Fast and Robust Face Tracking for CNN chips: application to wheelchair driving

Samuel Xavier-de-Souza, Michiel Van Dyck, Johan A.K. Suykens, and Joos Vandewalle
K.U.Leuven, ESAT/SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
Email: [samuel.xavierdesouza,
johan.suykens, joos.vandewalle]@esat.kuleuven.be

***Abstract*— An algorithm for fast and robust face tracking with the CNN Universal Machine is proposed in this paper. It is applied to a driving mechanism for a wheelchair with an on-chip implementation. A novel object tracking CNN visual algorithm is introduced and employed in the tracking of multiple face features. The speed and robustness of this method are achieved due to the parallelism in the visual algorithm, and the tracking of multiple face features. The tracking algorithm is designed to achieve a high frame rate and exploit the specific properties of face features. The face tracking method proposed here was implemented on a Bi-I stand-alone cellular vision system and applied to a wheelchair driving mechanism. The template operations were trained and/or fine-tuned in order to generate chip-specific robust templates. In order to improve performance in environments with varying illumination, an adaptive image capture procedure was also introduced. Our simulations with a 3D model wheelchair showed that the final algorithm is capable of performing tracking with a frame rate of 92 frames/sec, which is supposedly enough for real-time driving in most of the real life situations.**

I. Introduction

In this paper we propose a algorithm for tracking features of the human face using a chip implementation of the Cellular Neural Network Universal Machine (CNN-UM). This algorithm is then applied to a hands-free mechanism to drive a wheelchair.

Face tracking is a problem that have been actively studied in the recent years [1]–[3]. As part of an larger and more ambitious goal, face tracking has been intended to help on the realization of perceptual user interfaces. Other parts of such system include face detection, face recognition, gaze point estimation, and finally the translation of information into computer actions like mousing. While the aim of such system can be reasonably broad, it is mainly intended to aid people with a handicap to be able to use the computer. We believe that face tracking can also help these people with their mobility.

Cellular Neural Networks (CNN) [4], [5] technology seems a natural choice to implement on-board face tracking due to its high parallelism and reduced size. By placing a CNN visual system on board of a wheelchair and using an object tracking algorithm to track face features, the control of the driving of the wheelchair can be easily translated into movements of the face.

We propose a CNN visual algorithm that is fast and robust to track face features in order to control the driving of a wheelchair. For test purposes, we created a three-dimensional model of a wheelchair which reacts upon face movements of the user. He/She can also interrupt and restart the driving hands-free at anytime.

This paper is organized as follows. in Section II we describe the relevant aspects of the proposed algorithm and how we can improve its robustness. In Section III we define the problems involved in wheelchair driving. In Section IV we describe our implementation of the driving mechanism and its practical aspects. Finally, we conclude our thoughts in Section V.

II. Face tracking

We describe here the relevant aspects for our face tracking algorithm. We do not present any solution in the direction of face detection nor face recognition. Our algorithm relies on extra information at the initialization phase in order to locate the face in the initial image and start the tracking.

A. Tracking alternatives

The orientation of the human face can be calculated in real-time by tracking face features as individual objects. These objects must be clearly identified in the face. The most common choices are the eyes, nose breaches, mouth, hair and eyebrows. All these features stand out reasonably well in a frontal face image. The eyes have the advantage of being rather large. They are long visible as the head rotates and are rather similar among different people. Their disadvantages are the fact that they blink, that some people have to wear eyeglasses and that eyes and eyebrows lay sometimes very close to each other. Besides having mostly the same advantages as the eyes, the eyebrows do not blink. Their disadvantages are that the color differs from people to people and that sometimes they are very near the hair. An alternative are the nose breaches. They are very similar among different people and with a camera placement under the head they become clearly visible. Additionally, they lay rather isolated of other possible interfering tracking objects. Their disadvantages though are that they become invisible by rotating the head completely down, and that they are rather small. Moreover, they become invisible by people with a moustache. The advantage of the mouth is that when open it gives a large object that is good for tracking. However, it changes of form and a closed mouth is not a good tracking object. Further on, the mouth does not work as a good tracking object with people with beard and/or

moustache. At last, the hair is a very large object an so it is easy to find. The problem is that it borders to the background, which may interfere with the tracking algorithm. Moreover, it does not change of place in function of the rotation of the head.

The right choice for a tracking object depends thus mostly on the kind of face that needs to be tracked. By considering a face feature as a general object independently of its type, it is possible to develop a general method that can track these different features seamlessly. The algorithm we present here was developed with this objective.

B. Tracking window: a visual algorithm

In order to efficiently track one of the face features described above, we have developed a visual algorithm whose main principle is to make sure the object being tracked is always in the center of a window which floats along a larger streaming input picture. This approach is especially suitable for use with VLSI CNN-UM implementations because such a window can be made equivalent to the chip size.

This algorithm consists of two basic steps. First, at each current frame, the object to be tracked must be isolated from the rest of the image. After an image containing only this object is retrieved, it is calculated if the window has to be shifted, in which direction, and how much is this shifting.

Isolating the object to be tracked in the current frame can be easily performed as explained in [6]. The part of the object that overlaps in current frame with the isolated previous frame can be used to retrieve the object alone in the current frame. This operation can be performed using the recall template by using the current frame as input image and the previous frame with the isolated object as initial state image. At initialization, when a previous frame does not yet exist, it is necessary to locate the object and define an image with a marker at its location that serve as the previous frame for the first frame.

The second and novel step of this algorithm is to move the sliding window accordingly with the position of the object in the current frame. The objective is to make sure that the extracted object always stays within the tracking window. This is achieved by a procedure that uses the result of the intersection of 4 pixels with the shadow of the object. The same procedure is executed twice, once for the horizontal direction and once for the vertical one, at each current frame. What follows is a description of the procedure for horizontal component. It is analogue to the vertical one.

The first step is to apply the shadow operation on the object. The shadow of the object is checked on the bottom line in four pixel locations. Two of these pixels are located on each end of the line and the two other in the center of the line at equal distance from each other and from the other pixels. The intersections of the shadow with these four pixels are used to define the shifting. The objective here is to make sure that the object's shadow overlaps always with the two center pixels and avoid the overlapping with the other two. We devise two simple rules to define the necessary shifting.

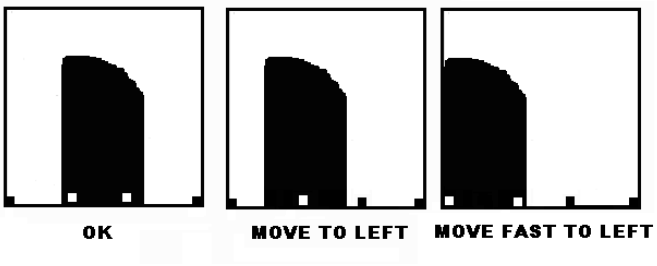


Fig. 1. The tracking window is shifted if necessary with two different values according to the intersection of four pixels with the shadow of the object. Here the two situations are presented for the horizontal shifting.

- If the shadow of the object intersects with only one of the center pixels, the window is shifted r pixels in the direction of this pixel.
- If the shadow touches one of the edge pixels, the window is shifted s pixels to the pixel direction.

See Fig. 1 for an illustration of this rules. The values for r and s are determined heuristically. r is obviously directly related to the actual speed of the object, which can only be instantaneously estimated. s is dependent on this speed but also on the size of the object and of the sliding window. Although the ideal values of r and s can be adaptively estimated accordingly to the speed of the object, constant values delivered sufficient performance for our implementation. See Section IV for details on these values. Note that despite the sequential reading of maximum 7 pixels, the other only two operations of this method can be fully implemented in parallel with CNN templates.

This simple procedure produces a very efficient way to track the given object by keeping the object in the center of the tracking window. Nevertheless, there are two fundamental issues to the algorithm as it was presented so far. First, the use of the recall operation to isolate the object in the current frame only works if the object always overlaps itself in consecutive frames. Fortunately, because this algorithm is designed for a CNN-UM and make use its high parallelism and fast speed, we can achieve sufficient frame rate to ensure the overlapping for most applications.

The second issue is related to the size of the object. In the method described above, the object must always fit in the tracking window and should not be smaller than the distance between the two center pixels in order to guarantee the correct shifting. In real-life applications, the object to be tracked varies constantly in size and form due to depth movements, lighting, rotation, etc. The following Section describes a methodology to overcome this problem.

C. Adaptive object resizing

Tracking an object that is constantly changing in size and form presents a problem to the method described above when it does not fit in the tracking window or when it becomes smaller than the distance between the two center pixels. In both cases, the direction of the shifting becomes impossible to be calculated. In order to solve this problem, it is necessary

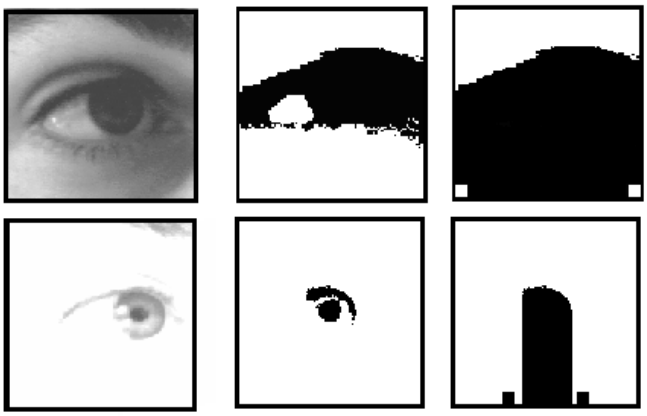


Fig. 2. The resizing algorithm changes the parameters the control the object's size according to the intersection of its shadow with four pixels. On the top row of the figure, there is too little light and a longer exposure time is needed; On the bottom row, there are too much light and a shorter exposure time is needed.

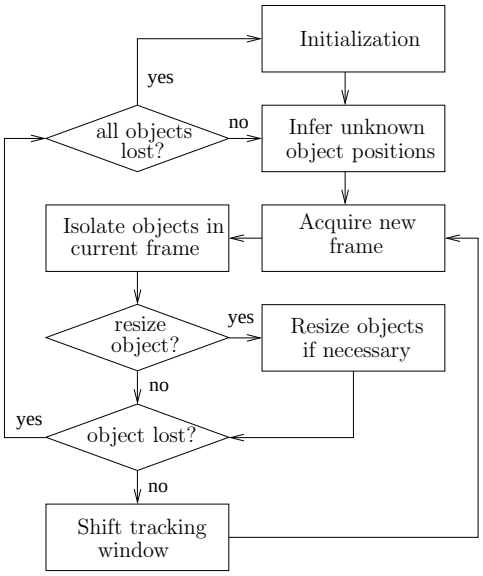


Fig. 3. An overview of the different stages of the multiple object tracking system.

first to monitor the size of the object and then apply some action to the image so that the object can be resized to an acceptable size. Such action can be e.g. either zooming, a change in lighting, adjusting the threshold level, adjusting the objective shutter speed, or a combination of them. In this paper we have chosen to adjust the shutter speed and the threshold value of the gray-scale to binary conversion.

These values are adapted using the readings of the pixels from the window shifting principle, i.e. no extra reading operation is required. If the object touches both edges of the tracking window, the two outer pixels are overlapped by the objects shadow, which means that the object has increased in size, e.g. because of too little illumination, and it no longer fits in the tracking window. In this case, the algorithm adjusts first the shutter speed. If the speed cannot be decreased anymore, the threshold value is then adjusted. On the other hand, when the object is being tracked and it decreases too much in the size such that it becomes smaller than the distance between the two center pixels, the object's shadow does not covers any of these pixels and thus the threshold and shutter speed values need to be adjusted in such a way that the object appears larger in the image. See Fig. 2 for a illustration of the two cases.

With such a procedure to complement the method described in the previous Section, we have a efficient algorithm to track objects which are subject to constant size variation in its image projection. Although such a procedure already increase significantly the robustness of the method, in the next Section we present a strategy which can deal with other occasional problems, e.g. partial or complete occlusion of the object.

D. Robust face tracking

In order to increase the robustness of face tracking method presented above, we propose the tracking of multiple face features instead of only one. This boosts the reliability of the whole process because if one of these objects gets lost from the tracking, it is often possible to retrieve it by the relatively fixed geometry of the face. For that, the positions of the objects that are still being tracked are used to estimate a marker for the lost object. It is clear that the more objects being tracked there are, the larger the robustness of the tracking system will be. Nevertheless, this is only true up to a certain level. Although the tracking of one single object can be done in parallel in the CNN-UM, the tracking of the different face features is sequential. The addition of more objects to the tracking results in a reduction of the maximum frame rate of the whole system. Consequently, it results in a reduction in the maximum speed in which the object being tracked can still be followed. Therefore, the number of objects to track simultaneously must be wisely traded off with the frame rate of the system so that the overall robustness prevails.

The combination of the window shifting and adaptive resizing algorithms with multiple tracking results in a very simple and efficient algorithm for face tracking. An overview flowchart of the tracking process which involves these three features is presented in Fig. 3.

III. Driving of a wheelchair

The challenge here is to implement a hands-free driving mechanism for a automated wheelchair in order to give the user the ability to move the chair forward, backwards, turn left, and right. Additionally, the user should be able to start or interrupt the driving at any time also without the use of hands. Finally, this mechanism should be robust enough to work in different environments with different illumination types.

In order to realize such a mechanism, we propose the use of the movements of the user's face to initialize, move, and interrupt the movements of the wheelchair. A procedure for the initialization and interruption of the movements needs to be created. This way, while driving the chair, the user can at any time stop the controlling, move his/her head freely, and restart

the driving at will. The mechanism needs thus two distinct states corresponding to active and passive tracking. When in passive tracking, the face of the user is being tracked but its movements do not result in any motion for the wheelchair. The system must be waiting for a start command which can be e.g. a sequence of pre-defined movements. The active tracking is the state which most of the face movements yield in motion to the wheelchair. At this stage, the system must be also aware of a pre-defined interruption command.

We have created a protocol to translate the face tracking into commands to the wheelchair. In the passive state, the system must wait for a fast motion of the head in the horizontal direction. While in active driving, the commands to go forward, go backwards, turn left, and turn right were associated with the movements of the head look up, down, to the left, and to the right, respectively. In order to stop the driving and go into the passive mode, the use must again wave his/her head in the horizontal direction. Note that this could be ambiguous with the turning commands, but there is a practical solution. The turning commands can be delayed in such a way that this fast horizontal movements would not be actually translate into wheelchair movements because the passive mode would have already taken oven. In general, this is a very simple and efficient protocol for testing purposes. Nevertheless, in a final implementation more sophisticated protocols might be more interesting to deal with circumstances that were uncovered here, e.g. when the user expresses a 'no' by moving his/her head in passive mode, without the intention to switch the state of the system to the active mode.

Another aspect that should be covered in a final implementation is the initialization of the tracking itself. This must be a fast and straightforward procedure in order the rapidly recover from a tracking failure. Although there are many options for such a procedure, we propose one here which we believe fits well many requirements. Aligned with the camera which captures the image of the user's face, we propose to place a light beam, e.g. oriented LED light, which shines every time a failure occurs, i.e. all the face objects being tracked are lost. Upon a failure, the user must thus place his eye in front of the light beam. The place in the image where this light shines is the place where the initial marker for the first frame is. After the eye is successfully tracked, the light beam goes off. In the sequence the positions of the other objects are calculated relatively to position of the eye. Besides providing an efficient way to recover from failure, this method also provides the user with the information about the status of the tracking through the status of the light beam.

Fig. 4 presents and overview of the main features of the wheelchair driving mechanism.

IV. Implementation and practical considerations

We implemented both face tracking visual algorithm and wheelchair driving protocol in a Bi-I system [7] with an ACE16k_v2 [8] CNN-UM chip placed at one of the optical inputs, and a higher definition CCD camera placed at the other

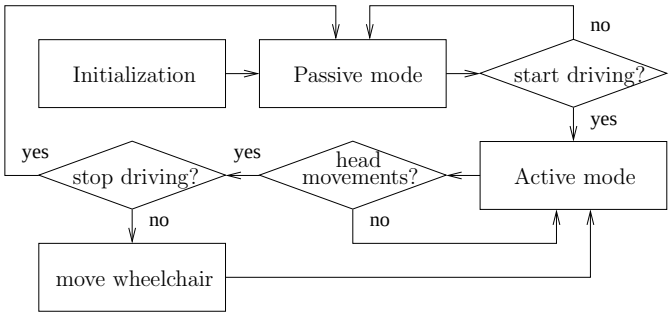


Fig. 4. Overview of the wheelchair driving mechanism.

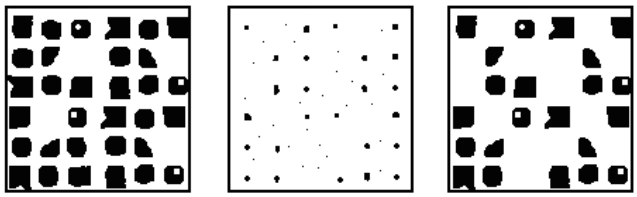


Fig. 5. Training set for the recall template. Input, initial state, and desired output are shown in the picture.

input. Besides the ACE16k_v2 and the CCD camera, the Bi-I also embeds a DSP.

During the implementation we have encountered a number of problems related to the CNN chip and the CCD camera. A description of these problems follows in the next Sections together with the solutions that were applied.

A. Tuning of CNN templates and chip-specific robustness

Unfortunately, templates designed to work in ideal CNNs are not guaranteed to work on analog VLSI CNN implementations [9]. The reason for that lies mainly on manufacturing mismatches, which for analog VLSI are around 10%. Therefore, in order to make use of these templates in a Bi-I, it is necessary to tune the template values for the specific chip to be used. With this purpose, we used the framework in [10] in order to tune the recall and shadow templates.

For the recall template, the images in Fig. 5 were used as training set. Only the tuning of the hardware parameters was necessary to achieve a good working template. Observe that many single dots were added to the marker image. This was necessary to make sure that noise in the initial state image would not recall undesirable objects from the input image.

The same training set design strategy was used for the shadow template, see Fig. 6. The addition of noise to the input and initial state images in the training phase avoided that resulting output images would also shadow noisy pixels. Moreover, in the tuning it is possible to optimize the time necessary to perform the operation. This way, our shadow operation can be performed in the shortest time possible for the chip, improving the overall speed of the tracking algorithm.

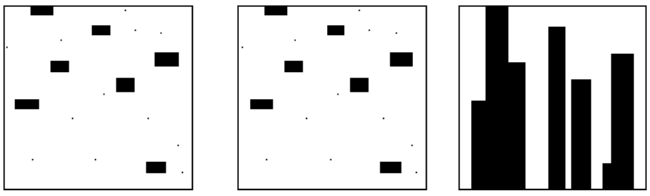


Fig. 6. Training set for the shadow south operation. Input, initial state, and desired output are shown in the picture.

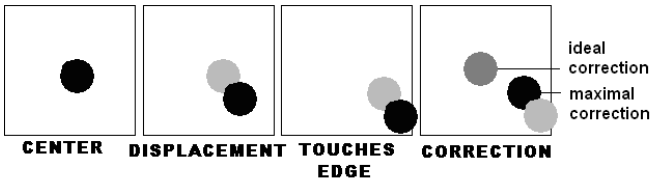


Fig. 7. Defining the maximal correction. Black: current frame, grey: previous frame. The ideal shifting cannot be applied due to the recall operation that always needs overlapping.

B. Maximal shifting and minimal frame rate

The window shifting algorithm described in Section II-B relies on the recall operation to precisely re-position the window in such a way that the object is always in the center of the image. On the other hand, the recall operation also relies on the right window shifting in order to ensure the overlapping of the object in the previous and current frame, see Fig. 7. Therefore, the values for the two different shifting values r and s need to be defined with precaution.

The value of r obviously need to be sufficiently small such that the object is stabilized in the center of the window upon object stop or even little movement. We have used a value of two pixels for the r shifting. When the object is moving with a larger speed than the r shifting can follow, it eventually reaches the border of the image. The s shifting is applied at this point. The ideal value for s would be to bring the object to center of the window. However, depending on the size of the object, the recall operation would promptly fail. This shifting thus must not be larger then the object diameter plus is actual speed. Empirically, good results were achieved with $s = 24$ for a window of size 128 pixels.

The recall operation also can cause the loss of the object when the frame rate becomes too slow. It can occur that wrong objects become also recalled. Upon an abrupt movement it can occur that the overlapping region of the object in two successive frames does not only contain the object being tracked but also another object nearby, e.g by a fast movement of the head in the vertical direction, it can happen that the eye in the previous frame does not only overlap the eye in the current frame but also the eyebrow. The eyebrow and the eye become member of the initial state marker and are tracked together. This in itself is not yet a problem for this application. However, if this eyebrow overlaps with the hair, now the eye, the eyebrow, and the hair become tracked. All these components do not fit in the tracking window and the tracking system would fail to work properly. In this situation, our system assumes that the objects is lost by checking if it cannot be resized back into the window.

C. Illumination issues

One of the most important aspects of processing images is that they need to be of good quality. There are two important aspects in order to acquire good images. The object being photographed must be sharp and the lighting must be appropriate. It may not be too dark nor too bright. Sharpness is generally no problem for the wheelchair case. It is assumed that the wheelchair user always sits on approximately the same distance of the camera. The lens must be adjusted only once to make the tracking objects look sharp. On the other hand, solving the illumination problem is more complex. Since the intention is that the wheelchair drives around, the light intensity might change with that. The parameters that modify the picture illumination must change according to the quantity of the light that is available at that moment. These parameters can be either the opening of the lens, the objective shutter speed, the sensitivity of the CCD or the threshold value for the conversion of gray-scale to binary pictures. The most obvious choice of a parameter to compensate different illumination is the adjustment of the shutter speed or, illumination time, i.e. the time which the CCD sensor is exposed to the light. In the Bi-I, the waiting for this time can be parallel to the CNN template operation for the previous frame. Since the CNN template operations for the object tracking are normally faster than the standard Bi-I illumination time, this becomes a bottleneck for the whole system and therefore must be optimized. Our first step in this direction was to set the value for the objective opening up to the maximum. This causes loss in depth sharpness, which imposes almost no problem since the user of the wheelchair sits always approximately at the same distance of the camera. The sensitivity of the CCD sensor and the binary threshold are other two additional values that we have optimized in order to decrease the illumination time. When close to extreme values, these parameters bring noise into the image and therefore must be carefully used.

D. Testing with a 3D wheelchair model

In order to test the driving system in real-time, we have created in Virtual Reality Modeling Language (VRML) a three-dimensional (3D) environment with a 3D model of a wheelchair. The goal was to make the 3D wheelchair move as an actual wheelchair according to our face tracking mechanism. The commands from the Bi-I were transmitted to the 3D wheelchair via the Matlab VRML toolbox. Fig. 8 depicts the VRML environment and an overview of our setup.

In order to increase robustness, we followed not one but two face features, the left and right eyes. The initialization procedure was performed by positioning the left eye in the middle of the initial tracking window, where the marker for the first frame was. After the left eye was found and locked for tracking, the right eye is immediately inferred and another marker is created at its inferred position.

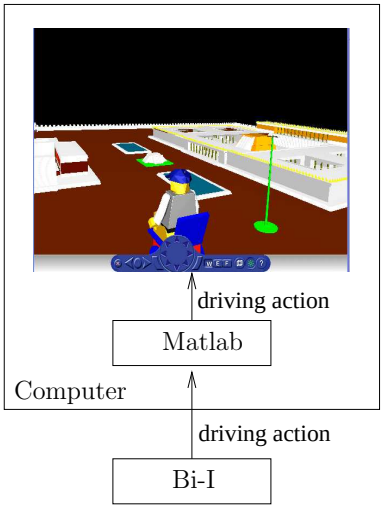


Fig. 8. Test setup for the real-time driving of a 3D model wheelchair using the head movements. Matlab is merely a communication channel between the Bi-I and the 3D environment.

We obtained a frame rate of 92 frames/s for the tracking of the two eyes. This frame rate decreases to an average of 35 frames/s when poor illumination is present. With ideal illumination, optimized memory management, and further tuning of the templates involved, we believe frame rates over 300 frames/s could be reached for a single object.

V. Conclusions

We have proposed a visual algorithm for face tracking which is fast and robust enough to be applied to a hands-free wheelchair driving system. The wheelchair user can drive the chair forward and backwards and steer it to the left or to the right accordingly to his/her head movements. Additionally, the user can interrupt or restart the driving at any moment by predefined head movements. The system also adapts to a different range of illumination intensities. Additional robustness could be achieved by tracking multiple face features at the same time. This way, if one of these features is lost, its position can be inferred accordingly with the position of the features that are still being tracked, then the lost feature can be reinserted into to the tracking algorithm. Tests of this system in a 3D simulation environment have shown that a actual physical implementation is feasible.

Acknowledgment

Research supported by: • Research Council K.U.Leuven: GOA-Mefisto 666, GOA-AMBioRICS, BOF OT/03/12, Center of Excellence Optimization in Engineering; • Flemish Government: ○ FWO: PhD/postdoc grants, G.0407.02, G.0080.01, G.0211.05, G.0499.04, G.0226.06, research communities (ICCoS, ANMMM); ○ Tournesol 2005 • Belgian Federal Science Policy Office IUAP P5/22. J.A.K. Suykens is an associate professor, and J. Vandewalle is a full professor, both with K.U. Leuven Belgium. SXS and MVD thank Dániel Hillier for the fruitful discussions.

References

[1] G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," *Intel Technology Journal*, no. Q2, p. 15, 1998.

[2] T. Darrell, B. Moghaddam, and A. Pentland, "Active face tracking and pose estimation in an interactive room," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'96)*, San Francisco, CA, June 1996, pp. 67–72.

[3] L.-P. Morency and T. Darrell, "Head gesture recognition in intelligent interfaces: the role of context in improving recognition." in *Intelligent User Interfaces*, 2006, pp. 32–38.

[4] B. E. Shi, P. Arena, and A. Zarandy, "Special Issue on CNN Technology and Active Wave Computing," *IEEE Trans. on Circuits and Systems-I*, vol. 51, no. 5, pp. 849–850, May 2004.

[5] T. Roska, "Computational and computer complexity of analogic cellular wave computers," *J. Circuits, Systems, and Computers*, vol. 12, no. 4, pp. 539–56, 2003.

[6] S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle, "Real-time tracking algorithm with locking on a given object for VLSI CNN-UM implementations," in *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and their applications*, Budapest, Hungary, Sep 2004, pp. 291–296.

[7] A. Zarándy and C. Rekeczky, "Bi-i: a standalone ultra high speed cellular vision system," *IEEE Circuits and Systems Magazine*, vol. 5, no. 2, pp. 36–45, 2005.

[8] A. Rodríguez-Vázquez, G. Liñán Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Gálan, F. Jiménez-Garrido, R. Domínguez-Castro, and S. Meana, "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Trans. on Circuits and Systems-I*, vol. 51, no. 5, pp. 851–863, May 2004.

[9] S. Xavier-de-Souza, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle, "Toward CNN Chip-specific robustness," *IEEE Trans. on Circuits and Systems-I*, vol. 51, no. 5, pp. 892–902, May 2004.

[10] D. Hillier, S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle, "CNNOPT: Learning dynamics and CNN chip-specific robustness," submitted for publication, 2006.