Parabolic Flight Reconstruction from Multiple Images from a Single Camera in General Position

Raúl Rojas, Mark Simon, and Oliver Tenchio

Institut für Informatik
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany

Abstract. This paper shows that it is possible to retrieve all parameters of the parabolic flight trajectory of an object from a time stamped sequence of images captured by a single camera looking at the scene. Surprisingly, it is not necessary to use two cameras (stereo vision) in order to determine the coordinates of the moving object with respect to the floor. The technique described in this paper can thus be used to determine the three-dimensional trajectory of a ball kicked by a robot. The whole calculation can be done, at the limit, with just three measurements of the ball position captured in three consecutive frames. Therefore, this technique can be used to forecast the future motion of the ball a few milliseconds after the kick has taken place. The computation is fast and allows a robot goalie to move to the correct blocking position. Interestingly, this technique can also be used to self-calibrate stereo cameras.

1 Introduction

The years 2004/2005 constitute a turning point in the small-size RoboCup league. Several of the competing teams used chip kickers to lift the ball above 15 cm high robots placed at a distance of 30 to 50 cm from the kicking robot. Unfortunately for the defending team, it is very difficult to position the goalie when such a kick occurs. The projection of the parabolic flight of the ball on the video images looks like a curve. It is not immediately clear, whether the ball is flying high or is just rolling along a curved trajectory. Fig. 1 shows the apparent trajectory of a ball kicked high from a corner to the opposite corner of the field. Fig.1 is the combined result of observing the field with two cameras. In this paper, we discuss only the mathematics for a single camera overlooking the field. Generalizing to several cameras whose fields of view do not overlap is straightforward.

It would be very useful for a defending team to be able to determine the parameters of the parabolic ball trajectory (that is, the direction of the kick and its height) immediately after the kick occurs. As we show, any three points

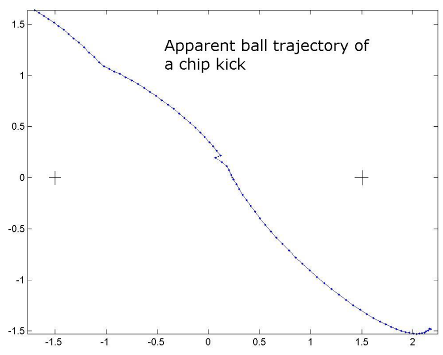

Fig. 1. Chip kick as seen with two global cameras in a small-size field (the kick starts in the lower right). The axis dimensions are given in meters. The cameras are located at the coordinates $(1.5, 0)$ and $(-1.5, 0)$. Their fields of view overlap only slightly in the middle of the field.

measured at any time in the apparent trajectory of the ball can be used to reconstruct the three-dimensional parabolic path. We do not need to apply methods from stereoscopic vision [1], [2].

Kim et al. [3] have studied a related problem: the reconstruction of the parabolic flight of a ball from a video of a soccer game. However, their method is based on using the two extremes of the parabola (when the ball touches the ground, at the start and at the end of the ballistic motion), and is not suitable for predicting future parabolic motion using a minimal number of points just after a kick. A variation of their method, in which they adjust a quadratic function to many alternative planes of motion, searching among them for an optimal fit, is too cumbersome and inefficient. The method described here, by contrast, is direct, does not require any search driven computation, and can be used for forecasting future motion using only three video frames just after the kick.

2 Projective Transformation and Reconstructed Path

We adopt the following conventions. The origin of world coordinates is on the floor. The camera is situated above the field. Figure 2 shows how the two coordinate systems (field and camera) are related. Without any corrective computation, the flying ball is interpreted as a "virtual ball" rolling on the ground along a curved path. In the general case, the three axis of the camera's system of coordinates are rotated relative to the field's coordinate axis. Let us denote by R the rotation matrix needed for transforming from world to camera coordinates, and by ℓ the translation vector from the origin of the world system to the camera coordinate system. Therefore, a point with world coordinates $p = (x, y, z)^{\mathrm{T}}$ has coordinates $q = R(p - \ell)$ in the camera's coordinate system. The inverse transformation, from camera to world coordinates, is therefore $p = R^{-1}q + \ell$.

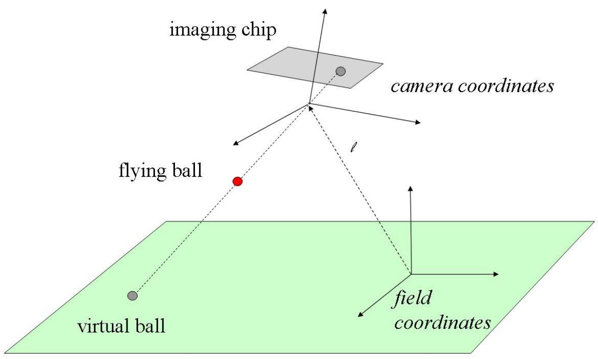

Fig. 2. Coordinate systems of the field and camera

Our computer vision system is self-calibrating. Just by selecting four points on the field, whose world coordinates are known, such as the corners of the field itself, the software computes automatically the rotation matrix R and the translation vector ℓ mentioned above (by solving a system of linear equations, see [2], [5]). Given the position of an object in a pixel of the imaging chip, the computer vision can then determine its position in world coordinates on the field.

In what follows, we assume that the successive projections of the ball on the field, with world coordinates $p_i = (x_i, y_i, 0)$, for $i = 1 \ldots$, have been transformed to the camera coordinate system using the transformation $R(p_i - \ell)$. This is a straightforward step since the matrix R and the vector ℓ are known from the initial calibration step. This also means that the camera can have any angle and displacement in relation to the world coordinates, and that the method described next is completely general.

We assume also, for the sake of the computation, and without losing generality, that the camera imaging chip is at a unit distance from the pinhole. The point where parabolic flight starts to be measured has camera coordinates (x_0, y_0, z_0). The velocity of the ball, after the kick, is given by the vector $v = (v_x, v_y, v_z)$. The parabolic flight of the ball is then described by the following parameterized path, in the camera's coordinate system:

$$(x, y, z) = \left(x_0 + v_x t + \frac{1}{2} g_x t^2,\ y_0 + v_y t + \frac{1}{2} g_y t^2,\ z_0 + v_z t + \frac{1}{2} g_z t^2 \right)$$

where t is the time elapsed, starting with the first data point ($t = 0$ for that point), and (g_x, g_y, g_z) are the components of the earth's acceleration in the camera's coordinate system (which can be tilted with respect to the vertical).[1]

[1] Gravitational acceleration varies with the latitude, because the earth is not perfectly spherical. The Geodetic Reference formula of 1967, used by geographers, is given by $g = -9.7803185(1 + 0.005278895 \sin^2 \phi - 0.000023462 \sin^4 \phi)$ m/s. Surface features of the earth are not considered in the formula.

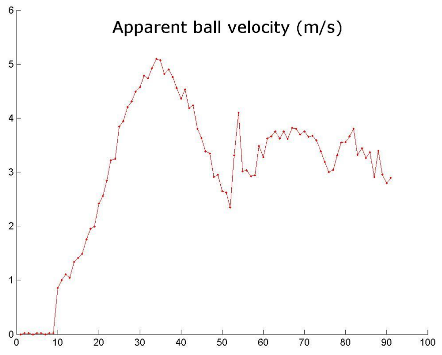

Fig. 3. Apparent velocity of the "virtual" ball from frame to frame in m/s

The components of the earth's acceleration with respect to the camera system can be computed using the rotation matrix R:

$$(g_x, g_y, g_z)^{\mathrm{T}} = R(0, 0, -9.8)^{\mathrm{T}}$$

Let us assume that the kick is never so high as to reach the camera plane (that is, $z \neq 0$). The projection of the position of the ball in the image plane of the camera (at a unit distance from the pinhole) is then

$$\left(\frac{x_0 + v_x t + \frac{1}{2}g_x t^2}{z_0 + v_z t + \frac{1}{2}g_z t^2}, \frac{y_0 + v_y t + \frac{1}{2}g_y t^2}{z_0 + v_z t + \frac{1}{2}g_z t^2}\right)$$

Assume now that m points in m images are given, where the "virtual" ball has been detected at times $t_1, t_2, \ldots, t_m$ (setting $t_1 = 0$). Let us denote the coordinates of the m points on the ground with respect to the camera system by $(x'_1, y'_1, z'_1), \ldots, (x'_m, y'_m, z'_m)$. Then, since "virtual ball" and real ball have the same projection on the camera chip, we have in general:

$$\left(\frac{x'_i}{z'_i}, \frac{y'_i}{z'_i}\right) = \left(\frac{x_0 + v_x\, t_i + \frac{1}{2}g_x t_i^2}{z_0 + v_z t_i + \frac{1}{2}g_z t_i^2}, \frac{y_0 + v_y\, t_i + \frac{1}{2}g_y t_i^2}{z_0 + v_z t_i + \frac{1}{2}g_z t_i^2}\right) \qquad \text{Eq.1}$$

We start processing the video sequence only when the ball is moving (which is easy to check). We denote by α_i the ratio x'_i/z'_i and by β_i the ratio y'_i/z'_i. From the expression above (and for the i-th point) we can derive two linear equations:

$$z_0\alpha_i + v_z\alpha_i t_i - x_0 - v_x t_i + 0 \cdot y_0 + 0 \cdot v_y t_i = -\frac{1}{2}g_z\alpha_i t_i^2 + \frac{1}{2}g_x t_i^2$$

and

$$z_0\beta_i + v_z\beta_i t_i + 0 \cdot x_0 + 0 \cdot v_x t_i - y_0 - v_y t_i = -\frac{1}{2}g_z\beta_i t_i^2 + \frac{1}{2}g_y t_i^2$$

We have two linear equations for six variables. Therefore, three points on the parabolic flight path provide enough equations (six) which can be used to solve

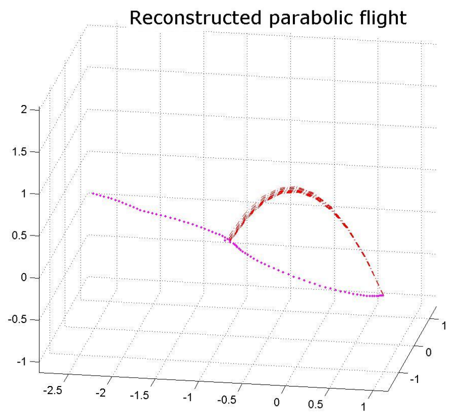

Fig. 4. Reconstruction of the parabolic flight (20 to 30 points on the trajectory were used)

the system. If we use more than 3 points, let us say m, then the general form of the system of equations we obtain is

$$D(z_0, v_z, x_0, v_x, y_0, v_y)^{\mathrm{T}} = d$$

where D (for data) is a $2m \times 6$ matrix and d is a $2m$-dimensional vector. Using the pseudoinverse D^+ of D we find the solution

$$(z_0, v_z, x_0, v_x, y_0, v_y)^{\mathrm{T}} = D^+ d$$

where $D^+ = (D^T D)^{-1} D^T$. The pseudoinverse allows us to use as many points for the calculation as we have already measured, because we are interested in producing an estimate of the flight trajectory of the ball as soon as possible, but also as precise as possible.

The computed initial position of the ball (x_0, y_0, z_0) and the velocity vector (v_x, v_y, v_z) must be transformed now from the camera's to the world's frame of reference. That is, we compute

$$(x_0^w, y_0^w, z_0^w)^{\mathrm{T}} = R^{\mathrm{T}}(x_0, y_0, z_0)^{\mathrm{T}} + \ell$$

and

$$(v_x^w, v_y^w, v_z^w)^{\mathrm{T}} = R^{\mathrm{T}}(v_x, v_y, v_z)^{\mathrm{T}}$$

where the superindex w means "world coordinates". Since R is a rotation matrix, its inverse is the transpose of R.

A word of caution: since we start the clock t_i when we get hold of the first data frame, it is necessary to start the calculation only when the ball is really moving fast on the field (a ball being pushed by a robot is usually not so fast). Fig. 3 shows the velocity of the ball on the ground, as computed from a series of frames of a real game. The moment after the ball has been kicked (frame 10) is readily identifiable. Therefore, it is easy to avoid performing erroneous calculations for a ball which has not been kicked (more about this below).

3 Resetting the Clock

We assumed in the previous sections that the clock starts running with the first available data point (after we detect that the ball was kicked). If we miss the exact frame in which the ball was kicked, we can start the clock later with any frame in which we see the "virtual" ball still on the air. When we compute z_0^w, we can compare it to 0 (points on the field have z coordinate equal to zero). If both are different, we can compute the time T it took the ball to reach the height z_0^w(relative to the field plane). Having this elapsed time, we can go "back in time" and find the exact point where the ball was kicked.

Let v_{z0}^w be the initial vertical velocity of the ball, at the moment of the kick. Then, if v_z^w is the vertical velocity in the frame we started recording data, we know that

$$v_z^w = v_{z0}^w - gT$$

where g is the downward earth acceleration, and

$$z_0^w = v_{z0}^w T - \frac{1}{2} g T^2 = v_z^w T + \frac{1}{2} g T^2.$$

Solving this quadratic equation, we find T. With T, we compute v_{z0}^w and the following corrections for all other parameters (the superindex f means *final*):

$$\begin{aligned} x_0^f &= x_0^w - v_x^w T & \qquad v_x^f &= v_x^w \\ y_0^f &= y_0^w - v_y^w T & v_y^f &= v_y^w \\ z_0^f &= z_0^w - (v_z^w T - \tfrac{1}{2} g T^2) & v_z^f &= v_z^w + gT \end{aligned}$$

4 Experimental Results

No robotic vision system is perfect. There is an intrinsic error in all measurements of the ball position. This noise, which is even larger when the ball is flying high, makes necessary to use more than three data points in order to make the best future projection of the ball's path. We have experimented with several different numbers of points. The best results were obtained (for our computer vision software) when more than 20 points were used for the ball trajectory. The camera was running at 52.7 frames/second. The ball was in flight for around 60 frames, so that we had to use one third of the data points to get a good reconstruction of the parabolic path, with subcentimeter accuracy.

Fig. 4 shows the result of the inverse computation described in this paper. Given the path of the ball projected on the ground, we computed parabolas based on 20 to 30 data points from as many video frames. The ball is in flight for around 50 frames. All parabolas are very near to each other.

Fig.5 shows a reconstruction computed with only ten frames, starting the computation at different moments (between frame 10 and frame 20). The figure is a screenshot of our vision software for the small-size league [4],[5], enriched with this new feature.

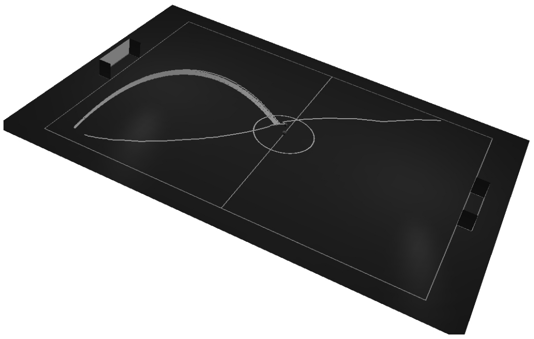

Fig. 5. Reconstruction of the parabolic flight (10 points on the trajectory were used, several different start frames were selected). The color screenshot is from our our small-size computer vision program.)

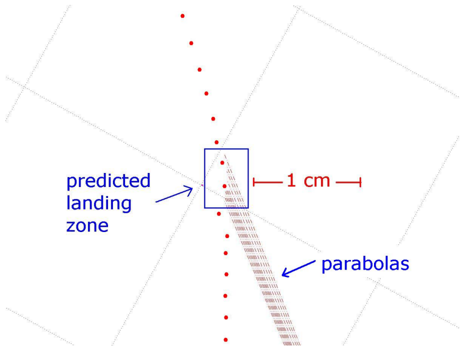

Fig. 6. Predicted impact position computed with 5 to 14 points from the trajectory. The predicted impact position is at the end of each parabolic path line. The isolated points are “virtual” images of the ball, before and after impact.

Fig.6 shows the predicted impact position of the ball after a chip kick, using different numbers of data points (from 5 to 14 frames). Each prediction is the tip of the predicted flight line. As can be seen, the spread of the prediction is smaller than 1 cm. Such an error, after a 2m chip kick, is almost negligible.

Fig.7 shows how the three components of the 3D-velocity of the ball converge to very precise values, when the number of points along the parabola is increased from 5 to 25. Already ten points produce a very good estimate of the velocity vector. The time needed, when using ten frames for the computation, is about one fifth of a second.

5 Reaction Time

The parabolic flight reconstruction would be only of theoretical interest if the goalie had not time to react. We show now that in many cases the goalie can

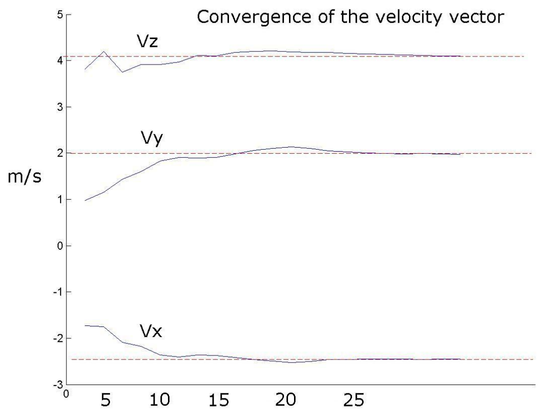

Fig. 7. Convergence of the three components of the velocity vector for different number of data points on the parabola

indeed intercept the ball. Whatever the distance from the goal line is, the ball has to be lifted above the goalie, that is, above 0.15 m. The time the ball is in the air in parabolic flight is given by $t = 2\sqrt{2h/g}$. Given a minimum height of 0.15 m, and the acceleration constant g, we obtain a minimum time of flight of $t = 0.35$ seconds. When the ball falls (for half of the flight-time) starting from rest at the highest point of a parabola just high enough to fly above a robot, the final velocity is $v_z = -0.175\,s \times g$, which is around -1.7 m/s. This is also the initial vertical velocity of the ball, when it is chip-kicked (with positive sign). Therefore, any slower ball will not fly high enough to go above the goalie or any robot on the field. This can be used as a fast check for a "dangerous" flying ball. Fig. 8 shows how fast the "virtual" ball is accelerated by our chip kick. The maximum apparent acceleration value is around 40 g's. The apparent acceleration remains positive as long as the ball is approaching the camera objective and its apparent speed continues to increase. When the ball starts falling, it seems as if the ball falls being accelerated by more than one g.

Now we can consider if a chip kick is stoppable. The delay from our vision system is about 100 ms (that is the time it takes an image to be accepted and processed)[4]. Assume that we use 5 frames for predicting the parabolic flight (first provisional estimation). At 52,7 frames per second, those 5 frames correspond to 95 ms. The robot has just 350 – 195 = 155 ms for moving to the approximate blocking position. If the distance is not very large, the robot could reach the correct position. Our robots travel at around 2 m/s. They can move 31 cm in 155 ms, but only after having been accelerated. The question becomes therefore how fast can the robots start moving from scratch.

First we would like to have a table of the flight time of the ball for different distances on the field. Let us assume that the kick is optimal: the ball goes only as high as it needs to go, and no higher. The ball can go forward as fast as desired, but the real limit for the ball is to go above the goalie and below the

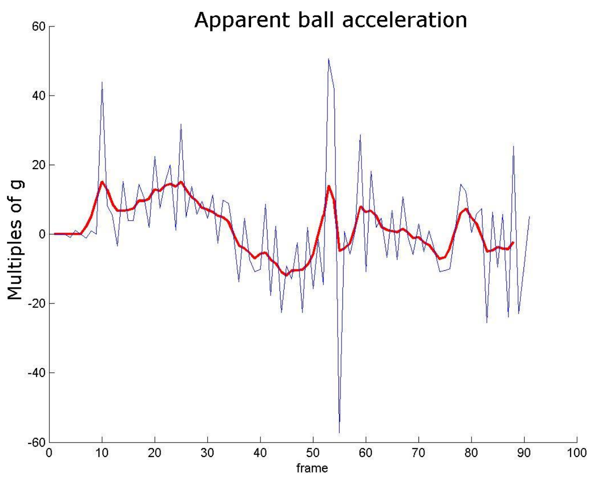

Fig. 8. Apparent acceleration of a ball kicked high (virtual ball acceleration)

goal bar (positioned at 15 cm). We do the following: we position the goalie at a distance r from the goal line. The goalie has a width of 18 cm and a height of 15 cm. The ball has to go under the bar. The ball has a diameter of 4 cm (in reality 4.3 cm, but let us simplify). We then compute the best parabolic shot that clears the robot and enters the goal box going below the bar. The time of flight is then plotted according to the coordinates on the field of the point where the ball is kicked.

The results obtained with this method can be seen in Fig.9. As stated before, there is a minimum time of flight of around 0.35 seconds. When the goalie is 10 cm away from the goal line, the minimum time for a chip kick over the goalie from far away (5 meters) rises to about 0.7 seconds, double the minimal time. However, teams cannot shoot so impressively from far away. They cannot adjust the angle of the shoot, which is fixed, and this produces suboptimal trajectories. Therefore, in practice, the real time of flight will be always much larger than the lower and optimal bound computed here.

We measured the velocity of some of our robots. Due to initial inertia, our goalie needs around 0.2 seconds to move 10 cm. An optimal shot from midfield requires around 0.5 seconds flight time. If we collect 10 data points before moving the robot (in 0.2 seconds), and if the vision delay is 0.1 seconds, we have 0.2 seconds left to position the goalie. If we need to move the robot less than 10 cm, stopping a chip kick is possible. Parabolic flight prediction has been incorporated now in our vision software. It is possible to stop all chip kicks coming from far away. Stopping nearer chip kicks depends on the position of the robot. Chip-kicks from anywhere near the goal area are unstoppable, if the goalie is not touching the line.

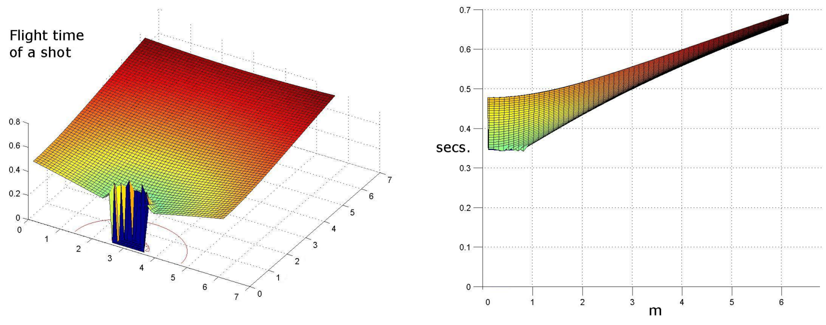

Fig. 9. Time of flight in seconds for optimally lifting the ball above the goalie and scoring (left image). The goalie has been positioned 10 cm away from the goal line. Lateral view of the same surface (right figure). The maximum distance is 6 meters (x-coordinate 6). The time of flight was set to zero 50 cm from the center of the goal box (a graphic artifact arises). Chip kicks from far away require 0.7 seconds. The minimum flight time for lifting the ball above the goalie is 0.35s.

6 Conclusions

This paper has shown that after a chip kick it is possible to reconstruct the three-dimensional flight path of the ball using a single camera overlooking a field. We have shown that the minimal velocity required for clearing a small-size robot is 1.7 m/s. This criterion can be used as a trigger for starting the code which computes three-dimensional paths. Hard flat shots can be discriminated by just looking at the "straightness" of the ball's path. A false alarm triggered by a robot pushing the ball in a curved path can be handled easily since no robot can be moving right behind a ball in parabolic flight. It can also be checked if the path corresponds to an acceleration compatible with the gravity constant. If not, the parabolic path can be discarded.

The method described in this paper can be used also in the mid-size league with omnidirectional or frontal cameras. Since the computer vision software for omnicameras produces an estimation of the position of the ball on the floor, as if we had a camera overlooking the field, the method described here can be used without any modification (provided the ball is visible under the horizontal plane of the omnicam).

There are other applications for our technique, for example, for obtaining the orientation of a camera overlooking a featureless area. If the camera is not pointing straight down, its orientation can be computed by tracking a ball thrown in front of the camera. The method described in this paper must be just extended to deal with a gravitational acceleration with unknown components (g_x, g_y, g_z) in the tilted frame of reference of the camera. In that case, Eq. 1 transforms into

$$(\alpha_i, \beta_i) = \left(\frac{x_0 + t_i\, v_x + \frac{1}{2} g_x t_i^2}{z_0 + t_i v_z + \frac{1}{2} g_z t_i^2}, \frac{y_0 + t_i\, v_y + \frac{1}{2} g_y t_i^2}{z_0 + t_i v_z + \frac{1}{2} g_z t_i^2} \right)$$

where (α_i, β_i) are the ball coordinates in the camera's chip. This expression provides two linear equations for each detected ball in each frame. For the nine unknowns we need at least five points to compute the value of (g_x, g_y, g_z), and from this, the angle of the camera axis with the vertical.

There is also an application for stereo-vision. If two stereoscopic cameras overlook a common field of view, and if their pose is unknown, an object thrown in parabolic flight can be used to determine the direction of gravity relative to each camera. The cameras do not need to be triggered simultaneously. Bringing the reconstructed parabolic flight curves for each camera in correspondence can then be used to calibrate the stereoscopic vision system. The numeric for such stereoscopic calibration without camera parameters is much simpler than that used by other groups [6], [7], and does not require matching points, nor the world coordinates of the corresponding points in space. This result is more general than Luong and Faugeras technique [8], and will be published elsewhere.

References

1. Howard, I.P., Howard, A.P., Rogers, B.: Binocular Vision and Stereopsis. Oxford University Press, Oxford (1995)
2. Forsythe, D., Ponce, J.: Computer Vision: A Modern Approach. Prentice-Hall, Englewood Cliffs (2003)
3. Kim, T., Seo, Y., Hong, K.-S.: Physics-based 3D position analysis of a soccer ball from monocular image sequence. In: ICCV 1998, 1998th edn., Bombay, India, pp. 721–726 (January 1998)
4. Simon, M., Behnke, S., Rojas, R.: Robust Real Time Color Tracking. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 62–71. Springer, Heidelberg (2001)
5. Simon, M., Wiesel, F., Egorova, A., Gloye, A., Rojas, R.: Plug & Play: Fast Automatic Geometry and Color Calibration for Tracking Mobile Robots. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, Springer, Heidelberg (2005)
6. Hartley, R., Gupta, R., Chang, T.: Stereo from Uncalibrated Cameras. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 761–764. IEEE Computer Society Press, Los Alamitos (1992)
7. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
8. Luong, Q-T., Faugeras, O.: Self-calibration of a stereo rig from unkown camera motions and point correspondences. In: Bensoussan, A., Verjus, J.-P. (eds.) Future Tendencies in Computer Science, Control and Applied Mathematics. LNCS, vol. 653, Springer, Heidelberg (1992)