camera trajectory; thirdly, as indicated in section 6, the flexibility. We are free to choose different combinations of objectives and constraints to optimize the camera motion. Concerning the initial values, we can use bisection or other search methods to find the optimality.


16. **Ma Yi [o.a.]** An invitation to 3-D vision [Bok]. - [u.o.]: Springer, 2003.


