A Neuro-inspired Visual Tracking Method Based on Programmable System-on-chip Platform

Shufan Yang*[1], KongFatt Wong-Lin*[2], James Andrew[1], Terrence Mak[3], T Martin McGinnity[4],

[1]School of Mathematics and Computer Science, University of Wolverhampton

[2]Intelligent Systems Research Centre, University of Ulster, UK

[3]Department of Electronics and Computer Science, University of Southampton, UK

[4.] School of Science and Technology, Nottingham Trent University, UK

***Corresponding author:**

*****Shufan Yang, School of Mathematics and Computer Science, Faculty of Science and Engineering, City Campus, Wolverhampton Phone: (+44) 01902-518-594, email: s.yang@wlv.ac.uk.

Acknowledgements: SY was supported by the Early Research Scheme Reward from University of Wolverhampton and National High Technology Research and Development Program from China. KFW-L and SY were supported by ASUR (1014-C4-Ph1-071).

Running title: A Neuro-inspired Visual Tracking Model

ABSTRACT

Using programmable system-on-chip to implement computer vision functions poses many challenges due to highly constrained resources in cost, size and power consumption. In this work, we propose a new neuro-inspired image processing model and implemented it on a system-on-chip Xilinx Z702c board. With the attractor neural network model to store the object's contour information, we eliminate the computationally expensive steps in the curve evolution re-initialization at every new iteration or frame. Our experimental results demonstrate that this integrated approach achieves accurate and robust object tracking, when they are partially or completely occluded in the scenes. Importantly, the system is able to process 640 by 480 videos in real-time stream with 30 frames per second using only one low-power Xilinx Zynq-7000 system-on-chip platform. This proof-of-concept work has demonstrated the advantage of incorporating neuro-inspired features in solving image-processing problems during occlusion.

Keywords: Visual object tracking, mean-shift, level set, attractor neural network model, occlusion, system-on-chip

1. INTRODUCTION

Due to the advantages offered by embedded devices for computational intensive applications, many algorithms in image processing are implemented in such hardware [1-4]. Nevertheless, one of the biggest challenges is to implement image processing algorithms onto rigid resource constraints with low cost and high computationally efficiency. For instance, a visual tracking using an optical flow algorithm has been optimised for VLIW DSP architectures [3]. However, it only achieves 5 fps in 200 MHz. A Particle filter based multiple objects tracking has been implemented on an FPGA [4]. But those implementations

are not suitable for general platform due to high power consumption and the requirement for expensive hardware resources.

Besides the rigid resource constraints in the programmable system-on-chip platform, the visual tracking is a difficult task, especially when objects are partially or wholly occluded in the visual field [5]. These occlusion issues in the object tracking are generally addressed using some forms of prediction or estimation methods [6]. For example, a common approach in visual tracking is to assume a constant motion or acceleration to project the position of object from previous frame to a new position during occlusion [7]. However, in realistic scenarios those assumptions are often violated due to cluttered background (e.g. similar colors of target and background) and dynamical changes of objects' shape during occlusion [7].

Visual systems in humans and animals, in general, can easily deal with such issues. Hence, it would be interesting to incorporate neuro-inspired features, especially into the programmable system-on-chip platform, to exploit their advantages. For example, attention-modulated coordinate systems that used in the visual tracking can modulate, enhance, retain and predict relevant visual information of the object [8 -10]. In a practical example, García et al. (2013) used the commercial Kinect camera incorporated with a human/animal-like inhibition of return behavior for detecting unknown visual objects in an office environment. However, visual object tracking in occlusion and cluttered environments still remains unsolved, especially with low-power hardware devices.

In this work, we incorporate a neural network model with traditional computer vision algorithms to solve the occlusion and cluttered scene issues in visual scenes. The neural network model will be conveniently utilized in the form of a specialized function block

within the system-on-chip platform for real-time computation (see below). In particular, an attractor neural network model is integrated with the mean-shift based object tracking algorithm and the level-set active contour method. The mean-shift method is used to calculate image density for tracking vectors while the level-set active contour method is for mapping the object's contour [11-12]. The attractor state of the neural field model is used to retain the current contour in the object's absence and provides fast convergence in the subsequent frame/iteration. Our integrated prototype model primarily has the advantage of employing a dynamical neural network to avoid the initialization process of the curve evolution and allow fast convergence, hence improving computational efficiency while reducing power consumption.

2. MATERIALS AND METHODS

During the visual object tracking tasks, two major processes are involved: occlusion detection and occlusion handling. Occlusion is detected when parts of the object features are obscured and not visible. Specifically, if the distance between two objects is decreasing and the size of one object is changing dramatically, the latter is considered as an occluded object (see Section 3.2 for more details). During the occlusion handling stage, the attractor neural network is used to alleviate occlusion issues, which is integrated with a mean-shift visual tracking method and a level-set active contour method.

Traditional mean-shift tracker suffers inaccurate representation of objects due to the constancy of the kernel bandwidth, which can result in an inaccurate representation. A new target location in the current frame is calculated using the mean-shift procedure, which computes the translational offset of the target location in each frame [12], which can lead to more inaccuracy when occlusion happens. To overcome such inaccuracy in the presence of

total occlusion, we combine the level-set based active contour and color histogram as an object representation. This is because the level-set based active contour and feature abstraction can effectively control topological changes, which is important for tracking moving objects in cluttered scenes. However, the curve evolution needs to be re-initialized at every new iteration or frame, which is computationally expensive. To overcome this, an attractor state of the neural network is used to retain the contour information and hence allows rapid convergence of the level-set based active contour in the new frame. In other words, the attractor property in the network dynamics can be used to "store" the location of an occluded object over time to track it. The initialization process will be switched on only when the neural network model is not in its attractor state. Overall, our proposed integrated approach maintains the advantage of convenient implementation of the mean-shift method while solving occlusion problems in the object visual tracking at a low computational cost.

2.1 Basic mean-shift tracking method

Before occlusion, the average size of the object's contour is calculated based on the visual object features using commonly used mean-shift algorithm for the object tracking [10]. The mean-shift procedure is employed to calculate a new target location in current frame based on the location of the target in the previous frame. The mean-shift framework described here follows to the implementation in Comaniciu et al. (2000) [13].

Mean-shift is an algorithm to track objects whose appearance is defined by histograms. In the context of tracking, A target model Q is defined using a set of data point $Q(x)$ in Euclidean space that describes the tracking object's associated colour. In this work, the color histograms are used as a feature space [12]. The Bhattacharya coefficient is used as a similarity measure to calculate mean-shift vectors. The bin size of histograms of oriented

gradients is 6. To incorporate image scaling with camera zooming, 4 different scales are computed using adjacent histograms. Therefore the overall value for a target model is 24 (6 × 4). *Q(x)* denotes the kernel density estimator, as defined by:

$$Q(x) = 1/24 \sum_{i=1}^{24} K(\frac{x_i - L}{h}) \quad (1)$$

where x_i represents pixel *i*'s value in the current frame; *L* is the coordinates of the center of the region of interest. *h* is the scale window that defines the scale of the targeted object, i.e. the number of pixels will be considered in the localization process. This density function *K(.)* determines the weight of nearby points for re-estimating of the mean density value. In Comaniciu et al. (2000)'s work, *K(.)* is the Epanechnikov kernel and is defined by [14].

$$K(x) = \begin{cases} 2 * {}^{1}/_{\pi}(1 - x) & x < 1 \\ 0 & x \geq 1 \end{cases} \quad (2)$$

The translation offset of the mean-shift vector Δx is computed by the following:

$$\Delta x = \left[\frac{\sum_{i=1}^{n} x_i K(\|(x_i - x_0)/R\|^2)}{\sum_{i=1}^{n} K(\|(X_i - X_0)/R\|^2)}\right] - x \quad (3)$$

where K(.) is a radially symmetric kernel as descripted in equation (2). The bandwidth R defines the tracked object region. The weight at pixel x_i is estimated assuming it follows a uniform distribution. The new position in the current frame is calculated by the mean-shift vector iteration according to Equation (3).

2.2 Level-set based active contour

The traditional mean-shift requires a targeted object to be a rigid shape, such as a circle or a rectangle. However, the mean-shift tracking approach can suffer from inaccuracy of object representations [12], because it does not entirely represent the object shape and may contain

non-object regions as part of the tracking object. Although some researchers have used deformation or similarity parameters techniques, those approaches were only used for tracking the affine parameters and unable to handle local deformation of the objects, especially when occlusion occurs [15].

In this work, we adapt the level-set based active contour approach into the object tracking. The level set method that was originally proposed by Sethian and Osher (1999) [17]. It was used to implement object segmentation by evolving a closed contour to the object's boundary [18-19]. Implicit level set function $\emptyset(\mathrm{x})$ encodes the signed distances of the pixels *x* from the tracking object boundary. The object's region is implicitly defined as the zero crossings in the level set grid. In this manner, the level set can change its topology of object contour while maintaining the form of a graph. Evolution of the contour is governed by computing the regional energy: curve evolution is first performed globally (Figure 1 (A)), and then locally modified each iteration until close to the desired object's boundary (Figure 1(B)). In order to derive a density estimator in the mean-shift tacking, Equation (2) has been modified to:

$$\mathrm{K}\,(\mathrm{x}) = \begin{cases} 1/\mathrm{n}(\sum_{i=1}^{n} \emptyset(E(C, c1, c2))) & \mathrm{x} < 1 \\ \mathrm{o} & \mathrm{x} \geq 1 \end{cases} \tag{4}$$

Mumford and Shah (1989) first proposed a method of segmenting the image into nonverlapping regions using an energy function. In our work, a special case of Mumford-Shah model is used to solve minimisation problem during segmentation, followed up the Chan and Vese (2002)'s approach. We assume that image consists of two regions that can be approximated using piecewise-constant intensities. For image Q(x), the energy function is :

$$\mathrm{E}(\mathrm{C}, \mathrm{c1}, \mathrm{c2}) = \int_{\mathrm{inside(C)}} \mathrm{U(x)}|\mathrm{Q(x)} - c1|^2\, \mathrm{dx} + \int_{\mathrm{outside(C)}} \mathrm{U(x)}|\mathrm{Q(x)} - c2|^2\, \mathrm{dx} + \ |\mathrm{C}| \tag{5}$$

where C is the approximated contour for the image Q(x); inside(C) denotes the inner region of contour C, and outside (C) denotes the outer region of the contour. The first term and second term in Equation (5) are used to bring the contour close to the image intensity distribution (Figure 1B), while the third term is a constraint term and smoothness of the contour. *C1* and *C2* represent the average value of the pixel inside C and outside C respectively. U(x) has localization property for fast curve evolution (See section 3 for details).

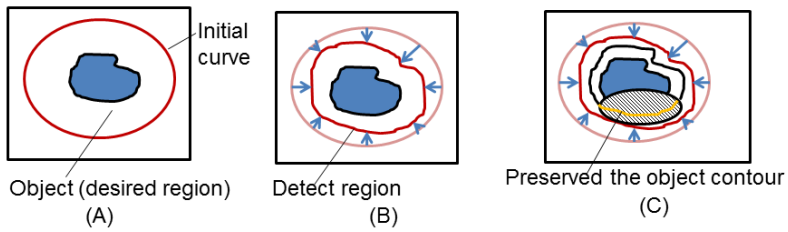


Figure 1. A schematic representation of implementing the level-set method for recovering occlusion in a visual space. (A) The initial curve is indicated by a red circle, which is defined by the average image density function. The region of dark blue represents the target object to be tracked. (B) The curve then evolves and gets closer to the boundary of the tracked object. (C) A shaded region represents the occluding object. The black curve is the curve evolution using a level-set approach. The orange curve is restored using the signed level-set function based on the attractor state of the neural network.

2.3 Occlusion handing using neural network dynamics

Traditional mean-shift tracking method does not consider occlusion effects and the inaccurate target model can easily lead to tracking failures. As illustrated in Figure 1(C), during occlusion, a completed boundary curve of the occluded object cannot be obtained. Here, we propose to model the non-rigid changes in the object's shape using an attractor neural network, which encodes the history of the object's contour.

After an occlusion is detected, an attractor neural network is employed to rapidly capture and then maintain the state of the tracked object's boundary over time. In our method, the energy function used in this model is integrated with the attractor states of neural network to make fast convergence for the occluded object. Also, the energy function of the curves can avoid the re-initialization process if the neural network is already in the attractor states. Hence, the computational efficiency is greatly improved. Consequently, computational power consumption can be reduced.

A standard neural network to discriminate and store sensory information over time was used [20, 21]. These types of recurrent networks are also called attractor neural networks, and under certain conditions, their dynamics can encode and store information (attractor state) in the absence of the external stimulus. This typically requires the network architecture to consist of short-range excitatory and long-range inhibitory lateral connections [22]. In particular, the strong short-range recurrent excitation allows localized and activated neural activity to be sustained when the stimulus is removed.

As shown in Figure 2(A), we demonstrate an example using three neural populations in the network model. Let $A_{exc1}(x,t)$ and $A_{exc2}(x,t)$ are excitatory variables at position x with a time t. $A_{inh}(x,t)$ is inhibitory variables at position x, which can be described, respectively, by [22]

$$\tau_{exc}\frac{\partial A_{exc1}(x,t)}{\partial t} = -A_{exc1}(x,t) + F_{inh}(x,t) + I_{ext1} \tag{6}$$

$$\tau_{exc}\frac{\partial A_{exc2}(x,t)}{\partial t} = -A_{exc2}(x,t) + F_{inh}(x,t) + I_{ext2} \tag{7}$$

$$\tau_{inh}\frac{\partial A_{inh}(x,t)}{\partial t} = -A_{inh}(x,t) + F_{exc1}(x,t) + F_{exc2}(x,t) \tag{8}$$

where I_{ext1} and I_{ext2} are the external inputs to the excitatory neural population A_{exc1} and A_{exc2}. τ_{exc} and τ_{inh} are time constants of the corresponding neural population, respectively. F_{exc1} , F_{exc2} and F_{inh} are the input-output functions to the excitatory and inhibitory population, which can be described by

$$F_{exc1}(x,t) = \int_{\Omega} J_{ie}(x-\acute{x})A_{exc1}(\acute{x},t) - J_{ee}(x-\acute{x})A_{inh}(\acute{x},t)\, d\acute{x} \quad (9)$$

$$F_{exc2}(x,t) = \int_{\Omega} J_{ei}(x-\acute{x})A_{exc2}(\acute{x},t) - J_{ee}(x-\acute{x})A_{inh}(\acute{x},t)\, d\acute{x} \quad (10)$$

$$F_{inh}(x,t) = \int_{\Omega} J_{ie}(x-\acute{x})A_{exc1}(\acute{x},t) + J_{ei}(x-\acute{x})A_{exc2}(\acute{x},t) - J_{ee}(x-\acute{x})A_{inh}(\acute{x},t)\, d\acute{x} \quad (11)$$

where the integration is over the 2-dimensional space Ω. The J′s are the synaptic coupling strengths which depend on the relative spatial distance x to $\acute{x}$. Specifically, J_{ie} represents the coupling strength from the excitatory neural population A_{exc1} to the inhibitory neuronal population, while J_{ei} represents the coupling strength from the excitatory neural population A_{exc2} to the inhibitory neuronal population. While J_{ee} represents the coupling strength from the inhibitory neuronal to excitatory neural population. Note that we have ignored self-inhibition within the inhibitory population. For simplicity, we allow the input-output functions F′s to be of threshold-linear type, and the τ′s to be the same for both excitatory and inhibitory populations. The exponential convergence towards the attractor state requires the following conditions to be satisfied:

$$1 < 2\, J_{ee} \sqrt{J_{ie}\, J_{ei}} \quad (12)$$

$$0.075 < J_{ie}\, J_{ei} < 1 \quad (13)$$

When the model exhibits persistent activity behavior, i.e. maintenance of the signal even in the absence of the presented stimulus (Figure 2 (B)), the neural activity is larger than baseline activity (zero) and the conditions (12-13) and bi-stable steady state solution can be obtained by solving $\frac{dA_{exc1}(x,t)}{dt} = 0.$. Conceptually, Equation (12) means that the self-

excitation J_{ee} needs to be sufficiently strong, while Equation (13) means that intermediate level of recurrent inhibition is required to suppress unwanted activations, generating a temporary "storage" behavior. The upper bound in Equation (13) is to prevent overly strong inhibition that eliminates any stimulus-based activation.

When an occlusion is detected, the cue stimulus is first provided to population A_{exc1}, resulting in an increase in activity A_{exc1} and then sustained. When the occlusion of the tracking object is detected, the cue stimulus is activated, causing A_{exc2} to increase and then sustained, while A_{exc1} is suppressed, due to inhibition. This is necessary to permit both locations of the targeted object to be tracked and maintained over successive frames.

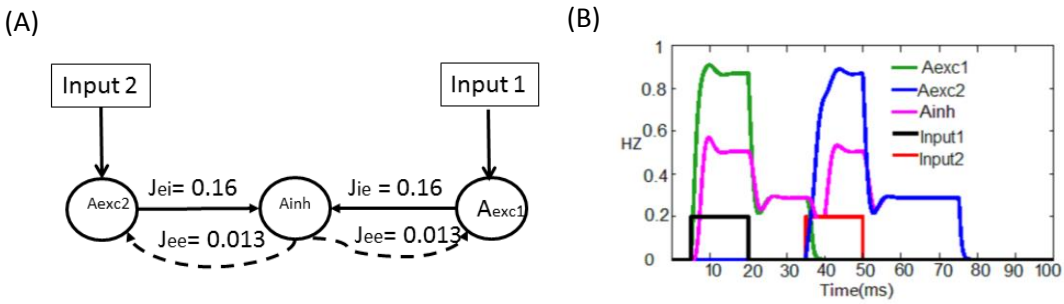


Figure 2. A simplified attractor neural network model to illustrate persistent activity for temporary memory storage. (A) For simplicity, the neural network considered here contains only 3 neural populations. The inhibitory connections are denoted by dash lines and excitatory connections by bold lines. (B) Sample persistent neural activities over time. The pulse input stimulus to the excitatory population A_{exc1} is provided from 5 to 20ms. When input stimulus is removed from the 2ms time point, the population activities relax into another albeit lower steady state or attractor. Note that although stimulus input (1) is received by population A_{exc1}, A_{inh} is also activated.

During curve evolution, we define F_{outx} as an attractor state in the neural network that stores the outer boundary of contour from the previous frame, while F_{inx} is another attractor state that stores the inner boundary of contour in the previous frame. C is the approximated detected region from the previous frame. As an illustration, in a one-dimensional case, the following region-evolving algorithm defined localisation property *U(x)* in the x direction (the algorithm for y direction is similar) as defined by:

$$U(\mathrm{x}) = \begin{cases} 1 & \text{if x is outside C but not inside } \mathrm{F_{inx}} \\ 3 & \text{if x is inside } \mathrm{F_{outx}} \\ -3 & \text{if x is outside } \mathrm{F_{inx}} \\ -1 & \text{if x is inside C but not inside } \mathrm{F_{inx}} \\ 0 & \text{otherwise} \end{cases}$$

Here, we choose the values of ±3 and ±1 as signed distances, which depends on the relative position of the initial contours to the target object. In this work, the normalized neural population activities and the associated states of curve evolution for this example are shown in Table 1.

Table 1 Normalized neural population activities and the associated value of level-set function

Normalize population activity in Aexc1	The value of level set function	Normalize population activity in Aexc2	The value of level set function
0 ~ 0.25	1	0 ~ 0.25	3
0.25 ~ 1	-1	0.25 ~ 1	-3

2.4 General steps for object tracking

In practice, given a pre-defined target model, our proposed algorithm can be summarised in the following steps:

1. Initialise the location of the target in the current frame based on its location in the previous frame.
2. Compute iterations of curve evolution for each object at an initial curve.
3. Activate neural network when occlusion is detected.
4. If the activity of neural network is in the attractor state, the curvature of target object will be computed based on this attractor state. Otherwise, there is no occlusion and the curvature of the target object will be calculated as usual. In this case, the neural network is not involved.
5. Compute the mean-shift vector using calculated weighted colour histogram of target object.
6. If the region of interest and centre of mass are similar, stop the evolution and return to step 1.

2.5 The System-on-Chip implementation

We employ a customized parallel architecture to implement our efficient object-tracking algorithm. The logical partitioning of the system is divided into software and hardware blocks performing video capture, subtracting background, generating active contour and modeling dynamical neural network. The system-on-chip design is partitioned into computational blocks with interconnection communication channels for performing data manipulation. This partition enhances the chances of corrected operation when the different parts of the algorithm are merged, while allowing new algorithm development to commence

in parallel with a hardware development. The following describes the architecture and design flow using a Xilinx Zynq-702 evaluation board. The FPGA design process follows a similar approach to our previous work [23]. The source code of this project can be down loaded from open source repository github (https://github.com/WOLVS/).

2.5.1 Design flow

We used one ARM Cortex-A9 core (at 667MHz) to implement image acquisition. Another processor core serves interrupts from the programmable logic. The programmable logic system partitioned into computational blocks with interconnected communication channels for performing data manipulation. This partition enhances the chances of corrected operations when different parts of algorithm are merged, while allowing new algorithms development to commence in parallel with a hardware development. It is essential to explore various visual tracking methods and feature abstractions in parallel in order to develop an efficient integrated system. As show in Figure 3, the process starts with checking the image boot load from a flash memory, then loading all the libraries for video decoding.

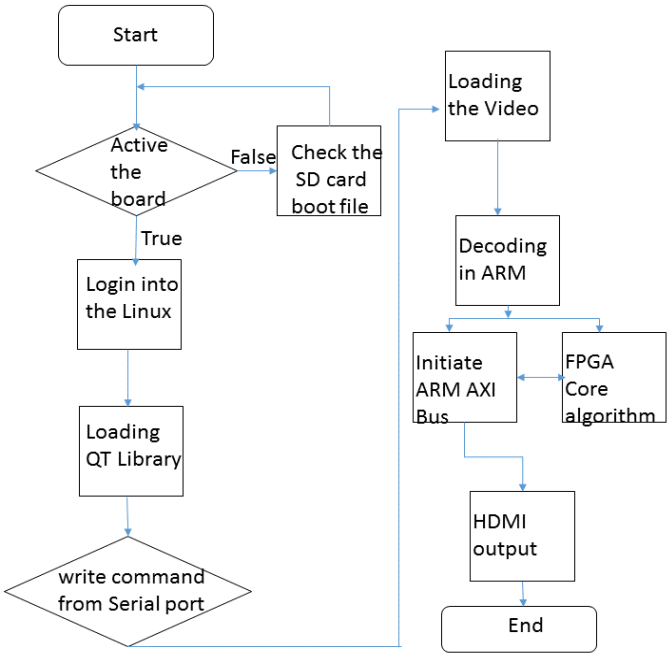


Figure 3. Design flow for ARM and Programmable IP integrate

2.5.2 Programmable IP implementation

As shown in Figure 4 (A), the foreground objects from the input frame are detected and are computed using the image density mean shift method. The frame counter is used to facilitate pipeline calculations. The Bhattacharyya coefficient is calculated in the mean-shift module and the new location is calculated after iterations using the loop box (see Figure 4(B) for more details). The attractor neural network block is used to keep the boundary of curve evolution at a given time period, and preserves the contour of the object during partial and full occlusion. For on-line debugging purpose, the command interpreter is designed to assist the tracking with interfaces for setting up pre-defined feature spaces of the targeted object. Thus, the "set" instruction sets the frame store to write back from memory and the "delay" instruction inserts a delay enabling screen effect that allows user to observe contour evolution during processing while the "relocate" instruction writes back entries in the frame store.

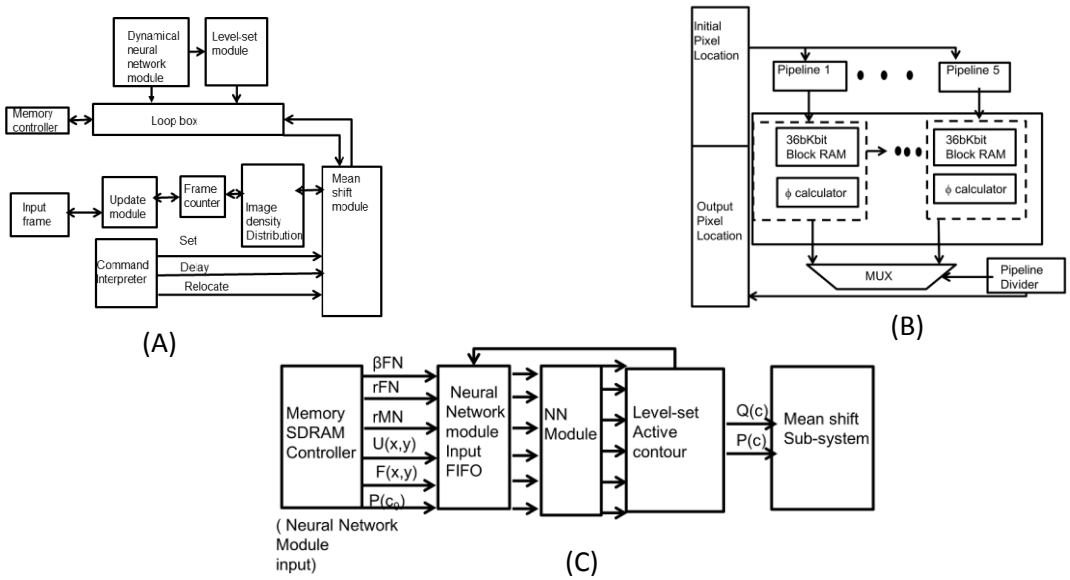


Figure 4. Block diagram of object tracking (A) Block diagram of the proposed method in system view. (B) Loopbox design for running multiple iterations in five pipelines design.

(C) Attractor neural network implementation: Communication protocol between the attractor neural network and level-set and mean-shift module

Figure 4(C) depicts the dynamical neural network block diagram. The NN module is used to mode the attractor neural network. The βFN and rFN are the input currents for two excitatory neuron populations; the rMN is the input current for the inhibitory neuron. U(x) and F (x) are initial contour starting points; $P(c_0)$ is the initiate mass density for the mean-shift system. Q(c) and P(c) are generated the mass densities for the mean vectors.

The system design block diagram can be found in Figure 5 (A) and Figure 5 (B) shows the floor-planning of the SOC data fusion platform. The neural network processing unit is located in between the two processing stages, in such a way that they are used as a fence to prevent a single data transfer from corrupting, show as the red fence in Fig.5. The dark blue fence is data acquisition and the light blue part is for HDMI interfaces. The neural network module and image processing modules are physically near to the processing stages with which they share most of their connections, lead to short paths and ultimately high stable clock frequency (200MHZ)).

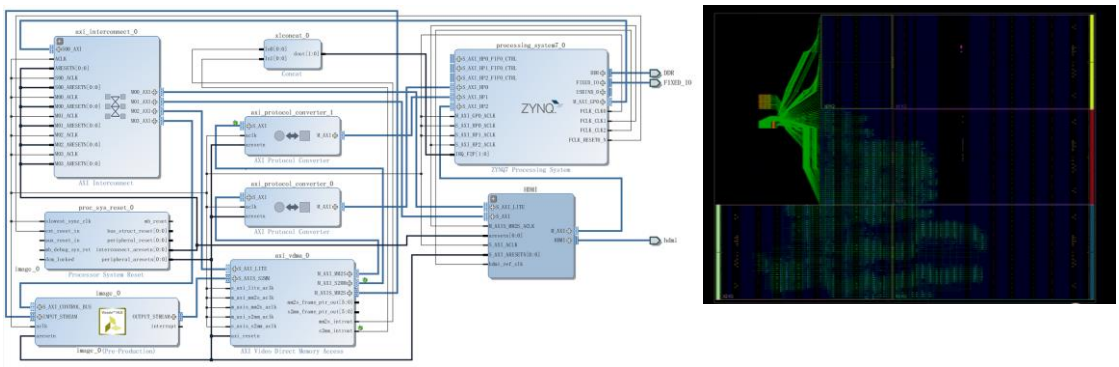


Figure 5 (A)Block Diagram of overall system with HDMI module included for visulasation. Image_0 module is the core module for neural network processing (B) XC702 SOC floor-planning.

3. RESULTS

This section presents three realistic experiments, which are used to test the ability of the system to correctly handle occluded object. Our hardware implementation test two resolution inputs: VGA resolution 640 by 480 (VGA) video and 1024 by 768 (XGA) video. The incoming video frames are processed, using our integrated model in a system-on-chip platform, within three main stages: (1) image data preparation; (2) active contour evolution combined with attractor neural network; and (3) update mean vectors.

3.1 Hardware resource analysis

The implementation of the dynamic neural network with standard floating-point arithmetic units on FPGA is rather straightforward. A summary of the resource usage for the whole design is shown in Figure 6. The first column shows the type of hardware resources, and each row shows the total number of logical resources used. A memory controller used in this design consisted of single bidirectional 128-bit port configuration and two additional FIFO buffers (1025 words deep). Using this module a constant data flow at the level of 30 frames per second achieved with the image resolution of 640 by 480. The maximum operating frequency of whole design was 200MHz, which is more than enough for processing video stream (pixel clock rate of 25 MHz).

Since the visual tracking system needs to access large data inside loops, the power consumption of this system is largely contributed by data transfer and memory access operations. The data operation consumes much less power than memory access. Our method minimizes the off-chip memory access using attractor neural network, since intermediate data can be represented by population efficiency and redundant memory access are reduced. Power consumption of the whole programmable IP reported by Xilinx XPower Analyzer for the device (On-Chip) was 0.03 W [24].

Resource Type	Modules		
	Mean Shift	Level-Set	Neural Network
LUT(k)	245.8	87.9	43.2
Block RAM(36kb)	112	207	80
DSP48	98	34	18

Table 1 Hardware resource usages and comparison

3.2 Occlusion detection

During occlusion, visual features of the occluded object are not observed and the object's contour is recovered by an attractor state of the recurrent neural network. Figure 6 shows an occlusion recovery example, where one synthetic object occludes a person. Before the occlusion (Figure 6(A)), contour evolution is based on visual features and the object's contour is complete.

(A) (B)

Figure 6. A synthetic object is occluding a walking person. (A) The first image of a sequence of images. (B) A synthetic object is the occluding object. The white box next to the image shows the extracted objects.

Figure 6(B) shows an occlusion example, where a synthetic object (black rectangle) occludes a walking person. During occlusion, the evolution using the visual feature results in an incomplete contour. The moving person and the synthetic block are two objects labelled as A and B. When the size of object A is dramatically changed while the size of object B is remained, we label object B as occluding object A. The Euclidean distance between the two objects A and B is labelled as $D_{A,B}$. The current size of object A is $\emptyset_A^t$. The average size of object A is $\emptyset_A^{ave}$. Occlusion detection is calculated using the value $\frac{1}{\exp(-|D_{A,B}|)+1} \times \frac{\emptyset_A^t}{\emptyset_A^{ave}}$; if this value is less than 0.25, we assume that object A is occluded. Figure 6(B) demonstrates the correct extraction of the two objects.

3.3 Moving objects with partial and full occlusion

Our focus is on the tracking of occluded objects where occlusions may be present and the camera may not necessarily be stationary. However, there are not many openly available datasets with these characteristics [26]. Hence, we conduct the experiment using two case studies: walking person and children playing. Note that the popular pedestrian's video clips in the Caltech database [26] would be too challenging for this work, of which one of objectives is for prototyping. This is mainly due to the detection module with weak response to illumination changes as well as the dramatic changed in size of people in the video clips (see Section 2).

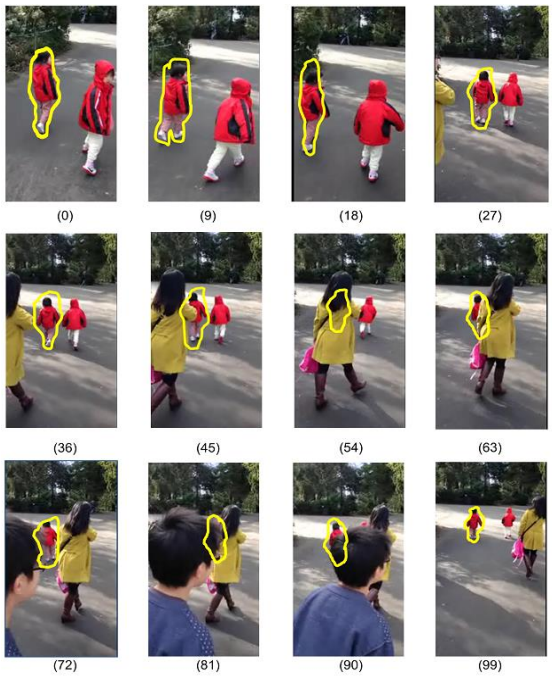


Figure 7. Tracking of object in a multiple people walking sequence with total occlusion. Under each image is labelled the frame number. The video clips can be viewed at the following link (https://www.youtube.com/watch?v=Kq5dxpiYyjs).

A multiple people walking sequence is illustrated in Figure 7. In this scene, partial occlusions occur repeatedly and there is total occlusion towards the end of the sequence before the person reappears. Hence, the detected region can be dramatically changed and even disappeared altogether. When the mean shift tracker fails to generate the mean vector, the curve evolution needs to be restarted and many computational steps are involved. In Figure 8, the results of tracking integrated with the attractor neural network can correctly depict the active contour of moving persons during partial occlusion by recovering its hidden parts (frames 45, 63, 72, and 81, in Figure 7). During total occlusion, the (yellow) curve predicts and evolves to locate the moving object in the preceding frame even if the real object is fully concealed (frames 54 and 90, in Figure 7). Significant partial occlusion and total occlusion also occur in the sequence. During total occlusion, the bottom-up feature

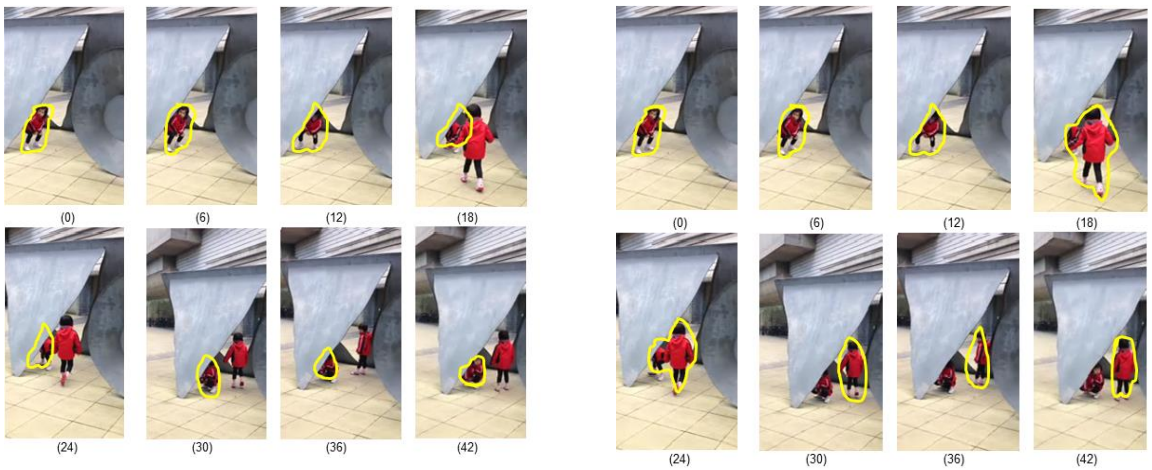
abstraction guides the contour evolution. After total occlusion the person is detected although the detected shape is not exactly the same as the object's contour.

3.4 Comparison

Sometimes the contour shape of the tracked object can be occluded or undergo profile variation, as shown in the above experiment. There are two challenges that make the scene an interesting example for object tracking. The first challenge is when a section of the contour is occluded unpredictably. The other challenge is a cluttered scene, in the sense that the moving object and the background have a similar random texture. A cluttered scene makes distinguishing between background and foreground difficult. A traditional mean-shift algorithm alone might not be able to track this particular type of scenario since the prior colour is computed inaccurately due to the cluttered scene [12].

Figure. 8 shows a realistic example of occlusion. In this scene, two children were playing, and partial occlusions occurred repeatedly. When occlusion occurs, there is less relevant information available to the model's visual inputs for tracking. However, our model is still able to track the targeted object by inferring from the geometric contour in the previous frames (Figure 8 (A)). In particular, the attractor neural network model is able to sustain the significant deformation occurring over time to enable the continuity of tracking. These combined mechanisms present considerable benefit when the occluded object is highly mobile. In comparison, the particle filter, which is one of the most common geometric models for tracking, is not as effective and robust in generating the same object representation [12]. The results are shown in Figure 8 (B). The poor accuracy of the particle

filter may be because it uses only the energy function to weight particles and evolve the curve over time.



(A) (B)

Figure 8. Tracking children playing to test for object tracking with occlusion. (A) By constraining the curve to retain a similar shape between consecutive frames, the tracked object with partial occlusion was maintained even in the presence of a similar (colour) object in the same frame. (B) Tracking method using particle filter, in which the tracked object was unintentionally switched to a similar object that has occluded the originally tracked object. (The particle filter experiment here used 35 particles.)

An important issue of visual tracking that often comes up in practice is that of the algorithm's computational efficiency. We compare the computational efficiency of our proposed method with the basic mean-shift tracking method [11] and particle filter method [12]. This is achieved by computing the average computational time used to track one frame of video. Table 2 shows the running time for the three methods in the system-on-chip platform. The particle filter method outperforms the tradition mean-shift method because the mean-shift method uses local max/min density value and minimise the distance in the current frame. It

is noted that our proposed integrated method requires less computational cost compared to the other two methods; about 10% lesser than the basic mean-shift method and 20% lesser than the particle filter method.

Table 2 Comparison of computational efficiency in terms of averaging time required to track one frame with 1024 by 768 (XGA format) and 640 by 480 (VGA format).

Video	Proposed method (s)	Mean-shift method (s)	Particle filter method (s)
Multiple people walking scene 1024 by 768 (Figure 8)	1.115	1.235	1.345
Children's playing scene 1024 by 768 (Figure 9)	0.731	0.812	0.936
Multiple people walking scene 640 by 480 (Figure8)	0.038	0.041	0.045
Children's playing scene 640 by 480(Figure 9)	0.033	0.036	0.039

4. DISCUSSION

There has been significant scientific debate regarding the appropriate incorporation of biological approaches to address computer image processing. Typically, neurobiologically realistic computational models are computationally costly [27]. In this paper, we strike a compromise between biologically inspired and efficient computation by adopting an attractor neural field model to the traditional computer vision method of curve evolution, allowing for a reduction in curve evolution iterations during object occlusion situations, and hence improving computational efficiency. Specifically, we have used traditional mean-shift tracking and level-set methods to track the contour of a moving person, and when partial or total occlusion occurs, the attractor states of a neural field model can store the contour

information in visual space to refine the evolution of the curve and preserve the object contour in the subsequent frame. This eliminated the computationally expensive re-initialization of the curve evolution at every new iteration or frame. Importantly, the system is able to process, in real-time using only one low-power Xilinx Zynq-7000 system-on-chip platform. Overall, our proof-of-concept work has successfully demonstrated the advantage of incorporating neuro-inspired features in efficiently solving image-processing problems.

Despite the positive results demonstrated in this work, there remain some limitations to our approach. For on-chip implementation, slow memory speed limits the feed-in frame rates in our system. Our current implementation also could not handle scenes with objects undergoing dynamical scale change (e.g. due to camera's zooming) and unpredictable motion (e.g. sharp turns and sudden stops). Furthermore, the basic idea of the background updating we have implemented is based on the assumption that the pixel value of background changes slower than those of the tracked objects. In many realistic scenarios, it is a valid assumption. However, tracking non-uniform color object can potentially become challenging. These issues will be addressed in the future, and extended methods can then be implemented on the challenging pedestrian datasets such as the Caltech pedestrian datasets.

In this work, although we have only adopted a neural field model with basic attractor features, in future work, we can extend it to other more complex visual tasks using more dynamic neural field capabilities. For example, our present work can easily be extended to more complex visual search or visual motor control paradigms [28]. By equipping the neural field model with adaptive mechanisms, it can produce anticipative and enhanced tracking capabilities especially on time-varying stimuli. Also, our framework can be expanded into multiple object tracking. With multiple objects to be tracked simultaneously, multiple nodes

can be added for multiple regions of occluded object. It should be noted that in this work, we have assumed that the tracked object is not moving out of the frame and the first image has the tracking object already in the visual scene.

To summarize, we have successfully integrated traditional mean-shift tracking and level-set methods with an attractor neural field model, and solved, as proof-of-concept, various occlusion problems during tracking moving objects. It opens up the opportunity of providing low-power neuro-inspired system-on-chip platform for the challenging visual object tracking application.

REFERENCES

1. Malamas, E. N., Petrakis, E. G., Zervakis, M., Petit, L., & Legat, J. D. (2003). A survey on industrial vision systems, applications and tools. Image Vis. Comput., 21(2), 171-188.

2. Nikitakis, A., Papaioannou, S. and Papaefstathiou, I. (2013). A novel low-power embedded object recognition system working at multi-frames per second. ACM Transactions on Embedded Computing Systems (TECS), 12(33), pp.39-58

3. Díaz, J., Ros, E., Pelayo, F., Ortigosa, E. M., & Mota, S. (2006). FPGA-based real-time optical-flow system. *Circuits and Systems for Video Technology, IEEE Transactions on*, *16*(2), 274-279.

4. Jin, J., Lee, S., Jeon, B., Nguyen, T. T., & Jeon, J. W. (2013). Real-time multiple object centroid tracking for gesture recognition based on FPGA. In: Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, article 80.

5. Nguyen, H. T., & Smeulders, A. (2004). Tracking aspects of the foreground against the background. In *Computer Vision-ECCV 2004* (pp. 446-456). Springer Berlin Heidelberg.

6. Lee, B. Y., Liew, L. H., Cheah, W. S., & Wang, Y. C. (2014). Occlusion handling in videos object tracking: A survey. In IOP Conference Series: Earth and Environmental Science (Vol. 18, No. 1, p. 012020). IOP Publishing.

7. Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. ACM Computing Surveys (CSUR), 38(4), 13.

8. Yantis, S., & Johnson, D. N. (1990). Mechanisms of attentional priority. Journal of Experimental Psychology : Human Perception and Performance.

9. Frintrop, S., Rome, E., & Christensen, H. I. (2010). Computational visual attention systems and their cognitive foundations: A survey. ACM Trans. Appl. Percept., 7(1).

10. García, G. M., Frintrop, S., & Cremers, A. B. (2013). Attention-Based Detection of Unknown Objects in a Situated Vision Framework. KI-Künstliche Intelligenz, 27(3), 267-272.

11. Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(5), 603-619.

12. Perez, P., Vermaak, J., & Blake, A. (2004). Data fusion for visual tracking with particles. In: Proceedings of the IEEE, 92(3), 495-513.

13. Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893). IEEE.

14. Epanechnikov, V.A. (1969). "Non-parametric estimation of a multivariate probability density". Theory of Probability and its Applications 14: 153–158. doi:10.1137/1114019.

15. Han M., Xu W. and Gong Y. (2004) An algorithm for multiple object trajectory tracking. In: Proceedings of the IEEE Computer Society Conference, 1: 864-871.

16. Chan, T. E., & Vese, L. A. (2001). A level set algorithm for minimizing the Mumford-Shah functional in image processing. In Variational and Level Set Methods in Computer Vision, 2001. Proceedings. IEEE Workshop on (pp. 161-168). IEEE.

17. Osher, S., & Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics*, *79*(1), 12-49.

18. Mendi, E., & Milanova, M. (2010). Contour-based image segmentation using selective visual attention. *Journal of Software Engineering and Applications*, *3*(08), 796.

19. Cremers, D. (2006). Dynamical statistical shape priors for level set-based tracking. IEEE Trans. Pattern Anal. Mach. Intell., 28(8), 1262-1273.

20. Cremers, D. (2013). Shape Priors for Image Segmentation. In: Shape Perception in Human and Computer Vision, pp. 103-117, Springer, London.

21. Rutishauser, U., & Douglas, R. J. (2009). State-dependent computation using coupled recurrent networks. *Neural computation*, *21*(2), 478-509.

22. Johnson, J. S., Spencer, J. P., Luck, S. J., & Schöner, G. (2009). A dynamic neural field model of visual working memory and change detection. Psychol. Sci., 20(5), 568-577.

23. Yang S., McGinnity T. M., and Wong-Lin, K. (2012). Adaptive Proactive Inhibitory Control for Embedded Real-time Applications. Front. Neuroeng. 5:10. doi: 10.3389/fneng.2012.00010.

24. Xilinx, 2014 Xilinx Xpower Analyzer [Online]. Available: <www.xilinx.com/products>.

25. Cehovin, L., Kristan, M., & Leonardis, A. (2013). Robust visual tracking using an adaptive coupled-layer visual model. IEEE Trans. Pattern Anal. Mach. Intell., 35(4), 941-953.

26. Dollár, P., Wojek, C., Schiele, B., & Perona, P. (2009, June). Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 304-311). IEEE.

27. Kruger, N., Janssen, P., Kalkan, S., Lappe, M., Leonardis, A., Piater, J., ... & Wiskott, L. (2013). Deep hierarchies in the primate visual cortex: What can we learn for computer vision?. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(8), 1847-1871.

28. Fung, C. C., Wong, K. Y., Wang, H., & Wu, S. (2012) Dynamical synapses enhance neural information processing: gracefulness, accuracy, and mobility. Neural Comput., 24: 1147-1185.