Proc. of WESIC 2001, University of Twente, The Netherlands, 27-29 June 2001

MOTION CONTROL WITH VISION

Peter Boots and Dick van Schenk Brill

Fontys University of Professional Education, IPA Research Centre
P.O. Box 347, 5600 AH Eindhoven, The Netherlands
Email: P.Boots@fontys.nl, D.vanSchenkBrill@fontsy.nl

Abstract

This paper describes the work that is done by a group of I3 students at Philips CFT in Eindhoven, Netherlands. I3 is an initiative of Fontys University of Professional Education also located in Eindhoven. The work focuses on the use of computer vision in motion control. Experiments are done with several techniques for object recognition and tracking, and with the guidance of a robot movement by means of computer vision. These experiments involve detection of coloured objects, object detection based on specific features, template matching with automatically generated templates, and interaction of a robot with a physical object that is viewed by a camera mounted on the robot.

1 The I3 project

The I3 project involves the implementation of a model for education in innovative engineering with industrial coeducation. The model (Bakker et al, 1999) was developed by three knowledge transfer centres affiliated to the Fontys University of Professional Education (technological departments). Those were the centre for Medical Technology, the Environmental research centre and the IPA (Integrated Production Automation) research centre. In short the model implies, that students from different departments will fulfil the last 18 months of their (polytechnic) studies fully in industrial practice. They have to work in project teams that are innovative, interdisciplinary and internationally oriented (the three I's). Within the project teams modern ways of working are applied as described in (Kollenburg et al, April 2000 & September 2000). There are multiple I3 student groups working simultaneously in different companies; one of which at Philips CFT (Centre for Industrial Technology) Eindhoven, department of Industrial Vision (IV). At this centre the I3 project team is involved in developments in the field of motion control with vision. The work of this team and their preliminary results will be described further in this paper.

2 Project background

The goal of this project at CFT-IV is to gain experience in controlling the motion of a robot by using a vision system. Normally, such a movement is controlled with sensors that measure the actual position of each individual joint. A feedback loop then locks the motors into predefined positions (called 'setpoints'). A setpoint generator is used for generating consecutive positions, e.g. for following a smooth spline path. The path followed by the robot arm is strictly defined by those setpoints, and therefore this approach, although very common, has some drawbacks:

- When the position or orientation of an object can vary from one situation to the other, it can be difficult to program the robot-movements in such a way that the object is always grasped correctly.
- When the situation is such that the object is moving randomly, it is difficult to follow the object with a robot.
- In order to achieve a reasonable accuracy, it is always necessary to calibrate the robot.
- Extensive computations are necessary for transforming real-world co-ordinates to robot joint positions.

These problems can possibly be solved when a vision system is used for controlling the robot instead of, or in addition to, the normal sensors.
This paper describes the main results achieved by the project team thus far.

3 Project goals

There are two practical cases that the project team is working on. These are described here shortly and elaborated upon in the following sections.

- The "RoboCup" project.
  RoboCup is a competition between teams of mechanical football-players. These teams must be able to play a football game against other robot teams, with of course some adapted rules. The goal is to create in the long run a team of robots that can play and win a football game against a human team.
  A vision system has to be developed that will be able to recognise and localise the ball, the goals, the players of the two teams and the borders of the playing field. With these data, the robot must be able to compute its own position, the speed of the ball, the best way to hit the ball, the best trajectory from the current ball position to the opponent goal, etc (see Iakovou 2001 for an extended description). The vision module is therefore part of a much larger system that is not considered here.
- The "JuggleBot" project.
  Here, the intention is to let a robot interact with an object, in this case a ball. The project will evolve in a number of steps of increasing difficulty: first the robot follows the ball, then it pushes the ball into a hole and finally it keeps the ball in the air by bouncing it with a small bat. The emphasis is on speed and accuracy.

As mentioned before, the benefit for Philips is that knowledge is gained in a number of fields related to vision based robot control.

4 RoboCup

4.1 Vision system

The vision system for the football robots consists of 2 cameras, a black and white camera and a colour camera, and 2 identical image acquisition boards equipped with a Philips TriMedia chip. The purposes of the 2 systems are described below:

- The colour camera is responsible for finding the objects, and determining their positions, based on colour. This can be done because all the important objects, the field, the lines, the ball, the goals, the teammates and the opponents, have distinct colours.

- The black and white camera receives the object positions from the colour module and starts tracking that object (e.g. the ball). The tracking is done by the B/W camera because of its higher capture and refresh rates.

Until now, emphasis has been put on the first part, the colour based object detection.

4.2 Colour spaces

In order to efficiently distinguish the objects based on their colour, it is necessary to choose a suitable colour representation. Colours are represented by a number of (mostly 3) scalar values. Possible representations are:

- RGB: 3 values for the intensity of the red, the green and the blue component of the colour. This is a natural representation because, on the one hand, colour cameras decompose every pixel in the image into these 3 component colours and, on the other hand, a CRT has 3 electron guns, one for each component colour. The values range from 0 to 255 (i.e. 1 byte) per colour. This representation, however, is not very suitable for colour based object detection, since changing lighting conditions generally causes changes in all 3 values, which is undesirable.
- HSI: Hue, Saturation and Intensity (sometimes called HSL, with Luminance instead of Intensity). These values are used in the control of an ordinary colour television, since this normally has controls for colour saturation and for the intensity. Low colour saturation means that a black and white picture is shown, high saturation usually gives an overly coloured picture. Low intensity means a completely black picture and high intensity gives a completely white picture. The hue gradient actually specifies the colour itself.
  For object detection, HSI is ideal since different lighting conditions normally have large effect on intensity, but little effect on hue and saturation, therefore one can restrict attention to these two components instead of 3 in the RGB case. The conversion from RGB to HSI is quite complicated, as this formula for the Hue shows
  ```
  H(R,G,B) =
      { if R=G=B, then 0
        else
        if R is maxcolor, then (G-B)/(maxcolor-mincolor)
        if G is maxcolor, then 2.0 + (B-R)/(maxcolor-mincolor)
        if B is maxcolor, then 4.0 + (R-G)/(maxcolor-mincolor)
      }
  ```
- YUV: For colour information transmission, mostly YUV is chosen. The Y component is comparable to the Luminance. U and V are used to reconstruct the colour information. YUV has similar advantages as HSI, albeit that the changes in U and V are slightly larger with changing lighting conditions than the changes in H and S. An advantage of YUV as opposed to HSI is that the conversion between RGB (the camera output) and YUV is easy to compute because of the linear relation between them, whereas the conversion from RGB to HSI is more complicated.

  $$\begin{aligned} Y &= 0.23 \cdot R + 0.58 \cdot G + 0.12 \cdot B \\ U &= -0.17 \cdot R + 0.33 \cdot G + 0.5 \cdot B \\ V &= 0.5 \cdot R - 0.42 \cdot G - 0.08 \cdot B \end{aligned}$$

For RoboCup, the YUV representation is chosen because of its good possibilities for object detection based on colours and its easy conversion from RGB. To increase processing speed, a hardware module is used for this conversion.

4.3 Algorithm for object detection based on colour

Every pixel in the image is converted into YUV representation, and then represented by a white dot in the 2-dimensional UV-plane (i.e. Y is neglected). U and V values range from -128 to 127. These dots in the UV-plane can also be represented by polar co-ordinates (angle and radius, relative to the origin (0,0)):

Angle = arctan(U/V)

Radius = $\sqrt{}(U^2+V^2)$

When the UV-plane is shown graphically, the resulting picture resembles a square pizza pie filled with all kinds of colourful ingredients. For this reason, the following algorithm is also called "pizza pie segmentation".

A feature of YUV representation is that pixels of the same colour are all inside a triangular segment in the UV-plane between a minimal and a maximal angle, much like a piece that is cut out of a pizza pie. Since the points near the origin of the UV-plane are almost grey and therefore contain almost no colour information, these are excluded from such a pizza pie segment.

Now, for every object that has to be localised, a corresponding pizza pie segment can be defined in the UV-plane. These parts are not overlapping since the colours in the scene are clearly distinguishable (see Figure 1). Experiments have shown that re-calibration is necessary when the lighting changes, because the positions of the white dots representing a certain object change when there is more or less light, and of course also when the colour of the light itself changes. In the future, a method will be designed for automatically determining where the pizza pie segments have to be.

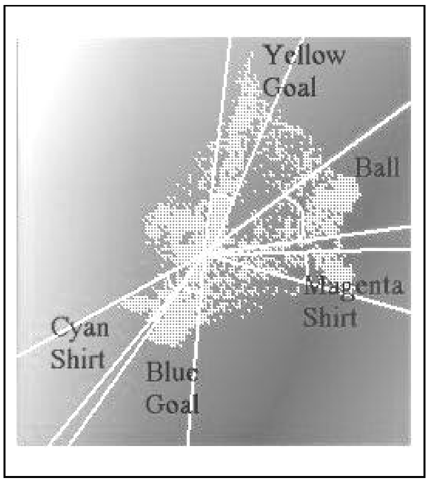

Figure 1 UV plane with pizza pie segments

All pixels of the original image that are projected inside one of the predefined pizza pie segments in the UV-plane are supposed to belong to the same object. In this way all pixels of one object are located and features like the area size (i.e. number of pixels in the object) and the centre of gravity of the object can be easily computed.

4.4 Reconstructing a partly occluded ball

When we concentrate on the ball (which is red), we can say that, when the ball is not occluded by any other object, it is visible as a red circle in the image and the centre of gravity of all red pixels coincides with the centre of the physical ball. This is a rather crude and unreliable way of determining the position of the ball, mainly because the results are distorted when the ball is partly occluded or badly lit. When only half of the ball is visible, the computed centre of gravity of all red pixels will have a considerable deviation from the real centre of the ball. Since, for a football-playing robot, it is of crucial importance that the position of the ball is known at every moment, a solution has to be found.

It is investigated whether the Hough-transform is suitable here. The Hough transform is a technique, originally designed for finding partly occluded straight lines in an image (first described in Hough 1962; see Davies 1997 for an elaborate treatment) and later extended to the case of partly occluded circles and other shapes. For this algorithm, first all edge points must be found by means of some other method, and then the algorithm can collect all points belonging to the same circle. Disadvantages are that it is a quite computation intensive algorithm and that, when a circle is searched, the radius of that circle must be known on beforehand. As a result, it is not possible to do this for every single image frame, but only at a certain regular rate. Furthermore, the radius of the circle to be searched must be determined one way or the other. A possibility is to take the perspective view of the camera into account. Since the camera is fixed to the robot at a certain height above the playing field, and assuming that the ball is always on the ground and never lifted, there is a relation between the height of the ball in the image, and the radius that the ball will have. The higher the ball is, the greater the distance from the robot, and hence, the smaller the radius. One can measure the correspondence (or compute it with a suitable mathematical model) and use this to set the radius of the circle that the Hough transform is looking for. The Hough transform can then reconstruct the complete ball-circle when that ball is partly occluded. When the ball is completely visible, the Hough transform is not necessary.

5 Jugglebot

5.1 Feedback loops

A typical control system for a servo motor is depicted in Figure 2. There are three feedback loops here, which will be discussed below:

1. The amplifier loop
2. The controller loop
3. The setpoint generator loop

5.1.1 Amplifier loop.

In this picture, the internal loop denotes the amplifier applying a certain current to the motor in order to get it moving. This current is measured about 10000 times a second in order to stabilise it to the value supplied by the controller. This loop is mostly implemented in hardware.

5.1.2 Controller loop.

The middle loop is meant to determine the current that is needed to get the motor to a certain position. The position and/or velocity of the motor (or an axis connected to the motor) is measured by the encoder and the difference between this actual location and the desired location (the setpoint) is calculated and fed into the controller. The controller decides what current should be applied to the motor in order to get it to the right position. The controller will try to balance the applied currents in such a way that the motor is in the setpoint position as fast as possible, taking into account the effects of overshoot. The actual position is measured by the encoder in certain time-intervals, with a length in the order of magnitude of 1 millisecond. In between two encoder-measurements, the current is constant. This loop is implemented in software.

5.1.3 Setpoint generator loop.

The outer loop is based on the setpoint generator (see below) that calculates the setpoints. The frequency with which setpoints change depends on the situation. The more setpoints an application generates, the better control it has over the exact movement, but the more computation time is needed. Consider an example in which an axis is in position A and has to move to position B at a distance of 1 meter. When the driving motor is controlled by only one setpoint (position B), only the controller will determine the behaviour of the axis. Since the controller reacts on the signals of the encoder, the exact movement will depend on external influences like the load. When two or more motors are moving in parallel the path is unpredictable. By generating multiple setpoints, the application has more exact control over the followed path. A setpoint may be determined by the desired position and/or velocity. When a setpoint is supplied to the controller, it will try to send the axis to these values as explained above.

5.2 Setpoint generator (SPG)

A setpoint generator is a software module that generates a sequence of setpoints on the basis of a limited number of parameters. Different setpoint generators may generate different setpoints when supplied with the same parameters. Two possible types are:

- A point to point generator will sent an axis from one point to another in the shortest possible time. The axis starts and ends with velocity zero. This generator will generate setpoints such that the axis starts with a maximal acceleration until the maximum speed is reached, then continues with that speed until nearly at the endpoint and then decelerates maximally to come to a halt exactly at the endpoint. The values for maximal acceleration, maximal deceleration and maximal speed of course have to be supplied to this setpoint generator in advance.
- A spline generator generates a "smooth" path through a number of points. The partial path from one point to another is called a spline segment. "Smooth" means here that every spline segment follows some polynomial profile and that adjacent

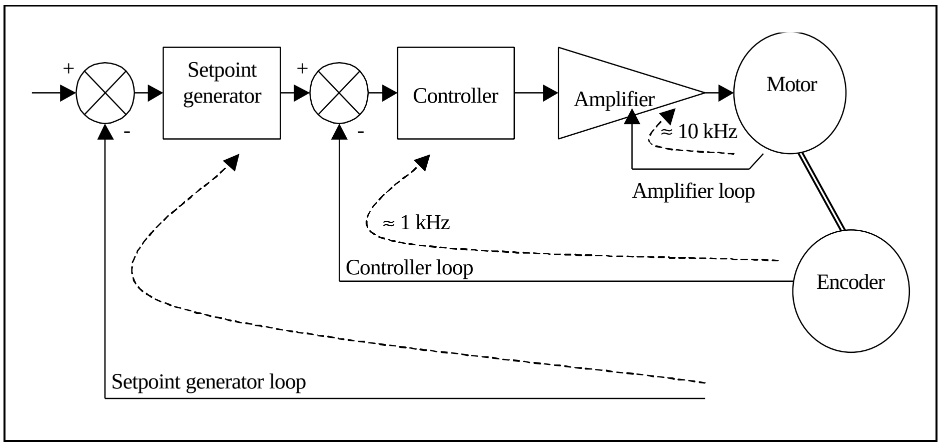

Figure 2 Feedback loops

segments satisfy certain continuity requirements (see Boots 1997). The degree of the polynomials and the exact continuity requirements may vary from one generator to the other. Thus, the spline generator receives a sequence of position- (and maybe speed-) values from some application, and generates setpoints in such a way that the axis moves smoothly from one point to the next.

5.3 The example robot

The Transposer robot that is used in the experiments has the following characteristics:

- The robot is of the 'jointed arm' type with 3 rotational joints.
- There are three motors: one for a rotation of the complete arm around the vertical axis, one for a movement of the effector in horizontal radial direction and one for a movement of the effector in vertical direction. The mechanics of the robot are designed in such a way that each of the last 2 motors in fact moves (the same) 2 joints.
- Each motor has a controller board that contains the three above mentioned feedback loops and generates the necessary currents for the motor. These controller boards are considered black boxes, i.e. no attempt is made to make changes in these feedback loops.
- The three controller boards are connected to each other and to a CPU board by a dedicated bus system. The CPU board is built around a processor of the M68000 family.
- The CPU board is connected to a PC by a serial link. It runs a special OS called MACOS.

The controller boards can be programmed in a number of ways, the two most relevant

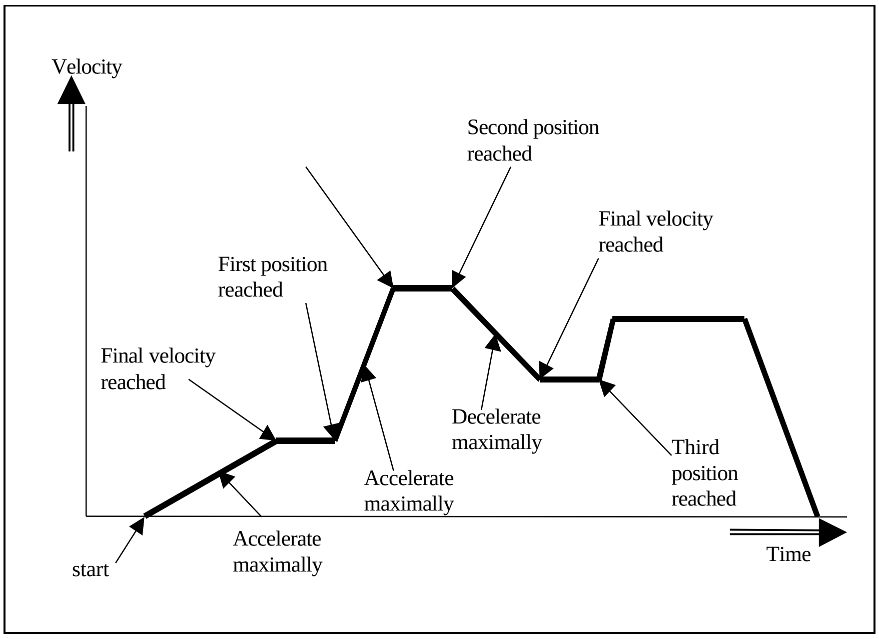

Figure 3 Position based speed profile

being:

- Position mode
  The distance, final velocity and maximal acceleration are specified. The built-in SPG generates a velocity profile as depicted in Figure 3.
  For every specified point, the SPG first accelerates maximally until the maximal velocity is reached, then continues with that velocity until the specified point is reached. At that moment the next parameters are read and the SPG generates a new similar profile for going to the next point.
- Speed mode
  Only speed and acceleration are specified and this results in a speed profile, which, unlike the previous one, does not react on reaching a certain position (since no position is specified at all), but is continuously moving with the final velocity until the application conveys the next set of parameters.

5.4 Software design

The software for this project is developed by several students concurrently, therefore the interfaces had to be defined unambiguously. Although most software at the IV department is written in the language C, this software is written in C++. A class model has been made which is still undergoing changes.

5.5 Object finding

The software is designed in such a way that the object location can be done with several different algorithms, by extending the class IPU (Image Processing Unit). Possible algorithms are:

- Feature tracking
  The object is located using some features like area size, main axes, etc. A standard library is used for finding these features.
- Template matching
  The object is found using template matching. An experimental algorithm is used for generating a suitable template automatically in the initialisation procedure of class IPU.
- Colour detection
  The pizza pie algorithm described in relation to the RoboCup project could also be used in this project. This approach, however, is not pursued.

5.6 Object tracking

One objective is to let the robot follow a randomly moving object. The camera will be mounted on the robot effector and the robot should move in such a way that the object is always almost in the centre of the image. In principle, when the position of the object in the image is found, when the mechanical dimensions of the robot arm are known exactly and when the actual position of each joint is known, it is possible to compute exactly in what direction and with what speed the motors should be moved in order to get the object in the middle of the image. The (not yet implemented) idea is to avoid computing this complex real-world to robot transformations and make the control as simple as possible, like a fly that is able to land on an apple without having to do complex computations. The simplified algorithm is:

- Find the object in the image using one of the techniques described in paragraph 5.5.
- Determine the deviation of the object position relative to the centre of the image.

- Steer the motors depending on this deviation. The direction of the deviation determines which motors have to move and the direction in which they have to move, and the distance from the centre determines the velocity with which the motors have to move. Instead of computing exactly the optimal values for each motor, these values are approximated with simple formulas.

This approach is viable because, assuming that the object is indeed kept in the middle of the image, the deviations from the centre will be small and hence the approximations will be close to the optimal values. In addition, the approximations are repeated in the same rate as the image capturing and in this way corrections will be made quickly when the robot is not moving in exactly the right direction.

6 Final remarks

The I3 projects are in principle 'never ending stories', which means that they are still running to date. Students are taking over each other's work. Currently the team is (a.o.) working on the use of a high-speed camera for motion control. Since the future of the project depends on the results gained on the current tasks and on the wishes and needs of Philips CFT, it is not possible to predict exactly in what way the project will evolve.

Working with a team like this has a lot of advantages, since the students can help each other and learn to co-operate with people from different disciplines and nationalities, but on the other hand it is difficult to find assignments in which the students can co-operate successfully.

We would like to thank the people at Philips CFT, IV department, for giving our students the opportunity for working in this technically advanced and inspiring environment. Finally, the students are thanked for their hard work and good results.

References

Bakker, R. M., Geraedts, H. G. M. & van Schenk Brill, D. (1999), A Model for Education in Innovative Engineering, *WESIC1999*, Newport, September 1999.

Boots, P.J.H.M. (1997), *A Spline Generator Interface for OMC SAC, Software Requirements Specification*, Philips CFT internal report.

Davies, E.R. (1997), *Machine Vision*, Academic Press, San Diego, USA

Hough, P.V.C. (1962), *Method and means for recognising complex patterns*, US PAtent 3069654

Iakovou, D. (2001), *RoboCup Vision Module*, Philips CFT internal report.

Kollenburg, P.A.M., Veenstra, H., van Schenk Brill, D., Ihle, H. & Kater, K. (April 2000), Integrated Product Development and Experiences of Communication, *TMCE2000*, Delft, April 2000.

Kollenburg, P.AM. van, Schenk Brill, D. van, Schouten, G., Mulders, P. Ochs, J.B. & Zirkel, M. (September 2000), Collaborative Engineering Experiences, *Engineering & Product Design Education Conference 2000*, Brighton, September 2000.