HARDWARE-ACCELERATED OBJECT TRACKING

Tobias Becker, Qiang Liu and Wayne Luk

Department of Computing
Imperial College London

Georg Nebehay and Roman Pflugfelder

Department of Safety and Security
AIT Austrian Institute of Technology

This work investigates hardware acceleration of object tracking by parallelising an algorithm for object classification involving decision trees. Object tracking is the process of recognizing and locating a particular moving object in the spatial as well as in the temporal domain of a video stream. One key application of object tracking is video surveillance, to provide an operator in a control room with novel tools for assessing complex events among hundreds of live videos. Object tracking can be achieved between two consecutive video frames through template-based or feature-based correlation of images. Although this approach is computationally efficient, it can be unreliable or unsuccessful, because the appearance of the object may drastically change or the object may become occluded.

As an alternative, one can apply object classifiers in subwindows that vary in scale, size and position [1]. A classifier describes an object based on a limited feature set. One example of such a feature set are 2-bit Binary Patterns that capture brightness variation in certain rectangular regions of an object's image. This feature set is mapped on an ensemble of decision trees known as a random forest [2] that can determine the probability of the object being present in a search window. A single 2-bit Binary Pattern gives a very weak indication that the sought object is present in the current search window, while the mapping of several features on one decision tree, and the combination of several trees, can identify objects with high confidence.

A classifier can be trained by supervised learning techniques: during the training phase the classifier is adapted by manually selecting the object of interest in one video frame; however, such human interaction is undesirable. Another approach starts from a given instance of the object's appearance [1]. Appearance changes are addressed through P-N learning [3], an online learning technique that combines a classifier learner with a Lucas-Kanade frame-by-frame tracker [4] and a random-forest-based detector. P-N learning identifies positive and negative instances of object appearances in the video stream and uses these instances to update the information captured by the decision trees.

Classifier-based object tracking is robust to drastic appearance changes and to total occlusions; however, it is computationally demanding and it is hard and sometimes impossible to achieve the required frame rates. We aim to alleviate this problem by accelerating the exhaustive search for potential object detections with reconfigurable hardware. Profiling of a software algorithm reveals that 90% of the total computation time is spent on the detector. We are therefore interested in developing a new custom hardware architecture for this part of the algorithm in order to achieve high frame-rates while lowering the power consumption at the same time. We also aim to perform a fully exhaustive search for object instances with pixel-by-pixel increments over each video frame, which will increase the robustness of the tracker.

The following describes the algorithm of our classifier-based detector. The incoming video frames have 8-bit greyscale representation and standard VGA resolution. We store incoming frames as integral images where each pixel value is the cumulative sum of pixel values in the rectangular image region to the left and above of the pixel. This image representation makes the summation of the pixel values in a rectangular region independent of the region's size: the operations are simple additions and subtractions of the pixel values at the region's corners. Binary Patterns are used for capturing horizontal and vertical brightness variations in an image region resulting in a 2-bit code. The 2-bit Binary Patterns can be computed based on simple additions and subtractions of a region's corner pixels as described above. A sub-window of a video frame is classified with a random forest consisting of ten decision trees, and each decision tree is traversed based on ten 2-bit patterns. The leaf-nodes of each tree specify the probability of an object match. Figure 1 illustrates this for three decision trees with three features each. The probabilities from all trees are averaged to yield a final probability; an unambiguous object identification is determined after the exhaustive search by hysteresis thresholding. This search is exhaustively repeated over the entire image with various search window scaling factors.

We now present a novel hardware architecture that implements the integral image conversion and classifier on an FPGA in order to accelerate the most compute-intensive part of the object tracking algorithm. The hardware architecture

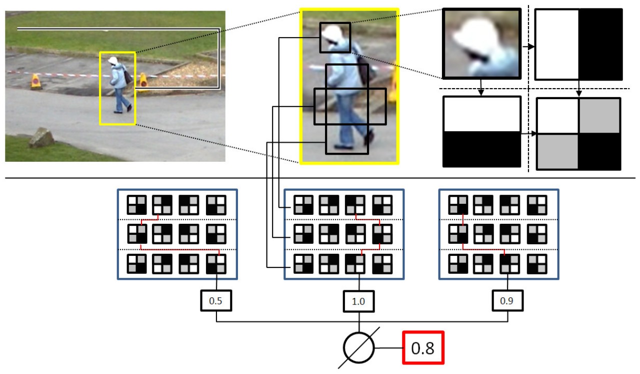

Fig. 1. An example classification of a sub-window. Three decision trees are used, each basing its decision on three feature locations. Probabilities at leaf nodes have been trained in advance. In this example, the sub-window represents the trained model with a probability of 80%.

is illustrated in figure 2. Here, we briefly introduce the design of classifier, since it is the bottleneck in the original software implementation.

To accelerate the classification process, we exploit the parallelism inherent in the random forest search: the ten decision trees with the same sub-window input can be traversed independently. We customise ten processing units (PUs) on the FPGA, each working on one decision tree and processing ten features associated with one tree in a pipeline. Each PU has four adders, eight subtractors and two comparators to calculate the 2-bit Binary Patterns for each feature in four clock cycles.

As shown in figure 2, each PU has a dual-port local on-chip memory storing all data elements of a sub-window. Therefore, the sub-window size determines the on-chip memory utilization. In the current implementation, the sub-window size is limited to 1024 elements that can be stored in one 36 Kbit block RAM (BRAM); if larger sub-window sizes are desired then more on-chip memory is needed. This distributed memory system allows all PUs to work independently. After the image is integrated in the integral module, the corresponding elements of a sub-window are transferred into these local BRAMs. Since eight data elements are needed to calculate the 2-bit Binary Pattern for one feature, four accesses to the dual-port local memory are required, completing in four clock cycles. This is well matched with the four clock cycle arithmetic operations in the PU. Thus, the memory access time is overlapped with computation time, reducing latencies. The hardware classifier outputs the tree traverse path, i.e. ten 2-bit patterns. These patterns are transferred back to the CPU to resolve the corresponding P-N values. The learning part of the algorithm which updates P-N values based on changing object appearances is also implemented on the CPU.

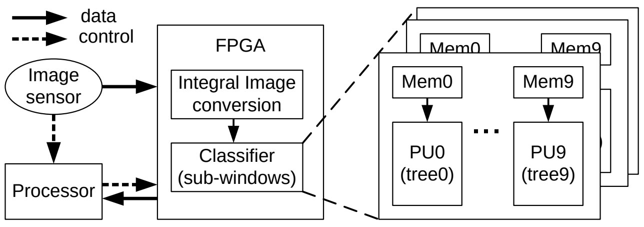

Fig. 2. Hardware architecture for object detection.

The resource utilization of one PU in the current implementation on a Xilinx Virtex-5 FPGA is one 36 Kbit BRAM and 2600 slices. Initial performance results show that with a clock rate of 125 MHz, the classification of one sub-window can be performed in 0.62 µs. This corresponds to a frame rate of 5 fps (frames per second) when performing a fully exhaustive search over VGA-resolution images. For comparison, our software implementation of the classifier on an Intel Xeon 2.4 GHz PC achieves a frame-rate of less than 1 fps when performing the same exhaustive search, which is over 5 times slower than the FPGA version. The performance can be further increased by processing several sub-windows simultaneously. This can be achieved by implementing multiple classifiers in parallel as illustrated in figure 2.

Two target architectures for our hardware architecture are a high performance compute node at Imperial College with an AMD Phenom quad-core CPU and Virtex-5 LX330T FPGA, and a Sony XCI-V100C Smart Camera with a VIA Eden processor and a Virtex-5 SX50T FPGA. On the Virtex-5 LX330T, 20 classifiers can be implemented in parallel and on the Virtex-5 SX50T, 3 classifiers can be implemented. This corresponds respectively to 100 times and 15 times speed-up over the original software implementation. The speed-up can be used for faster frame rates or processing higher resolutions video streams. Current power estimates for the Sony Smart Camera indicate that the computation can be performed with less than 20 W. This is significantly more efficient than the original PC implementation which consumes 112 W during processing.

1. REFERENCES

[1] J. M. Z. Kalal and K. Mikolajczyk, “Online learning of robust object detectors during unstable tracking,” in *On-line Learning for Computer Vision Workshop*. IEEE, 2009, pp. 1417–1424.

[2] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, October 2001.

[3] J. M. Z. Kalal and K. Mikolajczyk, “P-N learning: Bootstrapping binary classifiers by structural constraints,” in *Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 49 – 56.

[4] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1981, pp. 674–679.