Bachelor-Thesis

High-Speed Motion Tracking for Robot Control

Bachelor’s Thesis

Spring Term 2012

Supervised by:
Keith Gunura
Nandan Maheshwari

Author:
Dominik Eugster
## Contents

1 Introduction ........................................ 1

2 Problems ............................................. 3  
  2.1 Unavailability of Cheap Recording Software .......... 3  
  2.2 Tracking Monochrome Structures in any Background .... 3  
  2.3 Software-User Interaction ........................... 3  

3 Methods .............................................. 4  
  3.1 Image Access .................................... 5  
    3.1.1 Hardware .................................... 5  
    3.1.2 Software and Drivers ........................ 5  
  3.2 Image Processing ................................ 7  
    3.2.1 Choosing the appropriate Feature Extraction Algorithm .... 7  
  3.3 Speed-up using ROI ............................... 8  
  3.4 Data Limits ...................................... 10  
  3.5 Multicore Processing ............................. 10  

4 Results .............................................. 11  
  4.1 Recording ....................................... 11  
  4.2 Tracking ........................................ 12  
    4.2.1 On-line Tracking ............................ 12  
    4.2.2 Off-line Tracking ............................ 12  
    4.2.3 Background independence .................... 12  

A Quick User Guide .................................. 15  
  A.1 BIRLRecorder .................................... 15  
  A.2 BIRLTracker ..................................... 17  

Bibliography .......................................... 18
Chapter 1

Introduction

In robotics, especially working on walking or jumping robots, one needs to analyze a robot’s motion. For example, eigenfrequencies of an elastic part can be found by analyzing its motion. The method that was used at the lab before this work used long exposure with a photo camera. An example is depicted in Figure 1.1. For this goal, LEDs have to be mounted on the robot and the photo is taken in a darkened room. Powering and mounting the LEDs leads to additional work and could influence the motion of the robot.

To improve this situation, a software was developed containing the following features:

- real-time point tracking
- high-speed recording of a motion sequence
- non-invasive point tracking with path coordinates output

In this thesis, various methods are presented to achieve these goals with the help of a high-speed camera.
Figure 1.1: LED long exposure of a hopper robot. Only quantitative observations can be made. No temporal measurements can be taken.
Chapter 2

Problems

In this chapter, we present some of the problems that arise when dealing with a high-speed motion tracking problem.

2.1 Unavailability of Cheap Recording Software

Many relatively cheap high-speed cameras have a relatively low signal-to-noise ratio which can be problematic for feature extraction. Especially in dark environments, noise will be present. The Dalsa Genie used in this context provides almost noise-free images.

A huge drawback although is the GigE Vision interface used by this system. GigE is an interface standard based on TCP/IP. The standard is not open but can only be accessed by members of the Automated Imaging Association (AIA). This means that only members of the AIA can develop compatible products and no open source software will be available. The manufacturers usually provide their own visual libraries which can be quite expensive.

2.2 Tracking Monochrome Structures in any Background

Point tracking using color is one of the simpler routines in computer vision. The Dalsa Genie only provides 16bit monochrome images which makes this method unusable. The method could be easily adapted to monochrome images where background and robot are clearly separable by their brightness. As this technique needs artificial lighting, it is not usable in arbitrary background. Our software should be operable in any context with any lighting.

2.3 Software-User Interaction

Many tracking softwares will need some user interaction between the start of the recording and the tracking, for example to denote the point to track. In this case it will not be possible to track a point in real-time.
Chapter 3

Methods

Recording and/or tracking consists of different task described in Figure 3.1. The methods used to master these problems are described in the following.

![Diagram of processing pipeline](image)

Figure 3.1: processing pipeline of the tracking software. The methods used for these tasks are described in chapter 3

![Diagram of image access scheme](image)

Figure 3.2: Image access scheme of the used hard- and software. The method uses both commercial and open-source software.
3.1 Image Access

The used recording software and drivers will highly depend on the available hardware, in our case a Genie HM640. Its basic specifications are shown in Figure 3.3. Detailed descriptions can be found on the Dalsa webpage. Figure 3.2 shows the packages and interfaces described in the following.

3.1.1 Hardware

The usability of a camera is defined by multiple factors. Too much noise makes it hard to operate especially at low lighting. The Dalsa Genie is equipped with a modern and relatively expensive CCD sensor which provides a very good image quality. In Figure 3.4, image qualities of CCD and the much cheaper CMOS sensor are compared.

As parts of robots can reach quite remarkable speed, motion blur needs to be considered as it makes efficient object recognition impossible. The Dalsa Genie HM640 allows to reduce exposure time down to 1ms which reduces motion blur to a minimum. A direct comparison between the usual 10ms and 1ms of exposure time can as well be observed in Figure 3.4.

3.1.2 Software and Drivers

The camera comes with a drivers package which includes:

**Sapera LT CamExpert** Driver Package for Dalsa GigE vision devices

**Common Vision Blox by STEMMER IMAGING** Commercial computer vision library, available package including image retrieval and saving. Further packages such as feature extraction and video generation are not included in the basic Image Manager. The library is available for C++ developers.

Currently there are no alternatives to these libraries, so they were included in this project.
Figure 3.4: Image quality comparison of different camera types. The Genie HM640 allows much shorter exposure which reduces motion blur. In comparison to the CMOS sensor of the webcam, the CCD sensor of the Genie has a much higher signal-to-noise ratio.
3.2 Image Processing

A widely used, efficient and free computer vision library is OpenCV. A big number of algorithms is implemented in this library which is distributed under a BSD license. OpenCV has interfaces for C, C++ and Python programming. The whole code of this project is written in C++. As the representation of images are similar in OpenCV and CVB, the data pointer of the OpenCV image can be pointed to the location of the acquired CVB-image. Line and column increments are computed and sent to OpenCV.

### 3.2.1 Choosing the appropriate Feature Extraction Algorithm

A suited algorithm for this application will need:

- **highly distinctive feature descriptors** As we want the software to run in an arbitrary background, we need the descriptors to be very distinctive. Regions of the background may be similar to the feature tracked.

- **rotational invariance** A robot may have different orientations during the movement. The position should be tracked independently of orientation.

- **scale invariance** Searching to minimize the interaction between software and user, the implementation should allow to process recordings from different distances to the robot without further user input.

- **efficiency** For a real-time image analysis, the algorithm needs to be highly efficient.

As the comparison in Figure 3.5 shows, both SIFT and SURF would satisfy our needs. In different environments and lighting conditions SIFT is slightly more accurate than SURF as shown in [2] and [3]. As speed is crucial for high-speed tracking, SURF are chosen. This method has been implemented as a demo in the openCV documentation and is adapted to our needs.

SURF is a state-of-the-art scale- and rotation-invariant detector and descriptor that was developed at ETH Zurich and K.U.Leuven and presented at ECCV in 2006 [1].

---

### Table 3.5: Comparison of popular feature extraction algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Rotational Invariance</th>
<th>Scale Invariance</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template Matching</td>
<td>+</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Harris Corners</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>SURF</td>
<td>+</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>SIFT</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
</tbody>
</table>

Figure 3.5: Comparison of popular feature extraction algorithms. Due to its efficiency, SURF was preferred over SIFT. Less popular and more specific algorithms are omitted in this chart.

Figure 3.6: Patterns in typical resolution and their corresponding SURF descriptors, depicted as red circles. Working with a circle requires least computational power as only one descriptor is to be matched. Symmetry although disables orientation detection.
Figure 3.7: Schematic explanation of the Region Of Interest (ROI). In this work, the ratio between original image and ROI is called ROI size.

In principle, the tracking works for arbitrary patterns, but is most efficient for patterns which generate only one SURF descriptor. If only the position and not the orientation of the pattern is required, one can use a pattern containing rotational symmetry. The simplest pattern used in this context is a circular shape. Figure 3.6 shows different patterns with their corresponding SURF descriptors.

### 3.3 Speed-up using ROI

When recording a robot at high frame rate, the point of interest will probably be found in proximity of the previous position. A region of interest (ROI) can thus be defined around the expected position. The feature extraction algorithm will then only analyze the image within the defined ROI. If there is no success, the size of the ROI is then increased and feature extraction done on this bigger ROI. The scaling factor of 2 proved to be appropriate.

When setting the ROI size to a value of 0.1, the processed image will then be a factor 100 smaller than the original image. We expect a speed-up of 100 at feature extraction and feature matching.

The actual performance at different initial ROI sizes is discussed in the following. As the ideal value will highly depend of the point's velocity and the size of the pattern, three different case studies are performed and shown in Figure 3.8. In the three cases, the optimal value lies between 0.07 and 0.1. As expected, we gain a speedup of a factor of 100.

In a second version, the motion between three frames is assumed to be constant and the expected position computed from the two previous position. This allows even smaller ROI and thus increases speed. For the tested recording in this thesis, the effect was marginal, probably due to the relatively low velocity of the robot.
3.3. Speed-up using ROI

Figure 3.8: Performance study on ROI size. Three different recordings on different robots were analyzed. We can observe a speedup by a factor of 100 and an optimal ROI size of around 0.1.
3.4 Data Limits

Recording with a 640 × 480 pixel monochrome camera at 300 fps creates an amount of data that reaches 92MB per second. A today's CPU has no problem to cope with this amount of data, whereas traditional harddisks have a data rate of about 50 MB/s. When using the camera for robot control we may only need the current position of the object and it is thus not needed to save the image. The only constraints are therefore image access time and the speed of feature extraction and feature matching.

If although we want to save each image at a frame rate of 300 fps, the traditional harddisk will be too slow. In the following, three solutions to this problem are discussed.

SSD Flash-based solid-state-disks are modern disks with a minimum writing speed of up to 400 MB/s. The disadvantage of the SSD is their high cost in comparison to traditional HDD. They are mostly used in mobile devices.

RAM An easy possibility is to store all images in the RAM before writing them on the HDD. The main disadvantage is the size of the RAM: 1GB will be filled in about 3 seconds at a frame rate of 300 fps.

Internal memory High-end cameras often offer internal flash memory where acquired images can be buffered. Using this internal buffer, the recording is quite stable at 300fps but no reference about the buffer size of the HM640 was found.

First, the RAM solution was adapted in this work which was not very stable for longer recordings due to memory size. Using the internal memory of the camera, recordings over 20 seconds and more are easily feasible and this possibility was finally used.

3.5 Multicore Processing

Parallel computing must be considered for programs where speed is a major goal. One can imagine to process multiple images on multiple cores simultaneously. As the major speedup results from the ROI method, it is important that the current position of the pattern is predicted as precisely as possible. We therefore have a data dependency between subsequent images which is one of the major problems in parallel processing.

This possibility of parallelization has been tested for this software using openMP and proved to behave as expected. No significant speedup has been observed. To improve the efficiency of the code, the feature extraction and feature matching algorithm would need to be parallelized which exceeds this work.
Chapter 4

Results

For recording and tracking, a simple-to-use GUI was developed and presented in Appendix A. All performance studies were calculated on a Intel i3-380M Dual-Core Processor using 4MB DDR3 SD Ram and a 5400 rpm SATA HDD.

4.1 Recording

The developed User Interface allows to record videos in a simple way. The output is either in a video file, the uncompressed bitmaps or both. The software does not always run stable, the reason for this remained unclear. In one of two cases although the software works in the manner needed. We are able to use the full range of frame rates that the camera offers, i.e. up to 300fps. If enough space on the harddisk, recordings up to 30 seconds at 300 fps can be made. When recording for longer periods we assume the frame rate to be lower and therefore the number of images not exceedingly high. Videos of test recordings are found on the annexed CD.

Figure 4.1: Graphical interface of the recording tool
Chapter 4. Results

4.2 Tracking

![Tracker GUI](image)

Figure 4.2: Tracker GUI

4.2.1 On-line Tracking

The images acquired by the camera are instantly analysed and the current coordinates of the pattern written to a file. Using the presented methods, on-line image analysis works at a frame rate of up to 300fps as desired.

4.2.2 Off-line Tracking

The frames that were previously recorded as bitmap files are loaded and analyzed. The pattern coordinates of all previously processed images are connected to a piece-wise linear track. This track is painted on the current frame which is saved in a separate folder and optionally drawn into a video file. Loading and saving images slows the process down to about 100 files per second which is still quite fast.

The tracking software has been used in the work of M. Reis [4] on curved beam hoppers where the optimal setup of the hopper was to be found. Two of the trackings are shown in Figure 4.3.

4.2.3 Background independence

As described previously, the tracking software is required to work in any environment. The background independence has been examined for different environments and an example is shown in Figure 4.4. Even in this partly structured, noisy environment, the tracking algorithm works almost perfectly.
4.2 Tracking

Figure 4.3: Comparing different configurations of a curved beam hopper of the BIRL at ETH Zurich. The analysis was used to observe different behavior at different frequencies and different weight setups. [4]
Figure 4.4: Background independence of the algorithm: using SURF descriptors, tracking works perfectly even in a semi-structured environment.
Appendix A

Quick User Guide

A.1 BIRLRecorder

1. Make sure that the following software is installed on your system:
   - OpenCV Version 2.1 or later
   - Common Vision Blox
   - Sapera LT Drivers

   A CVB Getting Started Guide can be found on www.commonvisionblox.com.

2. Open CVB Management Console to adjust the following parameters:
   - Set the desired frame rate
   - Set the Buffer Size to 10000
   - When grab is enabled, the camera’s focus can easily be adjusted.
   - The brightness can as well be adjusted on the camera’s objective

   The Ethernet connection will not work when the connected to a virtual private network (vpn).

3. Close the Management Console and run the BIRL Recorder executable
   - Choose name and length of recording
   - Set the frame rate of the camera. This must be the same value you set in the Management Console.
• Choose the type of output you need. When recording an .avi file, set a lower frame rate for slow-motion. For further object tracking, it is advantageous to record to Bitmap images.

If the executable does not open, make sure that your camera is correctly connected to the Ethernet card of your computer and that no other application (for example the Management Console) is using the camera.

4. Click on 'Record' and wait until 'recording successful' appears. You will find the recording in your current folder.
A.2 BIRLTracker

1. Open the Tracker GUI and set the following parameters:

   - name of the recording you have already done
   - name of the pattern Image. This needs to be in the same folder as the executable and the recorded images.
   - Setting the frames that are to be analysed allows to choose only the sequence where the pattern is present in the recording.
   - avi output will trace a colored line along the extracted path. Select a lower framerate for slow-motion
   - bmp output will save all images of the video as bitmaps
   - txt output will write the camera coordinates of the pattern center to a textfile

2. click on 'Track' and wait until 'tracking successful' appears. You will find the output files in your current folder.
Bibliography


