Object Tracking With the Use of a Moving Camera Implemented in Heterogeneous Zynq System on Chip

Marcin Kowalczyk[1] and Tomasz Kryjak[1]

AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland
mkowalcz@student.agh.edu.pl, tomasz.kryjak@agh.edu.pl

Abstract. In this paper a hardware-software implementation of an object tracking system, which uses a moving camera is described. Selected algorithms: mean-shift, particle filter, KLT and so-called tracking by detection were analysed and evaluated. Particular attention was paid to the effectiveness of fast moving object tracking and the ability to implement the algorithm in a heterogeneous computing system. The selected solution was implemented in the Zynq SoC (System on Chip) device from Xilinx. Object position was used to control two servomotors, which constituted a pan-tilt mounting of the camera. Additionally, object position prediction was realised using a Kalman filter. The proposed system is able to process a 1280×720 @ 60 fps video stream in real time and track moving objects.

Keywords: object tracking, moving camera, Kalman filter, hardware-software co-design, Zynq SoC

1 Introduction

Object tracking with the use of a moving vision camera or thermal imaging camera is used, among others, in advanced video surveillance systems and many military applications. This type of systems consist of two main components: an object tracking algorithm and moving camera head control algorithm. In most cases, it is assumed that the aim is to keep the tracked object in centre of the frame.

Designing this type of vision system is a quite difficult task. Firstly, effective tracking requires the use of a method prone to scene lighting changes, rapid object movement, as well as size, shape and orientation changes. Moreover, in the case of a moving camera, image blur could also be a problem. Secondly, an important decision is the choice of the used computing platform. It should allow real-time implementation of the tracking algorithm and also be quite energy effective for many applications. Additionally, an easy integration with the moving camera head should be possible. According to the authors opinion, the above defined requirements are very well met by a Zynq SoC (System on Chip)

W. Mitkowski et al. (Eds), *Trends in Advanced Intelligent Control, Optimization and Automation*, Advances in Intelligent Systems and Computing 577, DOI 10.1007/978-3-319-60699-6_34

device, which combines in one housing reprogrammable logic (FPGA – Field Programmable Gate Array) and a processor system based on a dual-core ARM Cortex A9 unit.

In this paper the concept of using the Zynq SoC device for constructing a prototype of a moving smart-camera able to perform tracking of selected objects is investigated. The main contributions of this paper are:

- design of a fully operational prototype of a moving smart-camera (camera, Zynq computing platform, two servos, PID controller, Kalman filter),
- working system evaluated for two algorithms for 1280×720 @ 60 fps video stream.

The rest of this paper is organized as follows. In Section 2 previous work on object tracking implemented in FPGA devices is discussed. In the next Section 3 the proposed hardware-software system is presented. Its evaluation is shown in Section 4. The paper ends with a short summary and further research direction discussion.

2 Previous Work

The issue of implementing object tracking using a moving smart camera based on a FPGA or Zynq SoC device has been presented in several research papers. In the work [13] an original concept of a smart camera was proposed. It was based on a Spartan 6 FPGA device connected with 8 SRAM (Static Random Access Memory) banks. The camera was mounted in a housing, which allowed movement in two dimensions: rotation (360°) and tilt ($0 - 90^\circ$). Moreover, a tracking algorithm was proposed, which was based on edge movement analysis between consecutive frames. Finally, real-time processing for a 640×480 @ 20 fps video stream was obtained.

In the paper [4] a two camera system able to track circular shape objects was presented. Used algorithm was based on edge detection and circle fitting. The authors reported processing speed of over 1000 fps for a 816×600 pixels video stream. The system has been implemented in a Zynq device on the ZC 706 platform from Xilinx.

In a recently published article [7] a hardware-software tracking system using a pan-tilt camera was described. It used object detection based on colour and edge information. For servomotors control a PID controller was used. The whole system has been implemented in a Zynq device present on the ZedBoard development board. The authors report 58 fps processing speed, but did not provide information about the image resolution.

In the article [5] a vision system for a mobile robot implemented in FPGA was presented. The used algorithm was based on orange object detection. According to the description, the algorithm was realized on the PowerPC processor (with Linux operating system). No unambiguous information about the performance was provided (the used camera supported 640×480 @ 30 fps).

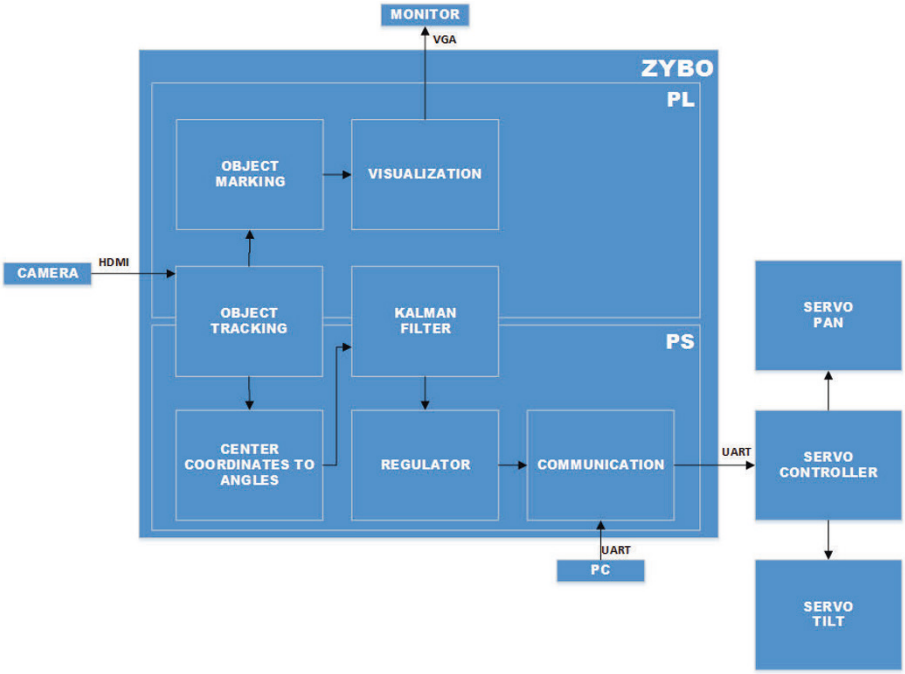


Fig. 1. Scheme of the proposed hardware-software system for object tracking using a moving camera.

In the work [3] a feature point (corners) based tracking algorithm was presented. Also the pyramidal Lucas-Kanade optical flow and Kalman filter were used. All computations were realized on a GPU (Graphics Processing Unit). The authors reported real-time processing for a 656×524 @ 30 fps video stream.

3 The Proposed Vision System

A general scheme of the proposed smart-camera prototype is presented in Figure 1. It consists of the following components.

- Xiaomi Yi Action YDXJ01XY camera – a 1280×720 @ 60 fps video stream was processed. It was transmitted to the computing platform via a HDMI (High Definition Multimedia Interface) connection. Additional, on top of the camera a laser pointer was installed. This allowed for better visualization of object tracking results.
- ZYBO development board manufactured by Digilent with Zynq 7010 (PS – processing system (dual ARM core), PL – programmable logic (FPGA)) from Xilinx. Moreover, the HDMI input, VGA output and serial port (USB to UART) were used.
- monitor – for tracking algorithm visualization,

– two digital servomotors PowerHD D-21HV connected into a PT (pan-tilt) configuration.
– servomotor control unit 'MiniMaestro' manufactured by Pololu (communication via UART).

In the following subsections a detailed description of the modules implemented in the Zynq device is provided.

3.1 Object Tracking

At the conceptual stage, the following object tracking algorithm were considered: simple tracking by detection (e.g. object segmentation using colour features),mean-shift [2], particle filter [10], KLT (Kanade, Lucas and Tomasi) [12].

During preliminary work the listed algorithms were evaluated on a sample video sequences containing a moving drone. At this stage the *Matlab* environment was used. The simple tracking by detection was carried out on the basis of the object colour. First RGB colour component ranges were defined and then a pixel was compared with them to check if it should be considered as a part of the object. In the next step, the centroid of the selected pixel was computed. The following equations were used:

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}} \tag{1}$$

$$m_{00} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{ij}, \quad m_{10} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} i \cdot x_{ij}, \quad m_{01} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} j \cdot x_{ij} \tag{2}$$

where: N and M are the image horizontal and vertical sizes, and x_{ij} is a binary pixel.

The point x_c, y_c was assumed as the tracked object's location. During real-life experiments a green object was used, so the following range of RGB components were applied:

$$R \in [0, 30], \qquad G \in [40, 255], \qquad B \in [0, 30] \tag{3}$$

The FPGA implementation of this method was quite straightforward. To obtain the object mask six comparators and some basic logical operations were required. The computation of m_{00}, m_{01} and m_{10} was done with simple accumulators. For the final division an iterative algorithm was used, as it was needed only once per single frame.

The following algorithm implementations were used: mean-shift [1], particle filter [11] and KLT (build-in in *Matlab*). The results analysis allowed to drew the following conclusions. The simple tracking by detection based on colour features worked correctly when the considered object was clearly distinguishable from the background. Its key advantage was the very simple FPGA implementation. An example tracking result is presented in Figure 2(a). It should be noted that in this version the method should only be used for testing and demonstration of the system. However, the concept could be applied, assuming the use of a more

(a) Object detection (b) Mean-shift

Fig. 2. Sample frames from the tracking algorithms evaluation on the drone sequence.

advanced object detection algorithm e.g. the well-known HOG (Histogram of Oriented Gradients) features and SVM (Support Vector Machine) classifier or even deep convolutional neural networks.

The mean-shift algorithm worked correctly only if the object moved relatively slowly. This was a direct consequence of the method's working principle. The new object location in the current frame is sought in a defined surrounding of the previous one. Moreover, due to the iterative procedure, the hardware implementation of this algorithm is quite complex [8]. An example tracking result is presented in Figure 2(b). Similar observations were also made for the particle filter algorithm. Furthermore, experiments described in the work [9] showed that an effective implementation of this algorithm on the ZYBO platform is impossible due to limited hardware resources.

The last of the evaluated algorithms – KLT – provided the best results. The tracking was correct, even in case of sudden movements. However, the hardware implementation of this algorithm (especially it's multi-scale version) is also quite complex [6] and is planned as part of the future research.

3.2 Centre Coordinates to Angles

The coordinates computed in programmable logic (using the tracking by detection or mean-shift method) had to be converted into angle control errors. This was done one the basis of input image resolution and camera's angle of view. It was assumed that the distance from the image centre is proportional to the angle error. This is generally not true, however for small error values it is a good approximation. Using the exact value would require the use of a look-up table (distance in pixels to angle assignment) or the determination of a mathematical function connecting those two values. Finally, the following equations were used:

$$\Delta_x\varphi = \Delta x \cdot \frac{Z_x}{S_x}, \qquad \Delta_y\varphi = \Delta y \cdot \frac{Z_y}{S_y} \tag{4}$$

where:

- Δx,Δy is the horizontal and vertical distance between the frame centre and current object position (can be positive or negative).
- Z_x, Z_y are the angle of views of the used camera.
- S_x, S_y are the horizontal and vertical frame resolutions.

3.3 Kalman Filter

In order to filter the position of the tracked object and try to predict the object movement when it is covered, the Kalman filter was applied to the output of the tracking algorithm (object location coordinates). For this purpose, the following state-space model of object movement and measurements was assumed:

$$x(k+1) = Ax(k) + \nu(k) \tag{5}$$

$$y(k) = Cx(k) + \omega(k) \tag{6}$$

$$A = \begin{bmatrix} 1 & 0 & h & 0 & \frac{h^2}{2} & 0 \\ 0 & 1 & 0 & h & 0 & \frac{h^2}{2} \\ 0 & 0 & 1 & 0 & h & 0 \\ 0 & 0 & 0 & 1 & 0 & h \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

where:

- $x(k)$ is the state vector containing: horizontal position, vertical position, horizontal velocity, vertical velocity, horizontal acceleration, vertical acceleration.
- h is the sampling period.
- $\nu(k) \sim \mathcal{N}(0, V)$ is the process noise.
- $\omega(k) \sim \mathcal{N}(0, W)$ is the observation noise.
- V and W are covariance matrices for respectively $\nu(k)$ and $\omega(k)$.

Equations (5) and (6) describe the uniformly accelerated motion in two directions (horizontal and vertical). The covariance matrices were chosen experimentally.

It was assumed that during object occlusion only the prediction phase of the Kalman filter will be conducted. Such approach was tested for the tracking by detection algorithm. The occlusion was detected using the number of object pixels visible in the current frame. If this value was below a preset threshold, the occlusion state was activated and the position of the tracked object was changed according to the used model.

3.4 Regulator

In the designed system the camera works as a sensor of the control error. In order to guarantee the correct positioning of the considered pan-tilt head, a regulator had to be implemented. It's output signals were used as servomotors setpoints.

Figure 3 shows a block diagram of designed control system where:

- $P(s)$ – Laplace transform of the position of the tracked object.

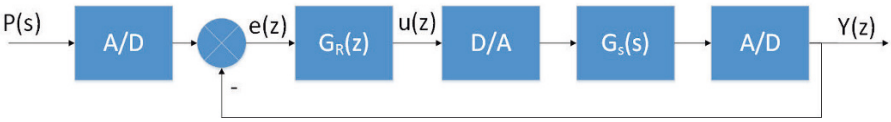


Fig. 3. Block diagram of designed control system.

- $e(z)$ – Z transform of the control error.
- $u(z)$ – Z transform of the regulator's output.
- $G_R(z)$ – transfer function of the used regulator.
- $G_S(s)$ – transfer function of the controlled object (servomechanism).
- D/A – digital to analogue converter (zero-order hold).
- A/D - analogue to digital converter.

In the current version of the system, it is impossible to directly measure the positions of servos. Thus, the used control algorithm had to be based on the output from the previous iteration. Therefore, it was decided to use the incremental PID (Proportional – Integral – Derivative) controller. It is described by the following equation:

$$u(k)-u(k-1) = P\cdot(e(k)-e(k-1))+I\cdot e(k)+D\cdot(e(k)-2e(k-1)+e(k-2)) \quad (9)$$

where: $u(k)$ – controller's output, $e(k)$ – control error, P, I, D – coefficients for the proportional, integral and derivative terms.

To be able to test the controller without the risk of damaging the pan-tilt head, it was decided to create a model of the used system. The analogue part of the system with D/A and D/A was converted to a digital state space model:

$$x(k+1) = A^{+}x(k) + B^{+}u(k) \quad (10)$$

$$y(k) = C^{+}x(k) \quad (11)$$

$$A^{+} = \begin{bmatrix} (1+\frac{h}{2T})e^{-\frac{h}{2T}} & he^{-\frac{h}{2T}} \\ -\frac{h}{4T^2}e^{-\frac{h}{2T}} & (1-\frac{h}{2T})e^{-\frac{h}{2T}} \end{bmatrix} \quad (12)$$

$$B^{+} = \begin{bmatrix} -2T(h+2T)e^{-\frac{h}{2T}} + 4T^2 \\ he^{-\frac{h}{2T}} \end{bmatrix} \quad (13)$$

$$C^{+} = \begin{bmatrix} \frac{1}{4T^2} & 0 \end{bmatrix} \quad (14)$$

where: h – sampling period, T – time constant of used servomechanisms.

Then, simulations were executed in *Matlab* and *Simulink* environments. Based on the model response, the following parameters were chosen:

$$P = 0.4, \quad I = 0.1, \quad D = 0.05 \quad (15)$$

Figure 4 shows the step response of the model (10), (11) with the described regulator for the following parameters of the model: $T = 0.05$, $h = 1/60$. The described regulator was implemented in the ARM processor core available in the Zynq device (c.f. Figure 1 – Regulator).

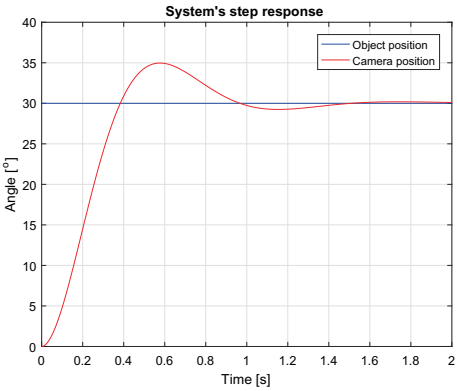


Fig. 4. Step response of system model with designed regulator.

3.5 Communication

To run the system it was necessary to establish communication between a PC, the Zynq device and the servomotor controller 'Maestro' (c.f. Figure 1). UART (Universal Asynchronous Receiver and Transmitter) communication protocol was used. Dedicated ARM core peripherals were utilized for this purpose. Moreover, a simple protocol was proposed. The command sent to the Zynq device from a PC was 5 bytes long. The first byte indicated the type of command and the following ones contained parameters. The system worked in two modes: manual and stand-alone. In the first one, the data received from the PC were forwarded to the 'Maestro' device. So, the head could be controlled directly from an UART terminal. In the second mode, for each received frame from the camera, a new setpoint was sent to the servos controller.

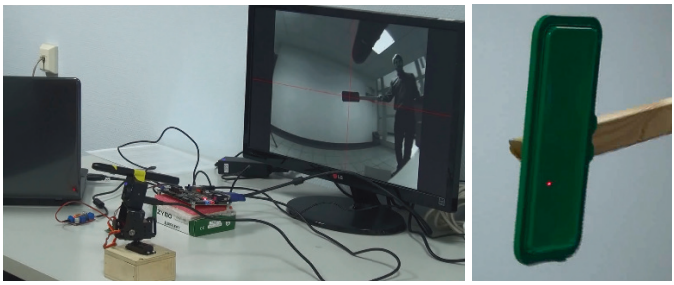
Support of the following commands was implemented:

- changing position of the servomechanisms,
- changing maximal speed of the servomechanisms,
- changing maximal torque of the servomechanisms,
- reading current setpoint from the controller,
- starting stand-alone operation mode (tracking),
- start of sending data to the PC (data acquisition mode).

Also communication between the reprogrammable logic (FPGA, PL) and the ARM processor (PS) was required. The *AXI-Lite* interface was used. When coordinates of the tracked object were available, an interrupt was triggered and the data read by the processor. Then the controller routine was executed and control for the servos provided.

3.6 Object Marking and Visualization

In the presented system both modules were used to analyse the correctness of the tracking algorithms. In the case of simple tracking by detection the centroid



(a) The prototype (b) Tracking result

Fig. 5. Real-life evaluation of the prototype.

was visualized by two intersecting lines – cf. Figure 2(a). Whereas, for the mean-shift algorithm a bounding box was used – cf. Figure 2(b). The markers were overlaid on the input video stream, which was then passed to the VGA display controller.

4 System Evaluation

In Figure 5(a) a photograph of the designed prototype is presented. In the foreground, the mobile camera (with two servos) with a laser pointer on top is visible. Behind (right) the ZYBO board and servo controller (left) are present. On the monitor, the tracking by detection result for a green object is shown. Moreover, in Figure 5(b) a sample tracking result is displayed (the laser beam points at the object).

The proposed system worked correctly. Test for two tracking algorithms (by detection and mean-shift) were conducted. A 1280×720 @ 60 fps video stream was analysed in real-time. Additionally, the performance of the Kalman filter is case of object occlusion was evaluated. Incorrect behaviour was observed only in case of slowly occurring occlusion. They resulted from gradual object size shrinking and thus change of the centroid location. This issue will be addressed in the nearest future.

5 Summary

In the paper the concept of a Zynq SoC based moving smart camera was presented. The proposed system, consisting of a camera, two servos, servo controller and computing platform has been positively verified for two tracking algorithms. The obtained results reveal very good properties of heterogeneous platforms for this type of applications. Computationally complex algorithms can be implemented in the reprogrammable part (FPGA) to obtain real-time performance and energy efficiency. One the other hand, relatively simple computations like

the controller or Kalman filter, as well as communication with other components on the system can be implemented in the processor system.

As part of future work it is planned to: implement the KLT and other more advanced tracking algorithms, add servo position and speed sensors to improve the positioning, realize wireless communication with the unit, as well as perform an in-depth analysis of possible regulators.

Acknowledgements The work presented in this paper was supported by the AGH University of Science and Technology project no. 15.11.120.879.

References

1. S. Bernhardt. Mean-Shift Video Tracking. https://www.mathworks.com/matlabcentral/fileexchange/35520-mean-shift-video-tracking. Last access: 20.12.2016.
2. D. Comaniciu and V. Ramesh and P. Meer Real-time tracking of non-rigid objects using mean shift Proceedings IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 142–149 (2000)
3. D.D. Doyle, A.L. Jennings, J.T. Black Optical flow background estimation for real-time pan/tilt camera object tracking Measurement 48, pp. 195207 (2014)
4. Z. Du, H. Lu, H. Yuan, W. Zhang, C. Chen, K. Xie A FPGA Based High-Speed Binocular Active Vision System for Tracking Circle-Shaped Target Advances in Multimedia Information Processing – PCM 2015: 16th Pacific-Rim Conference on Multimedia, pp. 365–374 (2015)
5. H. Hagiwara, K. Asami, M. Komori FPGA Implementation of Image Processing for Real-Time Robot Vision System Convergence and Hybrid Information Technology: 5th International Conference pp. 134–141 (2011)
6. W. Jang, S. Oh and G. Kim A hardware implementation of pyramidal KLT feature tracker for driving assistance systems 12th International IEEE Conference on Intelligent Transportation Systems, pp. 1–6 (2009)
7. C. Lyu et al., High-speed target tracking base on FPGA IEEE International Conference on Real-time Computing and Robotics (RCAR), pp. 272-276 (2016)
8. K. Mazur, T. Kryjak An embedded vision-based tracking system for autonomous robot navigation Measurement Automation Monitoring vol. 5 pp. 172–174 (2016)
9. T. Meresiski Implementation of an object tracking algorithm in heterogeneous Zynq device Master Thesis, AGH University of Science and Technology, Krakow, 2016
10. Amir Mukhtar and Likun Xia Target Tracking Using Color Based Particle Filter 5th International Conference on Intelligent and Advanced Systems (ICIAS) (2014)
11. S. Paris. Particle Filter Color Tracker. https://www.mathworks.com/matlabcentral/fileexchange/17960-particle-filter-color-tracker. Last access: 20.12.2016.
12. C. Tomasi, T. Kanade Detection and Tracking of Point Features Computer Science Department, Carnegie Mellon University (1991)
13. A. Zawadzki and M. Gorgon Automatically controlled pantilt smart camera with {FPGA} based image analysis system dedicated to real-time tracking of a moving object Journal of Systems Architecture, vol. 61, number 10, pp. 681–692 (2015)