Diploma Thesis IST-91

Nonparametric System Identification and Control for Periodic Error Correction in Telescopes

Edgar Klenske

Supervisors: Philipp Hennig (Max Planck Institute for Intelligent Systems)
Gregor Goebel (Institute for Systems Theory and Automatic Control)
Stefan Harmeling (Max Planck Institute for Intelligent Systems)

University of Stuttgart

Institute for Systems Theory and Automatic Control

Prof. Dr.–Ing. F. Allgöwer

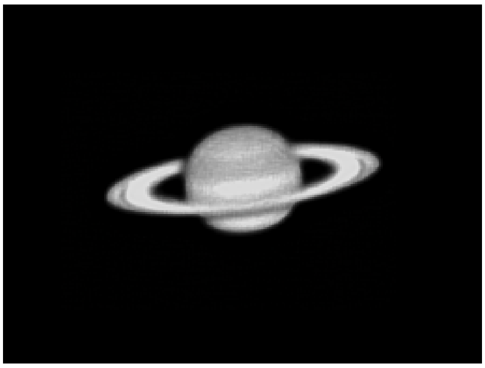
2012-07-19

Abstract

High quality photographs of astronomical objects require a photographic device to remain steady relative to the sky. Because Earth rotates, the telescope has to follow a circular trajectory as precisely as possible. The machinery built for this purpose is never absolutely perfect: Imprecisions in the worm drives of standard telescope mounts cause a *periodic* error in the pointing direction, resulting in image blur. Because the dynamics that lead to this error are often not known in advance, they cannot be modelled explicitly. Therefore, this problem is normally addressed with a model-free controller. In this work, Gaussian process regression with a periodic kernel is employed to identify the unknown dynamical system. As a nonparametric method, this regressor can learn even complicated nonlinearities and does not require the user to provide a parametric model. Experiments in simulation and hardware show improvements over standard approaches to autoguiding and periodic error correction. Beyond the application to telescopes, this study serves as an example for the utility of Gaussian processes in providing a flexible modelling class for various types of hardware.

Zusammenfassung

Um hochwertige photographische Aufnahmen astronomischer Objekte anfertigen zu können, sind Teleskope nötig, die sich relativ zum Himmel nicht bewegen. Weil die Erde sich dreht, muss das Teleskop einer kreisförmigen Trajektorie möglichst genau folgen. Die mechanischen Apparate, die für diesen Zweck konstruiert werden, sind nie perfekt: Ungenauigkeiten in den Schneckengetrieben, welche in den meisten Teleskop-Montierungen benutzt werden, führen zu einem *periodischen* Fehler in der Ausrichtung. Dies äußert sich dann in Form von verschwommenen Bildern. Weil die dem Fehler zugrundeliegende Dynamik oftmals nicht vorab bekannt ist, kann sie nicht explizit modelliert werden. Daher begegnet man diesem Problem normalerweise mit einem modellfreien Regler. In dieser Arbeit wird ein Gauß-Prozess mit einem periodischen Kern benutzt, um das unbekannte dynamische System zu identifizieren. Mit dieser nichtparametrischen Methode können auch komplizierte Nichtlinearitäten gelernt werden, der Benutzer muss dazu kein parametrisches Modell entwickeln. Experimente sowohl in simulierter als auch in realer Umgebung zeigen Verbesserungen gegenüber den normalen Ansätzen für die automatische Nachführung und die Korrektur des periodischen Fehlers. Neben der Anwendung auf Teleskope ist diese Arbeit ein Beispiel für die Anwendbarkeit von Gauß-Prozessen für die flexible Modellierung verschiedener Arten von Hardware.



“It is cool to use Gaussian processes
for what Gauss used to do once:
Tracking objects in the sky.”

NEIL LAWRENCE, APRIL 2012

Acknowledgements

First of all, I want to thank the whole Empirical Inference Group at the Max Planck Institute in Tübingen and especially Bernhard Schölkopf for lots of brilliant ideas and hours of valuable discussions. Also, I really enjoyed the opportunity to meet many inspiring people at the Machine Learning Summer School and to test my work on a perfect site for astronomical observations.

Thanks to Verena Klenske, Daniel Kauker and Kathleen Misajon for proof-reading parts of my thesis.

I want to thank my parents for their patience during the last few months and the invaluable support during the whole time at the university.

It took a lot of energy working on the project and writing this thesis, and I have to thank Verena Neumann for constantly supporting and encouraging me.

The greatest thank goes to my supervisors: To Philipp Hennig for teaching me scientific work and Gaussian processes, for much constructive criticism, for pouring lots of red ink into my drafts, and for keeping me on track during the whole time. To Gregor Goebel for keeping things running at the university and for hints on the control part. And to Stefan Harmeling for his ideas and help with the image processing part.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Telescope Guiding Problem

Aside from the optical fidelity of the telescope, the main challenge in astronomical photography is to achieve a stable image over the time of exposure. This problem is more challenging than one might think at first sight: To counteract Earth's rotation, the telescope has to follow a circular path across the sky. Otherwise, the light of the stars gets distributed to circular trails like in Figure 1.1.

Even for amateur-class telescopes and cameras, the precision of this path has to be on the order of a thousandth of a degree for several minutes, if individual stars should not be blurred over several pixels in the resulting image. Mechanical imprecisions in the drives used to create this motion make it hard to achieve this level of fidelity at a reasonable price. The worm drives that are used in most of the telescope mounts to provide the needed gear reduction show an error which recurs on every rotation of the worm. This error can be seen on long exposure images, as for example in Figure 1.2.

1.2 Traditional Autoguiding and Periodic Error Correction

Traditional approaches to this problem mostly rely on tedious adjustment protocols performed by the user: The error of the worm drive has to be measured over several periods and post-processed accurately. The processed error curve is used for open-loop-control and has to be stored in the telescope mount's software or firmware. It needs to be carefully

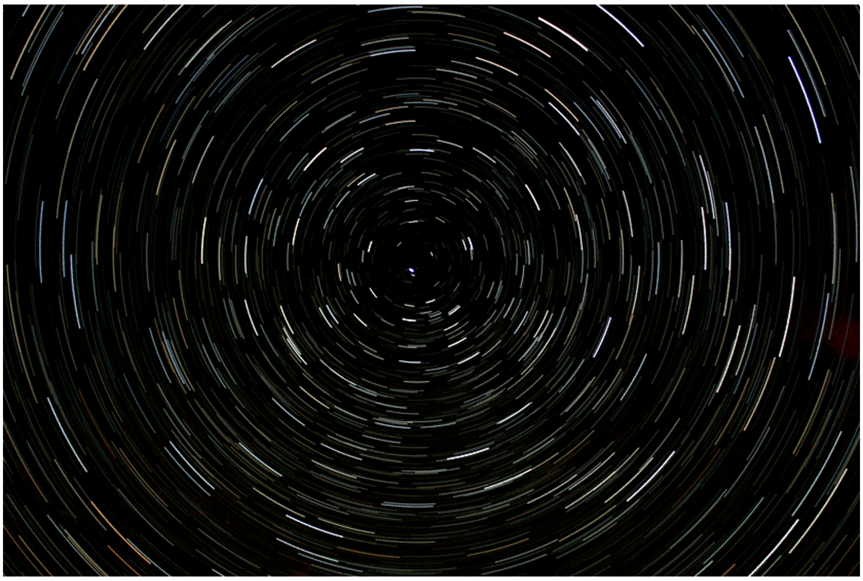


Figure 1.1: Star Trails

This figure shows a combined image of several long time exposures with a still standing camera. Image printed with kind permission of Robert Vanderbei[1].

synchronized to get good results. Because this is a lot of work, many amateur astronomers do not use the information about the periodic error at all and control their telescopes only on the measured error with an *autoguiding system*: A second camera is used to measure the error and a simple controller acts on the telescope's motors to keep the error small.

1.3 Controlling Unknown Dynamical Systems

Classic control relies on prior knowledge about the controlled system. An open-loop-controller needs the system's full dynamics and error inputs to work well. It fails if the system behaves nondeterministically or is affected by unmeasured errors. A closed-loop-controller in contrast is more robust: Because it uses the system's output for control, it can also handle certain nondeterministic effects. But also the closed-loop-controller works only with a sufficient amount of information about the system.

Controlling *completely* unknown dynamical systems is a hard problem. Every controller has to be based on assumptions about its environment. Even a learning controller has to assume that it is possible to learn at all. If it is impossible to learn *anything* from the past about the future, trying to do so is pointless.

To achieve good control performance, the system's parameters have to be known. This

[1] `https://www.princeton.edu/~rvdb/images/other/LaPalma.html`

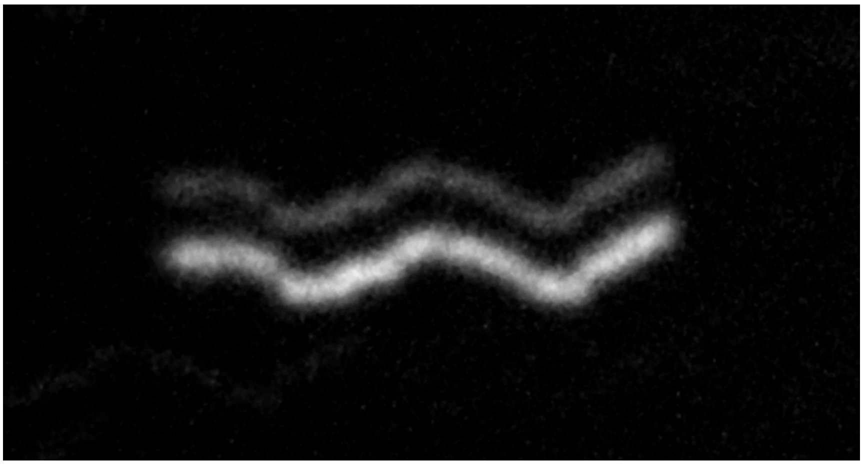


Figure 1.2: Visualized Gear Error

The periodic gear error (vertical) is visualized through slow horizontal drift. This image was combined from several 20 s exposures. It corresponds to 15 min overall exposure time and was made with the experimental setup described in Chapter 6.

involves a lot of accurate measuring before the controller can be designed. What happens if those parameters are not known or cannot easily be measured?

Most controllers only work well if the step time is short enough and the measurements can be made with a high temporal resolution. This gets even more problematic if high control precision is needed. What can be done if a fast measurement is not possible?

One answer to such problems is to identify the system's behaviour by a regression method and control upon it. The advantage over a classic controller is that this approach generalizes: Not only one known system can be controlled, but also other systems that show similar behaviour with different parameters. Thus, this kind of controller can save a lot of design time: The controller can be constructed for a whole class of systems and can learn the individual parameters, it does not have to be newly designed for each single system.

In this work, a controller is built upon this idea: *Creating a controller that learns about the system while controlling it and that uses the learned information to improve the control performance.* This is done by working on a real problem, control of the pointing direction of a telescope. This application shows some problems which are hard to solve with simple controllers: The dynamics of the mechanical system are not fully known, the measurements are relatively slow and high precision is needed.

1.4 The Research Paradigm of this Work

This thesis presents a new approach to cope with this problem: After modelling the telescope system (Chapter 2), a light-weight and precise image registration method is developed to measure the error from star images (Chapter 3). Having a working measurement, a periodic kernel is used with Gaussian process regression to construct an effective nonparametric system identification method for periodic error correction problems (Chapter 4). This provides a model for a nonlinear predictive control algorithm (Chapter 5). While an experimental evaluation on an astronomical setup is offered (Chapter 6), the results of this work generalize to other control problems with periodic errors in which parametric models are not readily available.

Chapter 2

Problem Setting

2.1 The Telescope System

Telescopes for astrophotography are normally attached to a mechanical device, a telescope *mount*. The mount is used to hold the telescope still for photography or to move the telescope when the user wants it to point to another direction. As shown in Figure 2.1, such a mount normally has two electrically movable axes: *right ascension* and *declination*. Two axes are necessary to be able to point in any desired direction. If the telescope is aligned properly, there is no coupling between these two axes. It is therefore possible to consider them as separate systems. In this work, the focus lies only on one axis, the right

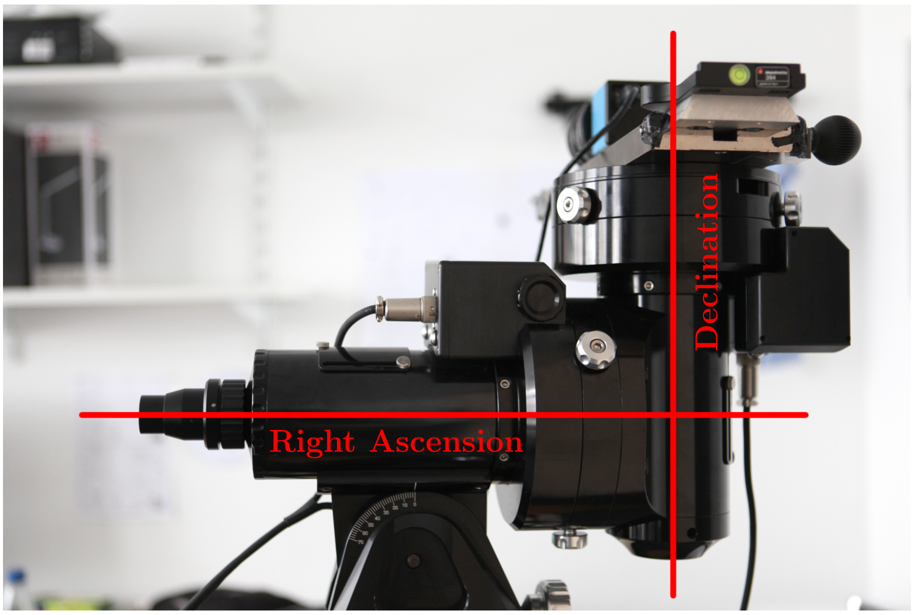


Figure 2.1: An Exemplary Telescope Mount

A telescope mount has two electrically movable axes, marked in red in the image. These axes are orthogonal.

ascension, which compensates for Earth's rotation. This axis has to move one full turn in approximately 24 hours, which is called *sidereal speed.* Because this is much movement, the right ascension axis is affected by the periodic error of the worm drive.

Even if this compensation is active, an imperfect alignment of the "right ascension" axis with Earth's rotation axis often causes drift effects. With sufficient effort for the alignment, those effects get small and slow compared to the periodic error. Hence, they are omitted for the modelling.

2.1.1 Overview & Sub-Systems

Figure 2.2 gives a schematic overview on components of a standard telescope system. It consists of four main parts:

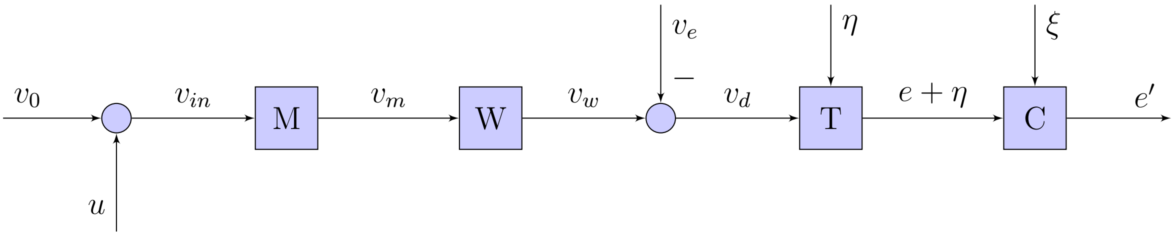


Figure 2.2: Overview of the Uncontrolled Telescope System

This figure shows the relationships between the four sub-systems motor (M), worm drive (W), telescope (T) and camera (C). For details on the sub-systems and meaning of the symbols, see page 6f..

Motor (M), an actuator.

$$v_m = g(v_{in}) = g(v_0 + u(t)) \tag{2.1}$$

The motor speed v_m is dependent on the input v_{in}. The function $g(\cdot)$ represents the motor's characteristic curve. It can be nonlinear, but has to be monotonic. v_0 is the input that is meant to compensate for Earth's rotation. u is the control input of the system, it slows down or speeds up the motor.

Worm Drive (W), a nonlinear dynamical system with one internal state.

$$v_w = k_w \cdot (1 + f(x)) \cdot v_m = k_w \cdot \left(1 + f\left(\int_0^t v_m(\tau) \cdot d\tau\right)\right) \cdot v_m \tag{2.2}$$

The output-speed of the gear v_w is ideally the motor speed v_m multiplied by the gear's transmission ratio k_w, but it is additionally affected by the periodic error $f(x)$. This effect has its origin in the worm gear: The transmission ratio is proportional to the lead of the worm, which is never perfectly constant. $f(x)$ is a nonlinear and *unknown* function of the worm position x, but it is nevertheless periodic with an upper and lower bound. The function $f(x)$ is a central part of this work: In Chapter 4 it is subject to a regression method in order to reach better control performance.

Telescope-Mount (T), an integrator.

$$e = \int_0^t v_d(\tau) \cdot d\tau + \eta = \int_0^t (v_w(\tau) - v_e) \cdot d\tau + \eta \tag{2.3}$$

As the telescope detects only the difference between Earth's real movement v_e and the movement of the gear v_w, the mount is an integrator for the difference speed v_d. Because the absolute difference between telescope and sky should ideally be zero, this integration directly leads to the error e. There is also random atmospheric movement η in the signal, which complicates further considerations.

Camera (C), a sensor for the pointing direction.

$$e' = e + \xi \tag{2.4}$$

Although the measurement is involved (as shown in Chapter 3), its effects can be modelled as noise, which will be approximated as Gaussian.

Figure 2.3 shows the complete system with controller K and the mathematical description for each block. Both noise terms η and ξ are combined to a new term ζ for simplification.

2.1.2 Simplifications to the Model

In order to simplify the system's description, the difference speed v_d needs further investigation: With Equations 2.1 and 2.2, the complete term for v_d is

$$v_d = k_w \cdot \left(1 + f\left(\int_0^t g(v_0 + u(\tau)) \cdot d\tau\right)\right) \cdot g(v_0 + u(t)) - v_e \tag{2.5}$$

The telescope mount already has a constant motor input v_0 to compensate for the effect of v_e. Assuming the standard speed v_0 was chosen correctly, the difference speed should be

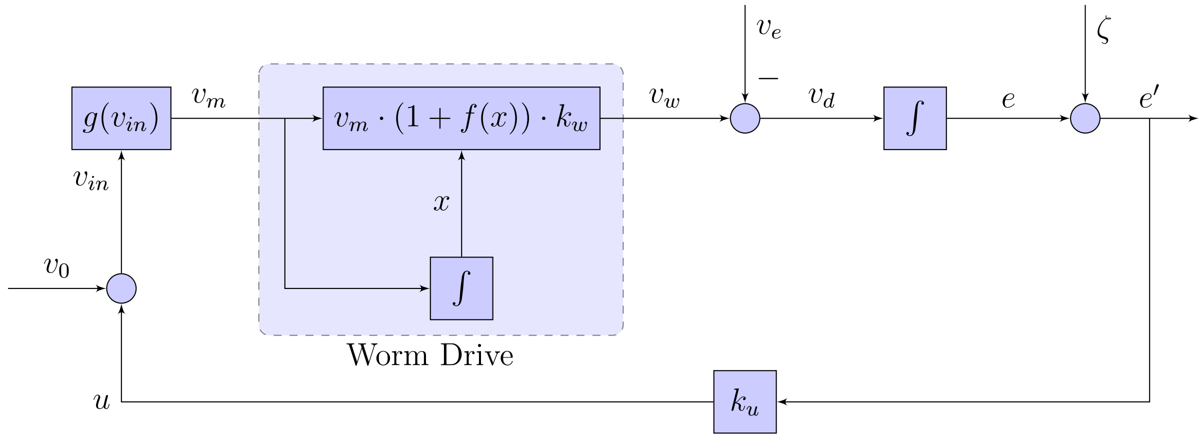


Figure 2.3: Overview of the Controlled Telescope System

This block diagram shows the mathematical background of the functional blocks. For details to the functions and the meanings of the signals, see text.

zero ($v_d = 0$) when no periodic error is present ($f(x) = 0$) and without any control input ($u = 0$). This results in

$$v_e = k_w \cdot g(v_0) \tag{2.6}$$

which can be inserted into Equation 2.5 to obtain

$$v_d = k_w \cdot (1 + f\,(x)) \cdot g(v_0 + u(t)) - k_w \cdot g(v_0) \tag{2.7}$$

A first order Taylor expansion of $g(v_0 + u)$ is used to get

$$v_d \approx k_w \cdot \underbrace{g(v_0)}_{\text{constant}} \cdot f(x) + k_w \cdot (1 + f(x)) \cdot \underbrace{k_g \cdot u}_{\text{linear term}} \tag{2.8}$$

where k_g is the slope $\left.\frac{\mathrm{d}g(v_0+u)}{\mathrm{d}u}\right|_{u=0}$ of the linear term.

Note that the gear function f depends on the control signal u, but at the same time acts as a scaling factor on u. This would create a recursion in the control algorithm, which makes the calculation of the optimal solution intractable. To ease this problem, the periodicity of f is taken into account. From the periodicity follows

$$E[f(x)] = 0 \tag{2.9}$$

where $E[f(x)]$ is the expected value of the function $f(x)$ at a random location. This implies that also $E[k_g \cdot u]$ is zero, because $k_g \cdot u$ is meant to compensate for the effect of $f(x)$. Because the periodic error is usually small compared to the nominal speed $g(v_0)$, $f(x) \ll 1$

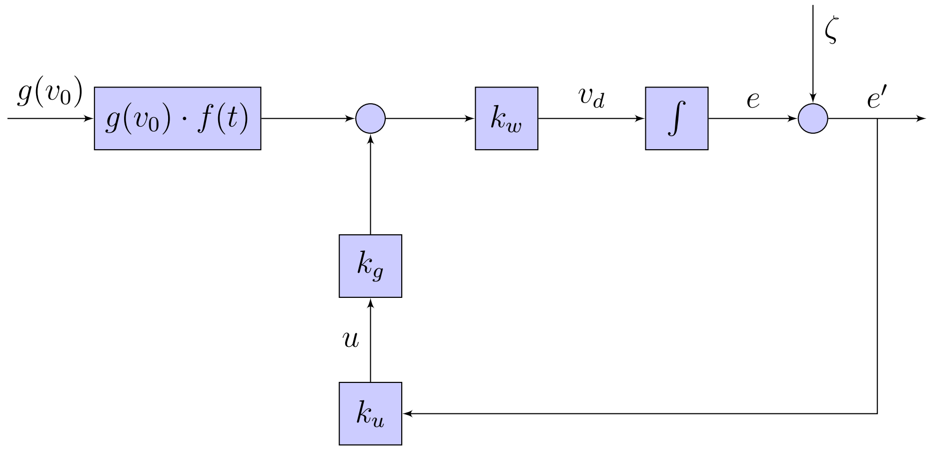


Figure 2.4: Overview of the Simplified Telescope System

From Figure 2.3: The effects of v_0 and v_e cancel each other out. The feedback path with controller K is isolated from $f(t)$.

can be considered. This gives

$$v_d = k_w \cdot g(v_0) \cdot f(t) + k_w \cdot k_g \cdot u \tag{2.10}$$

This simpler model is depicted in Figure 2.4. It shows the decoupling of the gear function $f(\cdot)$ from the controller. The resulting feedback-loop has a simple structure, which makes it easy to design a controller.

2.2 Pointing Accuracy & Error Sources

2.2.1 Hardware Limits

In order to build a reasonable control system, the hardware has to be checked for limitations. It is sensible to keep them in mind when defining a desired accuracy for the controller.

The Telescope has an optical resolving power of 0.38 arcs. [Mea]

The Imaging Camera in this setting has a resolution of 0.43 arcs[1]. [Canb]

The Guiding Camera has usually lower resolution and a smaller telescope as the main camera. In this setting it has a resolution of 2.40 arcs[1]. [The]

Of course, these values are specific to the used hardware, but other setups in the same price range (upper amateur class) should show numbers of the same order of magnitude.

[1] See Chapter 6 for derivations of these numbers.

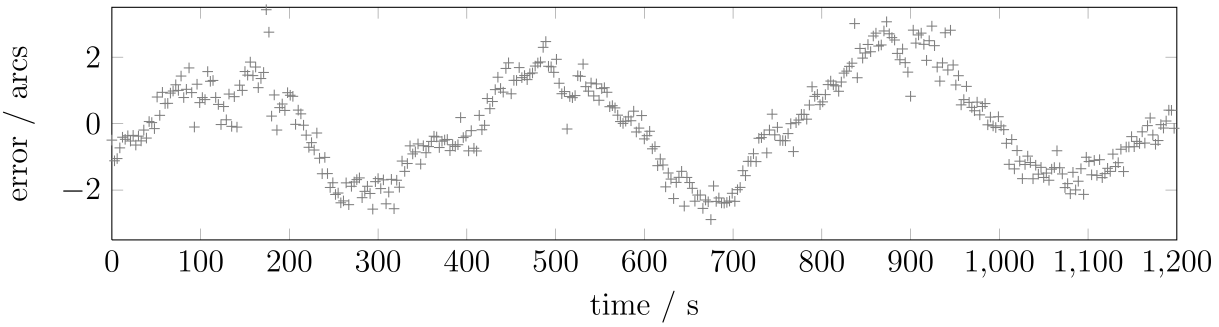


Figure 2.5: The Uncontrolled System's Error

This figure shows a typical measurement of the error in the uncontrolled system. The static drift is corrected for by subtracting the linear component.

The strongest limitation in resolution is the overall limit for the whole setup. As described in Chapter 3, the measuring resolution of the guiding camera can be increased using post-processing. This resolution enhancement cannot be used for the main camera, because the intended use is imaging, which works directly on the pixel information. Thus, the limiting factor in this setup is the resolution of the used imaging camera. Therefore, the objective for the controller is to keep the error below 0.43 arcs.

2.2.2 Error Sources

The Worm Drive shows errors as for example in Figure 2.5. Three turns of the worm are visible and show the periodic structure of the data. The amplitude is around 2.5 arcs, therefore the peak-to-peak error of the mount is approximately 5 arcs.

The Atmosphere is constantly in motion. Therefore, the light that travels through it, is subject to distortions. The magnitude of this effect, the *seeing*, is around 1 arcs under good conditions. Only a few places show better conditions [MVV97].

The Measurement adds noise to the real error. This noise has a mean of 0.02 pixel with a standard deviation of 0.01 pixel. For further details, see Chapter 3.

2.3 Conclusion

Because the Earth is rotating, the stars are moving – from the observers point of view – with a constant velocity around the sky. This is compensated in modern telescope mounts, but the mechanical part is not perfect: It shows periodic pointing errors.

In order to analyse the telescope system, a block diagram of the uncontrolled system was created, consisting of the functional blocks and the signals between them. The block diagram shows how the different parts of the system – motor, worm drive, mount and camera – work together.

The mathematical modelling includes two nonlinearities: The worm drive is modelled as a proportional element with time-dependent gain and the motor can have a nonlinear input function. For the ease of system identification, the system was simplified by omitting weak relationships.

The outcome of this procedure is a clearer representation of the telescope system and the mathematical background. With the simplifications, the two coupled nonlinearities are independent. One was further simplified by a linearisation approach, the other has a constantly changing working point and therefore cannot be easily linearised. This is why a sophisticated approach for the system identification is needed.

Chapter 3

Image Registration

3.1 Measuring the Tracking-Error from Camera Images

A controller needs an accurate measurement of the system's output. In this application, the error is the distance between the actual and the desired pointing direction. As not only mechanical problems but also atmospheric aberrations have an effect on this, it is reasonable to measure the error directly through an optical system.

This optical system is often a second telescope, the *guiding scope*. It is mounted next to the main telescope. Therefore it always points – disregarding small deformations in the telescope tube – in the same direction as the main telescope. With a second camera, the *guiding camera*, the movement of the pointing direction relative to the sky can be measured in terms of displacement in the consecutive images from the guiding camera.

A reference image is taken at the beginning of the experiment. In order to measure, the translation between reference and current image is calculated. This is known as *image registration*.

3.2 Standard Methods for Image Registration

Normally, planar image registration is done with cross-correlation

$$cc_{q,r}(i,j) = \frac{1}{M \cdot N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(q(m+i, n+j) \cdot r(m,n) \right) \tag{3.1}$$

between images q and r, where $M \cdot N$ is the image size. The translation is the coordinate with highest correlation, $\underset{(i,j)}{\operatorname{argmax}}\, cc_{q,r}(i,j)$. This procedure gets a lot faster if done in the frequency domain, because the mathematical operation is simpler: Since the form of Equation 3.1 is closely related to the form of convolution, it can be done with a single matrix multiplication

$$\mathcal{F}(cc_{q,r}) = Q \cdot R^* \ , \tag{3.2}$$

where Q is the Fourier-transform of q and R^* is the complex conjugated Fourier-transform of r. [Tö05]

As motivated in Chapter 2, the goal for the controller is an accuracy of 0.43 arcs. With respect to the image sensor on the guiding camera, the accuracy for the image registration needs to be in the order of 10^{-2} pixel. Normal cross-correlation only provides an accuracy of 1 pixel.

More accuracy for the cross-correlation approach can be reached by upsampling the used images, but then the procedure gets time consuming and memory intensive[1]. There are already algorithms that handle the upsampling and registration in an efficient way, for example [GSTF08]. But for the use in this work, they are comparably slow, which is shown in Section 3.3.5. Because of these limitations, most software for autoguiding only use a small window around one star for the error measurement.

3.3 A Triangulation Method for Image Registration

Because the use of only one star means not using all available information and cross-correlation is too slow at the required resolution, the image registration in this work is done with a specifically developed technique.

3.3.1 Related Work

The development of the presented method was inspired by the idea behind *astrometry.net*. It is a software to infer the pointing direction of an image of the sky by finding star constellations. A short overview of the method and what it is used for, can be found in [HBL+08].

[1] If upscaled by factor 100, a black and white image of resolution 1280×960 pixel would need approximately 11.4 GB of memory.

The basic idea is to take the positions of the stars (which are easy to detect) and generate geometric features from the positions relative to each other. *astrometry.net* uses a set of four stars to generate four measures. These form a kind of "hash", what means these constellations can be recognized by comparing those four numbers. This hash is then used to find a certain constellation from an image in a huge catalogue of known constellations.

The setting in this thesis is similar: The stars from one image have to be found in the other. Therefore, a comparison of geometric features is sensible. On the other hand, the setting differs: No catalogue with millions of constellations has to be searched, only the constellations in the second image. Also, the computing power is limited and fast measurements are needed, why smaller geometric structures are used.

3.3.2 Finding Stars

As it is generally not possible to determine the complete geometric transformation between two images, a common approach is to use the spatial relocation of *tie points* for an approximation [GW02]. First, a set of suitable tie points has to be found. As this algorithm is applied to star-images, it searches for the stars' centres.

The procedure begins with a noisy image s of the night sky, Figure 3.1 shows an example. The noise is reduced by subtracting the mean $\bar{s}$ and γ times the standard deviation σ_s from the intensity values $s_{i,j}$

$$s'_{i,j} = s_{i,j} - \bar{s} - \gamma \cdot \sigma_s \tag{3.3}$$

The parameter γ controls the aggressiveness of the noise reduction: With a too small γ, there will be a lot of noise left, with a too big γ the dim stars will vanish. As the pixel values cannot drop below zero, this serves as a threshold. The result of this operation is an image that is mostly black and has only some regions left with intensity values above zero. These regions contain the stars, if γ was chosen right.

Because of blurring in atmosphere and optics, the stars are not mapped to one single pixel each. They are mapped to a small region consisting of a number of neighbouring pixels on the camera's sensor. An image with applied threshold and resulting regions is shown in Figure 3.2.

From a region of interest, there is one more step to obtain the centre of the star, which afterwards represents one tie point. A weighted centroid of the connected pixels is an

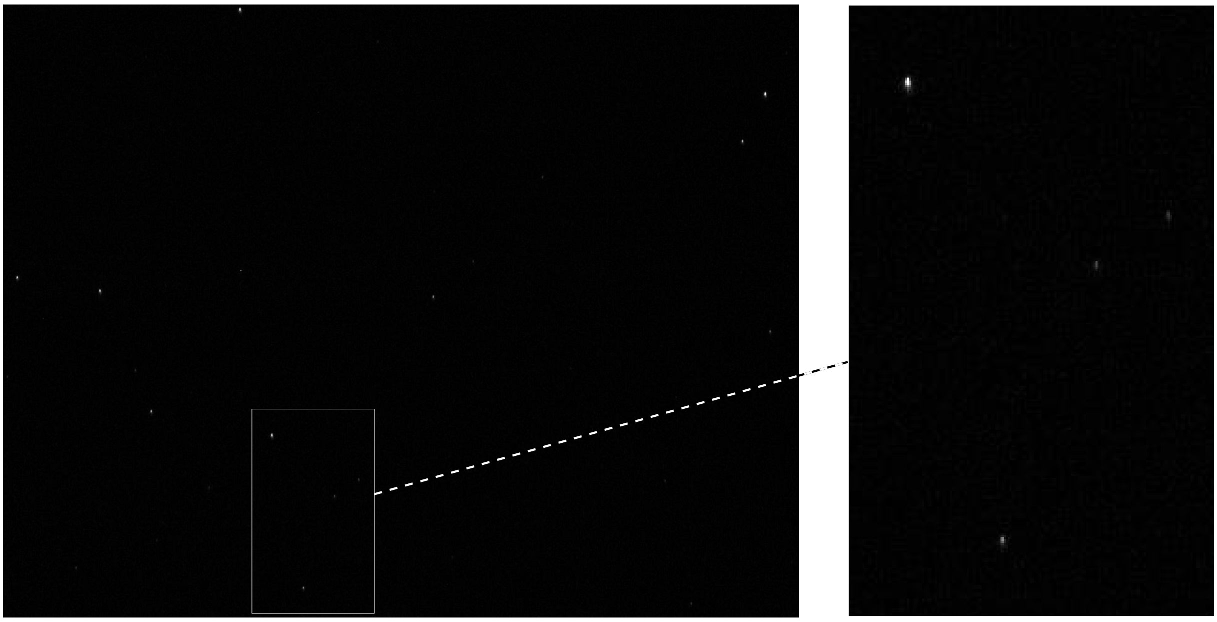


Figure 3.1: Raw Image from the Guiding Camera

This figure shows a raw image from the guiding camera. The original image size is 1280 × 960 pixel.

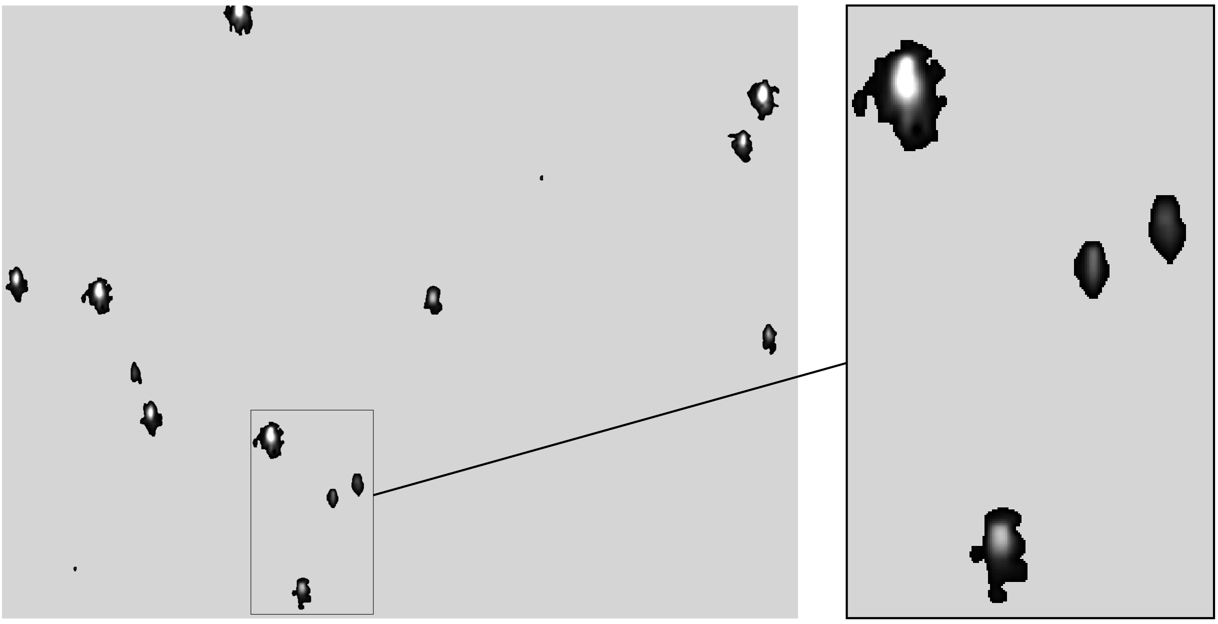


Figure 3.2: Isolated Stars

From Figure 3.1: A threshold is applied. The grey area is below the threshold. The black-and-white patches are the star-regions. *For better visibility, the stars are artificially enlarged.*

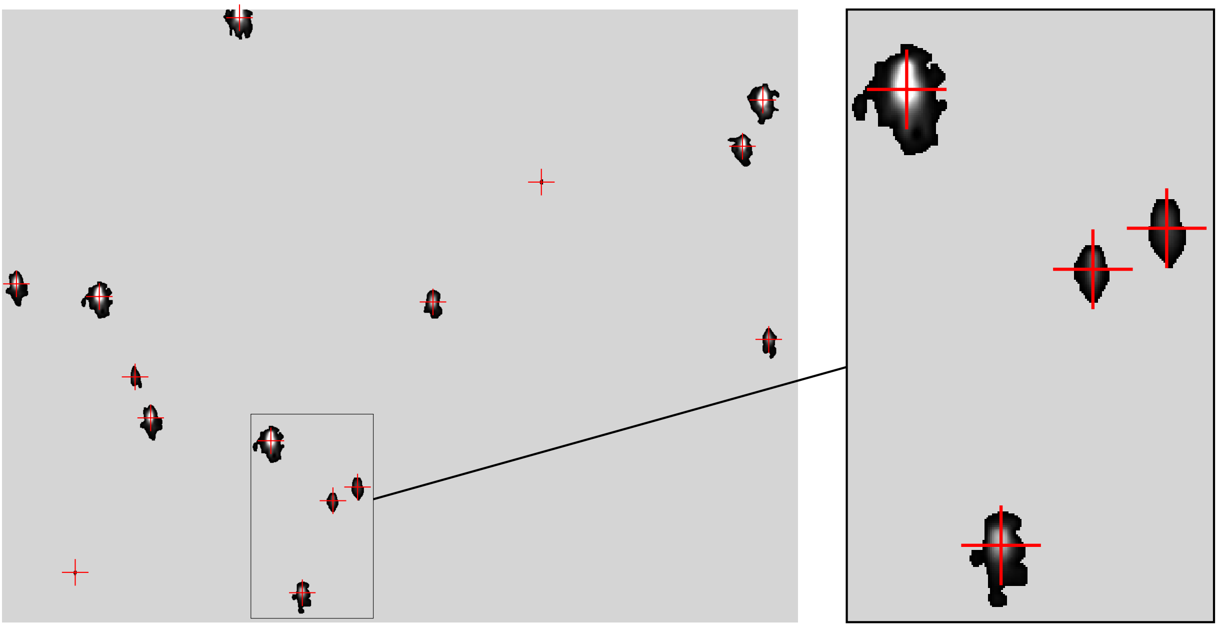


Figure 3.3: Isolated Stars with Centroids

From Figure 3.2: The weighted centroids of the stars are calculated and marked with a red cross. *For better visibility, the stars are artificially enlarged.*

approximation for the star's position

$$C = \frac{1}{N_k} \sum_{i,j}^{s_{i,j} \in R_k} (i,j) \cdot s_{i,j} \tag{3.4}$$

where R_k is the set of pixels in the region, N_k is the number of elements in R_k and $s_{i,j}$ is the intensity value of the pixel. An example for the position of the centroids with respect to the star image is shown in Figure 3.3.

3.3.3 Matching Triangles

With the star centres, tie points are available to infer the transformation between the images. But before this can be done, the points of both images have to be matched. To do the matching, the idea of geometric measures is useful: After building a set of triangles from the points, their similarity can be used to find corresponding triangles. This provides the matched tie points.

To find the same triangles in both the reference frame and the current frame, all pairwise unique triplets of points are used to construct triangles. If less triangles are used, experiments showed that the matching often fails. One set of constructed triangles is shown in Figure 3.4.

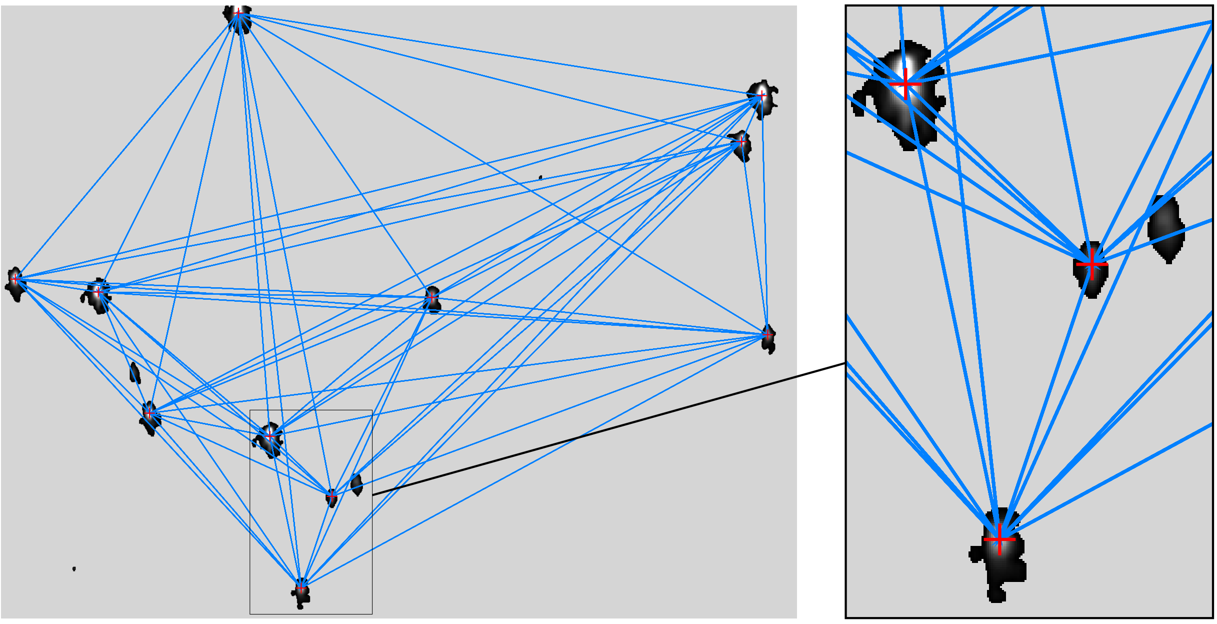


Figure 3.4: Stars with All Triangles

From Figure 3.3: For the eleven brightest stars, all triangles between the centroids are calculated and visualized. *For better visibility, the stars are artificially enlarged.*

For the recognition of the triangles, the right geometric similarity measures have to be chosen. Because this algorithm runs on star images, the scale of the images does not change, but there might be a rotation in it. Therefore, a rotation invariant but scale variant measure is used. One such measure is the equal oriented congruence of triangles (side-side-side) [BSMM05]: Two triangles are matched, if the ordered lengths of their sides are equal.

Figure 3.5 shows the used features of the triangles. The features f_1, f_2 and f_3 are the length of the sides of the triangle in descending order. In addition, the positions of the vertices $V_{\{1,2,3\}}$ and the angle δ between the triangle's base and the coordinate system are stored. This is useful for calculating the transformation later on.

For every pair of triangles from the reference frame and the current frame, the elementwise ℓ_2-distance of their features $f_{\{1,2,3\}}$ is calculated. For example the distance of the first feature

$$d_{f_1} = |f_1(t_r) - f_1(t_c)| \tag{3.5}$$

where t_r denotes one triangle from the reference frame and t_c one from the current frame. If all distances are below a threshold, the triangles are assumed to belong to the same stars in both images. For each found triangle, the vertices can be matched from one image to the other to get a set of corresponding tie points. This process can be seen in Figure 3.6, where ten out of eleven stars are matched. It also shows an advantage of this method: Its

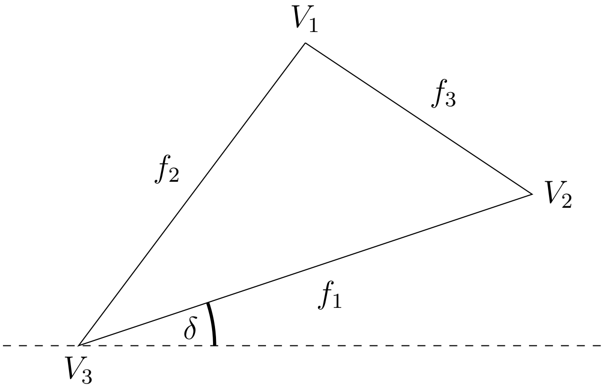


Figure 3.5: The Triangle-Features

This figure shows an overview of the triangle-features. The dashed line shows the zero angle of the coordinate system.

robustness against lost or covered stars, which can be seen in the inset. If, for example, a small cloud disrupts the measurement, a single-star autoguiding would be lost, while this version can stay on track.

To prevent the association of wrong triangles or of the right triangle in the wrong orientation, it is useful to drop triangles when there are other triangles with similar features or triangles which are self-similar (isosceles or equilateral). Although it is possible to lose too many triangles in this way, experiments showed that this practice does not lead to problems.

In the triangle-matching with the complete set of triangles lies the main computational limitation of this method: With growing number of points, the set of triangles to compare increases exponentially. Therefore, the number of points registered has to be limited. We consider eleven points to be a good compromise between accuracy and computation time.

3.3.4 Inferring the Image Displacement

With a list of corresponding tie points, the planar transformation between the current and the reference image can be inferred. As any planar transformation between two points with a given translation can be constructed from a rotation around an arbitrary centre and a corresponding translation, there is no single solution. The explanation used here is a rotation around the image-centre, which then defines the translation.

The rotation can be computed by calculating the difference

$$d_\delta = \delta(t_r) - \delta(t_c) \tag{3.6}$$

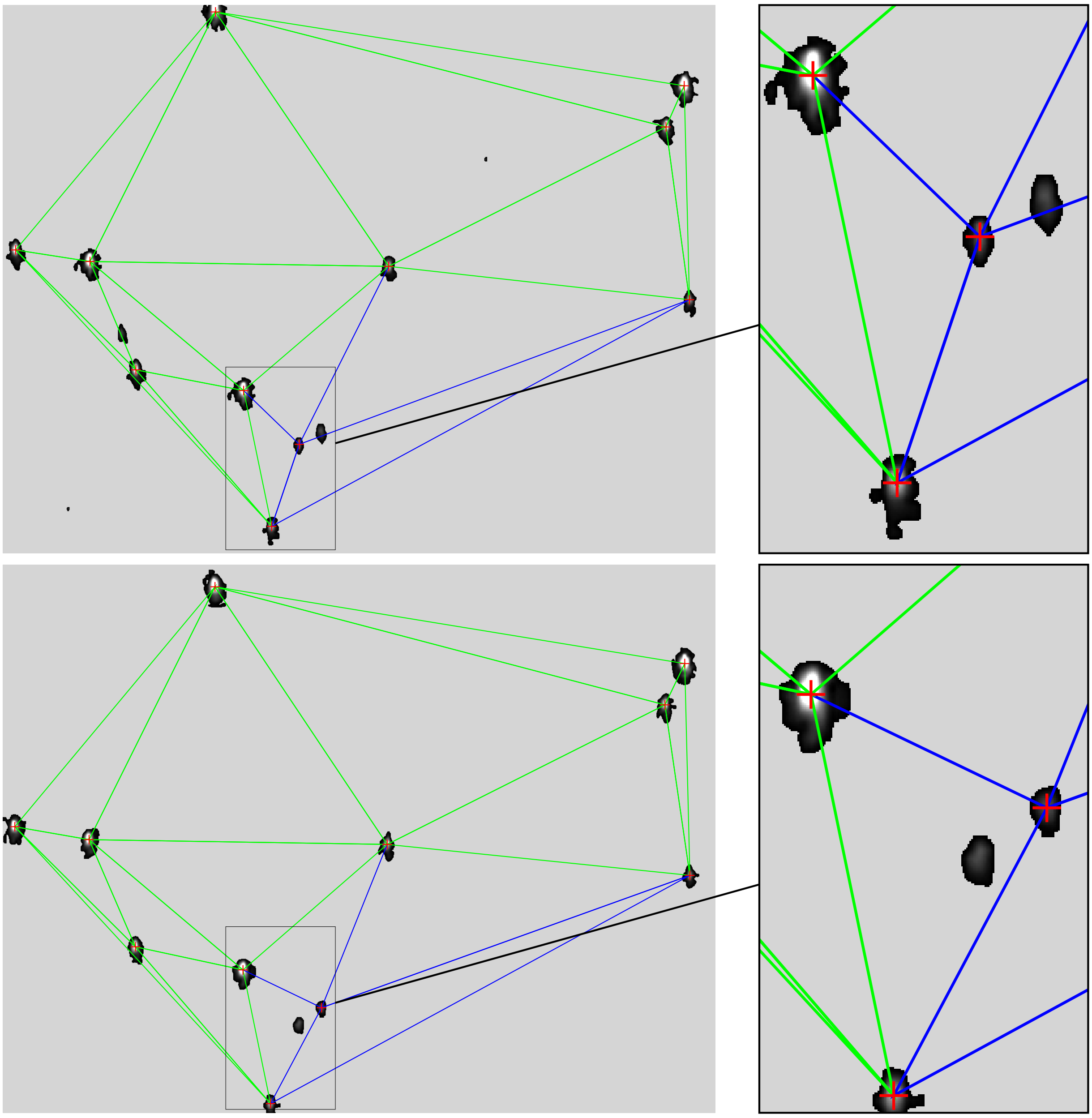


Figure 3.6: Matching Triangles in Two Images

This figure shows the triangle matching for two different images with 12 s time difference. The green triangles are matched from the upper to the lower image. The blue triangles are not matched. In the inset one can see that two stars have swapped because of variations in measured brightness. *To see the effect, not all triangles are shown in this image. For better visibility, the stars are artificially enlarged.*

for each pair of matched triangles (t_r, t_c), where δ is the angle of the triangle with respect to the coordinate system. The tie points from the current frame can then be rotated back with the mean angle $\bar{d_\delta}$. This leaves the points and their correspondences with only a translation between them.

From the the vertices' translations

$$d_{V_i} = V_i(t_r) - V_i(t_c) \tag{3.7}$$

for each pair of matched triangles (t_r, t_c), the overall translation is the mean $\bar{d_V}$.

This procedure requires only one step, no iterations are needed as with other techniques like the iterative closest point algorithm [BM92]. Therefore, it is reliably fast.

3.3.5 Empirical Evaluation

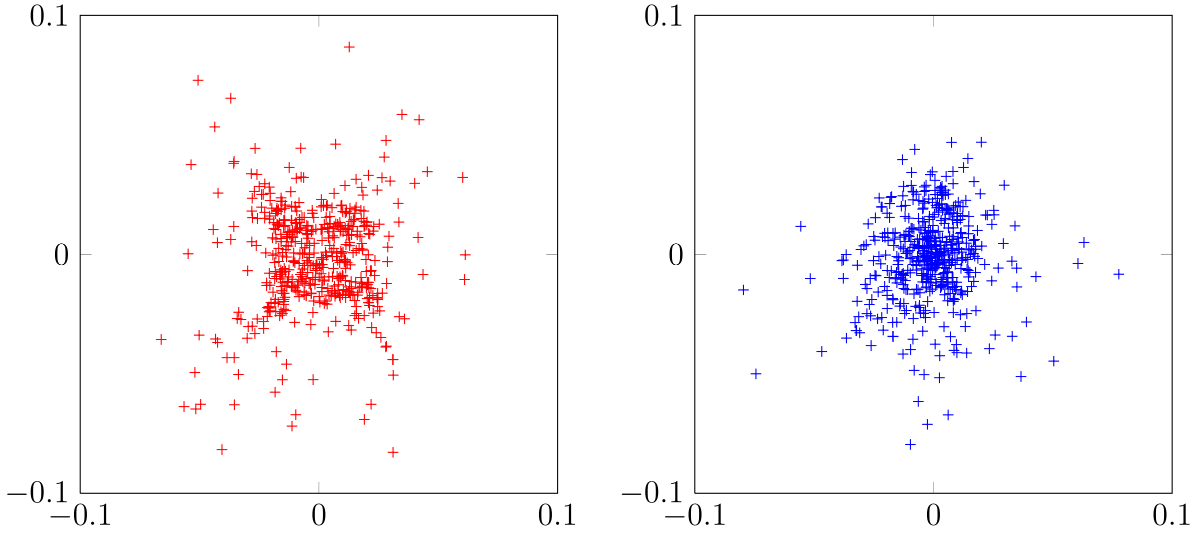
The presented triangulation-based image registration technique was tested against a highly engineered cross-correlation approach from [GSTF08] that uses the fast Fourier transform [CT65] for reasons of speed. A randomly picked image from a set of night-sky images was shifted with sub-pixel accuracy to a random translation between -1 and $+1$ for each direction. Then, the image was registered with both the triangulation and the cross-correlation method. The error measure is the difference between true translation and the result from the image registration.

From 500 random shifts and registrations, the mean error for the cross-correlation setup was 0.0245 pixel with a standard deviation of 0.0159 pixel. The triangulation-based setup showed a mean error of 0.0196 pixel with a standard deviation of 0.0141 pixel. The mean runtime of the cross-correlation method was 1598 ms per image registration, compared to 373 ms with the triangulation-based method.

In Figure 3.7 the scatter plot of the error is shown. The error distribution of the cross-correlation shows artefacts in attenuating the bisectrices, whereas the error distribution of the triangulation-based method looks more Gaussian.

3.4 Conclusion

For the application in astrophotography, measuring the pointing error with an optical device is practical. Because the error is measured as shift between two images, image registration



(a) Cross-Correlation, Error / pixel

(b) Triangulation, Error / pixel

Figure 3.7: Scatter Plots of the Image Registration Evaluation

Panel (a) shows the measurement error of the cross-correlation method, Panel (b) shows the same for the triangulation method. Each point marks the registration error for a randomly shifted star image.

is employed for this. Unfortunately, standard methods like cross-correlation are too slow at the required resolution.

After presenting inspiring related work, a new method for registering star images was developed step-by-step: First, interesting regions are extracted by applying a threshold on the pixel intensity. Second, the star positions are estimated by calculating the weighted centroid for each region. After knowing the stars' positions, a set of triangles is constructed and geometric measures are calculated.

With the geometric measures, the triangles from one image can be compared to those from the other. If the similarity is high enough, the vertices of the triangles are matched. These matched points serve as tie points for the reconstruction of the translation between the images.

With the algorithm presented in this chapter, the pointing error of a telescope can accurately be measured. As shown in the experimental comparison, the presented algorithm is faster and more accurate than a comparable cross-correlation approach.

Chapter 4

System Identification

4.1 Context: System Identification Methods

System identification is the huge field of building mathematical models from measured data. This is useful because with more information about it, a certain system can be controlled better. Although there is a distinction between different kinds of systems (linear/nonlinear), almost all physical systems are nonlinear [Lju10]. More important is the distinction between linear and nonlinear *models*.

Another classification is into parametric and nonparametric methods: Parametric models have a fixed set of basis functions (for example polynomials) that are parametrized by a set of parameters. These parameters are subject to the system identification and are adjusted during the process to best fit to the system. Nonparametric models have an infinite-dimensional space of basis functions, for example the Gaussian bell (see Section 4.2.2).

There is also a classification in different shades of grey for the type of system identification method, reaching from pure physical models (white) to models that are "just flexible function surfaces" (black-box) [Lju10].

In this work, the identified system is nonlinear. For the identification, a nonparametric framework is used, but the function class is not completely flexible, because some physical information is encoded in the basis functions (see Section 4.2.2 for more detail). Therefore, it can be seen as almost black box model. Note that this system identification is only used for one part of the model (the gear function). Other parts are identified through a physical model.

4.2 Background: Gaussian Process Regression

Gaussian process regression is a general framework for nonparametric, nonlinear regression. It is based on calculating probabilities according to Bayes' theorem. A comprehensive introduction can be found in [RW06].

The Bayesian View

This is just a brief recall of Bayes' theorem, based on [Bis06]. A full introduction can also be found there or elsewhere (for example in [RN03] or [Bar11]).

The theory of Gaussian processes uses the Bayesian framework, which is a way of thinking about and calculating uncertainties. Assume there is an observed dataset $\mathcal{D}$ and a set of possible models $\mathcal{M}$ to produce the data. In most cases, the question is, which latent model produced the measured data. To answer this question, the probability distribution over the models *given* the data $p(\mathcal{M}|\mathcal{D})$ is needed. This conditional probability (*posterior*) can be represented with Bayes' theorem

$$p(\mathcal{M}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{M}) \cdot p(\mathcal{M})}{p(\mathcal{D})} \tag{4.1}$$

where $p(\mathcal{M})$ is the *prior* distribution over the models, $p(\mathcal{D}|\mathcal{M})$ is the *likelihood* for the data to be produced by the models, and $p(\mathcal{D})$ is the normalization constant (*marginal*).

This can be seen as follows: *Before* the inference system gets measured data, the belief over the models is distributed according to the *prior*. When measured data comes in, those parts of the prior distribution are emphasized that are *consistent* with the data. This gives the belief over the distribution of models *after* the data was taken into account, which is called *posterior*:

$$\text{posterior} \propto \text{likelihood} \cdot \text{prior} \tag{4.2}$$

The Gaussian Process

A Gaussian process $\mathcal{GP}(f; \mu, k)$ is an infinite-dimensional probability measure over *functions* $f : \mathbb{X} \multimap \mathbb{R}$ over a metric space $\mathbb{X}$, such that every finite, $D-$dimensional linear restriction to function *values* $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{X}^D \multimap \mathbb{R}^D$ is a Gaussian distribution $\mathcal{N}(f(\boldsymbol{x}); \mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}))$. It is parametrized by two objects: A mean function $\mu : \mathbb{X} \multimap \mathbb{R}$, and a covariance function $k : \mathbb{X} \times \mathbb{X} \multimap \mathbb{R}$. The mean has a relatively straightforward role; it simply shifts predictions.

The covariance function's responsibility is more intricate. It can be interpreted as a similarity measure over function values in $\mathbb{X}$ and controls the shape of the Gaussian process belief in the space of functions. It has to be chosen such that the finite-dimensional matrix $k(x, x') \in \mathbb{R}^{|x| \times |x'|}$ is positive definite for all $x, x' \in \mathbb{X}$ and is also known as the *kernel.*

The main utility of Gaussian process *priors* arises from the closure of the Gaussian exponential family under multiplication: If data points $\boldsymbol{y}$ are observed at locations $\boldsymbol{x}$ with Gaussian noise ε

$$y_i = f(x_i) + \varepsilon \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{4.3}$$

then the *posterior* distribution under the Gaussian process prior and the Gaussian likelihood encoding this observation is also a Gaussian process, with the posterior mean and covariance

$$\mu(x_*) = \boldsymbol{k}^{\intercal}_{x_*\boldsymbol{x}} K^{-1}_{\boldsymbol{x}\boldsymbol{x}} \mathbf{y} \tag{4.4}$$

$$\Sigma(x^*, x_*) = k_{x^*x_*} - \boldsymbol{k}^{\intercal}_{x^*\boldsymbol{x}} K^{-1}_{\boldsymbol{x}\boldsymbol{x}} \boldsymbol{k}_{\boldsymbol{x}x_*} \tag{4.5}$$

where $\boldsymbol{k}_{x_*\boldsymbol{x}} = k(x_*, \boldsymbol{x}) \in \mathbb{R}^{|x_*| \times |\boldsymbol{x}|}$ is the kernel projection of the observations' effect at the evaluation location(s) x_*, and $K_{\boldsymbol{x}\boldsymbol{x}} = k(\boldsymbol{x}, \boldsymbol{x}) \in \mathbb{R}^{|\boldsymbol{x}| \times |\boldsymbol{x}|}$ is the kernel Gram matrix. [RW06]

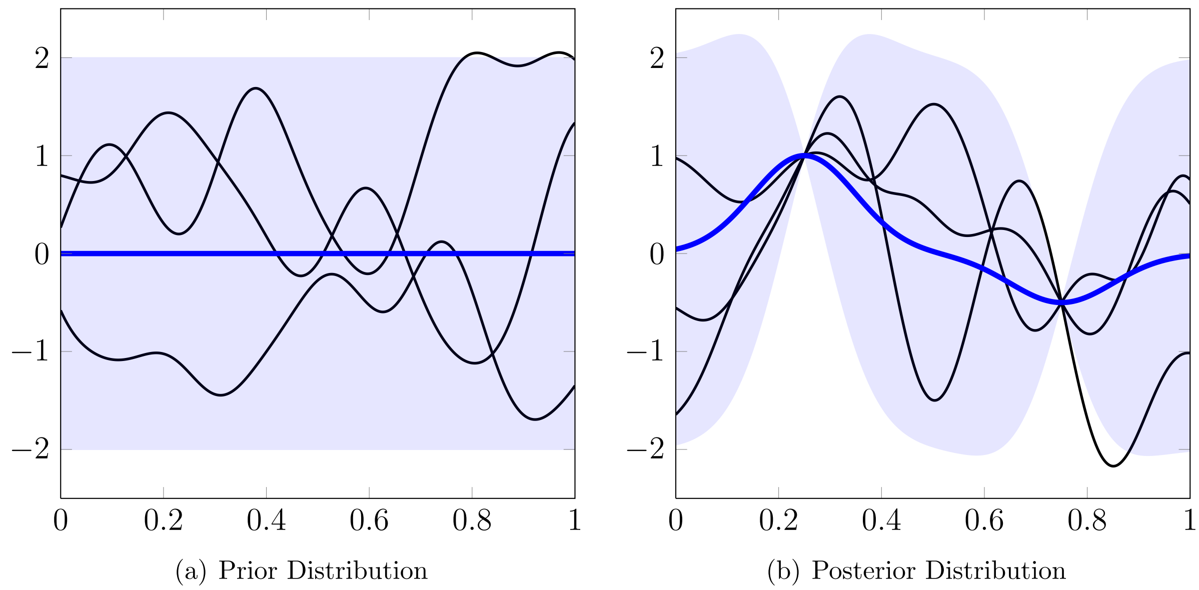
4.2.1 Inference & Prediction

One property of Gaussian processes is that it is actually possible to see "what is going on". Consider an example of a Gaussian process with a *squared exponential* covariance function. This is standard, there is more information about it in the next section. The mean function is zero, which is also common.

In Figure 4.1(a), a Gaussian process before doing inference (*prior distribution*) is shown. The mean is zero as expected, the variance is constant everywhere. The randomly sampled functions illustrate how functions from this prior *can* look like.

Figure 4.1(b) shows the *posterior distribution.* This shows the effect of Equations 4.4 and 4.5 on the distribution: With the inference, the mean is "pulled" towards the measured value at locations near the measurement (Equation 4.4). The *un*certainty, which is expressed as covariance $k_{x^*x_*}$, is reduced near the measurements (Equation 4.5). This is a sensible view on this – where more information is available, the uncertainty is lower.

According to Equation 4.2, the prior defines the space of functions that is considered to



(a) Prior Distribution

(b) Posterior Distribution

Figure 4.1: Samples from Prior and Posterior of a Gaussian Process

Panel (a) shows the prior distribution of a Gaussian process and three randomly drawn samples from it. Panel (b) shows the posterior distribution and three samples after observing two data points. In both plots the mean prediction is shown as blue line and twice the pointwise standard deviation is shown as shading. *Figure adopted from* [RW06].

be suitable for the problem. During the inference, this space gets restricted in those regions where data was seen. From the prior space of functions, only those functions with a high likelihood for having produced the data are considered to be consistent with it. This gives the posterior distribution.

4.2.2 Kernel Engineering

Before a Gaussian process can be used to learn a specific function, the right kernel for the problem has to be chosen.

The most widely used kernel is the "squared exponential" (sometimes also "Gaussian" or "radial basis function (RBF)") kernel, even if this name is misleading because it is an exponential quadratic. In the space $\mathbb{X} = \mathbb{R}^1$ it is of the form

$$k_{\text{SE}}(x_i, x_j; \theta_{\text{SE}}, \ell_{\text{SE}}) = \theta_{\text{SE}}^2 \exp\left(-\frac{(x_i - x_j)^2}{2 \cdot \ell_{\text{SE}}}\right) \tag{4.6}$$

where θ_{SE} and ℓ_{SE} are the hyperparameters. Those parameters have the following roles:

- ℓ_{SE} is the characteristic length scale. It is proportional to the expected number of zero crossings of the function values [RW06] and can be seen as complexity measure:

Functions with a short length scale are flexible and can encode much information, functions with a longer length scale are less flexible.

- θ_{SE} is the signal variance. It defines how strongly the function values with a certain input-distance can differ from each other. This can be seen as the amplitude of the function.

The squared exponential kernel induces a Gaussian process belief whose uncertainty rises with Euclidean distance from observation locations up to a bound. Samples from the corresponding Gaussian process have only one unique length scale controlled by the scaling parameter ℓ_{SE} [RW06].

One way of checking if a chosen kernel is the right kernel for a certain application, is to sample from the prior. This corresponds to picking functions randomly from the space of functions that is defined by the prior. The samples should look like data from the real system – then the data can be explained by the prior. Trying this with the data from the telescope system and a squared exponential kernel (with different length scales) brings samples like in Figure 4.2:

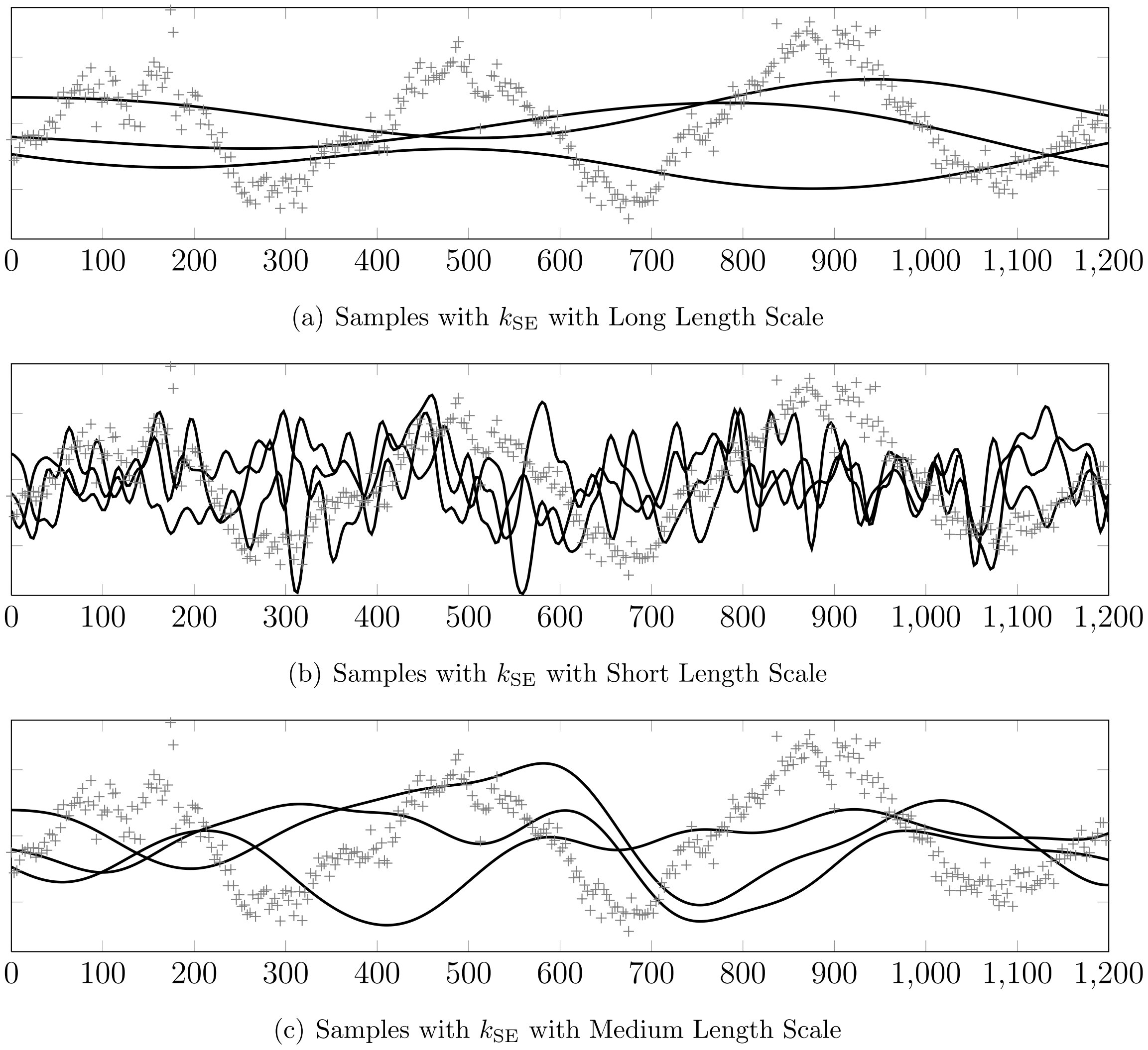
- Long length scale, Figure 4.2(a): The function values cannot change fast enough, they could not follow the data.

- Short length scale, Figure 4.2(b): The function values change too quickly. Large scale structure would not be captured.

- Medium length scale, Figure 4.2(c): The samples have a similar shape as the measured data. It looks like the right length scale.

When the right length scale is determined, the next step is to infer the posterior from the prior and the data to see if predictions are possible. Figure 4.3 shows that the posterior mean follows the rough structure around the training points. But after the last training point, the mean drops back to zero. A prediction of the periodic structure upon this kernel is not possible.

This means, it is not ideally suited for modelling a periodic function. The squared exponential only predicts well in areas where enough data is already available. But as with all control problems, since measurements in the future are not causal, only data from the past is available. The convenient property of *periodic* functions is that their future does indeed resemble the past.



(a) Samples with k_{SE} with Long Length Scale

(b) Samples with k_{SE} with Short Length Scale

(c) Samples with k_{SE} with Medium Length Scale

Figure 4.2: Samples for k_{SE} with Different Length Scales

Panel (a) shows three samples from a Gaussian process with squared exponential kernel and long length scale. Panel (b) shows three samples with short length scale. Panel (c) shows three samples with medium length scale. In all plots the grey data points are measured data from a real system.

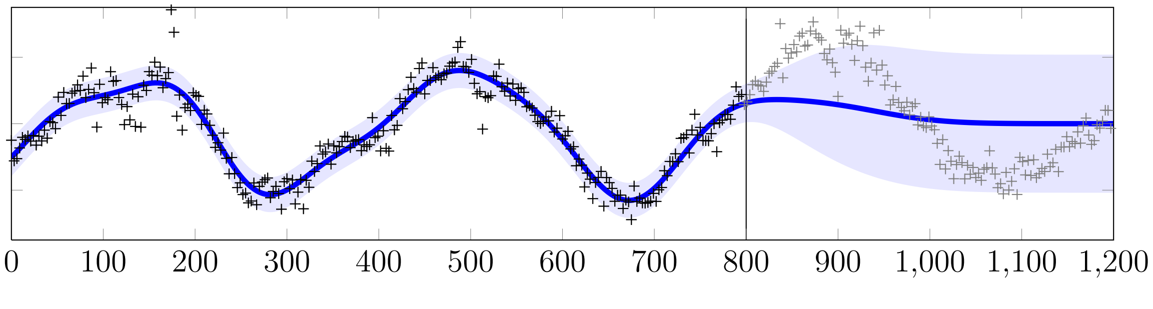


Figure 4.3: Posterior distribution for k_{SE}

This figure shows the posterior distribution with a squared exponential covariance function after observing measured data. The black points are training data, the grey points are test data. The mean prediction is shown as blue line and twice the pointwise standard deviation is shown as shading.

This aspect is captured with a periodic kernel function. The idea of mapping Euclidean distances through a periodic function to construct periodic beliefs was first described by MacKay [Mac98], and further studied by Rasmussen and Williams [RW06]. They propose a kernel on one-dimensional input spaces $\mathbb{X} = \mathbb{R}$ of the form

$$k_{\text{periodic}}(x_i, x_j; \theta_{\text{p}}, \ell_{\text{p}}, p) = \theta_{\text{p}}^2 \exp\left(-\frac{2\sin^2(\frac{\pi}{p}(x_i - x_j))}{\ell_{\text{p}}^2}\right) \tag{4.7}$$

The parameters $\theta_{\text{p}}, \ell, p$ of this kernel have nicely interpretable, though non-trivial roles:

- θ_{p}^2 is the signal variance.
- p is the period length.
- ℓ_{p} controls the length scale of samples, similarly to the length scale of the squared exponential kernel. For $\ell_{\text{p}} \to 0$, samples are periodic, but function values at points other than multiples of p apart are completely uncorrelated. For $\ell_{\text{p}} \to \infty$, samples have the same value everywhere.

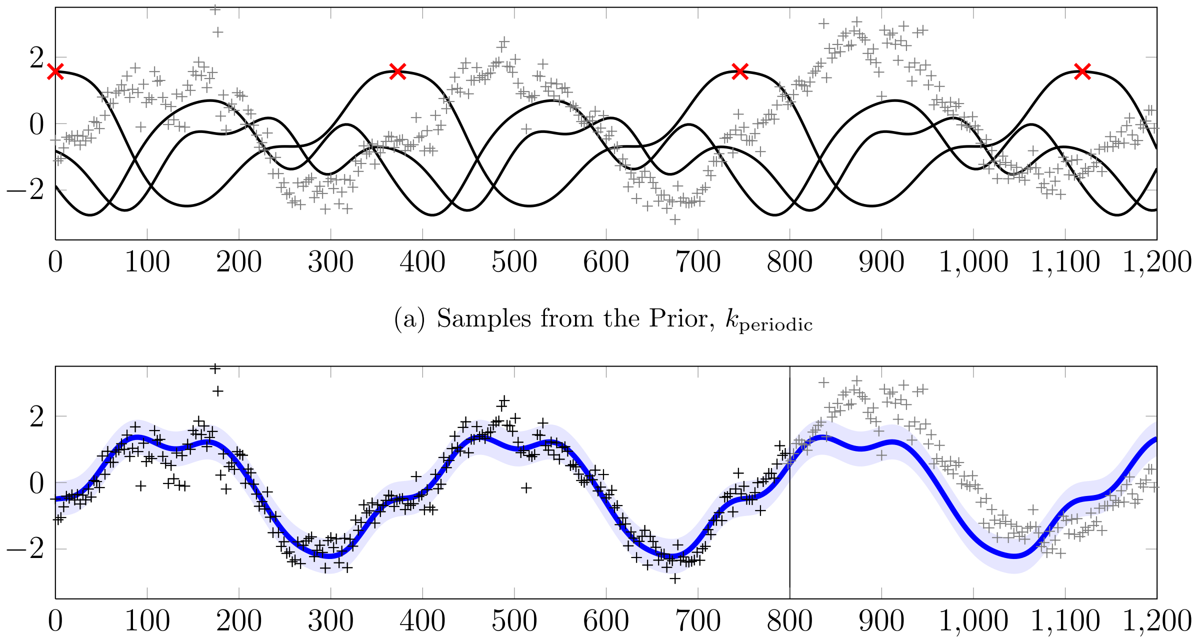
Equation (4.7) is still a relatively restrictive kernel. It assumes that the true function is perfectly periodic over all times. The effect is visible in Figure 4.4(a): The points that are one period length apart from each other have exactly the same value.

Since this is not true for the real physical system (see Figure 4.4(b)), this property needs to be relaxed. To address this issue, the fact that kernels form semirings is used: sums and products of kernels are also kernels [RW06]. Therefore, a composite kernel is constructed by setting

$$k_{\text{comp}}(x_i, x_j) = k_{\text{SE}}(x_i, x_j; \theta_{\text{SE}}, \ell_{\text{SE}}) \cdot k_{\text{periodic}}(x_i, x_j; \theta_{\text{p}}, \ell_{\text{p}}, p) \tag{4.8}$$

This kernel considers two input locations similar if they are similar under *both* the squared exponential *and* the periodic kernel. If $\ell_{SE} > p$, this allows encoding a *decay* in the periodic behaviour over several oscillations. This means that the kernel puts higher weight on the most recent observations than on older ones. This effect can be seen in Figure 4.5.

What is still left unmodelled are the small nonperiodic variations on top of the periodic structure. Because they are not periodic, they cannot be modelled with a short length scale of the periodic term in Equation 4.8. They cannot be modelled with a short length scale in the squared exponential term either, because this would encode a too strong decay in the periodicity. Therefore, a third part is needed in the composite kernel. Because the length



(a) Samples from the Prior, k_{periodic}

(b) Posterior Distribution, k_{periodic}

Figure 4.4: Samples and Posterior Distribution for k_{periodic}

Panel (a) shows three samples from a Gaussian process with periodic covariance function, the red marks are one period apart from each other. The grey data points are measured data from a real system. Panel (b) shows the posterior distribution with a periodic covariance function after observing measured data. The black points are training data, the grey points are test data. The mean prediction is shown as blue line and twice the pointwise standard deviation is shown as shading.

scale of these variations is not known exactly, we chose a rational quadratic term for these effects:

$$k_{\text{RQ}}(x_i, x_j; \theta_{\text{RQ}}, \ell_{\text{RQ}}, \alpha) = \theta_{\text{RQ}}^2 \left(1 + \frac{(x_i - x_j)^2}{2\ell_{\text{RQ}}^2 \alpha}\right)^{-\alpha} \tag{4.9}$$

It was proposed by Matérn [Mat60] and can be seen as generalized version of the squared exponential kernel with a *mixture* of characteristic length scales, controlled by the parameter α [RW06].

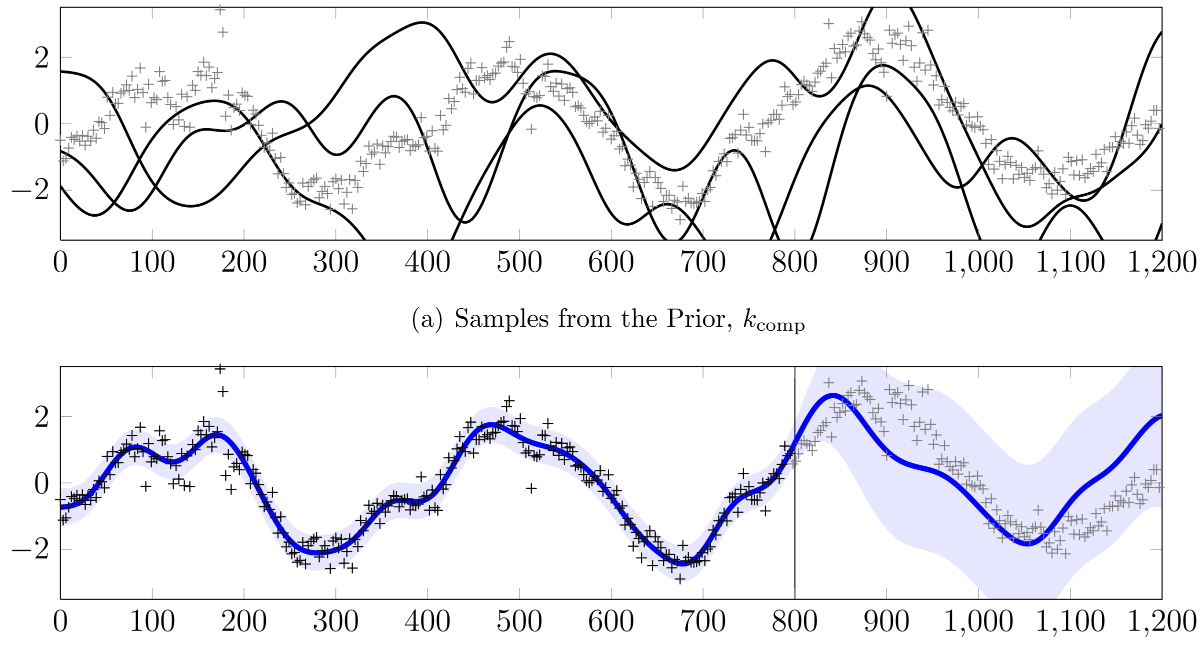
The complete kernel that is used for the modelling is then

$$k_\star(x_i, x_j) = k_{\text{SE}}(x_i, x_j; \theta_{\text{SE}}, \ell_{\text{SE}}) \cdot k_{\text{periodic}}(x_i, x_j; \theta_{\text{p}}, \ell_{\text{p}}, p) + k_{\text{RQ}}(x_i, x_j; \theta_{\text{RQ}}, \ell_{\text{RQ}}, \alpha) \tag{4.10}$$

Samples from this kernel and the inference on the telescope data are shown in Figure 4.6.

4.2.3 Hyperparameter-Optimization

The kernel from Equation 4.10 has eight hyperparameters. To get a good prediction for the learned function, those parameters need to have sensible values. Some parameters cannot



(a) Samples from the Prior, k_{comp}

(b) Posterior Distribution, k_{comp}

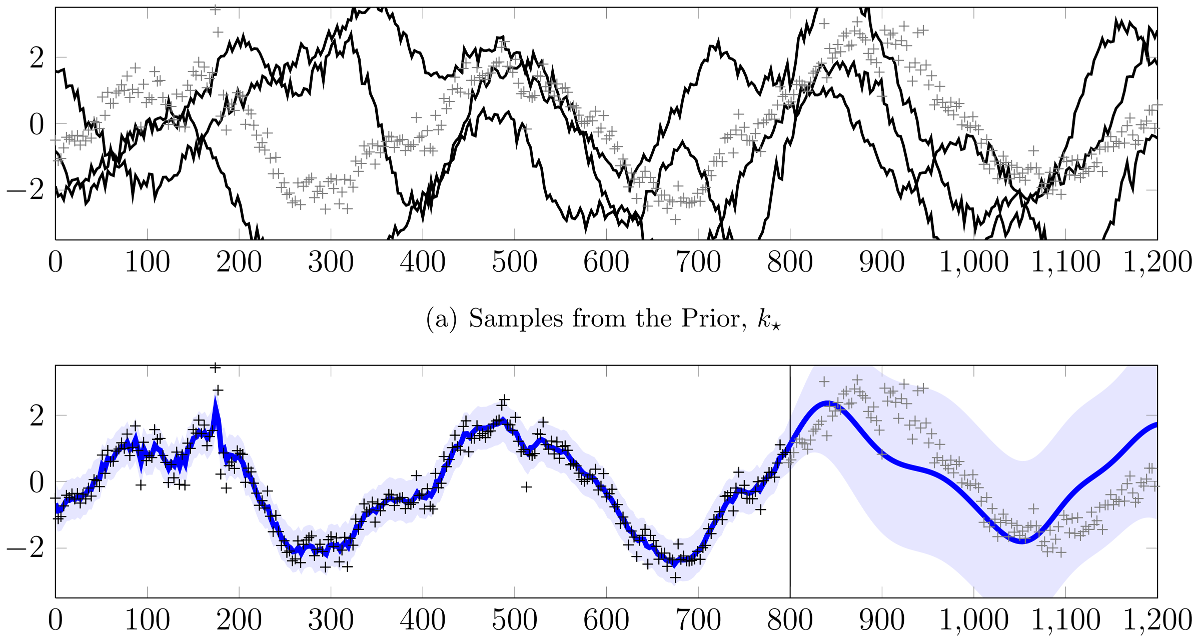
Figure 4.5: Samples and Posterior Distribution for k_{comp}

Panel (a) shows three samples from a Gaussian process with the composite covariance function k_{comp}. The grey data points are measured data from a real system. Panel (b) shows the posterior distribution with the composite covariance function after observing measured data. The black points are training data, the grey points are test data. The mean prediction is shown as blue line and twice the pointwise standard deviation is shown as shading.

be chosen beforehand, because they are dependent on the used hardware. They have to be learned at runtime. The period-length of the function is an example for this: Because the type of the telescope mount is not known in advance, the period length cannot be set to a fixed, known value.

To adjust the hyperparameters at runtime, the negative likelihood for data to be produced by these hyperparameters has to be minimized over the parameter space. This can be done with a numerical optimizer, for example `minimize.m`[1], which is an implementation of the conjugate gradient optimizer [HS52]. Problematic is that the space of hyperparameters is not likely to be convex. There are a lot of local minima and the optimizer often converges to a point in the space where the result, in terms of the resulting space of functions, is not desired. For example, one local minimum could be a set of hyperparameters where the posterior mean is zero everywhere and the noise is so high that every variation in the measurements is explained to be noise. Another local minimum could be a very short length scale and a high signal variance on one covariance function, so that the posterior mean goes

[1] Implemented by Rasmussen, available online:
`http://www.gaussianprocess.org/gpml/code/matlab/util/minimize.m`



(a) Samples from the Prior, $k_\star$

(b) Posterior Distribution, $k_\star$

Figure 4.6: Samples and Posterior Distribution for $k_\star$

Panel (a) shows three samples from a Gaussian process with the complete covariance function $k_\star$. The grey data points are measured data from a real system. Panel (b) shows the posterior distribution with $k_\star$ after observing measured data. The black points are training data, the grey points are test data. The mean prediction is shown as blue line and twice the pointwise standard deviation is shown as shading.

through every single measured value but does not predict anything into the future but a zero mean.

The problem of unwanted local minima can be addressed with priors on the hyperparameters. Even when they cannot solve all such problems, the priors help by making the likelihood function "more convex". The priors can be seen as a penalization for hyperparameters which leave the region of sensible values for them. For example, a Gaussian prior can be used for the period-length: It is not sensible to assume the complete range of $\mathbb{R}^+$ to be equally likely for it. A human would know that a period length of 1 s or 2 h is not that kind of function that is addressed here. If a sensible range for a hyperparameter is known, a prior can be used to encode this knowledge. There should be *much* prior mass on the expected ranges of the hyperparameters, but also put *some* mass on everything that is possible.

In the case of the telescope system, a Gaussian prior of the form

$$\frac{1}{\sqrt{2\pi} \cdot \sigma_h} \cdot \exp\left[\left(\frac{h-\mu_h}{\sigma_h}\right)^2\right] \tag{4.11}$$

is used, where h denotes the current value of the hyperparameter, μ_h the mean and σ_h the standard deviation.

4.3 Conclusion

System identification is the idea of inferring from measurements to a system's underlying model. Because it is hard to provide a mathematical description for the error function of the telescope gear, a nonparametric regression method is used: Gaussian process regression.

The regression depends strongly on the used covariance function (or "kernel") and its hyperparameters. In this setting, the most common squared exponential kernel does not work, because its predictions are weak. Therefore, a more uncommon kernel is used, a periodic one.

Because the purely periodic kernel is too strict, the mathematical properties of kernel functions are used to form a composite kernel with weakened strictness. It captures the periodic effects but can also handle nonperiodic long term effects.

Not every hyperparameter can be chosen in advance. Therefore, they have to be adjusted during runtime. This can lead to problematic effects due to the nonconvexity of the parameter space. By the use of priors on the hyperparameters, this problems can be eased.

With the right kernel, Gaussian process regression is a powerful tool to infer certain nonlinear effects of the telescope system. In the next chapter this is used to obtain better control performance.

Chapter 5

Control Engineering

Short Notation

Note that in this chapter a short notation for discrete time signals is used:

- Values that are measured or calculated at time t are denoted with subscript T

$$e(t) = e_T$$

- A time difference of one finite time step Δt corresponds to 1 in the subscript notation

$$e(t - \Delta t) = e_{T-1}$$

$$e(t + \Delta t) = e_{T+1}$$

- The measurement marks the *beginning* of a control period. Thus, the control signal u_T is calculated with the measurement e_T and influences the next measurement e_{T+1}.

5.1 Control Configuration

The used controller is a state feedback controller in the setup from Figure 2.4. Disregarding the noise, it is of the form

$$u = k_u \cdot e \tag{5.1}$$

For the closed loop follows

$$\dot{e} = -k_u \cdot k_g \cdot k_w \cdot e \tag{5.2}$$

which leads to a characteristic polynomial $\lambda + k_u k_g k_w = 0$. In continuous time, the closed loop is stable for all values of $k_u > 0$, if the input gain k_g and gear reduction k_w are restricted to be positive [LW05].

Nevertheless, care has to be taken when choosing the controller gain k_u. Because the controller operates in discrete time steps, k_u cannot be chosen arbitrarily high. If the value is too high, the system will overshoot and become unstable.

The discrete-time state-equation of the system is

$$e_{T+1} = e_T + u_T \cdot \Delta t = e_T - k_u \cdot k_g \cdot k_w \cdot e_T \cdot \Delta t \tag{5.3}$$

The value for k_u that puts the error back to zero in one step is $\frac{1}{k_g \cdot k_w \cdot \Delta t}$, which is also called dead beat control [LW05].

To use this result, k_g and k_w have to be known. At first, assume that the product $k_g \cdot k_w$ is known to be ≈ 1. This is reasonable, because the telescope vendors try to provide this normalization by scaling the input accordingly.

As the system is constantly disturbed by the effect of $f(t)$, the P-controller is permanently tracking behind the error that is created by $f(t)$. If the effect of $f(t)$ could be *predicted*, it would be possible to increase the control performance by correcting the error beforehand. This is possible with the regression and prediction from Chapter 4.

5.2 Control with Prediction of the Error

One option for the controller is to use the direct prediction of the error. The basic idea is that the controller builds a model for the error to “look up” the future error. This would be the error that would happen if u was kept at zero. With this information about the future error, it is possible to construct the value of u that minimizes the future error.

First, the previous effects of the controller on the system have to be added to the measured error:

$$\tilde{e}_T = e_T + \sum_{\tau=0}^{T-1} u_\tau \cdot \Delta t \tag{5.4}$$

where $\tilde{e}$ is the estimate for the uncorrected error. For the next time step

$$\tilde{e}_{T+1} = \underbrace{e_{T+1}}_{\to 0} + \sum_{\tau=0}^{T} u_\tau \cdot \Delta t \tag{5.5}$$

the error e_{T+1} is set to be zero. The estimate $\tilde{e}_{T+1}$ in this equation is the mean prediction from the Gaussian process. Extracting the last element from the sum gives

$$\tilde{e}_{T+1} = \sum_{\tau=0}^{T-1} u_\tau \cdot \Delta t + u_T \cdot \Delta t \tag{5.6}$$

This can be solved for the next control signal u_T

$$u_T = \frac{1}{\Delta t} \left(\tilde{e}_{T+1} - \sum_{\tau=0}^{T-1} u_\tau \cdot \Delta t \right) \tag{5.7}$$

which is also a dead beat controller, but for the *future* error.

This controller works, the results are discussed in Chapter 6. As expected, the control performance of the controller with prediction is better than the performance of a controller without prediction.

However, note that if there is a flaw in the model of the control dynamics, this approach introduces an error which will accumulate over time (see the sum in Equation 5.7). Such a model error could be an assumed system gain k_g that differs from the real value. The effect of the controller on the system is then not known perfectly, and therefore the error $\tilde{e}$ cannot be reconstructed reliably. This leads to a discrepancy between the *real* and the *expected* effect of the controller. Because this difference accumulates, the gap between model and reality increases over time, giving bad control performance.

To avoid this problem, a second approach is presented in the next section.

5.3 Control with Prediction of the Derivative of the Error

To address the problems of the error-based control, instead of the error, the derivative of the error is considered for the regression. This can be done, because the number of available data points is high enough. The model is expanded to contain the motor function $g(u)$ in the feedback path of the system. Thus, a sum over the control signal is no longer necessary and the input function $g(u)$ can be learned as well as the system gain k_w.

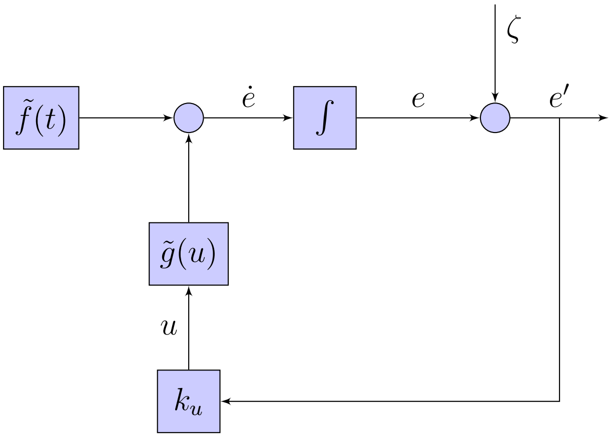


Figure 5.1: The Corrected Telescope System

This figure shows the corrected version of the telescope system according to Equation 5.8.

5.3.1 Correcting the Model

Some simplifications from Chapter 2 are too strong. When Equation 2.10 is changed to contain the complete remainder $\hat{g}(u)$ of the Taylor expansion, the term for the dynamics is

$$\dot{e}(t,u) = d(t,u) = \underbrace{k_w \cdot g(v_0) \cdot f(t)}_{\tilde{f}(t)} + \underbrace{k_w \cdot \hat{g}(u)}_{\tilde{g}(u)} \tag{5.8}$$

This representation can capture a nonlinear function g and an unknown k_w. Nevertheless, the property of $\tilde{f}(t)$ and $\tilde{g}(u)$ being isolated is kept, which is important for the control structure. The block diagram for the corrected model is shown in Figure 5.1.

5.3.2 Extending the Kernel

In order to identify both the free dynamics $\tilde{f}(t)$ and the input-function $\tilde{g}(u)$, a Gaussian process with two-dimensional input is needed. Because both parts contribute to the measured movement of the mount, they have to be learnt jointly. A covariance function $k(t_i, t_j, u_i, u_j)$ with two-dimensional input space $\mathbb{X}^2$ is needed. It should produce the same results as the original covariance function when the input u is zero and should model an input function $\tilde{g}(u)$ when t is fixed. This can be achieved by taking the sum of two distinct covariance functions

$$k(t_i, t_j, u_i, u_j) = k_\star(t_i, t_j) + k_{\mathrm{SEu}}(u_i, u_j) \tag{5.9}$$

where $k_\star$ is the periodic kernel presented in Equation 4.10 and k_{SEu} is a second squared exponential kernel, in u-space. This is possible because both parts that contribute to the

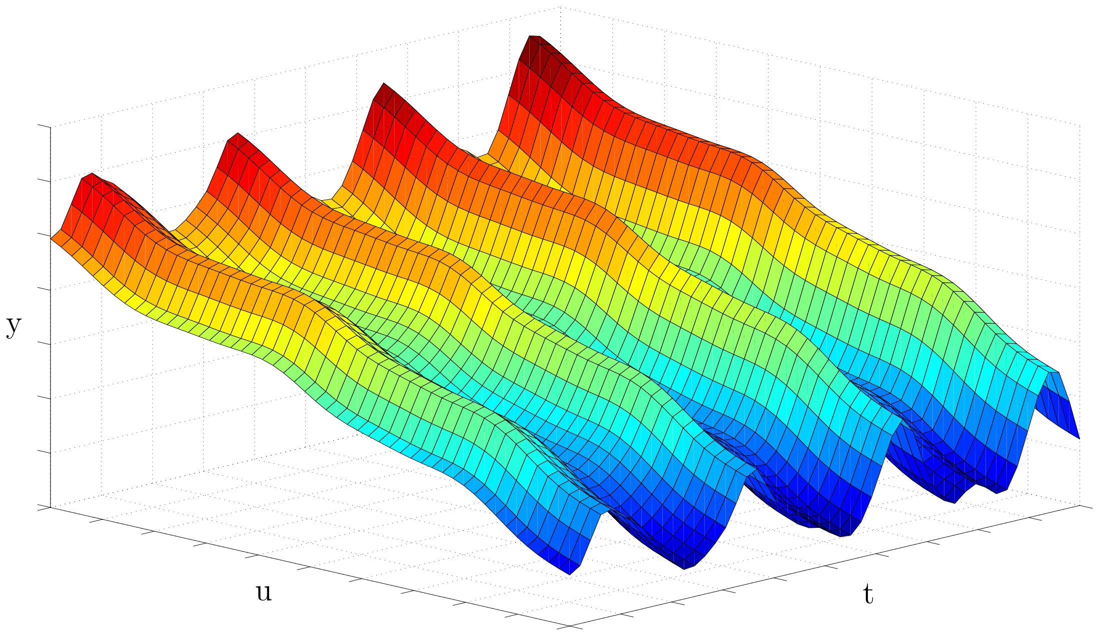


Figure 5.2: A Sample from the Extended Kernel

This figure shows the shape of a sample from a Gaussian process with the extended kernel according to Equation 5.9. For a better view, the smoother kernel k_{comp} is used instead of $k_\star$. The variables t and u form the input space, y denotes the function value.

dynamics are separable.

The meaning of such a construct of covariance functions is that two points in input space are considered similar, if they are similar under one *or* the other kernel. One sample of this combined covariance function is shown in Figure 5.2. The function values are plotted over t-u-space. The sample is approximately periodic in t-dimension and a nonlinear function in u-dimension.

5.3.3 Control Algorithm

The updated model allows inference on the complete the dynamics $d(t, u)$. Re-consider the definition of the dynamics

$$\dot{e}(t, u) = d(t, u) \tag{5.10}$$

whose finite difference approximation is

$$d(t, u) \approx \frac{e_{T+1} - e_T}{\Delta t} \tag{5.11}$$

which vanishes if

$$\underbrace{e_{T+1}}_{=0} = d(t,u) \cdot \Delta t + e_T \tag{5.12}$$

A local linearisation of $d(t,u)$ in u gives

$$0 = \Delta t \cdot (d(t,u_0) + \nabla_u d(t,u)|_{u_0} \Delta u) + e_T \tag{5.13}$$

This can be solved for Δu

$$\Delta u = -\frac{1}{\nabla_u \, d(t,u)|_{u_0}} \left(d(t,u_0) + \frac{1}{\Delta t} e_T \right) \tag{5.14}$$

which is a step of Newton-Raphson optimization towards the control signal u minimizing the mean predicted error. Iterative application of this rule asymptotically approaches the solution at quadratic rate [BSMM05].

Note that the term e_T is measured, whereas $d(t,u_0)$ and $\nabla_u \, d(t,u)|_{u_0}$ are predicted by the Gaussian process regressor. $d(t,u_0)$ is calculated according to Equation 4.4, the derivative $\nabla_u d(t,u)$ is obtained by derivating $\boldsymbol{k}_{u^*,\boldsymbol{u},x^*,\boldsymbol{x}}$ with respect to u^*

$$\nabla_u d(t,u) = \left(\frac{\partial}{\partial u^*} \boldsymbol{k}_{u^*,\boldsymbol{u},x^*,\boldsymbol{x}} \right)^{\mathsf{T}} K^{-1}_{\boldsymbol{u},\boldsymbol{u},\boldsymbol{x},\boldsymbol{x}} \cdot \boldsymbol{y} \tag{5.15}$$

The experiments showed that this controller does not improve markedly on the first one (see Chapter 6 for details), at least not in the standard case with known k_w. The qualities of this controller lie somewhere else: It is robust in terms of changing the system gain k_w or the input function $g(u)$. This function can be nonlinear, because the controller calculates the proper control signal with the Newton-Raphson method. It even can handle a negative input response: The user of this system could swap the motor's movement direction or the camera orientation and would nevertheless achieve good results.

5.4 Conclusion

This chapter proposed two different controllers for the telescope system. Both are based on the state feedback controller. In the continuous case, the controllers are always stable, but not in the discrete case: The control-gain has to be chosen with respect to the system gain, which might be unknown.

The first controller uses an estimate for the error of the uncontrolled system to predict the future error with Gaussian process regression. It works, but runs into problems when some parameters of the system are changed or unknown.

The second controller setup fixes this. In order to use it, the model of the system was corrected and the kernel extended to a two-dimensional input. With these changes, a second controller was built. It uses the Gaussian process regression to predict the future dynamics and uses Newton-Raphson optimization for calculating the control signal. This controller can adapt to different systems as not only the free dynamics but also the input function and the system gain are learned. It is a self-tuning P-controller that shows robustness, even in cases that would normally require the user to flip a switch.

Chapter 6

Experiments

6.1 Simulation

Because the telescope system moves slowly, tests on real hardware are time consuming. To accomplish a 30 minute test-run with three different controller-setups, the sky has to be perfectly clear for at least 2 hours. Perfect nights for astronomical observations are rare in Southern Germany. So, to allow rapid software prototyping, a simulation of the entire system was developed.

6.1.1 Simulink Setup

The simulation built for this purpose was built in MATLAB/SIMULINK. The basis is a SIMULINK version of the block diagram in Figure 2.3. Where possible, built-in function-blocks were used for efficiency. Complicated parts like the controller were implemented in m-files that are called from SIMULINK-blocks.

The function $\tilde{g}(u)$ was implemented as a linear function with variable slope k_g. The nonlinear gear function $\tilde{f}(x)$ was implemented as a one-dimensional lookup-table with linear interpolation. To make the problem realistic, the gear function was not sampled from a Gaussian process but was taken from a real measurement. The telescope mount is an integrator with initial value of zero. The simulated measurement is done by adding zero mean Gaussian noise with a standard deviation of 0.12 arcs. In the simulation, the measurement sample time of 3 s was chosen according to the real system. The used solver is `ode45` (based on Dormand-Prince [DP80]).

The simulation was created with the goal to produce data that looks similar to real

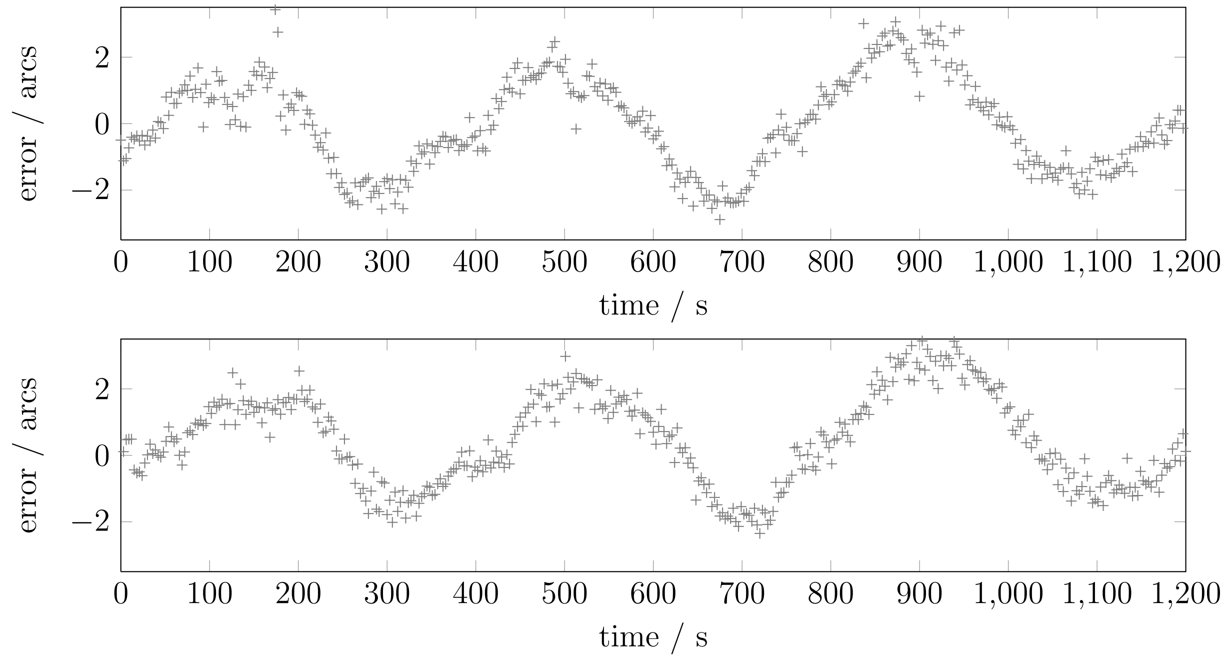


Figure 6.1: Measured Data from Simulation and Reality

These plots contain error data of the telescope system. One dataset is measured with the real telescope setup, the other is simulated.

data. Figure 6.1 shows the results for one real and one simulated run.

6.1.2 Experimental Setup

Four different controllers were compared in the simulation:

- The error based controller according to Section 5.2
- The derivative based controller according to Section 5.3
- Two basic P-controller as reference, optimized for two different settings

All controllers have a limited output $-1 \leq u \leq 1$.

The optimal control gain k_u for the P-controllers was determined with a grid search for different system gains k_g. $k_u = 1.05$ is optimal for $k_g = 0.5$ and $k_u = 0.5$ is optimal for $k_g = 1.0$.

To check if the controllers behave well under changes of k_g, each controller was tested with seven different values for k_g, including large and negative gain.

6.1.3 Simulation Results

Each experiment was run three times with different seed for the noise generator, but all experiments had the same set of seeds. The result of this tests is shown in Table 6.1. Note that the error values from the simulation are the error between telescope mount and reference position *before* adding measurement noise.

k_g	P ($k_u = 1.05$)	P ($k_u = 0.50$)	GPC-1	GPC-2
0.2	0.3483 ± 0.0064	0.6260 ± 0.0078	0.4684 ± 0.0066	**0.2225** ± 0.0197
0.5	0.2255 ± 0.0060	0.3053 ± 0.0061	0.2805 ± 0.0115	**0.2195** ± 0.0082
1.0	0.2982 ± 0.0070	0.2256 ± 0.0059	**0.1993** ± 0.0108	0.2233 ± 0.0075
2.0	1.6126 ± 0.0487	0.2879 ± 0.0069	**0.1983** ± 0.0090	0.2324 ± 0.0072
5.0	5.0864 ± 0.0899	4.8624 ± 0.0075	0.4117 ± 0.0058	**0.2693** ± 0.0082
−1.0	unstable	unstable	unstable	**0.2328** ± 0.0168
100.0	unstable	unstable	unstable	**0.2841** ± 0.0074

Table 6.1: Simulation Results for Different Controllers

The table shows mean ± standard deviation of the root-mean-squared error [arcs] from three runs of the simulation. The left column shows the used system gain k_g, the controller type is denoted by the abbreviations in the top row: simple P-controllers (P) with different gain k_u, one-dimensional Gaussian process controller (GPC-1), two-dimensional Gaussian process controller (GPC-2). To give the controllers time for learning, the error values are taken from $t = 200\,\mathrm{s}$ until $t = 1200\,\mathrm{s}$.

The controllers with the Gaussian process prediction show better control performance. This is sensible, as they use more information about the system than the other controllers. Interesting is the robustness under variations of the system gain k_g. All controllers which rely on a fixed and known gain have problems and become unstable on large or negative values for the input gain.

In contrast, the Gaussian process controller with two-dimensional input can adapt to large changes in the system gain. Especially the way the system reacts to negative gains is valuable: When a negative gain can be learned, the system is robust against issues like mounting the camera the other way round or accidentally swapping the turning direction of the motors. It also does not need any form of switch the user has to flip or a subroutine to determine the moving direction.

6.1.4 The Effect of Different Gear Functions

To analyse the effect of the prediction and the gear function further, different kinds of known gear functions were used for the simulation. As shown in Table 6.2, there are almost no differences. If the function is known beforehand and the controller uses the Gaussian process only for interpolation and not for extrapolation, the performance hardly differs.

The performance itself is a lot better when the function is known, but that is not really surprising.

$f(x)$	known	unknown
constant	0.1126 ± 0.0063	0.2050 ± 0.0124
sine wave	0.1126 ± 0.0062	0.2073 ± 0.0119
real gear	0.1140 ± 0.0062	0.2233 ± 0.0075

Table 6.2: Simulation Results for Different Gear Functions

The table shows mean ± standard deviation of the root-mean-squared the error [arcs] from three runs of the simulation. The used gear function is shown in the left column, the top row denotes whether the gear function was known in advance or not. Note that the flexible composite kernel is used for all simulations. Therefore, the results for simple gear functions are not as good as one might think. To give the controllers time for learning, the error values are taken from $t = 200\,\mathrm{s}$ until $t = 1200\,\mathrm{s}$.

6.2 Hardware

The goal of this thesis is that the provided concepts work on real hardware. Consequently, a full telescope setup for astronomical imaging was used to test the built controller in a realistic setting. The used setup is shown in Figure 6.2.

6.2.1 Hardware Specifications

The telescope setup used for the experiments consists of the following hardware:

Telescope: Meade LX200 ACF 12”: Schmidt-Cassegrain mirror telescope, 305 mm aperture, 3048 mm focal length (f/10 focal ratio), 0.380 arcs resolving power. [Mea]

Mount: Astro-Physics Mach1GTO German Equatorial Mount: Motors and gears included, ±3.5 arcs typical periodic error. [Ast]

Imaging Camera: Canon EOS 5D Mark II: Approximately 6.4 µm pixel size. [Canb]

Guiding Telescope: Canon EF 400 DO: Camera lens, 400 mm focal length. [Cana]

Guiding Camera: The Imaging Source DMK 41AU02.AS: 4.65 µm pixel size. [The]

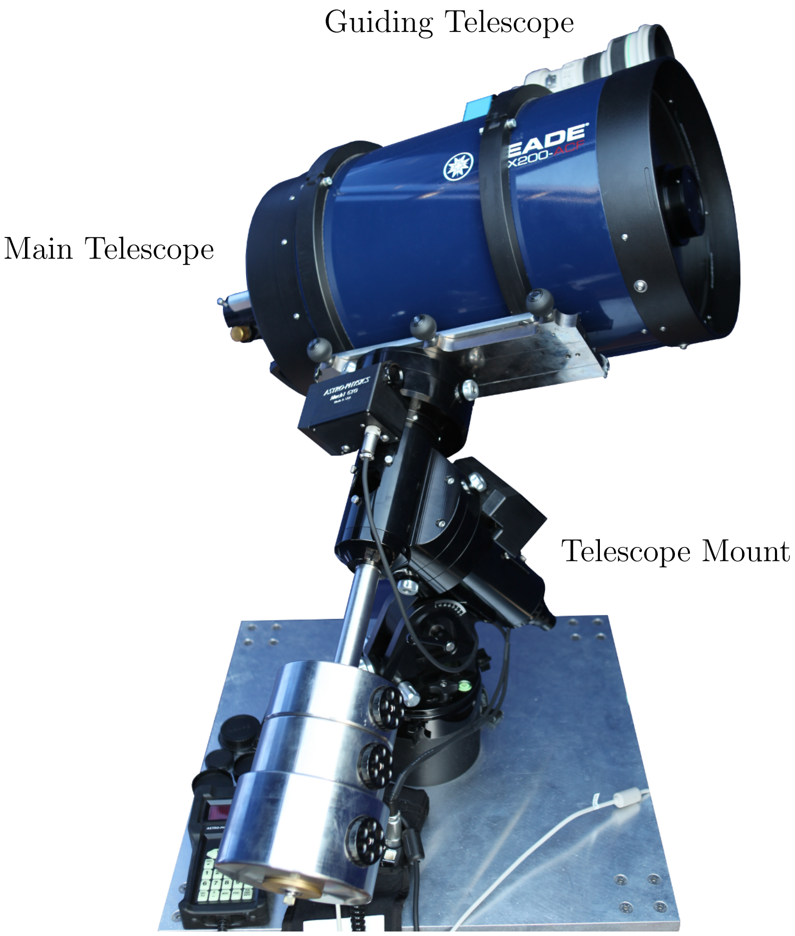


Figure 6.2: The Telescope Setup

This figure shows the mechanical and optical parts of the used hardware setup.

The pixel resolution of the telescope/camera combinations can be determined via calculating the ratio of the translation on the chip and the focal length. With the small-angle approximation, this is

$$\text{resolution}\left[\frac{\text{rad}}{\text{m}}\right] \approx \frac{\text{translation}}{\text{focal-length}} \tag{6.1}$$

With the pixel size and $1\,\text{rad} \approx 206\,265\,\text{arcs}$

$$\text{resolution}\left[\frac{\text{arcs}}{\text{pixel}}\right] \approx 206\,265\,\text{arcs} \cdot \frac{\text{pixel-size}\,[\frac{\text{m}}{\text{pixel}}]}{\text{focal-length}\,[\text{m}]} \tag{6.2}$$

which results in a resolution of $2.40\,\frac{\text{arcs}}{\text{pixel}}$ for the guiding camera and accordingly $0.43\,\frac{\text{arcs}}{\text{pixel}}$ for the imaging camera.

6.2.2 Hardware Setup

The telescope mount is located in a dome, on a stable rack on the roof of a building. The main telescope is attached to the mount and the guiding telescope is mounted "piggyback" on top of it. The complete system is balanced with counterweights and precisely aligned to the celestial north pole.

The interfaces between the parts of the system are more complicated, Figure 6.3 gives an overview. Matlab connects to the guiding camera through Ascom, which is a hardware driver layer for astronomical devices (both mounts and cameras). It can be used directly from Matlab. In Matlab, the pointing error is calculated from the image and the model is updated with the measurement. After that, the control algorithm is invoked to compute the next control signal.

The mount does not provide the ability to adjust the movement speed in a continuous way. It allows only to change the speed by a fixed amount, like half the sidereal speed. To move with an arbitrary speed, the mount therefore needs pulses of variable length. This can not be provided by Matlab, since it requires a second thread. A script was written in Python that creates those pulses. Matlab sends the desired speed via user datagram protocol (UDP) to the Python-script. This calculates the corresponding pulse-width, adjusts the speed of the mount and resumes the original speed after the pulse-width is reached.

The speed adjustment is done with a small piece of hardware (GPUSB) that can be accessed with Ascom and produces a signal on the 6-wire autoguiding port (ST-4), which is available on many telescope mounts.

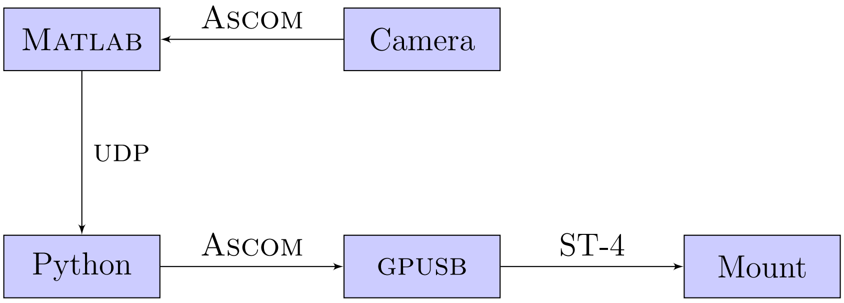


Figure 6.3: Hardware Overview

Connections between different hardware and software components in the controlled system. The boxes are the used components, the arrows show the used information channel.

6.2.3 Experimental Setup

The hardware was used to make images of an astronomical object: Messier 3, a globular cluster of stars at a distance of approximately 33 871 lightyears. After 10 min for settling of the system, every 30 seconds a 20-second-exposure was done, 30 in total per run.

Parallel to the exposures with the imaging camera, all images from the guiding camera were stored. From these images, the pointing error of the telescope could be re-calculated after the run was complete. With a runtime 1200 s and a measurement cycle of 3 s, this makes a total of 400 measurement points per run.

This was done for three different controller setups:

- Without active controller
- The error-prediction based controller according to Section 5.2
- A P-controller with Kalman-filter

6.2.4 Field Test Results

The set of experiments were run several times, but only once the external conditions (clouds, mechanical behaviour, seeing) were stable enough for direct comparison. Because it is not possible to measure without noise, the numbers in this section are from measurements *with* noise (as opposed to the simulation results in Section 6.1.3).

The results are shown in Table 6.3. The root-mean-squared error of the Gaussian process controller is 75 % smaller than without a controller and 22 % smaller than with the Kalman-filtered P-controller.

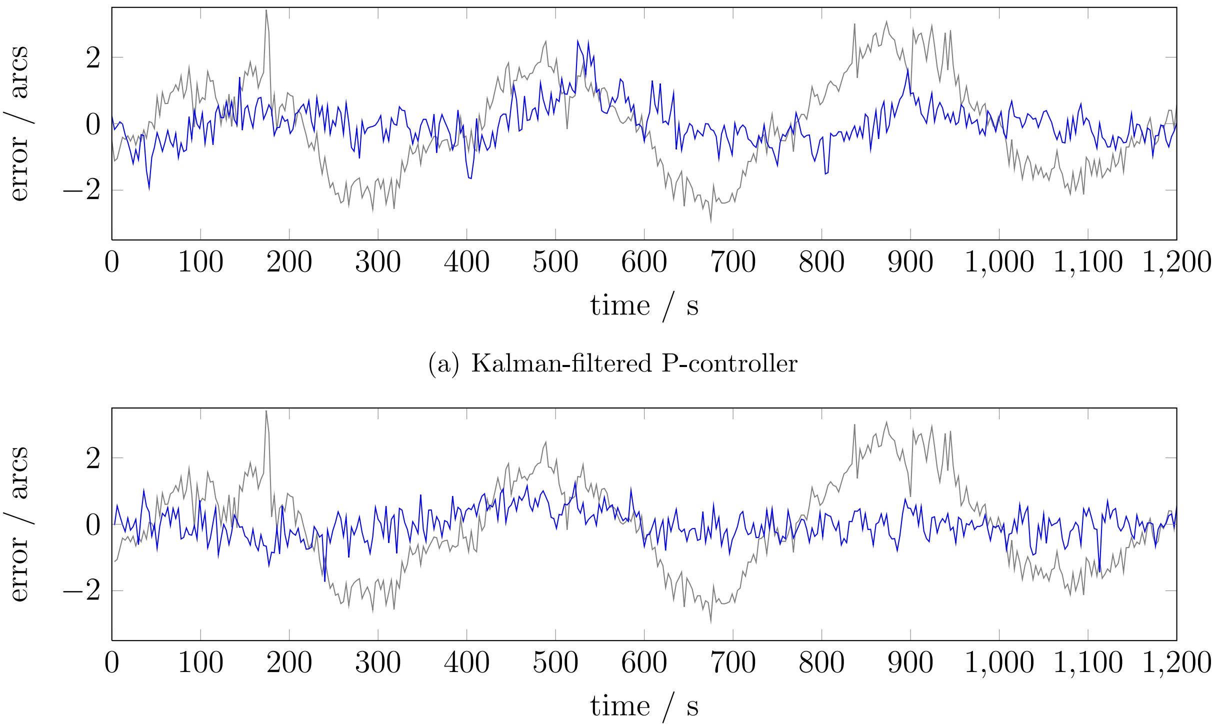
None	KF-P	GPC-1
1.4845	0.4842	0.3775

Table 6.3: Experimental Results of the Field Test

The table shows the root-mean-squared error [arcs] for different controller types, measured with the telescope system. To give the controllers time for learning, the error values are taken from $t = 200\,\mathrm{s}$ until $t = 1200\,\mathrm{s}$.

The plots of the error during the experiments are shown in Figure 6.4. They show little residual structure for the Gaussian process controller.

The images from the main camera provide an alternative, more qualitative means of comparison, Figure 6.5 compares these images. The image from the uncontrolled system



(a) Kalman-filtered P-controller

(b) Gaussian process controller, 1-D

Figure 6.4: Error Plots from the Hardware Experiments

Panel (a) shows the measured error of the telescope system with Kalman-filtered P-controller. Panel (b) shows the same, but with the one-dimensional Gaussian process based controller. In both plots, the error of the uncontrolled system is shown in grey.

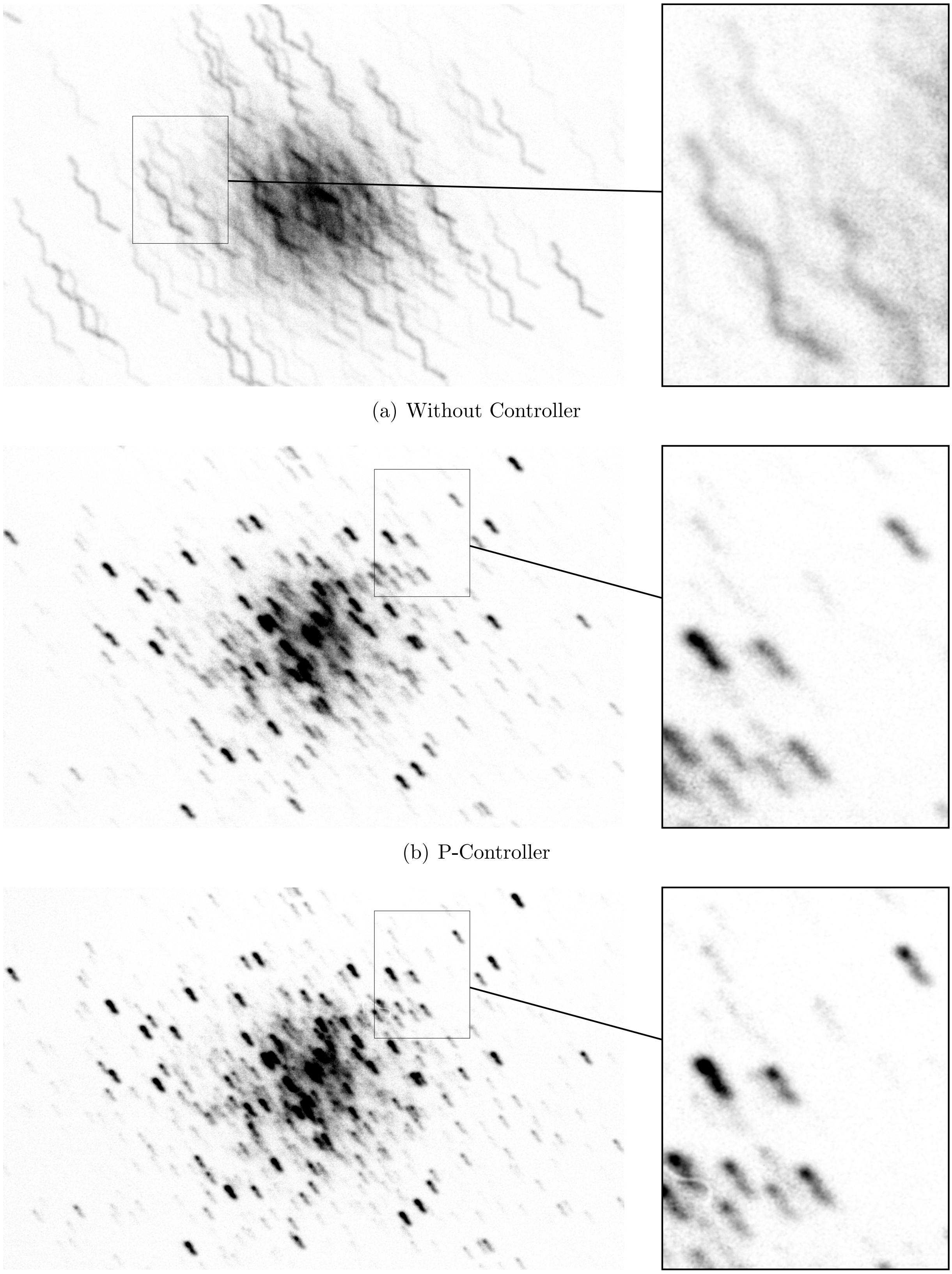
shows the periodic movement: Instead of as points, stars appear as wavy lines. The difference between Gaussian process controller and Kalman-filtered P-controller is small but noticeable.

6.3 Conclusion

To test if the controllers built in the last chapter work, a simulation was developed in Simulink as direct implementation of the block diagram of the system. The simulation was built upon measured values for the gear function, so that a simulated run looks similar to a real experiment.

After introducing the used controllers, the results from the experiments were presented. The improvements of the Gaussian process based controllers over basic controllers was shown. Also, the effect of different gear functions and prediction levels was presented: If the function is known in advance, the performance is good, no matter what gear function is used.

The improvements of the Gaussian process based controller do not only hold for the simulation case. The controllers were also tested on a set of hardware, a complete setup for astrophotography. The setup is involved, and explained in detail in the section. The results from the test in the real environment showed that the Gaussian process based controller also works in hardware. Furthermore, it was employed to make images of an interesting astronomical object, which is the intended use of such a setup.



(a) Without Controller

(b) P-Controller

(c) GP-Controller

Figure 6.5: Images of Messier 3 with Different Controllers

All images show Messier 3, the used controller differs. Panel (a) shows the result without controller, Panel (b) shows the result with a Kalman-filtered P-controller and Panel (c) shows the result with the Gaussian process based controller. The images show the summed-up pixel intensity values of 30 subframes (Panel (a)) respectively 15 subframes (Panels (b) + (c)) with 20 s exposure time each. The images are inverted and reduced to black and white for better comparison. The drift is not fully compensated, so that the movement of the right ascension axis is easier to see.

Chapter 7

Summary & Conclusions

For long exposure images of faint objects in the sky it is crucial that the stars do not move with respect to the telescope. Because Earth rotates, from the observers point of view the stars follow a circular trajectory around the sky. Thus, machinery is required that moves the telescope along this path. The mechanical part always has limited precision. This leads to a periodic error in the pointing direction, resulting in blurred star images.

Although closed-loop-controllers that rely on a guide star are standard in astrophotography, the periodic error itself is mostly addressed with time-consuming error recording and adjustment protocols. Because this is complicated, it is often omitted. We chose another way and asked the question: Is it feasible to use Gaussian process regression to infer a model of the periodic error during runtime?

Summary

System identification is a general problem in control. In this work, a method popular in the machine learning community, Gaussian process regression, was employed for system identification. The application is a realistic problem, correcting the gear error of a telescope. Interesting aspects are that the error shows a periodic structure and that the system identification is done while the controller is already active.

A mathematical model of the telescope mount was developed. The nonlinear parts were isolated during the modelling to work them out separately. One nonlinear function was linearised for simplification, the other one could not be linearised because of a constantly changing working point. We used a black box system identification method for that part.

The system identification for the nonlinear gear function is done with Gaussian process regression during runtime. Because the standard setup only gives weak predictions for the future, a specialized kernel function was developed. Based on a periodic kernel, a composite kernel was built that can capture periodic effects, changes in the periodicity over the time and small nonperiodic effects.

With the prediction from the Gaussian process, a state-feedback controller was built to keep the error small. Because the first approach did not work as expected, the model of the system was changed. The kernel was extended to be able to model gear function and input function jointly. Experiments in Simulink showed that this extension is more robust. Furthermore, the performance of the designed controller is better than the performance of a compared P-controller with optimized gain. Hardware experiments showed that this controller design with nonparametric system identification during runtime is feasible.

Aside of the control part, an image registration technique was developed for fast measurement of the error. This was necessary because the standard approach is too slow at the needed resolution. The developed method is based on finding tie points in the images by means of geometric measures. Experiments showed that it is faster and more accurate than the standard technique, but it is specialized to a certain kind of images.

The goal of this work was to build a controller that learns about the system while controlling and that uses the learned information to improve the control performance. This was successfully done, and also the goal of keeping the pointing error of the telescope system below the camera's resolution of 0.43 arcs was achieved. In the field test, the Gaussian process based controller had only a root-mean-squared error of 0.38 arcs.

Conclusions & Personal Remarks

This thesis shows that it is not only possible to build a controller upon a nonparametric model, but also that the performance of the resulting controller is respectable. It improves upon an optimized P-controller and is self-tuning on a wide range of unknown input gains. This shows the benefit of Gaussian process regression for system identification on unknown hardware. Due to the flexible model and the feasibility of regression and prediction during runtime it is possible to design a controller for a whole class of hardware.

The key ingredients that helped achieve this are:

Gaussian process regression: Serves as a flexible modelling class for nonlinear problems but also allows the explicit encoding of prior knowledge like the periodicity of the learned function.

Kernel engineering: With a standard kernel it would not be possible to make strong predictions in this setup, and with merely a periodic kernel the predictions would not be sensible. It is a valuable insight that choosing the right kernel (or kernel combination) for a certain problem leads to better results than taking a classic kernel "off the shelf".

Image registration by triangle matching: Can be used as an extension for almost all autoguiding software in astronomy. Because the widely used software for autoguiding use only one star measurements, they tend to be fragile. Finding the position by geometric measures and several stars is more robust and works even when the field of view is partly covered (with a cloud, for example).

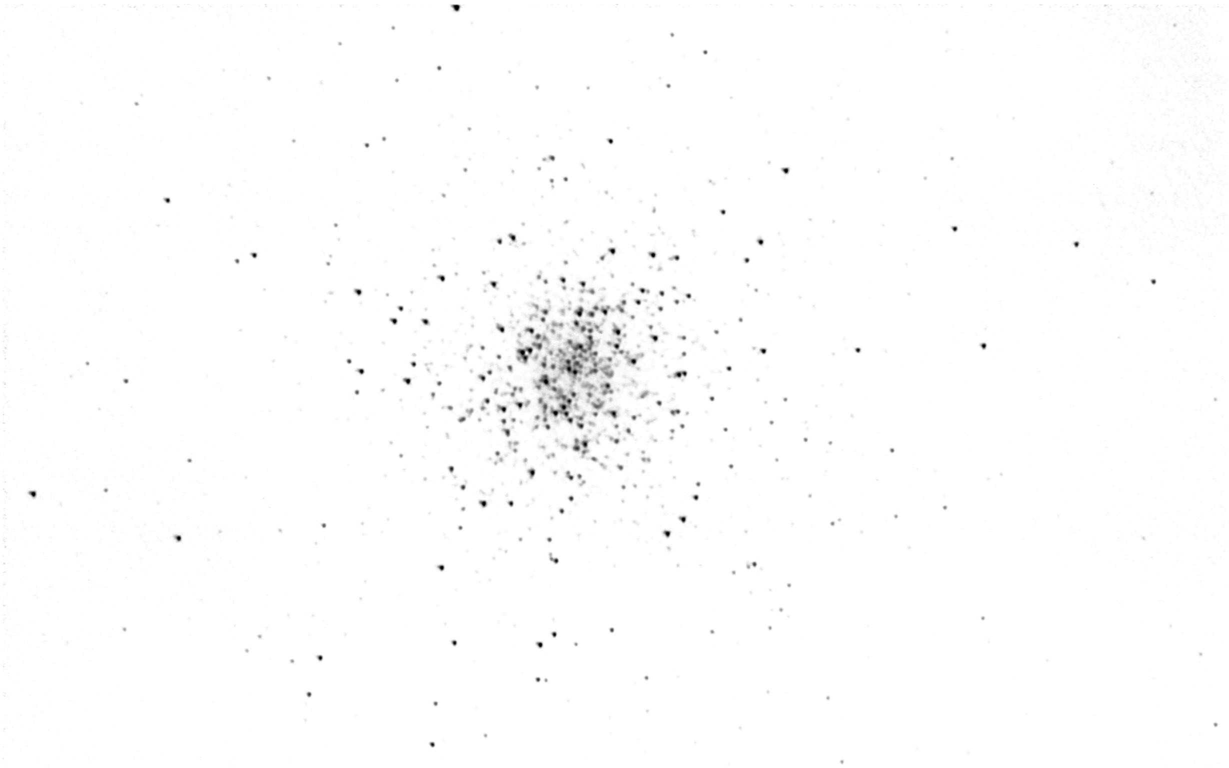
This work is also of practical use: With more time for engineering we can build an autoguiding system that is robust against many kinds of technical flaws, like wrong parameters for the guiding speed or inverse motor movement. This is a convenient feature, because setting up and debugging a telescope system is a time-consuming activity. With a system that is simple to set up, there is more time to capture the beauty of the universe, like the star cluster Messier 3 in Figure 7.1.

Besides the use for telescopes, we are looking forward to use methods from machine learning for all kinds of control problems. Much work remains to be done until the control and communities are connected as they should be.

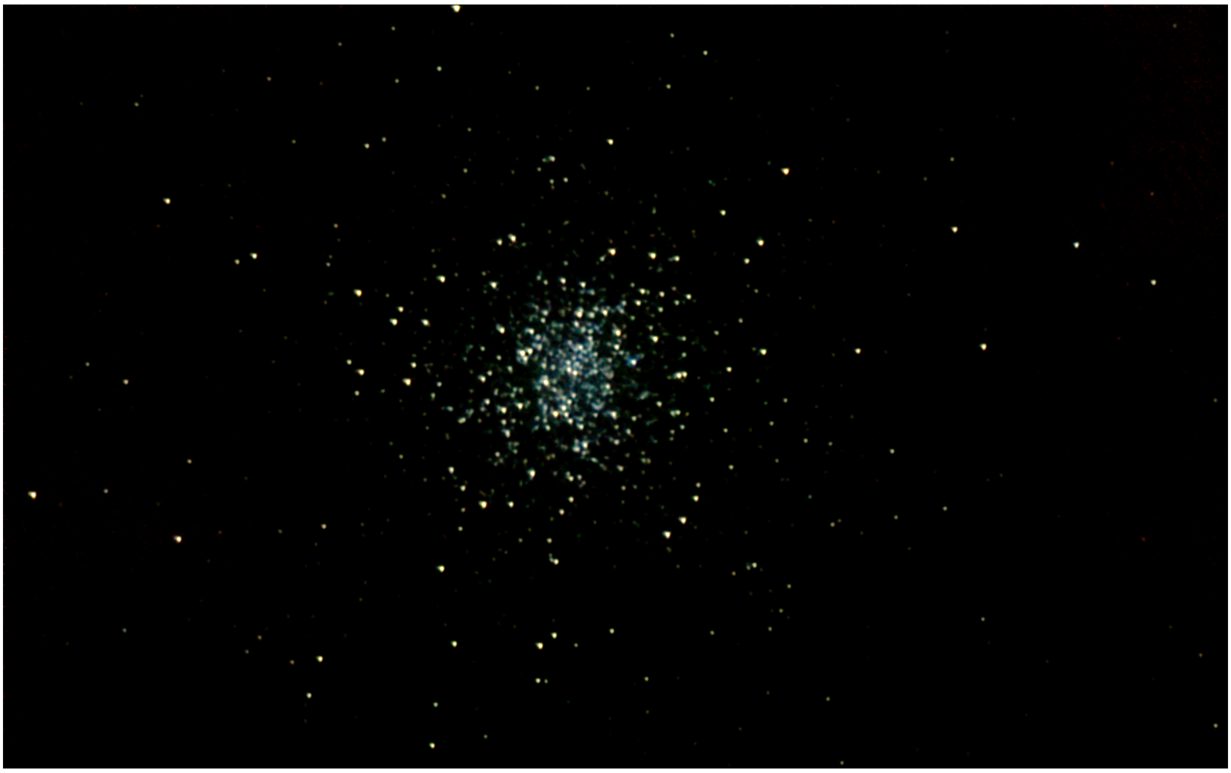
[1] http://www.astronomie.be/registax/

[2] http://cyanogen.com/maxim_main.php

[3] http://www.gimp.org/



(a) Inverted Monochrome Image



(b) Full Colour Image

Figure 7.1: Messier 3 in all its Beauty

This image (once inverted, once in colour) shows Messier 3. The image was combined from 15 sub-frames with 20 s exposure time each. The guiding during the exposures was done with the Gaussian process based controller. The raw images were post processed in several steps: Stacking and wavelet filtering with RegiStax[1]; denoising and deconvolution with MaxIm DL[2]; colour balance and contrast/brightness with GIMP[3].

Bibliography

[Ast] Astro-Physics, Inc., "ASTRO-PHYSICS Mach1GTO German Equatorial Mount with GTOCP3 Control Box," accessed: 2012-06-29. [Online]. Available: http://www.astro-physics.com/products/mounts/mach1gto/mach1gto.htm

[Bar11] D. Barber, *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2011.

[Bis06] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[BM92] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[BSMM05] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig, *Taschenbuch der Mathematik*, 6th ed. Verlag Harri Deutsch, 2005.

[Cana] Canon Europe Ltd, "Canon EF 400mm f/4 DO IS USM," accessed: 2012-07-18. [Online]. Available: https://www.canon-europe.com/For_Home/Product_Finder/Cameras/EF_Lenses/Telephoto/EF_400mm_f4_DO_IS_USM/

[Canb] ——, "Canon EOS 5D Mark II," accessed: 2012-07-18. [Online]. Available: https://www.canon-europe.com/For_Home/Product_Finder/Cameras/Digital_SLR/EOS_5D_Mark_II/

[CT65] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Computation of the Complex Fourier Series," *Mathematics of Computation*, vol. 19, pp. 297–301, Apr. 1965.

[DP80] J. Dormand and P. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[GSTF08] M. Guizar-Sicairos, S. Thurman, and J. Fienup, “Efficient subpixel image registration algorithms,” *Optics Letters*, vol. 33, no. 2, pp. 156–158, 2008.

[GW02] R. Gonzalez and R. Woods, *Digital image processing.* Prentice Hall, 2002.

[HBL$^+$08] D. W. Hogg, M. Blanton, D. Lang, K. Mierle, and S. Roweis, “Automated Astrometry,” in *Astronomical Data Analysis Software and Systems XVII*, vol. 394, Aug. 2008, p. 27.

[HS52] M. R. Hestenes and E. Stiefel, “Methods of Conjugate Gradients for Solving Linear Systems,” *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436, Dec. 1952.

[Lju10] L. Ljung, “Perspectives on system identification.” *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

[LW05] H. Lutz and W. Wendt, *Taschenbuch der Regelungstechnik.* Verlag Harri Deutsch, 2005.

[Mac98] D. MacKay, “Introduction to Gaussian processes,” *NATO ASI Series F Computer and Systems Sciences*, vol. 168, pp. 133–166, 1998.

[Mat60] B. Matérn, “Spatial Variation: Stochastic Models and their Applications to Some Problems in Forest Surveys and Other Sampling Investigations,” *Meddelanden från Statens Skogsforskningsinstitut*, vol. 49, pp. 1–144, 1960.

[Mea] Meade Instruments Corp., “LX200 Series - Specifications,” accessed: 2012-06-29. [Online]. Available: http://www.meade.com/lx200/specifications

[MVV97] C. Munoz-Tunon, J. Vernin, and A. M. Varela, “Night-time image quality at Roque de los Muchachos Observatory,” *Astronomy & Astrophysics Supplement Series*, vol. 125, pp. 183–193, Oct. 1997.

[RN03] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Pearson Education, 2003.

[RW06] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[The] The Imaging Source Europe GmbH, “The Imaging Source DMK 41AU02.AS,” accessed: 2012-06-29. [Online]. Available: http://www.astronomycameras.com/products/usb/dmk41au02as

[Tö05] K. D. Tönnies, *Grundlagen der Bildverarbeitung.* Pearson Studium, 2005.

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich, Edgar Klenske, die vorliegende Diplomarbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäß entlehnte Stellen als solche kenntlich gemacht habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum Unterschrift