Real-Time Tracking Algorithm with Locking on a given Object for VLSI CNN-UM implementations

Samuel Xavier-de-Souza, Johan A.K. Suykens, Joos Vandewalle

K.U. Leuven, ESAT-SCD-SISTA,
Kasteelpark Arenberg 10,
B-3001 Leuven (Heverlee) Belgium,
e-mail: samuel.xavierdesouza@esat.kuleuven.ac.be

ABSTRACT: An algorithm to lock on and track objects is presented in this paper. The algorithm is designed to take advantages of the highly parallel architecture and speed of the VLSI implementations of the Cellular Neural Network Universal Machine (CNN-UM). Given a dynamical image containing different objects and initial information about which object to track, this algorithm is able to lock on the object and calculate its instantaneous coordinates within the image. CNN chip-specific robust templates were generated in order to implement the method on a real CNN-UM chip. The experiments and the results of analysis show that a CNN-UM chip running this algorithm can be used in real-time tracking applications being able to achieve good performance.

1. Introduction

Object tracking allows the analysis of objects and persons' movements in video images, making it possible to calculate their position in time, the direction and speed of their movements, and whether or not they will meet or collide. Object tracking with locking on a given object permits one to collect this information for the locked object regardless of the presence of ambiguous similar objects in the scene. The applications are vast and range from the field of security and traffic analysis to sport events examination.

Real-time constraints for object tracking can be met by using a Cellular Neural Network Universal Machine (CNN-UM) VLSI implementation [1, 2, 3]. Such devices are able to deliver very high processing speeds owing to its highly parallel and *analogic* array processing. Moreover, some devices have the ability to acquire images directly from the cells by means of optical sensors, which drastically shorten the input acquisition latency.

This paper addresses the design of an algorithm to solve real-time object tracking via CNN-UM silicon devices with possibility of locking on a given object. The algorithm uses chip-specific robust CNN templates [4] in attempt to overcome misbehavior of these devices.

After a detailed description of the algorithm in Section 2, a brief introduction to chip-specific CNN robust templates is made in Section 3, followed by an analysis of performance and speed of the proposed algorithm in Section 4.

2. Tracking with Locking (a Visual Algorithm)

The objective of this algorithm is to provide the coordinates of an object given a sequence of image frames containing one or more objects and initial information about which object to lock on. The proposed algorithm contains two main steps that need to be executed for each new frame. The first step is to isolate the object which has to be tracked. This step denotes the locking feature of the algorithm. The second step is related to the calculation of the coordinates of the chosen object within the image. It provides the position of the center of the object with relation to the upper left corner of the image.

2.1 Locking on the chosen object

In order to provide the tracking with locking, it is necessary to isolate the chosen object from other objects in the current frame. This step has as input the current frame and the chosen object isolated in the previous frame. It gives as output the isolated object in the current frame. An important remark for this step is that the shifting of the object between two consecutive frames can not exceed its transversal length in the given shifting direction. This remark yields a very hard constraint for the algorithm since its performance scales proportionally with the transversal lengths of the object. Such a constraint may require extremely high processing speed and an abridged image acquisition latency. These features are natural characteristics of some VLSI CNN implementations. Further discussions about this remark will be given in Section 4..

Consider a image frame F_i as being the ith frame in a sequence $i = 1, 2, 3, ...$, where each element of F_i is a binary element (pixel) with value 0 (*false/white*) or 1 (*true/black*). The sets of adjacent *true* elements represent objects in the image. The image frame O_i will be the image of the chosen object alone in the same position as in the image F_i. Given a new frame F_i, and the previous image of the chosen object O_{i-1}, the calculation of O_i is given by the following equations:

$$O_i = F_i \text{ AND } M_i,$$

where M_i is the marked absolute difference between F_i and O_{i-1} calculated by

$$M_i = \text{recall}(A_i, O_{i-1}),$$

where $recall(R, S)$ is a operation that reconstructs only the objects of R that are marked with elements of S, and A_i is the union (OR logic operation) of F_i and O_{i-1} calculated by

$$A_i = F_i \text{ OR } O_{i-1}.$$

During the initialization of the algorithm, when O_{i-1} does not exist yet, O_{i-1} needs to be initialized with an image containing a mark, denoted by one or more elements set to *true*, to be placed in the exact correspondence with any element of the chosen object in F_i, and with all other elements set to *false*. An illustrative example of this step of the algorithm is shown in Fig. 1.

2.2 Obtaining the coordinates of the object

The objective of this step is to find the coordinates of the center of the object with relation to the upper left corner of the image. The input is the image of the chosen object alone, O_i,

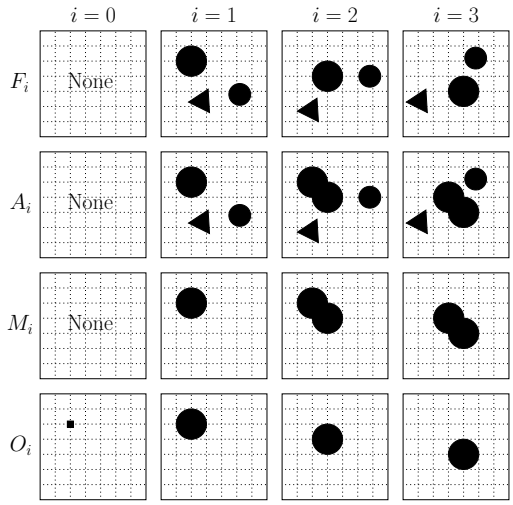

Figure 1: Locking on the object

given by the previous step. The concept of center here coincides with the mass center of the object growing to a rectangle[1]. The projections of the object to the bottom and left sides of the image, representing the length and height of the rectangle, are used to obtain the horizontal and vertical coordinates respectively. The center of each projection is found by pyramiding[2] the projection and then erasing everything but the top. The top is then shadowed horizontally or vertically, according to the dimension of the coordinate. The result is used as a mask for a massive diffusion CNN template operation [5] having as initial state a real-valued image with degrading in the orthogonal direction to the given projection, namely -90 degrees related to the projection. At the end of the diffusion, every element of the resulting image (now real-valued) is expected to be proportional to the given coordinate. Fig. 2 shows the evolution of this step for the frame O_3 of the example of Fig. 1.

Owing to difficulties to implement the *diffusion* template in VLSI CNN-UM chips, an alternative algorithm is also proposed to replace this template operation. After the top of the pyramided projection is found, it will be now shadowed in the same direction of the projection and the result shadowed in the orthogonal direction, namely -90 degrees related to the projection. A search in the resulting image is then performed to find the position of the last element with value *true* of the resulting image, starting from the less significant position value in the row or column containing the projection. The complexity of this search is $O(\log N)$, where N is the size in pixels of the related image dimension. Fig. 3 shows the evolution of these last operations for the example of Fig. 2.

In order to apply this two-steps approach to lock on and track a object, one may need to pre-process the input images (e.g. with adaptive thresholding, edge detection, hole filling, etc.). Such pre-processing can eventually be done with the CNN-UM architecture too, but is not described here. Another constraint for the algorithm is that the object needs to remain within the image frame in order to be tracked. However, by using the resulting coordinates information one can make sure the object remains within the view area of the camera [6].

[1]This approach was used instead of the well known recursive rotative peeling templates due to the smaller number of operations necessary to find the center.

[2]Pyramiding up: $A = [2.10\ -0.35\ 2.10\ 0.13\ 2.95\ 0.13\ 0.66\ 0.67\ 0.66]; B = 0; z = 4.7.$

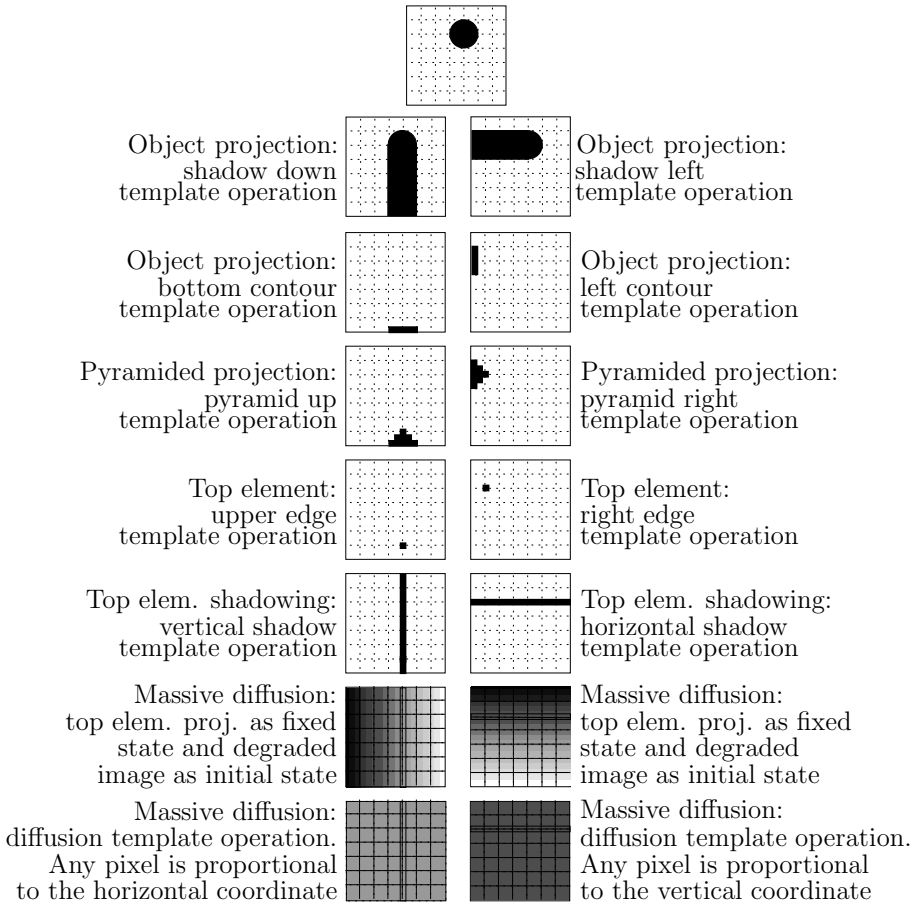

Figure 2: Obtaining the coordinates of the object

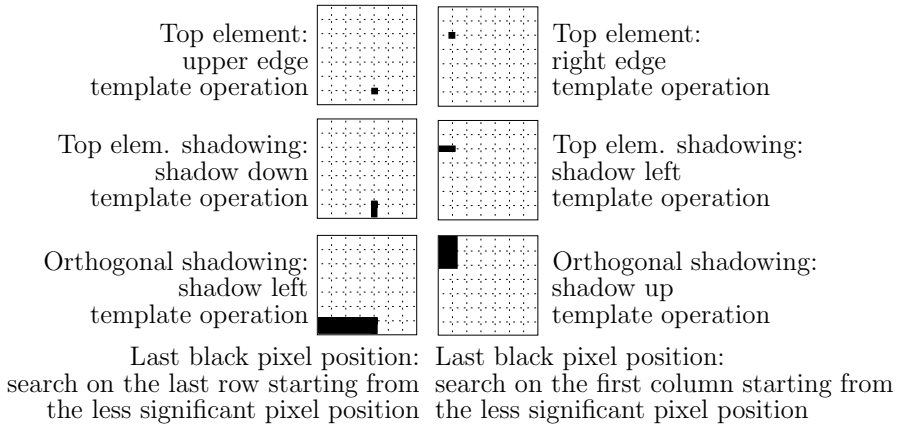

Figure 3: Obtaining the coordinates of the object without the diffusion template operation

3. Chip-Specific Robust Templates

Despite of the existence of a large range of designed CNN template operations, a reasonable amount of them does not work correctly when executed on VLSI CNN-UM implementations [7]. The reasons for that are many but mainly due to manufacturing failure to reproduce

in silicon the exact model of the cellular neural network. There exist also in the literature many robust versions of these templates that, independently of any implementation, increase the chances of correct functioning of CNN devices. However, even the most chip-independent robust template can not always provide correct behavior on VLSI CNNs. An attempt to solve these misbehavior may involve chip-specific template optimization [8] or even searching for chip-specific robust templates [4]. Both approaches try to avoid manufacturing parameter misplacements being the latter intend to avoid post-manufacturing causes of misbehavior as well.

In the light of the last observations, all templates used in the proposed algorithm for object tracking with locking were optimized as described in [4] to be used in a ACE4K device [2] in order to find their chip-specific robust versions.

4. Speed and Performance analysis

In order to estimate performance and speed, an experimental setup was established with a ACE4K CNN-UM chip (64x64 cells) installed in a DSP board hosted in a digital computer. The images were acquired on-the-fly by a video camera connected to the computer. The algorithm described here has three main parts: image acquisition, locking on the object, and calculation of the coordinates. Being the last part subdivided in horizontal and vertical coordinate calculations. Due to limitations of the camera used in the setup, the maximum frame rate reached was approximately 35 frames per second. Assuming that a faster camera is used or that the images are acquired by a optical input in the chip, which input latency is around 50 microseconds, the algorithm proposed here could reach up to 370 frames per second. The table below shows the average time delays for each part of the algorithm.

Algorithm step	Average delay
Reading from camera	25.20ms
Locking on object	0.50ms
Vertical coordinate calculation	1.00ms
Search for last black pixel in leftest column	0.17ms
Horizontal coordinate calculation	0.82ms
Search for last black pixel in bottom row	0.17ms
Total without reading from camera	2.66ms
Total	27.86ms

As mentioned before, fast frame rate is fundamental to the performance of the algorithm. This has specially effect in the locking step. The maximum object speed which the algorithm can follow is proportional to the length of the object's transversal cut in the same direction of the movement, l, and to the delivered frame rate, r. This speed is given by the following equation

$$S_{max} = (l-1)r \text{ pixel/second},$$

Thus, the worst case scenario is when the object is moving in the same direction of its smallest transversal cut.

5. Conclusion

This paper provides a detailed algorithm to lock on and track a object in a video image. The locking allows the object to be tracked even with the presence of other similar objects in the scene. The method may be used in a wide range of applications requiring real-time constraints as hard as 370 frames per second with the use of direct optical image acquisition. The generation of chip-specific robust templates avoided much of the misbehavior of the applied templates on the CNN-UM chip and made possible the implementation of the proposed algorithm in such device.

Acknowledgment Dr. Joos Vandewalle is a full professor at the Katholieke Universiteit Leuven, Belgium. Research supported by: • Research Council KUL: GOA-Mefisto 666, BOF OT/03/12, several PhD/postdoc & fellow grants; • Flemish Government: ○ FWO: PhD/postdoc grants, G.0407.02 (support vector machines), G.0080.01 (Collective Behavior and Optimization), research communities (ICCoS, ANMMM); ○ IWT: PhD Grants, ○ Tournesol 2004 - Project T2004.13. • Belgian Federal Science Policy Office IUAP P5/22 ('Dynamical Systems and Control: Computation, Identification and Modeling'). Johan A.K. Suykens is associate professor with K.U.Leuven.

References

[1] T. Roska and L.O. Chua. The CNN Universal Machine: an Analogic Array Computer. *IEEE Trans. Circuits and Systems*, 40(II):163–173, March 1993.

[2] G. Liñán, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. ACE4k: An analog I/O 64x64 visual microprocessor chip with 7-bit analog accuracy. *International Journal of Circuit Theory and Applications*, 30(2-3):89–116, 2002.

[3] G. Liñán, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. Architectural and basic circuit considerations for a flexible 128×128 mixed-signal SIMD vision chip. *Analog Integrated Circuits and Signal Processing*, 33:179–190, Nov. 2002.

[4] S. Xavier de Souza, M.E. Yalcin, J.A.K Suykens, and J. Vandewalle. Toward CNN Chip-specific robustness. *Accepted for publication in IEEE Trans. Circuits and Systems-I*. ESAT-SISTA Internal Report 03-104, Katholieke Universiteit Leuven, Belgium (see pub engine @ http://www.esat.kuleuven.ac.be/šistawww/cgi-bin/pub.pl), 2003.

[5] V. Gál and T. Roska. Collision Prediction via the CNN Universal Machine. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'2000)*, pages 105–110, Catania, Italy, May 2000.

[6] A. Gacsádi and P. Szolgay. An anologic CNN algorithm for following continious moving objects. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'2000)*, pages 99–104, Catania, Italy, May 2000.

[7] S. Xavier-de-Souza, M.E. Yalcin, J.A.K. Suykens, and J. Vandewalle. Automatic Chip-Specific CNN Template Optimization using Adaptive Simulated Annealing. In *Proceedings of European Conference on Circuit Theory and Design (ECCTD'03)*, Krakow, Poland, Sep 2003.

[8] P. Földesy, L. Kék, Á. Zarándy, and G. Bártfai T. Roska. Fault-Tolerant Design of Analogic CNN Templates and Algorithms—Part I: The Binary Output Case. *IEEE Trans. Circuits and Systems*, 46(2):312–322, February 1999.