

High-speed vision tracking system and the study of limitations in a cost-effective implementation

Robert Lukierski

Abstract

This paper proposes the concept of an efficient prediction method along with an appropriate image processing, focused on usage in a low-cost vision-based control application. Described method concerns with communication latencies, the lack of hard real-time traits and flaws of mechanical device construction that are present due to insubstantial resources available. In spite of the issues mentioned above, the high-speed operation of a system is the most important objective; in our setup the processing speed of 80 fps is achieved. Accurate measurements of the image processing algorithm and control approaches are also provided. A discussion on possible improvements (both applicable to the hardware and algorithms) is presented.

1 Introduction

Currently, vision systems are well-known. It is not only a research field, as it used to be. It is being applied successfully in the manufacturing industry, in consumer products and every other field where it can make human work easier. Object tracking is such subject, especially when the high speed operation is demanded. Human beings apply vision-based control with fine results, but when high speed events occur our perception is very limited. Because of the advances in technology, higher speeds can be achieved more easily, but it still requires uncommon cameras with a sophisticated image processing hardware.

In the past, there were many attempts intended to handle the high speed real-time control task in pong/table tennis game. One of the first was conducted by Russel Andersson in the late 80's [1]. His vision-control system, at the time it was developed, was the state of the art, although it was extremely expensive. He introduced high speed operation (60 Hz) of the stereo-vision system for ball tracking in a table tennis game with an industrial robot. He employed an ASIC chip, developed for image moments calculation and an "Expert Controller" [2], used for sophisticated physics-based predictions. Subsequent major improvement in the area of robot table tennis was accomplished at Osaka University by the group lead by Fumio Miyazaki [3]. Their ongoing research, since 2001, on the topic involves "k-d tree" approach ; the wall bouncing problem [4] and the paddle dexterity. During their work multiple generations of devices were constructed to verify presented ideas.

In this paper a low-cost device is used both as a design goal and as a verification platform; also, we employ not so elaborated vision hardware. Nonetheless, the task remains still complex. Such limitations indicate new challenges to the topic, as cost awareness affects other requirements and design principles. Constructed robot system simulates one of the first computer games - “The Pong”. The aim of the robot is to defend a ball shot initiated by a human opponent.

2 System

2.1 Device

Proposed system contains:

- Mechanical device – a robotic “flipper”,
- Digital controller,
- Mono vision camera,
- Image processing software on a PC.

The device (Fig. 1a) is constructed on a table, only the PC is an external component to the system. The active playable area dimension is 800×600 mm. The area is surrounded from two sides by bounds. Opposite sides are used for human opponent’s operation and the “flipper” mechanism. The latter is a precisely CNC-machined device that comprises the “flipper”, which is travelling on a guide by linear ball bearings. The “flipper” is connected to a belt passing through two appropriate gears, where one is rotated by a 120 W brush DC motor and another serves as a belt tightener.

This straightforward mechanical device is one of the most expensive elements of the system, however this component of the assembly causes many problems, e.g., limited control signals, the “flipper” inertia, motor delay constant, etc.

The motor is powered and controlled by a 12 V / 15 A H-bridge based on MOSFET transistors; also a Hall-Effect-based current sensor (ACS712) is build in. To read the “flipper” position a magnetic rotary encoder (AS5040) is mounted on a powered gear shaft.

For elaborated tracking system the object is, obviously, an essential part. In our case a golf ball had been chosen, mainly due to the mass of the ball. This, along with the usage of mono-vision camera, was because of the preliminary requirement that implied 2D system design.

The robot (Fig. 1b) is controlled by a microprocessor system, based on a NXP LPC2378 microcontroller of the ARM7TDMI family. Sophisticated firmware was designed and implemented in C++. It employs a real-time operating system, a MODBUS protocol stack, the “flipper” controlling code and modules for collecting measurements, statistics and debugging. The “flipper” is controlled by the PID regulator at 1 ms cycle. Major flaw of this subsystem is the latency on a RS-232/MODBUS communication link with the PC.

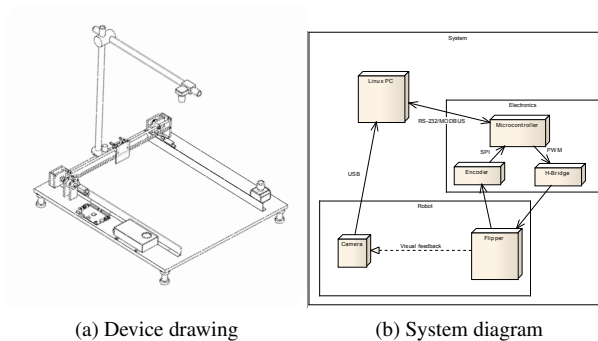


Figure 1: The system

An IDS uEye UI-1225LE-C camera along with the PC forms a vision system. The camera contains a CMOS sensor and utilises the USB 2.0 interface. The camera is capable of acquiring 80 frames per second at the highest resolution (752×480) and faster at reduced resolutions. Developed PC software is running on Ubuntu GNU/Linux operating system (not equipped with real-time features). In described arrangement the camera is running at approximately 80 frames per second, so exposure time is so low that proper lighting conditions must be met for correct system operation. To reduce the dimensions of our system a wide-angle lens is used, but such feature has also disadvantages – it intensely affects the video image, introducing barrel shaped distortions. The PC based system lacks also hard real-time traits, which is often the key for a successful control system. Even the camera operation is non-deterministic; speed of 80 fps (12.5 ms period) is only an estimate – mainly due to the OS and the USB mode of operation. The only one subsystem that complies with hard real-time guidelines is the digital controller.

2.2 Distortions

Barrel shaped distortions can be eliminated by undistorting the image with the camera’s intrinsic parameters, estimated from the camera calibration process, e.g., the chess-board pattern method [5]. This approach gives satisfactory results and the ability to calculate easily object’s position in real world coordinates, but has one vital deficiency – computation effort that leads to undistorting the image takes from 20 ms to 35 ms (even LUT based version of this method, see Table 1), which is not acceptable in such high-speed system.

The region of interest, defined by quadratic functions (Fig. 2b), can fairly well approximate active area’s region on the image, similar technique was used in [6] for correcting distortions in the image of a RoboCup field. Object’s position evaluated in camera sensor coordinate system (pixels) can be transformed into real world coordinates using *a priori* information about active area’s physical dimensions.

For the sake of image processing time optimisation, another concept is employed within. The processing of complete image consumes additional milliseconds unnecessarily, so the right-hand side of a “computable” ROI may be limited to the first found

trace of a ball plus some epsilon that is equal to the average ball diameter, estimated previously from data sets. This optimisation also eliminates most of unwanted disturbances and noises that may emerge from human opponent’s motions at the time of collision. This method implies an assumption that the ball is the first leftmost object in the scene.

The ROI upper boundary is defined in (1) and the lower boundary in (2).

$$Y_1(x) = a_2x^2 + a_1x + a_0 \quad (1)$$

$$Y_2(x) = b_2x^2 + b_1x + b_0 \quad (2)$$

Real area physical dimensions are defined as $A = (w, h)$, left and right ROI bounds are represented by k_1 and k_2 respectively. The ball position in real coordinate system, with its origin in right bottom corner, is defined as (r_x, r_y) expressed by (3) and (4).

$$r_x = \frac{(k_2 - p_x)w}{k_2 - k_1} \quad (3)$$

$$r_y = \frac{(Y_2(p_x) - p_y)h}{Y_2(p_x) - Y_1(p_x)} \quad (4)$$

2.3 Prediction Algorithm

In the elaborated system the mechanical construction is a serious bottleneck, both in the terms of strict time constraints and reliability. The “flipper” is controlled by the PID algorithm, correctly implemented and tuned (using Ziegler-Nichols tuning rules [7]), albeit it had to be weakened, because of motor heating problems. Its dead-band is quite wide, about 40 mm in total, applied to limit the number of changes in the H-bridge state machine, what causes massive current flow between motor’s brushes.

Due to such drawbacks “flipper’s” mean acceleration is only approximately $3.22 \frac{m}{s^2}$ and the maximal speed is around $1.6 \frac{m}{s}$. “Flipper’s” travel time from the midpoint to the one of the opposite ends is determined to be about 150 ms, which may be not sufficient for desired high speed object tracking – the observed maximal ball velocity was about $6 \frac{m}{s}$.

Such circumstances make visual servoing task challenging and therefore smart and fast prediction algorithm is required to forecast object’s trajectory. A predictive approach may also reduce the computational effort and limit the number of control commands that have to be sent to the microcontroller. Prediction based on a physical model requires *a priori* information on various parameters of the environment and the object itself; also, such calculations may become too complex to perform during the short time frame.

After the beginning of the motion the first three consecutive object positions are recorded (Fig. 2a), their coordinate system is converted to the real world dimensions by the ROI concept described in the Section 2.2 and they are used as input data for the algorithm. Object’s trajectory is expressed by the first order polynomial (5), fitted by the least squares approximation.

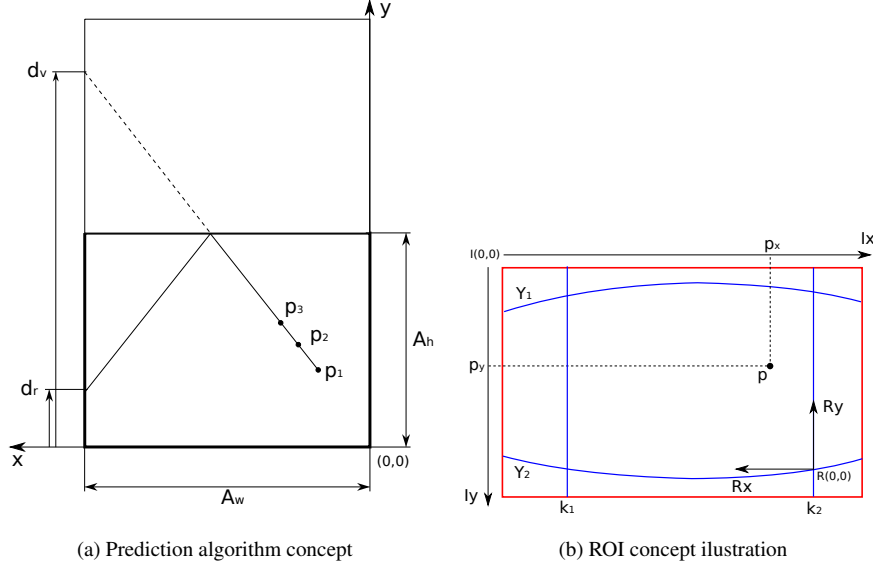


Figure 2

$$y(x) = a_1x + a_0 \quad (5)$$

The trajectory equation is evaluated for x value that represents the position of the left border of the active area (the “flipper” mechanism position). The result is the prediction of “virtual” suitable position of the “flipper” that must be converted to the physical position by the formula (6).

$$d_r = \begin{cases} d_v \bmod A_h, & \text{if } \lfloor \frac{d_v}{A_h} \rfloor \text{ is even;} \\ A_h - (d_v \bmod A_h), & \text{if } \lfloor \frac{d_v}{A_h} \rfloor \text{ is odd.} \end{cases} \quad (6)$$

The algorithm, described above, conforms with the simplifying assumptions that the high-speed object movement is linear and the collision incidence angle is equal to the reflection angle.

2.4 Image Processing

Image processing techniques are utilised to extract ball’s position from an image. The high-speed operation brings various difficulties like low brightness (caused by short exposure time), noises (especially in CMOS cameras) and image distortions (described previously in this section). However, the demands for the image processing stage are high. The image processing algorithm has to be rapid, even for high data throughput, should have ability to deal with distortions, should feature high accuracy of object’s position estimation and be quite robust for factors like lighting conditions and

various disturbances of other origins. The ability to deal with the distortions, with providing the necessary accuracy, is essential because of the prediction algorithm, which is very sensitive to the input data errors (they can lead to the invalid value of the slope parameter for the linear function and in effect large prediction errors).

The method that is comprised of multiple processing techniques is proposed. The algorithm operates only on a ROI defined by the quadratic bounds described above, for each pixel its luminosity is calculated and a difference between pixel brightness on the previous frame and the current one is computed. The difference is used as a method of motion detection, with appropriate thresholding it is able to reduce noises and other unwanted disturbances like background or nonuniform lighting. Only on the area where this difference is significant (motion occurred) and the current pixel luminosity is above the threshold (for the sake of simplification the ball is white in contrast to the black background) a centroid can be computed. It is expressed in (7) and (8).

$$x_c = \frac{1}{N} \sum_A x \quad (7)$$

$$y_c = \frac{1}{N} \sum_A y \quad (8)$$

where A is the set of pixels that conform to the criteria presented above (in the ROI, large difference between frames and the brightness above the threshold), N is the cardinal number of this set, i.e., the number of pixels computed. The point defined by (x_c, y_c) is the centre of mass of the detected object.

3 Results

3.1 Image Processing Efficiency

To illustrate limitations, caused by short time frame in the high-speed operation, the measurements of computation effort, for different image processing algorithms, are presented in the Table 1. The implementation is based on Intel's OpenCV library [8]; tests were run on Intel's T5600 1.83 GHz processor using the Linux operating system. The measurements were collected by using precise POSIX RT timers.

The image processing algorithm proposed in the Section 2.4, without any ROI computes in 7.777 ms on average with the standard deviation of 0.483 ms.

3.2 Communication Latency

Despite image processing, another important factor for the system operation is the latency time in the MODBUS communication link. The command for writing a 16-bit register over the 115200 bps RS-232 link takes on average 8.102 ms with the standard deviation of 1.066 ms.

Table 1: The comparison of image processing algorithms

Algorithm	Average [ms]	σ [ms]
RGB to luminosity conversion	2.457	0.055
Hough transform	34.198	1.02
Image moments calculation	2.022	0.051
OpticalFlow LK	28.895	0.678
Gaussian smooth	4.818	0.233
Undistorting RGB image	33.512	0.92
Undistorting RGB image by “remap”	20.812	0.481

3.3 Measurement Methodology

The research was based on a set of test cases that were recorded at 80 fps with their accurate timestamps (POSIX RT timers). This strategy had been chosen, because the system involves phenomena, which nature is above human perception. Also, the requirement for measurement repeatability is required in the scientific method. Without an “on-line” operation the software has to simulate camera operation, like delays or even dropping frames when necessary. The simulation concerns only the camera, the device is still connected and operational, as it would be in the normal mode of operation.

The measurements of device parameters are performed in the microcontroller and they are transferred to the PC at the end of each test case. The parameters measured in the microcontroller during the time of a shot are:

- Duration time,
- Total travelled distance,
- Minimal and maximal “flipper” speed,
- Average and maximal power on the motor,
- Energy consumption,
- The “flipper” position at the beginning and at the end of a test case.

3.4 Experiments

The measurements were taken both for the elaborated prediction algorithm and for a continuous regulation of the “flipper”.

The important parameters that illustrate the abilities and the restrictions of each control approach are:

- Energy consumption,
- Total travelled distance,

- The number of lost frames,
- Error in the final “flipper” position.

The total travel distance (Table 2a) is in almost all cases lower for the prediction method, because device operation is not affected from numerous set point changes. These changes also impact the number of lost frames (Table 2b), which is mainly caused by applying a MODBUS command with its latency time.

Table 2

(a) Total travel distance (in milimeters)			(b) Number of lost frames		
Test case	Continuous	Prediction	Test case	Continuous	Prediction
1	28.4	0	1	7	2
2	114.4	117.8	2	5	2
3	165.3	163.9	3	9	1
4	595.5	47.1	4	13	2
5	81.7	66.9	5	4	2
6	41.1	14.6	6	5	3
7	52.3	0	7	4	1
8	258.8	200.6	8	8	1

The differences in the energy consumption (Table 3a) are evincible, due to set point changes in the continuous approach the motor current raises significantly, especially when the motor’s movement direction is being reversed.

Table 3: Energy consumption (in joules)

Test case	Continuous	Prediction
1	12.668	0.006
2	54.359	45.078
3	69.877	56.854
4	158.087	20.500
5	48.340	35.202
6	22.236	14.724
7	35.654	0.114
8	71.478	61.465

There are two factors that describe an error for the predictive approach, both can be used as a quality criterion. The first one is the error between the last observed ball position and the final “flipper” position, read from the device by MODBUS. In an identical manner the error from the continuous approach is computed. The second factor is the error between the last observed ball position and the point estimated from the prediction algorithm only. The second factor does not include disturbances from

Table 4: Absolute error between the last ball position and the final position of the “flipper” (in milimeters)

Test case	Continuous	Prediction	Prediction 2
1	22.02	5.29	0.79
2	9.88	13.28	1.52
3	2.13	4.87	16.37
4	35.09	289.69	290.59
5	58	15	23
6	0	4	4
7	57.8	57	34
8	2.1	4	9

mechanics, like PID regulation with its wide dead-band, it is computed only on the data collected in the PC.

Absolute errors (Table 3b) can estimate how many shots were handled; the “flipper” width is 100 mm, so if an absolute error is less than a half of this dimension it can be treated as a success.

Table 5: Mean squared error

Factor	All test cases	Number 4 ignored
MSE for continuous	1065.92	1042.29
MSE for prediction	10956.79	533.43
MSE for prediction 2	10811.93	293.27

For the mean square error calculations (Table 4) the test case no. 4 is omitted, because it is a quite complex scenario with trajectory too ambiguous for applied algorithms.

Below the analysis of two interesting test cases is presented. The images (Fig. 3a and 4a) are displayed inversed for better readability. The plots (Fig. 3b and 4b) illustrate the prediction algorithm operation in time. On the plots (Fig. 3c and 4c) the difference in the execution time between both control approaches may be noticed.

Six of the eight test cases end with a success, so overall effectiveness of the system can be estimated roughly at about 75 %, but such result should be confirmed by a larger number of tests.

4 Conclusions and Future Work

4.1 Conclusions

In this paper the prediction method for the visual tracking of a high-speed object was proposed. The method deals with impediments caused by the limitations introduced

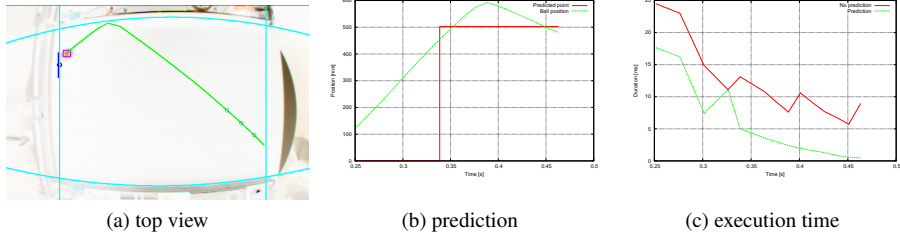


Figure 3: Test case 5

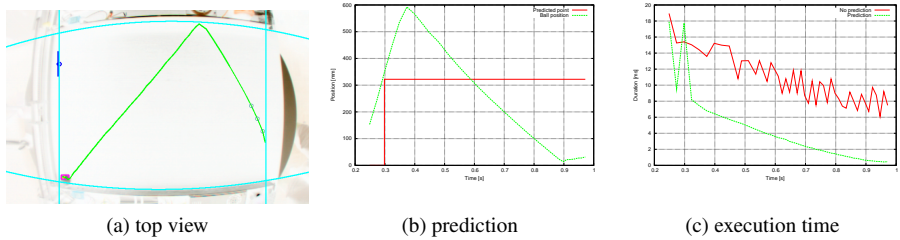


Figure 4: Test case 4

by a cost-effective device design, such as the lack of real-time features, communication latencies and low quality input data (distortions, noise). The system was verified in practise, engaging industrial high-speed camera, CNC machined device and an application specific digital controller. Experiments and proper measurements were performed to exemplify the demands for high-quality prediction in the elaborated system.

4.2 Future Work

In spite of many device defects, an FPGA based implementation of the system can be considered. This may bring the real-time operation, reduce communication latencies and, in effect, overcome the bottlenecks of the mechanics. Although, the FPGA design will significantly increase the total cost of the device and will need more elaborated and longer human work.

Acknowledgements

The author would like to thank dr Marek Wnuk, the supervisor of my Master Degree thesis, for valuable discussions and support. This paper contains main part of the results belonging to this Master Thesis.

References

- [1] Russell L. Andersson. *A robot ping-pong player: experiment in real-time intelligent control*. MIT Press, Cambridge, MA, USA, 1988.
- [2] R.L. Andersson. Understanding and applying a robot ping-pong player's expert controller. *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1284–1289 vol.3, May 1989.
- [3] F. Miyazaki, M. Takeuchi, M. Matsushima, T. Kusano, and T. Hashimoto. Realization of the table tennis task based on virtual targets. *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 4:3844–3849 vol.4, 2002.
- [4] M. Takeuchi, F. Miyazaki, M. Matsushima, M. Kawatani, and T. Hashimoto. Dynamic dexterity for the performance of "wall-bouncing" tasks. *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2:1559–1564 vol.2, 2002.
- [5] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.
- [6] Mark Simon, Sven Behnke, and Ral Rojas. Robust real time color tracking. In *4th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences), Lecture Notes in Computer Science*, pages 239–248. Springer, 2000.
- [7] J G Ziegler and N B Nichols. Optimum settings for automatic controllers. *Trans. ASME*, (64):759–768, 1942.
- [8] Intel. OpenCV library, <http://www.intel.com/technology/computing/opencv/>.