2009 Ninth International Conference on Intelligent Systems Design and Applications

GPU-Based Road Sign Detection using Particle Swarm Optimization

Luca Mussi, Stefano Cagnoni
Dipartimento di Ingegneria dell'Informazione
University of Parma
Viale G. Usberti 181a, I-43124 Parma, Italy
{mussi},{cagnoni}@ce.unipr.it

Fabio Daolio
Information Systems Institute (ISI) - HEC
University of Lausanne
Internef 135, CH-1015 Lausanne, Switzerland
fabio.daolio@unil.ch

***Abstract*—Road Sign Detection is a major goal of Advanced Driving Assistance Systems (ADAS). Since the dawn of this discipline, much work based on different techniques has been published which shows that traffic signs can be first detected and then classified in video sequences in real time. While detection is usually performed using classical computer vision techniques based on color and/or shape matching, most often classification is performed by neural networks. In this work we present a novel approach based on both sign shape and color which uses Particle Swarm Optimization (PSO) for detection. Remarkably, a single fitness function can be used both to detect a sign belonging to a certain category and, at the same time, to estimate its actual position with respect to the camera reference frame. To speed up execution times, the algorithm exploits the parallelism offered by modern graphics cards and, in particular, the CUDA™ architecture by nVIDIA. The effectiveness of the approach has been assessed on a synthetic video sequence, which has been successfully processed in real time at full frame rate.**

I. Introduction

Real-time automatic road sign recognition is one of the goals of Advanced Driver Assistance Systems (ADAS) [1], [2]. It can both improve safety, letting the driver concentrate more on driving, and help navigation, by providing necessary information the driver could otherwise miss. The task of road sign recognition is usually divided into the subtasks of detection and classification; the latter is often performed by artificial neural networks, while the former is carried out mainly following color-based or shape-based approaches [3].

In this paper we present a mixed approach. Sets of 3D points, which belong to the contour of the road sign model to be detected and describe its shape, are sampled and projected onto the image plane according to a transformation which maps points in the camera reference frame onto the image, and matched with the actual image content. The likelihood of detection is evaluated using a similarity measure based on color histograms [4]. This procedure actually estimates the pose of an object based on a 3D model and can be utilized with any projection system and any general object model. One of the advantages over other model-based approaches is that this approach does not need any preliminary pre-processing of the image (like color segmentation) or any reprojection of the full 3D model [4]. Another difference from similar previous work [5], along with the aforementioned similarity measure, is that, in our case, the space of possible poses is searched by means of Particle Swarm Optimization (PSO), an effective bio-inspired population-based metaheuristic.

Due to the large amount of resources needed to run the algorithm and to compute the fitness function, a classical sequential implementation of the system would probably result in excessively long execution times. To achieve real-time processing, we implemented our system on graphics hardware within the nVIDIA CUDA™ environment [6], to take advantage of the computing power offered by the massive parallel architectures available on present consumer video cards. As will be shown, thanks to the parallel nature of PSO, this choice is very effective, since the final system is able to manage several swarms at the same time, each of which can detect a specific kind of road sign.

This paper is organized as follows: Section II briefly introduces PSO; Section III addresses the problem of road sign recognition, motivating our approach and offering further details on how shape and color information is used in the fitness function. In Section IV the GPU-based implementation of the system is briefly described and, finally, in Section V we report some results obtained on a synthetic video sequence containing a warning and a regulatory sign.

II. Particle Swarm Optimization

Particle Swarm Optimization is a simple but powerful optimization algorithm, introduced by Kennedy and Eberhart in 1995 [7]. In the last decade many variants of the basic PSO algorithm have been developed [8] and successfully applied to many problems in several fields [9], image analysis being one of the most frequent ones. In fact, image analysis problems can be often reformulated as the problem of optimizing an objective function, directly derived from the physical features of the problem. Beyond this, PSO can often be more than a way to "tune" the parameters of another algorithm, but directly the source of the solution. For example, [10], [11], [12], [13], [14] use PSO to directly infer the position of an object that is sought in the image.

PSO searches for the optima of a function, termed *fitness function*, following rules inspired by the behavior of flocks

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.88

of birds looking for food. A population of particles, each of which encodes a point in the fitness function domain, move within the search space. At each move, the fitness of a particle, associated with its new position, is evaluated. In their motion, particles preserve part of their velocity (inertia) while undergoing two attraction forces: the first one, called cognitive attraction, attracts a particle towards the best position it visited so far, while the second one, called social attraction, pulls the particle towards the best position ever found by the whole swarm. In basic PSO, the following velocity and position-update equations are applied for each particle:

$$\begin{aligned} V_i(t) &= w \cdot V_i(t-1) \\ &+ C_1 \cdot R_1 \cdot [X_{i_b}\ (t-1) - X_i(t-1)] \\ &+ C_2 \cdot R_2 \cdot [X_{i_{gb}}(t-1) - X_i(t-1)] \quad (1) \\ X_i(t) &= X_i(t-1) + V_i(t) \quad (2) \end{aligned}$$

where the subscript i refers to the i-th dimension of the search space, V is the velocity of the particle, C_1, C_2 are two positive constants, w is the inertia weight, $X(t)$ is the particle's position at time t, $X_b(t-1)$ is the best-fitness position reached by the particle up to time $t-1$, $X_{gb}(t-1)$ is the best-fitness point ever found by the whole swarm; R_1 and R_2 are two random numbers from a uniform distribution in $[0, 1]$.

III. Road Sign Detection

Suppose that an object of known shape and color may appear within the field of view of a calibrated camera. In order to detect its presence and, at the same time, to precisely estimate its position, one can follow this algorithm (see also [4]):

1) Consider a set of key contour points, of known coordinates with respect to a reference position, and representative of the shape and colors of the object.
2) Translate (and rotate) them to a hypothesized position visible by the camera and project them onto the image.
3) Verify that color histograms of the sets of key points match those of their projection on the image to assess the presence of the object.

Simple objects characterized by just a few regions of homogeneous colors, as in the case of road signs, can be described by sets of key points which lie just near the color discontinuities, with points which belong to the same set being characterized by the same color. Again, once all these points are projected onto the image plane one must verify that colors of each set of corresponding points in the image match the original ones. A further set of points, lying just a bit outside the object silhouette, can help verify whether the object border has been detected: this would be confirmed in case colors in corresponding points of the image are significantly different from those of the object.

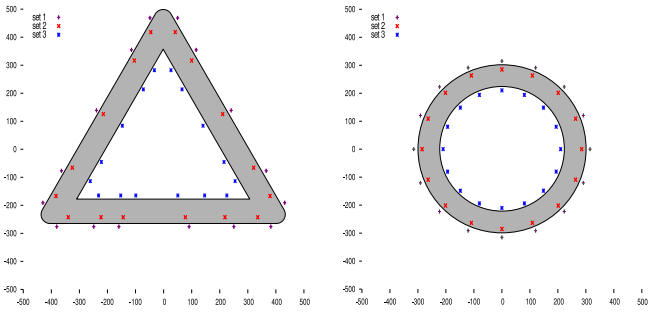

Figure 1. The three different sets of points used to represent a warning sign (left) and a regulatory sign (right). All coordinates are expressed in millimeters.

In Fig. 1 we show two kinds of traffic signs, warning (left) and regulatory (right), along with the sets of points we use to represent them. For each sign model, we consider three sets of 16 points: one lies just outside the external border (therefore, on the "background"), one on the red band just inside the external border and one on the central white area, as close to the red border as possible. Please note that, for the regulatory signs, we use points uniformly distributed around its circular shape, while for the triangle we consider more points near the corners than along the sides. This choice enhances the differences in the fitness values corresponding to the two kinds of signs.

If a calibrated camera is available on a moving car, along with an estimate of the position and rotation of a road sign inside the camera's field of view, the above sets of points can be roto-translated to this position and then projected on the images acquired, to verify the likelihood of the hypothesis. All is needed is a method to make hypotheses and refine them until the actual position of a sign is found.

To this aim we apply PSO, as introduced in section II. In our method, each particle encodes an estimation of the position of the sign with four values, representing its x, y and z offsets, as well as its rotation around the vertical axis (yaw) in the camera reference frame. A single swarm can then be used to detect the possible presence of a sign belonging to a particular category within the image. In case one is found, its position with respect to the car will be also determined. At the moment, no classification of the content of the signs which have been detected is peformed to distinguish between different signs of the same class.

PSO is run at each new frame acquisition for a predefined number of generations. At the moment no tracking is performed except, implicitly, for the fact that we do not reinitialize the particles' position (only velocities are randomly re-set) on the next frame when the best fitness of the last iteration is good enough (hopefully meaning that a sign has been correctly found). In the next subsection we describe the fitness function we use in our PSO-based approach in details.

A. Fitness Function

Let us denote the three sets of points used to describe each kind of sign (see Fig. 1) as $S_1 = \{\boldsymbol{s}_i^1\}$, $S_2 = \{\boldsymbol{s}_i^2\}$ and $S_3 = \{\boldsymbol{s}_i^3\}$, with $\boldsymbol{s}_i^x \in \mathbb{R}^2$ (they all lie on the xy plane) and $i \in [1, 16]$. Based on the position of one particle and on the projection matrix derived from camera calibration, each set of points is roto-translated and projected onto the current frame obtaining the corresponding three sets of points lying on the image plane $P_1 = \{\boldsymbol{p}_i^1\}$, $P_2 = \{\boldsymbol{p}_i^2\}$ and $P_3 = \{\boldsymbol{p}_i^3\}$.

To verify whether the hypothesized position is actually correct, three color histograms in the RGB colorspace, one for each channel, are computed for each set P_x with $x \in \{1, 2, 3\}$. Let us denote them as $\boldsymbol{H}_x^c$ where, again, $x \in \{1, 2, 3\}$ and $c \in \{R, G, B\}$ specifies the channel for which the histogram is computed. More formally, we can state that:

$$H_x^c(b) = \frac{1}{n} \sum_{i=1}^{n} \delta(I_c(\boldsymbol{p}_i^x) - b) \tag{3}$$

where $c \in \{R, G, B\}$ specifies the color channel, $x \in \{1, 2, 3\}$ identifies the set of points, $b \in [1, N_{bin}]$, N_{bin} being the number of bins in the histogram, n represents the number of points per set (sixteen in our case), the function $\delta(n)$ returns one when $n = 0$ and zero otherwise and, finally, $I_c(\boldsymbol{p}) : \mathbb{R}^2 \rightarrow \mathbb{R}$ maps the intensity of channel c at pixel location $\boldsymbol{p}$ to a certain bin index. Note that the term $\frac{1}{n}$ normalizes the histogram ($\sum_{b=1}^{N_{bin}} H_x^c(b) = 1$). Moreover, three additional histograms denoted with $\boldsymbol{H}_{ref}^c$ are used as reference histograms for the red band. At this point the Bhattacharyya coefficient (ρ) [15], which offers an estimate of the amount of overlap between two statistical samples, is used to compare the histograms.

$$\rho(\boldsymbol{H}_1, \boldsymbol{H}_2) = \sum_{b=1}^{N_{bin}} \sqrt{H_1(b) H_2(b)} \tag{4}$$

The Bhattacharyya coefficient returns a real value between 0 (no overlap at all between the two histograms), and 1 (the two histograms are identical). Finally, if we use

$$S_{x,y} = \frac{\rho(\boldsymbol{H}_x^R, \boldsymbol{H}_y^R) + \rho(\boldsymbol{H}_x^G, \boldsymbol{H}_y^G) + \rho(\boldsymbol{H}_x^B, \boldsymbol{H}_y^B)}{3} \tag{5}$$

to express the similarity of the two triplets of histograms computed for the sets of points x and y we can express the fitness function as

$$f = 1 - \frac{k_0\ (1 - S_{1,2}) + k_1\ (1 - S_{2,3}) + k_2\ S_{1,ref}}{k_0 + k_1 + k_2} \tag{6}$$

where $k_0, k_1, k_2 \in \mathbb{R}^+$ are used to weigh the contributions of the three distances appearing in the above formula. Here we want the histograms computed on the first two sets of points to be as different as possible, hypothesizing that, in case the sign had been "hit", the background colors nearby the sign would be significantly different from those of the red band. Similarly, we want the histogram computed for the points of the red band to be as different as possible from the one computed on the white area. Finally, we want histograms H_1^x to resemble as much as possible the reference histograms H_{ref}^x for the red band surrounding the sign. Histograms of regions having colors that are slightly different from the model, possibly because of noise, produce high values of $S_{1,ref}$. The fitness function f will therefore be close to 0 only when the position of one particle is a good estimate of the sign pose in the scene captured by the camera.

IV. Implementation on GPU

CUDA™'s parallel programming model is designed to allow the programmer to partition the main problem in many sub-problems that can be solved independently in parallel. Each sub-problem may then be further decomposed in many modules that can be executed cooperatively in parallel. In CUDA™'s terminology, each sub-problem becomes a *thread block*, which is composed by a certain number of *threads* which cooperate to solve the sub-problem in parallel. The software modules that describe the operation of each thread are called *kernels*: when a program running on the CPU invokes a kernel, a unique set of indices is assigned to each thread to denote to which block it belongs and its position inside it. These indices allow each thread to 'personalize' its access to the data structures and, in the end, to achieve problem parallelization and decomposition.

To exploit the impressive computation capabilities offered by CUDA™ effectively and develop a parallel version of PSO, the best approach is probably to consider the main phases of the algorithm as separate tasks, each of which must be parallelized separately: this way, each phase can be implemented by a different kernel and the whole optimization process can be performed by scheduling repeatedly all the basic kernels needed to perform one generational update of the swarm. Since the only way CUDA™ offers to share data among different kernels is to keep them in global memory, the current status of our PSO must be saved there. Data organization is therefore the first problem to tackle to exploit the GPU read/write coalescing capability. Our data design, besides encouraging this aspect, permits to run several swarms at the same time simply by playing with the thread indices.

We generate pseudo-random numbers directly on the GPU using the Mersenne Twister [16] kernel provided by the CUDA™ SDK: this way the CPU load is virtually zero. A brief outline of our kernels follows.

A. Bests Update

For each swarm, a thread block is scheduled with a number of threads equal to the number of particles in the swarm. Firstly, each thread loads in shared memory both

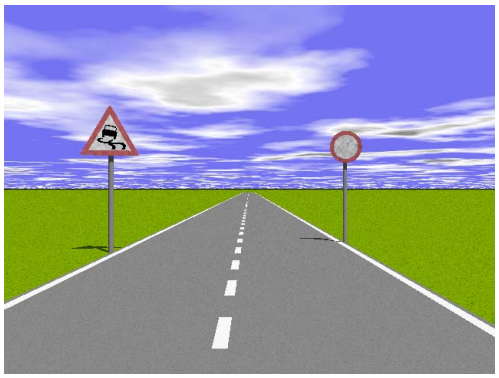

Figure 2. Sample frame taken from our synthetic video sequence simulating a country road scenario with two differently shaped roadsigns.

the current and the best fitness values of its corresponding particle. Then, the personal best is updated, if needed. Successively, parallel reduction is performed to determine the current best fitness value in the swarm, that is then compared with the best value found so far, to be possibly updated.

B. Position Update

A computation grid, divided into blocks of 32 threads, has the duty to update the position of all particles being simulated. Each thread updates one element of the position and velocity arrays, irrespective of the particle (or the dimension) to which it corresponds. At the beginning the particle's current position, personal best position, and velocity are loaded, after which the classical PSO equations are applied to the values.

C. Fitness Evaluation

This kernel is scheduled as a computation grid composed by one block for each particle being simulated (irrespective of the swarm to which it belongs). Each block comprises a number of threads equal to the total number of points that describe a sign (three sets of 16 points each) so that the projection of all points on the current image is performed in parallel. Successively, each thread contributes to building the histograms described in section III-A: the thread index determines to which set/histogram the projected point under consideration belongs, while the sampled color value determines which bin is to be incremented. Finally the fitness value is computed according to (6) where, once again, the parallelism among threads is exploited to compute similarities among histograms.

V. Experimental Results

As a first test we decided to evaluate our system on synthetic video sequences. To do so, we simulated a 3D rural environment with a road and a pair of traffic signs using the public-domain raytracer POV-Ray[1]. We relied on the Roadsigns macros by Chris Bartlett[2] to simulate the signs and on some ready-to-use objects by A. Lohmüller and F.A. Lohmüller [3] to simulate the road. Bumps and dirtiness were added to the traffic signs in order to simulate more realistic conditions. In Fig. 2, a sample frame from one of the synthesized sequences is shown. As time passes, the simulated car moves forward zigzagging from left to right. At the same time, as they get closer to the car, the two signs rotate around their vertical axis. We introduced rotations to test the ability of our system to estimate the actual roto-translation between the camera and the sign that is detected. In fact, in our case, each particle moves in $\mathbb{R}^4$ and its position represents the x, y and z offsets of the sign as well as its rotation on the vertical axis (yaw).

Experiments were run on a PC equipped with an Intel Core2 Duo processor running at 1.86GHz with a moderately-priced GeForce 8800 GT video card by nVIDIA corporation, equipped with 1GB of video RAM. The PSO parameters were set to $w = 0.723$, $C_1 = C_2 = 1.6$, and two swarms composed by 64 particles were run for up to 50 generations per frame: the first swarm seeks for regulatory signs (characterized by a circular shape) while the second one looks for warning signs (of triangular shape). The coefficients appearing in (6) were empirically set as follows: $k_0 = 1.2$, $k_1 = 1.0$, $k_2 = 1.4$.

The implementation of the whole system within the CUDA™ architecture described above permitted to achieve very good execution times, despite the computation intensiveness of the algorithm. In fact, a sequential version of the algorithm required about 80 ms of processing time per frame (limiting the maximum acceptable frame rate to about 12 fps), whereas the parallel version achieved a speed-up of about 20 times requiring 4 to 6 ms to process each frame. This means that we could either locate signs in videos running up to $150/160$ frames per second, or that, at 25 frames per second, more than 30 ms per frame would still be available to perform sign recognition, after a sign has been detected.

Fig. 3 shows three different frames obtained at the very beginning, in the middle and nearly at the end of the sequence, respectively. The contrast of the image has been reduced to better highlight the swarm positions. Points superimposed in white over the images represent the current hypothesis about the sign position made by the best individual, while black points depict current hypotheses of all other individuals. In Fig. 3.a it is possible to see the two swarms during the initial search phase: in this case both are on the wrong target despite being already in the proximity of a sign. Fig. 3.b and 3.c show how the two swarms correctly converged on their targets. This always happened after a small amount of frames, usually less than 20, meaning that

[1] visit http://www.povray.org/ for more information

[2] http://lib.povray.org/collection/roadsigns/chrisb 2.0/roadsigns.html

[3] http://f-lohmueller.de/pov_tut/objects/obj_500i.htm

Figure 3. Output of a run of our road sign detection system at the very beginning (a), at middle length (b) and near to the end (c) of a synthetic video sequence.

the system is able to detect a sign in less than one second since its appearance.

For a more detailed performance analysis, Fig. 4 shows results obtained in estimating the actual position of the signs with respect to the camera position. Fig. 4 (top-left) shows the actual x position and the estimated one (mean and standard deviation over one hundred runs), versus the frame number, for both the warning (green line) and the regulatory (red line) signs. As can be seen, after a brief searching phase, always lasting less than twenty frames, the horizontal position for the two signs is correctly detected and tracked until the end of the sequence with a precision of the order of centimeters. The sinusoidal trend of the two position reflects the zigzagging behaviour of the car we simulated. The top-right part of the figure shows results for the y coordinates. This time the actual position is constant since the simulated car has a constant pitch. Again, after the initial search phase, the estimated position is correct, with errors of just few centimeters. Similar considerations can be done for the bottom-left graph of Fig. 4, which reports results of depth (z coordinate) estimation. Despite the long distance from the car (about fifteen meters in the beginning) estimates are very good with errors of less than half a meter. The error is mostly due to the distance between the two most external sets of points, which introduces a tolerance in centering the target. Tightening this distance could improve the precision of the results but, at the same time, would make it more difficult to obtain high fitness values for signs which are far from the car: this happens because, when the sign is far away from the car, the projections of these two sets of points on the image almost overlap, producing color histograms which are very similar. Finally, in the bottom-right part of the figure we show results of yaw estimation. In this case results are not as precise: it is possible to see how the rotation of the warning sign is estimated rather precisely, although in the first third of the simulation the standard deviation is very high, while the rotation of the regulatory sign seems to be randomly estimated. Since we are dealing with small angles this must be due, again, to the tolerance in locating the signs introduced by the distance between the two external sets of points.

VI. Conclusions and Future Directions

We have shown that PSO, provided with the right fitness function, can be effective in detecting traffic signs in real-time. Experimental results on synthetic video sequences showed that our system is also able to correctly estimate the position of the detected signs with a precision of about ten centimeters in all directions, with depth being (obviously) the less accurate. As for future experiments with real-world images, we think that switching from the RGB to the HSV colorspace would permit to obtain results similar to those presented here for synthetic sequences.

The very next step in our research will be to exploit the accurate estimation of the position of the sign offered by our method to rectify its image by means of a simple Inverse Perspective Mapping (IPM) [17]. This means it will always be possible to obtain a standardized frontal view of the detected sign and thus to easily classify/recognise its content irrespective of its actual orientation and distance.

A future step will also be to test our method in detecting multiple signs of the same kind simultaneously. This might be done in essentially two different ways: by using the same swarm to detect many targets by re-running the optimization process several times on the same image, masking image regions where signs have been previously found or by running many different competing swarms at the same time on the same frame, that look for the same sign shape. For this solution to work inter-swarm repulsion forces could be introduced, in order to prevent different swarms from converging on the same target.

References

[1] A. Broggi, P. Cerri, P. Medici, P. P. Porta, and G. Ghisio, "Real time road signs recognition," in *Proc. IEEE Intelligent Vehicles Symposium (IV'07)*. Washington, USA: IEEE, Jun. 13–15, 2007, pp. 981–986.

[2] Jose M. Armingol et al., "IVVI: Intelligent Vehicle based on Visual Information," *Robotics and Autonomous Systems*, vol. 55, no. 12, pp. 904–916, 2007.

[3] Y.-Y. Nguwi and A. Kouzani, "A study on automatic recognition of road signs," in *Proc. IEEE Conference on Cybernetics and Intelligent Systems (CIS'06)*. Washington, USA: IEEE, Jun. 7–9, 2006, pp. 1–6.

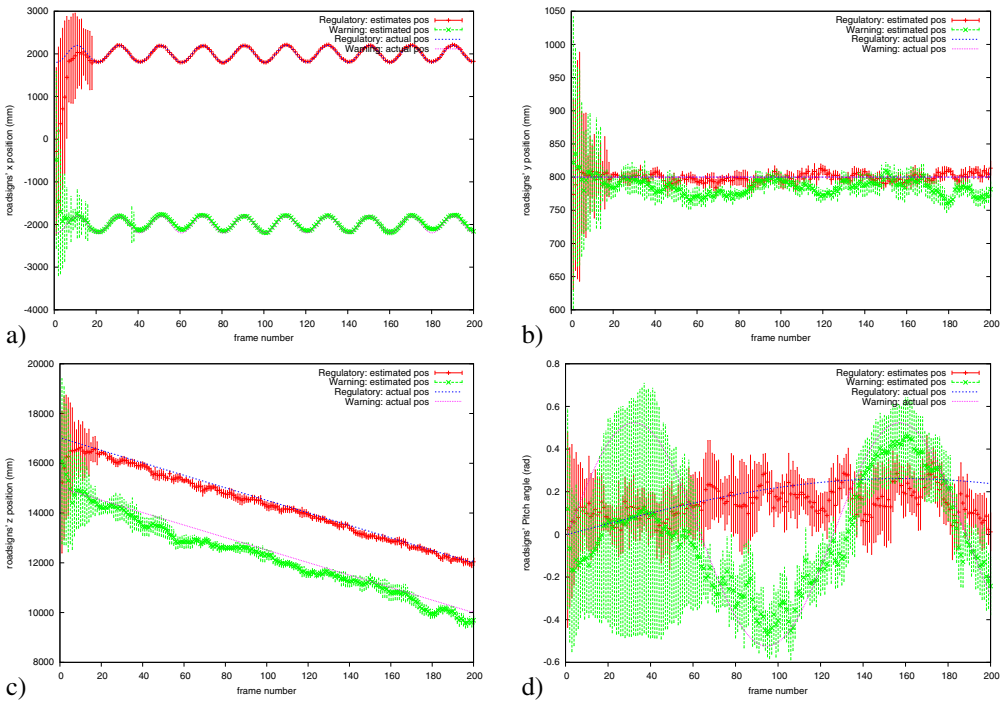

Figure 4. Position estimation errors: (a) shows the estimation of the horizontal position (x coordinate) for both signs, (b) shows the vertical position estimates, and (c) the depth estimates. The estimations of sign rotations around the vertical axis are shown in (d).

[4] M. Taiana, J. Nascimento, J. Gaspar, and A. Bernardino, "Sample-based 3D tracking of colored objects: a flexible architecture," in *Proc. of the British Machine Vision Conference (BMVC'08)*. Worcs, UK: BMVA, Sep. 1–4, 2008, pp. 1–10.

[5] A. de la Escalera, J. M. Armignol, and M. Mata, "Traffic sign recognition and analysis for intelligent vehicles," *Image and Vision Computing*, vol. 21, no. 3, pp. 247–258, 2003.

[6] *nVIDIA CUDA Programming Guide v. 2.3*, nVIDIA Corporation, Sep. 2009. [Online]. Available: http://www.nvidia.com/object/cuda_develop.html

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE International Conference on Neural Networks*, vol. IV. Washington, USA: IEEE, Nov. 27–Dec. 18, 1995, pp. 1942–1948.

[8] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[9] Riccardo Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, no. 1, pp. 1–10, 2008.

[10] Xiaoqin Zhang, Weiming Hu, S. Maybank, Xi Li, and Mingliang Zhu, "Sequential particle swarm optimization for visual tracking," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. Washington, USA: IEEE, Jun. 24–26, 2008, pp. 1–8.

[11] L. Mussi and S. Cagnoni, "Artificial creatures for object tracking and segmentation," in *Applications of Evolutionary Computing: Proc. of EvoWorkshops 2008*. Berlin, Germany: Springer, Mar. 26–28, 2008, pp. 255–264.

[12] S. Cagnoni, M. Mordonini, and J. Sartori, "Particle swarm optimization for object detection and segmentation," in *Applications of Evolutionary Computing: Proc. of EvoWorkshops 2007*. Berlin, Germany: Springer, Apr. 11–13, 2007, pp. 241–250.

[13] Luis Anton Canalis, Mario Hernandez Tejera, and Elena Sanchez Nielsen, "Particle swarms as video sequence inhabitants for object tracking in computer vision," in *Proc. IEEE International Conference on Intelligent Systems Design and Applications (ISDA'06)*. Washington, USA: IEEE, Oct. 16–18, 2006, pp. 604–609.

[14] Y. Owechko and S. Medasani, "A swarm-based volition/attention framework for object recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition - Workshops (CVPR'05)*. Washington, USA: IEEE, Jun. 20–25, 2005, pp. 91–91.

[15] T. Kailath, "The divergence and Bhattacharyya distance measures in signal selection," *IEEE Trans. Commun. Technol.*, vol. 15, no. 1, pp. 52–60, 1967.

[16] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.

[17] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle," *Biological Cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.