P4 – A Scalable Real-time Ping Pong Point Protocolar

Ulrich Schwanecke, Peter Brendebach, Nadia Haubner, Christian Rathemacher, Frank Walkowski, Friederike Wild, Kai Winter

Wiesbaden University of Applied Sciences
Fachbereich Design Informatik Medien
schwanecke@informatik.fh-wiesbaden.de

20. September 2007

Abstract

The analysis of sporting games is of major interest for many different people; it helps supporting a coach or a referee and supplies supporter with interesting facts about a certain athlete or team. While the manual analysis can be very cumbersome and time-consuming, automatic monitoring opens a new range of applications. In this paper we present a scalable real-time 3D tracking system for ball games. Our system traces the trajectory of the ball by an optical multi-camera tracking procedure. Albeit the tracking algorithm is based on a multi-camera architecture it can estimate the 3D position of the ball even if it is seen by only one camera. Combined with a prediction-correction step this makes our ball-tracking procedure very robust and reliable. The determination of the balls 2D position in each video frame is based on fast color segmentation and union-finding algorithms. The 3D coordinates of the observed ball are calculated by a triangulation process. We demonstrate the capability of our approach by implementing a system for real-time tracking and recording of table tennis games.

1 Introduction

The request for automatic monitoring of sporting games did not come about just because of debatable decisions of referees like e.g. in case of England's controversial third goal in the 1966 Football World Cup Final. Systems, automatically tracing the position of the ball (and the players) can support coaching activities before and after competitions. A lot of systems analyzing sporting games, using various kind of technologies have been developed recently (see e.g. [1, 2, 3, 12, 14]). Some of them are based on position-tracking systems relying on microwave transponders attached to the ball and the players. An example for such a system is the tracking-system developed by Cairos Technologies[1] and Fraunhofer IIS. Most of the monitoring systems, like e.g. the systems of ProZone[2] or Sport Universal[3], are based on computer vision technology. Thereby the playground, the players and the ball are observed using a bunch of cameras. All systems collect a hugh amount of data and enable exhaustive statistics that can be used by coaches or broadcast companies.

In some sports, like e.g. tennis most advanced technological aid has become a standard equipment for referee. One of the first commercial real-time systems was Cyclops, introduced to the Wimbledon Championships in 1980. This system, especially build for tennis games, monitor the service line to determine wether a serve was in or out. A recent successor of Cyclops is the real-time tracking system Hawk-Eye[4], actually used in cricket, tennis and snooker games. Hawk-Eye not only spots a special place, like e.g. the service line, but track and record the complete three-dimensional path of

[1] Cairos Technologies: www.cairos.com
[2] ProZone: www.pzfootball.co.uk
[3] Sport Universal: www.sport-universal.com
[4] Hawk-Eye: www.hawkeyeinnovations.co.uk

the ball. Another system used during tennis matches is MacCam[5]. Based on slow-motion cameras it allows to replay close or controversial line calls. The two recent systems are expensive high-end systems which are primarily used by television networks to visualize the trajectory of balls in flight.

In this paper we describe the software system P4 (Ping Pong Point Protocolar); a system to find a ping-pong ball within a video stream, extract and validate its 3D-position and display the result in a virtual 3D scene (figure 1 depicts this scenario). We show that P4 is capable to trace the ball and monitor the score in real-time using of the shelf computer and web-cams.

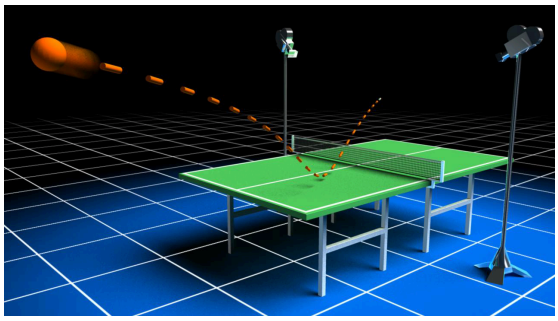

Figure 1: Our scenario: A ping-pong game is observed by several video cameras, tracking the 3D position of the ball in real-time.

The next section discuss the calibration of our installation. Section 3 give an overview of the system design of P4 and discuss each step of the ball-tracking procedure in detail. Section 4 briefly describes some important implementation details of our software prototype. An Evaluation of this application is given in section 5. In the last section we conclude and give an outlook on future work.

2 Calibration

To accurately evaluate the 3D position of a ball with respect to the table tennis plate out of the camera images the intrinsic and extrinsic camera parameters have to be determined in a calibration step. The calibration of the intrinsic camera parameter - especially precise lens undistortion - can be done using the GML MatLab Camera Calibration Toolbox [6].

The extrinsic camera parameters i.e. position and orientation of each camera with respect to the table tennis plate and each other could be determined by manual measurements. But this would be very cumbersome, error prune and time-consuming, especially due to the fact that this has to be done every time the setup is modified. Therefore we implemented an automatic calibration process using the multi-marker functionality of the video tracking library ARToolKit [11]. The calibration is based on a physical marker in one corner of each side of the ping pong table. These markers define a coordinate systems M (see figure 2).

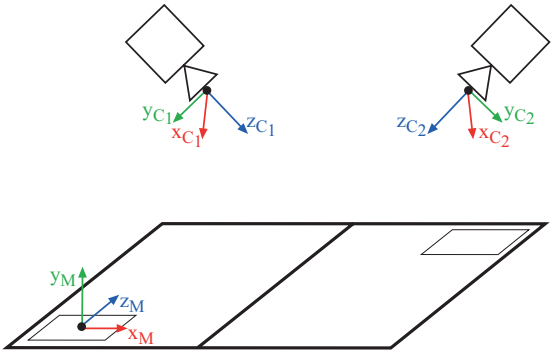

Figure 2: After calibration, based on two optical markers attached to the table tennis plate all coordinates are transformed from camera coordinates (C_1, C_2) into world coordinates (M).

If all cameras see at least on marker, ARToolKit evaluate the position and orientation of each camera relative to M. In all calculations described in the following sections we use M as world coordinate system.

3 System design

Our tracking system mainly consists of three parts synchronized by a *Main–component*. These different parts are

- *Segmentation*, i.e. finding the ball in the video-imagery. The centroid of the image

[5]MacCam: www.fastcamreplay.com

[6]GML MatLab Camera Calibration Toolbox: http://research.graphicon.ru/calibration/gml-matlab-camera-calibration-toolbox.html

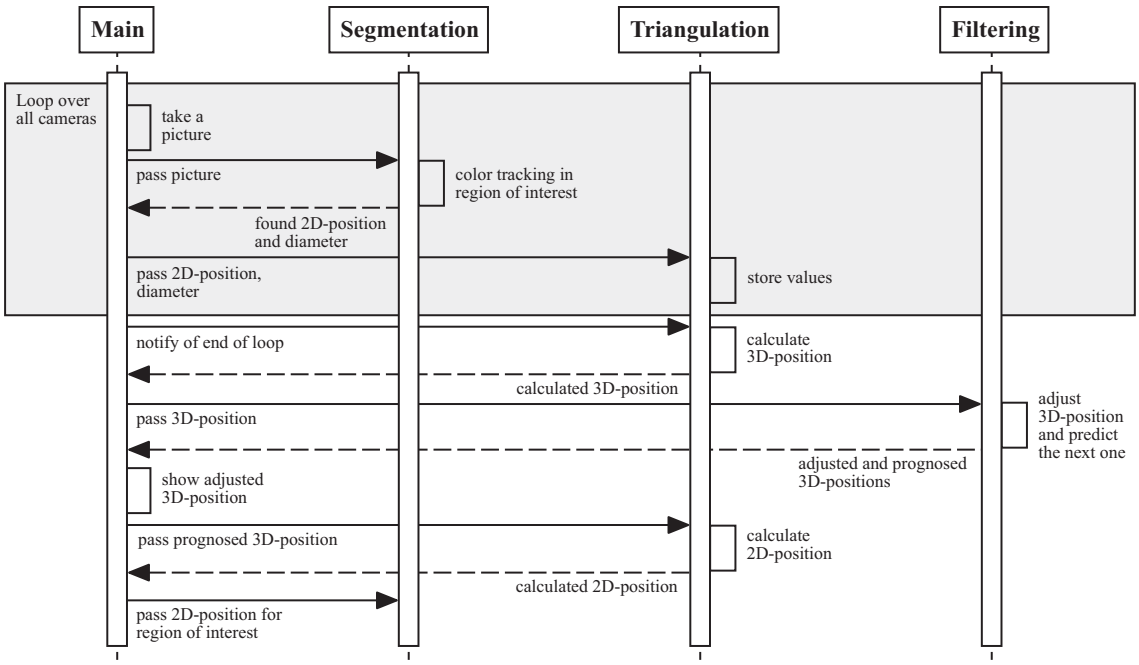

Figure 3: The design of our System. The figure depicts the interaction between *segmentation*, *triangulation* and *filtering*.

of the ball and the diameter of this image are found by a fast color tracking algorithm.

- *Triangulation*, i.e. calculating 3D coordinates out of the 2D coordinates obtained from *Segmentation*.

- *Filtering*, i.e. using a physical model of the movement of the ball to correct the 3D coordinates obtained from *Triangulation*.

Figure 3 illustrate the different parts of our system and their interaction while analyzing video images with identical time-stamp. After all images belonging to one time-stamp are analyzed by the *Segmentation–component* the *Triangulation–component* calculates the 3D-position of the ball. The result is passed to the *Filtering–component* to adjust the 3D-coordinates based on a physical model of the ball. The predicted 3D-coordinates are also used to define a region of interest (ROI) for the next image of each camera. For the next time-stamp the *Segmentation–component* start looking for the ball in the ROI. The usage of the ROI speed up the ball-finding process dramatically.

The calculated 3D-position is used to visualize the ball in a 3D scene and for additional game analysis like e.g. score counting. The trajectory of the ball is written to a file. Therefore every rally can be shown as a replay and evaluated afterwards.

3.1 Image segmentation

The first step towards reconstructing the 3D position of a ball out of video images is to find its mapping. In P4 this is done by a segmentation algorithm based on color classes. The algorithm calculate the two-dimensional centroid and bounding box of the image of the ball.

For color segmentation many algorithms such as e.g. linear color thresholding, nearest neighbor classification or color space thresholding are known (see e.g. [5, 4, 9]). Due to our real-time requirements we use a fast segmentation algorithm presented in [6] that efficiently classifies each pixel of an image in up to 32 color classes. The algorithm is initialized by creating color-classes defined by the color of the ball we are looking for. Segmentation is composed of three parts:

- Decide for each pixel if it falls in one of the defined color-classes.
- Merge all connected pixels in regions with a tree based union-find algorithm.
- Sort the regions with respect to size.

In P4 we are using orange table tennis balls and assume that the largest region of "orange" pixel correspond to the ball we are looking for.

Color Space and Color classes

Color strongly depend on lighting conditions. Thus the choice of an appropriate color space is of particular importance for color based segmentation. For a reliable color segmentation it is important to divide the color information from the luminance information. In P4 we are using the YUV color space which encode the luminance and color information separately.

A color-class is defined by six thresholds defining a box in YUV space. This method is called linear thresholding because a color-class hat linear boundaries (planes). Figure 4 show an image of a table tennis ball and the corresponding color values in YUV space. Regard that the variation in the Y component is bigger than in the U and V components. A coarse color-class *"Orange Ball"* is plotted with light dashed lines.

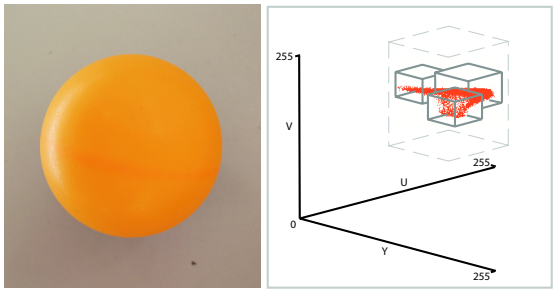

Figure 4: Table tennis ball and corresponding colors in YUV space together with various color-classes.

Because of the irregular shape of the color distribution linear thresholding can result in wrong segmentations using only one color-class. Therefore we divide the color-class *"Orange Ball"* in several smaller classes (see figure 4) and check each of them during segmentation.

Segmentation

Segmentation test for each pixel if its color fall into the rectangular box defined by a color-class. This test can be accomplished very efficiently by only two binary-`AND` operations. Therefore the color-classes are encoded in three Look-Up-Tables (LUTs) `Yc`, `Uc` and `Vc`; one for each color component. Using 8-bit color quantization each LUT contains 255 different values. All entries between the index positions given by the thresholds defining a color-class are set to one, all others to zero. Wether a pixel with color `c=(Y, U, V)` falls into the defined color-class or not can be checked by using `Y,U,V` as indices into the LUTs:

```
pixel_in_cc = Yc[Y] AND Uc[U] AND Vc[V]
```

Additionally it is possible to encode several color-classes in the same three LUTs without enhancing the complexity of the algorithm. Instead of building new LUTs for each color-class they can be combined using the bit position of an array element. Using 32-bit integer values as array elements allow to check for membership of 32 distinct color classes at once. In our application we are using three different color-classes to segment the ping pong ball. Every pixel falling in one of these classes is labeled as *"Ball-Pixel"*.

Find connected regions and calculate region information

After segmentation, each pixel is labeled to be a *"Ball-Pixel"* or not. Next, connected *"Ball-Pixels"* have to be combined to regions. This is done in two steps for performance reasons.

1. Attach pixels horizontally. For each row of the image a run-length-encoding (RLE) is accomplished during segmentation.
2. Merge horizontal lines vertically. On the classified RLE image a tree-based union-find algorithm with path compression is employed (for details see [6]).

It can be shown that this merging method employing tree-based union finding with path compression offers performance that provides a hard algorithmic bound (see e.g. [16]).

Merging connected pixel result in several regions. By means of the tree-based representation of each region their size and barycenter can be evaluated efficiently. We assume the greatest region to be the wanted ball. Size and barycenter of this region are the input values for the following triangulation process.

3.2 Triangulation

The intrinsic and extrinsic parameters of all cameras used are known from the calibration step. Thus, the 3D-position of the ping pong ball can be aquired from the camera images by triangulation (see e.g. [7, 8]). To get reliable results the cameras should be positioned in a way that the ball can be seen by as many of them as possible in every state of the game.

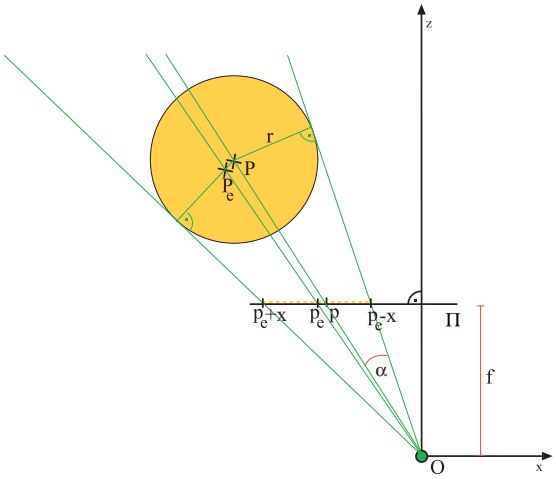

Figure 5: Evaluating the 3D position of a ball using one camera. Note that the center P of the ball is not projected to the barycenter p_e of the image of the ball.

Without additional information it is impossible to reconstruct the 3D position of the ball out of one camera image. There are two ways to solve this problem.

- use additional information; e.g. the diameter of the ball.
- use information from additional cameras.

Both methods are equivalent in theory but due to inaccuracies resulting from the limited resolution of a camera and the errors made determining the size of the diameter of the image of the ball, the first one is more inaccurate than the second one.

Nevertheless the first method is used also in P4 to provide a fallback solution if the ball is occluded for all but one camera and to support one camera as a minimum requirement. Knowing the projection p of the center of the sphere P and the boundary point $p_e - x$ some simple geometry shows that the distance d between P and camera origin O is given by $d = \frac{r}{\sin\alpha}$ with

$$\tan\alpha = \frac{(p - p_e + x)f}{p(p_e - x) + f^2}. \tag{1}$$

If the center P of the ball does not lie on the z-axis, the projection of the ball is an ellipse and the projection of the center of the ball unfortunately is not the center p_e of this ellipse (see figure 5). Due to performance reasons we do not correct this position-distortion. Indeed, using webcams with average image quality this correction is not necessary because the distortion is very small compared to other erros like e.g. calibration errors and noise. For a detailed discussion of the position-distortion of an ellipse centre under perspective projection see [19]. Instead of (1) we assume $p = p_e$ and use

$$\tan\alpha = \frac{xf}{p_e^2 - p_e x + f^2}.$$

Knowing the distance d between O and P the 3D coordinates of the center of the ping pong ball can be approximated. If $\vec{p}_e$ is the vector starting at O and pointing to p_e we obatin $P = d \cdot \vec{p}_e$.

As aforementioned, calculating the depth coordinate of a sphere out of the image of only one camera using the known sphere diameter is error-prone. Better results can be achieved by using additional cameras. Figure 6 illustrates the situation using two cameras. The optical centers of the cameras are O and O'. The center P is projected to p and p' onto the image sensors Π and Π'. Just like in the reconstruction process using one camera, for each camera a vector is set up starting at the optical center and pointing at P. These vectors define rays intersecting in the center of the table tennis ball. Regard that in practice the error made by using p_e and p'_e instead of p and p' is very much smaller than depicted in figure 6. This mainly rely on the fact that the diameter of the ball is

very much smaller than the distance between camera and ball.

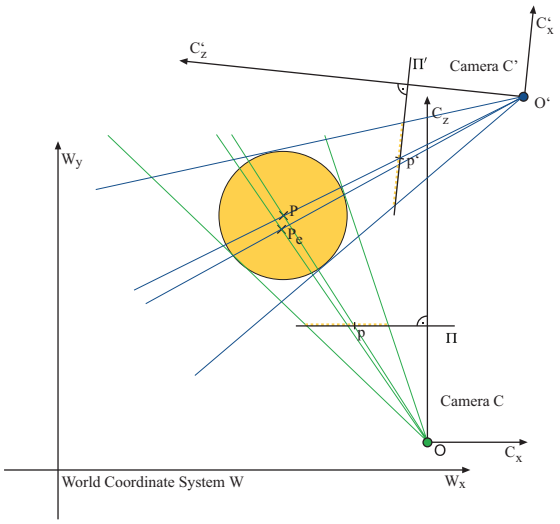

Figure 6: Buildup using two cameras

Generally the two rays will not intersect but define skew lines. Therefore we determine the least square solution, i.e. the closest point to both lines. In fact we are determine the least square solution for the intersection of all lines defined by all camers used.

3.3 Filtering

Finding the ping pong ball in a 2D camera image is affected by measurement and approximation errors. As a result, there is a difference between the triangulated 3D position and the actual position of the ball. Moreover, it is possible that several ball detections may fail, e.g. because of occlusion. To achieve a trajectory as close as possible to the actual trajectory, it is necessary to correct inaccurate positions in an appropriate way. Absent measurements should be replaced with likely occuring positions. Knowing the probable next 3D position of the ball also can help to accelerate the segmentation process. We will relate on this point in the next section.

We are using a discrete Kalman filter [10] as a prediction-correction filter which is the optimal estimator for a linear system for which the noise ist zero mean, white and Gaussian. Even if the noise is not Gaussian it provide good results in the majority of cases. A comprehensive and comprehensible introduction to Kalman filtering can be found in [17]. For an introduction to optimal estimation and stochastic models see e.g. [13, 15, 18].

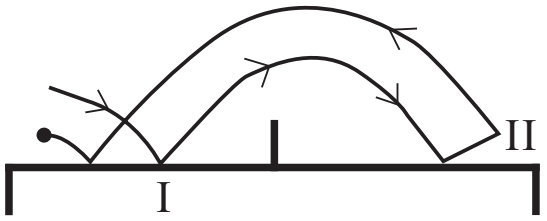

Figure 7: A poing pong game approximately consists of a sequence of bomb-trajectories.

A point pong game approximately consists of a sequence of bomb-trajectories (see figure 7). The position of the ball changes in the direction of the x-, y- and z-axis. The motions in the direction of the x- and z-axis are uniform. The position changes depending on the velocitiy and the last time step respectively. In the direction of the y-axis gravity affects the position of the ball additionally. Therefore we obtain

$$\begin{aligned} x_k &= x_{k-1} + v_{x_{k-1}}\Delta t \\ y_k &= y_{k-1} + v_{y_{k-1}}\Delta t - \tfrac{1}{2}g\Delta t^2 \\ z_k &= z_{k-1} + v_{z_{k-1}}\Delta t \end{aligned} \tag{2}$$

In the prediction step of our Klaman filter the *a priori* estimate of the system state $\hat{\mathbf{x}}_k^-$ and the error covariance matrix $\mathbf{P}_k^-$ are calculated out of the *a posteriori* values $\hat{\mathbf{x}}_{k-1}^+$ and $\mathbf{P}_{k-1}^+$ as follows.

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \mathbf{A}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{B}\mathbf{u}_{k-1} \\ \mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}^+\mathbf{A}^T + \mathbf{Q} \end{aligned}$$

The state $\mathbf{x}_k$ of our system and the system matrix $\mathbf{A}$ describing the uniform motion to the next state without any outside influences, are defined by equation (2) as

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{x}_k = \begin{pmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{pmatrix}_k .$$

As an outside influence the gravity affects the ball in the negative direction of the y-axis. Thus

the control matix $\mathbf{B}$ and controll input $\mathbf{u}_k$ are obtained from equation (2) as

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{u}_k = \begin{pmatrix} 0 \\ -g \\ 0 \end{pmatrix}.$$

For the matrix $\mathbf{Q}$, modelling a fixed uncertainty of our system we choose

$$\mathbf{Q} = \text{diag}\{10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}\}.$$

In the correction step of our Kalman filter a Kalman gain $\mathbf{K}_k$ is calculated that adequately weights the measurement $\mathbf{z}_k$ over the prediction. As a result of the combination of the *a priori state* and the weighted measurement, the *a posteriori* state $\hat{\mathbf{x}}_k^+$ and the error covariance $\mathbf{P}_k^+$ can be computed:

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}^T \left(\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}\right)^{-1} \\ \hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \\ \mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \end{aligned}$$

The measurement $\mathbf{z}_k$ consists of the current position of the ball. The matrix $\mathbf{H}$ which relates the system state to the current measurement only relates the position of the state and neglects the velocitiy which is not measured. Therefore we get

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \text{ and } \mathbf{z}_k = \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix}_k.$$

We estimate the measurement error to be $0.04cm$ and thus set

$$\mathbf{R} = \text{diag}\{1.6 \cdot 10^{-3}, 1.6 \cdot 10^{-3}, 1.6 \cdot 10^{-3}\}.$$

Start position, velocity and the error covariance matrix are not known. We start the filtering process with $\hat{\mathbf{x}}_0^- = (0, 0, 0, 1, 1, 1)$ and $\mathbf{P}_0^-$ being the identity.

The system matrix $\mathbf{A}$ describes a bomb-trajectory which changes in positve direction along the x-axis (regard to figure 2 were the world coordinate system M is depicted). However, during a match the flight direction of the ball changes permanently. We use some simple heuristics to adjust the system matrix to miscellaneous situations. When the ball collides with the board (see `I` in figure 7) we multiply the velocity towards the y-axis with -1. This collision can be checked easily because we know the exact position of the table tennis board from the calibration step. When the ball is hit by a player (see `II` in figure 7) we multiply the velocity towards the x-axis with -1 and set the velocity in z direction to zero, that is we do not make any assumptions in what direction the player hit the ball. The identification wether the ball is hit by a player is based on observing the x-direction of the movement of the ball.

The system matrix $\mathbf{A}$ has to be changed as the ball changes direction. The flight of the ball in negative direction along the x-axis can be described e.g. by

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & -\Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

One crucial point using a Kalman filter is the available amount of measurements per second. If there are not enough measurements per second, the Kalman filter starts jittering and the corrected ball positions do not match the acutal ball trajectory. In tests our Kalman filter works fine if we get at least twenty measurement values per second.

3.4 Performance optimization

The most time-consuming step is the segmentation process. The highest performance cost originates in checking each pixel in an image. Assuming that an object to track is greater than one pixel in the image, it is not essential to check every pixel. To save costs on this operation, P4 allows to check not every pixel but every second, third, etc. pixel. Omitted pixels are added to a color-class if the precursor- and the successor-pixel are in the same color-class.

Another approach to reduce calculation time is the use of a *region of interest* (ROI) (see figure 8) that is, searching the ball only in a small area instead of the whole image. The ROI can be determined using the predicted 3D coordinates of the ping pong ball obtained from the

filtering step. Therefore we project the 3D coordinates into the image area of each camera to obtain the corresponding 2D position of the image of the center of the ping pong ball. The center of the ROI is given by this 2D position. Due to the adjustment of our filter, the prognosed position and therefore the ROI is very close to the actual position. This makes it possible to choose the size of the search region nearly as small as the expected maximal size of the projected ball. Additionally the size of the ROI could be adapted using the information about the diameter and distance of the ball. In practice we received good results with a fixed size of about 40 pixels. If the segmentation fails in finding the ball in the ROI it falls back to researching in the entire image.

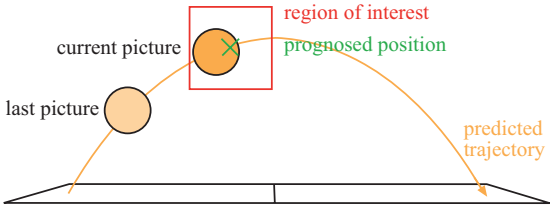

Figure 8: A region of interest is prognosed by the filtering step.

Knowing the instrinsic as well as the extrinsic parameters of all cameras, point correspondence geometry can be used to speed up segmentation. Suppose we had found the image p of the center P of the ball in the view of a camera with optical center O and image plane Π. The ray defined by O and p is imaged as line l' in the view of another camera with optical center O' and image plane Π' (see figure 9). l' is called the epipolar line for p.

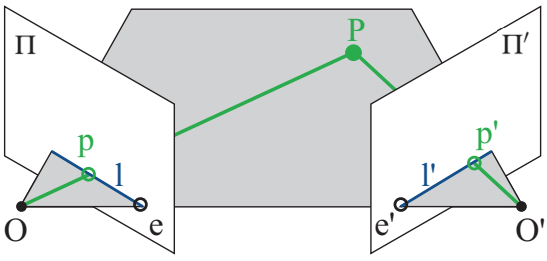

Figure 9: If we know the projection p of P onto Π we only have to search for the projection of P onto Π' along the epipolar line l'.

l' is also given as the line l defined by the epipolar e and the point p imaged in the view of the camera with optical center O' and image plane Π'. All epipolars are known from the calibration step. Therefore we can use them to speed up the search in the segmentation step. If we had found the image of the center P of the ping pong ball in the view of a specific camera we only have to search along the epipolar lines in the views of the other cameras. In practice we obtain errors from measurement and calculation and therefore the images of P will not fall exactly on the epipolar lines. Thus we are searching in a ROI around the epipolar lines.

Additionally we use the epipolar lines to confirm measurements. If the distance between an epipolar line and a measured point that is supposed to be the view of the center of the ping pong ball exceed a threshold we discard this measurement.

4 Implementation

Because P4 was intended to be a software prototyp we desided to implement the system using the scripting language Python. The usage of a scripting language facilitates fast development. The most time-consuming part is the segmentation. It is implemented in C and imported into our application using the Python module *cTypes*.

The ARToolKit library was embedded in P4 using the Python wrapper *PyARTK*. To access video-capture devices we are using the Python extension *VideoCapture* for Win32 systems. This part is the only one of our system that is platform dependent.

The calculated 3D coordinates of the ball are used to realize a point protocolling system and a replay function. To protocol the score, every collision of the ball with the board is interpreted and evaluated according to the rules of a ping pong game. To enable replay functionality all 3D coordinates are streamed into a file.

The replay functionality visualizes the ball together with a three-dimensional virtual table tennis plate. P4 allows to control the speed of the replay and to watch the scene from an arbitrary point of view. The rendering is based on OpenGL and the corresponding Python binding *PyOpenGL*.

Figure 10 shows a screenshot of the user interface of P4 utilizing three webcams. The upper half displays the video-stream of each connected camera. The physical marker used to calibrate the cameras is highlighted by a yellow frame. This indicates that the corresponding camera had been calibrated successfully. The red frame depicts the ROI used in the segmentation step. The yellow lines displayed in the second and third video image are the epipolar line for the image of the center of the ping pong ball found in the view of the first camera.

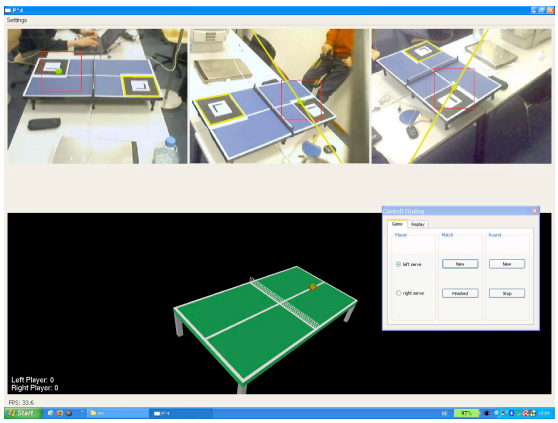

Figure 10: Screenshot of P4 using three cameras.

The lower half of figure 10 shows the virtual table tennis plate together with the ping pong ball. The 3D coordinates of the depicted ball are obtained by evaluating the corresponding video images shown in the upper half of figure 10.

5 Evaluation

We evaluated our program using an increasing number of cameras with varying resolutions. Table 1 summarize the measurement-rates indicated in measurements per second (mps) obtained by using a Centrion dual core 1,8GHz with 1GB RAM and Logitech Quickcam Messenger webcams.

Regard the speed-up obtained by the usage of a ROI. The ROI we used featured a fixed size of 40×40 pixels. The dramatic drop of the measurement-rate by switching from one to two cameras seems to rely on the video-capture library we used. This library pretend to be not intended to access multiple cameras optimally.

Nr. of cameras	Resolution			
	320×240		640×480	
	ROI		ROI	
	no	yes	no	yes
1	60	220	24	80
2	37	62	10	18
3	29	49	-	-

Table 1: Measurements per second on a Centrino dual core 1,8GHz with 1GB RAM.

Another problem is caused by a design limitation of the USB technology. USB is not the optimal bus for connection several cameras to a computer. When a USB camera start streaming video, it reserves as much bandwidth as possible. Thus the next camera trying to start streaming maybe could not get enough resources because al bandwidth has been already reserved. We obtained a system message declaring that the bandwidth of the USB controller has exceeded if we tried to use three cameras with a resolution of 640×480. Substituting the USB cameras by firewire cameras could avoid this problem.

6 Conclusion and Future Work

We presented P4, a prototype for realtime monitoring of ping pong games using standard webcams. Our system is based on the combination of an efficient color segmentation algorithm and a filtering step correcting the measurements and predicting the trajectory of the ball to further speed-up the calculational costly segmentation process. P4 is designed to work with an arbitrary number of cameras. The calculation of the 3D coordinates of the ball is based on triangulation. Our system can calculate an approximation of the 3D coordinates using information from only one camera. Together with the filtering process this make our system very robust, reliable and scalable concerning the number of cameras used.

A ping pong ball can achieve velocities up to 110 miles per hour or about 2 meters per frame

using cameras providing about 30 frames per second. This fact and the limitation of the USB technology demand the usage of better cameras for instance based firewire technology.

In our prototype we used *ARToolkit* to calibrate the extrinsic parameters of the cameras and obtained good results. Since *ARToolkit* was not designed for calibration purposes the calibration could be improved be implementing a new calibration step using e.g. information about the know size of the table tennis plate.

References

[1] T. Bebie and H. Bieri. Soccerman - reconstructing soccer games from video sequences. *1998 International Conference on Image Processing, 1998. ICIP 98. Proceedings.*, 1:898–902, October 1998.

[2] Michael Beetz, Bernhard Kirchlechner, and Martin Lames. Computerized real-time analysis of football games. *IEEE Pervasive Computing*, 4(3):33–39, 2005.

[3] Johan Borg. Detecting and tracking players in football using stereo vision. Master's thesis, Linkoeping University, February 2007.

[4] Carla E. Brodley and Paul E. Utgoff. Multivariate decision trees. *Mach. Learn.*, 19(1):45–77, 1995.

[5] T. Brown and J. Koplowitz. The weighted nearest neighbor rule for class dependent sample sizes. *IEEE Transactions on Information Theory*, pages 617–619, September 1997.

[6] James Bruce, Tucker Balch, and Manuela Veloso. Fast and cheap color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061–2066, October 2000.

[7] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, international edition, 2003.

[8] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[9] Ramesh Jain, Brian G. Schunck, and Rangacher Kasturi. *Machine Vision*. McGraw Hill Higher Education, 1995.

[10] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[11] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, 1999.

[12] M. Lames and G. Hansen. Designing observational systems to support top-level teams in game sports. *International Journal of Performance Analysis*, 1(1):85–91, 2001.

[13] Frank L. Lewis. *Optimal Estimation with an Introduction to Stochastic Control Theory*. John Wiley & Sons, Inc., 1986.

[14] Toshihiko Misu, Seiichi Gohshi, Yoshinori Izumi, Yoshihiro Fujita, and Masahide Naemura. Robust tracking of athletes using multiple features of multiple views. In *Journal of WSCG*, volume 12, February 2004.

[15] Dan Simon. *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*. John Wiley & Sons, Inc., 2006.

[16] Robert Endre Tarjan. *Data Structures and Network Algorithms*. Society of Industrial and Applied Mathematics, 1983.

[17] Greg Welch and Gary Bishop. An introduction to the kalman filter. In *SIGGRAPH 2001 Course*, 2001.

[18] X. Li Y. Bar-Shalom. *Estimation and Tracking: Principles, Techniques, and Software*. YBS Publishing (1993 by Artec House, Inc.), 1998.

[19] Guangjun Zhang and Zhenzhong Wei. A position-distortion model of ellipse centre for perspective projection. *Measurement Sci. Technol.*, 14:1420–1426, 2003.