# Segmentation Performance in Tracking Deformable Objects via WNNs

Mariacarla Staffa[1], Silvia Rossi[1], Maurizio Giordano[2], Massimo De Gregorio[3], Bruno Siciliano[1]

*Abstract*— **In many real life scenarios, which span from domestic interactions to industrial manufacturing processes, the objects to be manipulated are non-rigid and deformable, hence, both the location of the object and its deformation have to be tracked. Different methodologies have been applied in literature, using different sensors and techniques for addressing this problem. The main contribution of this paper is to propose a Weightless Neural Network approach for non-rigid deformable object tracking. The proposed approach allows deploying an on–line training on the shape features of the object, to adapt in real–time to changes, and to partially cope with occlusions. Moreover, the use of parallel classifiers trained on the same set of images allows tracking the movements of the objects. In this work, we evaluate the filtering/segmentation performance that is a fundamental step for the correct operation of our approach, in the scenario of pizza making.**

## I. INTRODUCTION

In our daily life, robots have to deal with the manipulation of numerous objects, which are non-rigid or deformable, such as, for example, clothes and foods in domestic applications, rubber tubes, sheet metals, cords, paper sheets in manufacturing processes, as well as soft tissues, including muscles and skin, in robotic assisted medical operations. Despite this fact, only a few efforts have been made in investigating the tracking of deformable (or non-rigid) objects during manipulation. Hence, deformable object tracking by a robotic system still offers an important challenge to the community. The main problem of tracking these objects is that their shape changes during the manipulation, and then both the location of the object and its deformation have to be controlled.

The object tracking problem consists in reconstructing the trajectory of objects along the sequence of images. It is considered a basic problem in many computer vision applications and it is inherently difficult, especially when applied to real world conditions, where unstructured forms are considered for tracking, real time responses are required for adapting the robot movements in time, computational capabilities are limited to on-board units and where problems of brightness and non-stationary background can affect the performance of the elaborating system. Moreover, in case of non-rigid objects, the task of dynamic tracking becomes even more challenging. The state of the art of tracking deformable objects is still rather far from the real applicability within robotic applications. Recent projects have been proposed, in the last few years, trying to cope with this kind of problem. The recent RODYMAN[1] project proposes, for example, the development of a unified framework for dynamic dexterous manipulation control, considering mobile platforms able to manipulate non-prehensile non-rigid objects, trying to fill the gap in the current state of the art. In order to achieve dexterous manipulation abilities, a fundamental step is to provide robots with the ability to efficiently track the objects to be manipulated. Both dynamic object tracking and manipulation become, in fact, the most complex categories of robotic tasks, which, if solved, could increase the opportunities for a wide adoption of robots within human co-habited environments.

The tracking capabilities we want to achieve have to be very flexible to identify all the different shape instances, but, at the same time, highly specific, in order to not misclassify the tracked target. In general, Neural Networks can be exploited in these contexts, since they can express highly nonlinear decision surfaces, and thus they are mostly used to appropriately classify objects presenting a high degree of shape variation. In this paper, we propose a particular Weightless Neural Network (WNN) approach, a WiSARD–based system, used as shape detector for tracking deformable objects during manipulation. Classifiers based on WNNs have been shown to be effective, very flexible and easy to use [1]. Moreover, this particular WNN system has the property of being noise tolerance and capable of learning step–by–step the new appearance of the moving object on a dynamic background without needing a model of the object to track. In the proposed approach there is no distinction between the recognition process of an object and the tracking, since the latter is treated as a classification process. In this framework, the image filtering process, used to train classifiers, is a fundamental step for the correct operation of the tracker. In this paper, we evaluate the tracking performance obtained with the use of four different filters to provide the foreground extraction, in the scenario of pizza making.

## II. RELATED WORKS

A wide class of approaches in object tracking explicitly assumes a model of the tracked objects. Usually, these methods provide robust solutions (e.g., they can cope with partial occlusions), but require the use of considerable computational resources in the object recognition process. Kalman filters and Gaussian distributions are often used as

[1]Mariacarla Staffa, Silvia Rossi and Bruno Siciliano are with Department of Electrical Engineering and Information Technology, University of Naples Federico II, Via Claudio, 21, 80125, Naples, Italy {mariacarla.staffa, silvia.rossi, bruno.siciliano}@unina.it

[2]Maurizio Giordano is with the Istituto di Calcolo e Reti ad Alte Prestazioni, CNR, Via P. Castellino, Napoli, Italy maurizio.giordano@cnr.it

[3]Massimo De Gregorio is with the Istituto di Cibernetica "Eduardo Caianiello", CNR, Via Campi Flegrei 34, Pozzuoli, Italy massimo.degregorio@cnr.it

[1]http://www.rodyman.eu/

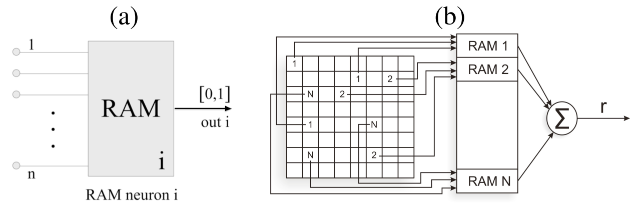

Fig. 1. RAM-neuron (a) and a WiSARD discriminator (b).

approaches to track the individual modeled objects [2]. More recently, particle filters have been introduced to estimate non-Gaussian, non-linear dynamic processes [3]. However, most of these successful approaches focused their efforts on the recognition, pose determination and tracking of 3D rigid objects, while only few approaches considered deformable objects. In tracking deformable objects, some attempts have been proposed in order to have a flexible model to track the objects [4] and to represent the elasticity and deformation characteristics during the physical interaction. In some of these cases, the authors consider a pre-defined initial shape to be manipulated into a deformable contour model. In [5] authors use physical, although very general, models and a set of constraints on the model to estimate the state of objects (e.g., a rope or a flag). In particular, they assume that a set of $K$ points with 3D coordinates can be associated with the object and completely define its current configuration.

Our proposed solution is more in line with features or appearance–based approaches, as the method of [6], where non-rigid objects are tracked based on visual features such as color and/or texture, object contours, regions of interest. In [6], a statistical distribution is used to characterize the object of interest. The approach is based on mean shift iterations to find the target candidate that is the most similar to a given target model. In [7] authors presented a feature method for tracking both rigid and deformable objects, and human beings shapes in video sequences. The proposed tracking algorithm segments object regions based on motion and extracts some feature points to track by using optical flow with online training. Conversely, in our work we track the complete appearance of the object. In [8] authors use a train of discriminative classifiers in an online manner to separate the object from the background with a model, which evolves during the tracking process as the appearance of the object changes. Finally, in [9] the estimation of non–rigid object is obtained by means of energy minimization and graph cuts.

## III. A WiSARD approach for Deformable Objects

WiSARD systems are a particular type of WNNs, that can be developed directly on reprogrammable hardware [1]. A WiSARD is composed of a set of classifiers, called *discriminators*, each one assigned to learn binary patterns belonging to a particular category. Therefore, a WiSARD has as many discriminators as the number of categories it should be able to distinguish. Each discriminator consists of a set of RAM nodes, which store the information of occurrences of binary patterns during the learning stage. All RAMs of a discriminator have the same size, i.e. the same number of locations. Given a binary pattern of size $s$, the so–called *retina*, it can be classified by a set of WiSARD discriminators, each one having $X$ RAMs with $2^n$ locations such that $s = X \times n$. Since each RAM location is uniquely addressed by an $n$-tuple of bits, the input pattern can be partitioned into a set of $n$-tuples, each one addressing one location in the set of RAMs. $n$-tuples are pseudo-randomly selected and biunivocally mapped to RAMs (see right part of Figure 1), in a way that the input binary pattern is completely covered. The mapping is fixed and it is the same for all discriminators. The WiSARD training phase is described as follows:

1) *Inizialization* - all RAMs locations for each discriminator are set to 0.
2) *Training Set Selection* - a training set, formed by binary patterns (or *training samples*) of the same size, is selected. Each sample of the training set is known to belong to, and thus to represent, only one category.
3) *Training* - for each training sample the discriminator assigned to the belonging category is selected. The psuedo-random mapping is used to extract, from the binary pattern of the sample, the $n$-tuples of bits; each $n$-tuple is a unique address of a RAM location of the discriminator, whose content is set to 1.

Once the training of patterns is completed, if the location of a RAM is 0, then its address (in binary notation) never occurred across all samples in the training set at the corresponding $n$-tuple of bits in the retina; otherwise it occurred at least in one sample.

The WiSARD classification works as follows:

1) *Test Set Selection* - a test set, formed by binary patterns (or *test samples*) of the same size, is selected. We want to know the belonging category of each sample.
2) *Classification* - on each test sample, the psuedo-random mapping is used to extract, from the binary pattern, the $n$-tuples of bits; all addressed contents are read and summed by an adder ($\Sigma$) obtaining a number $r$, the so–called *discriminator response*, which is equal to the number of RAMs that output 1.

It is easy to see that $r$ necessarily reaches the maximum $X$ if the input pattern belongs to the training set. $r$ is equal to 0 if no $n$-tuple of bits in the input pattern appears in the training set. Intermediate values of $r$ express a kind of "similarity measure" of the input pattern with respect to the patterns in the training set. The $\Sigma$ adder enables this network of RAM nodes to exhibit (just like other ANN models based on synaptic weights) generalization and noise tolerance [10].

DRASiW [11] is an extension to the WiSARD model. Instead of having RAM locations set to 1 if accessed under training, they are incremented by 1 at each access. Thus, at the end of the training, RAM contents store the number of occurrences (frequency) of a specific $n$-tuple of bits across training samples. One should notice that the new domain of memory content values (non negative integers) does not induce a different behavior with respect to regularly trained

RAM–discriminators if the $\Sigma$ adder counts the number of addressed memory locations whose content differs from 0. DRASiW allows a mechanism of backward projection then reduction of RAM contents onto the retina that can be used to generate prototypes (i.e., graphical representations) of learned categories, the so–called "mental" images (MIs).

## A. *DRASiW for Object Tracking*

The proposed tracking activity is performed as follows (see Figure 3). At the beginning, the system is fed with an image representing the object (with its initial shape and position) to be followed. The image, as well as the video stream, is preprocessed by a filter as described in Subsection III-B. The filtered image is used to train the DRASiW system. The DRASiW system, we propose, is formed by a set of discriminators, each one looking at different parts of the frame image of the video (see Figure 2). We can distinguish left, right, up and down discriminators, respectively, to classify (i.e., to track) left, right, up and down displacements of the tracked object. Except for the retina (input field) of the central discriminator, all the other retinas are placed all around the initial position. Doing so, each discriminator is identified by its relative coordinates. The displacement of all the retinas forms what is called *prediction window*. In particular, since we consider a prediction window precision of 10 pixels, we will use $21 \times 21 = 441$ discriminators (included the central one). Let $(0,0)$ be the retina coordinates of the central discriminator ($d_{0,0}$) and $h$ and $w$ the size of the prediction window. The whole set of discriminators are labeled as $d_{n,m}$, with $n \in [-w/2, w/2]$ and $m \in [-h/2, h/2]$.

The generic discriminator $d_{i,j}$ is going to be responsible for detecting the object in case its new position is identified by $(i,j)$ in the prediction window. The higher is the response the more probable the object is in that part of the prediction window. Finally, the new position of the central retina will be set to $d_{i,j}$ to track the object. The system always trains itself with the image on the retina of the discriminator that outputs the best response. Hence, in order to avoid RAM memory location saturation, we introduce a forgetting mechanism (bleaching [11]) that allows DRASiW to store in its MI an updated shape of the tracked object. In particular, all the sub-patterns of the new image on the retina are combined with those of the MI (this means increasing their frequencies in the RAM contents). On the other hand, those subpatterns which were not addressed by the image on the retina are decremented $(-1)$. So doing, DRASiW system will always get an updated MI of the object shape it is tracking. Furthermore, with the MI stored during time, we can produce a sort of object shape history, which can be used to extract some information about the cinematic/dynamic model of the object to be manipulated.

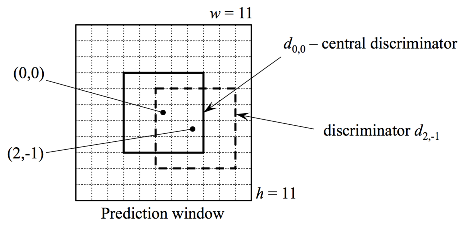

Fig. 2. Prediction window and discriminator retinas.

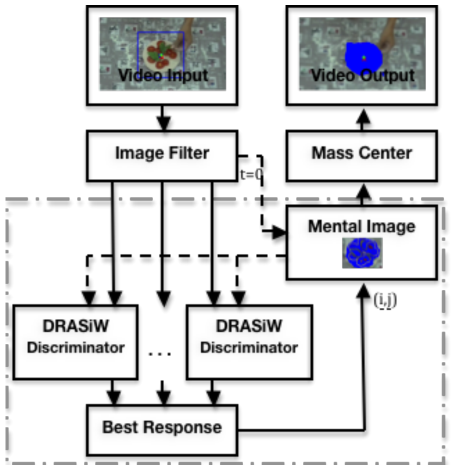

Fig. 3. Overview of the proposed tracking process.

## B. *Filtering the Input*

Before the tracking starts, the object to be tracked is selected by the user by drawing a bounding box (the blue rectangle in Figure 6). Such bounding box, representing the *retina*, can also be evaluated by implementing a motion–based segmentation. In order to transform the input video frame image in a suitable format for DRASiW discriminators (i.e., a black and white image) we used the 4 following filters.

*a) GrabCut filter:* it applies the well–known GrabCut method [12] to separate the foreground and background regions of the image enclosed in the retina box. The method evaluates the color distribution of the target object and that of the background in the retina using a Gaussian mixture model.

*b) Gregor filter:* it is a novel filtering technique we developed for the purpose (see Figure 4). First a *focus area* covering the target object is identified. The focus area is a box (the red rectangle) with the same center of the retina and with size equal to a $\alpha\%$ fraction of the retina size. The filter uses the focus area to compute the histogram representing the pixel color (HSV) frequencies in the focus. The histogram is then ordered and cut to leave only the more frequent pixel colors representing the $\beta$ percentage of the color frequency histogram. The selected colors are used to mark pixels in the bounding box as foreground, while the rest are marked as background (binarization).

*c) Kalman filter:* it applies the well–known Kalman method [13] to label as foreground the pixels in the retina whose RGB channel are within a certain range around a reference pixel color.

*d) Threshold filter:* it uses a dynamic threshold computed at runtime according to the Otsu algorithm clustering-based image threshold method [14] in order to segment foreground/background pixels in the retina.

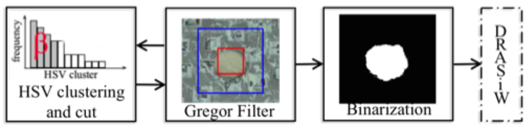

Fig. 4. Gregor filter functioning.

## IV. A Case Study in Pizza Making

As a benchmark to test our approach for tracking deformable object, we adopt the *Pizza Making* case study. Despite its apparent simplicity, this task represents a very challenging test bed, both for object tracking and for robotic manipulation. Figure 5 shows a simulation of the RODYMAN robot in manipulating a pizza. Pizza is a non-rigid deformable object that can assume whatever shape we want. Hence, it is not possible to define a model for the tracking. In this context, the robot should be able to dynamically identify the pizza dough and robustly track it without prior knowledge. In Figure 6 we show some snapshots from the main considered actions (or sub-tasks) for making pizza: the manipulation, the stretching until the pizza is completely extended and the seasoning (we do not consider the tossing in this first testbed). During all these phases temporary occlusions can occur due to the manipulation, while stationary occlusions occur during the seasoning process. We test the performance of different filters in the proposed DRASiW-based tracking system, in order to evaluate the ability of the MI to keep a 2D representation of the pizza dough up-to-date and of tracking such moving deformable object in time.

### A. *Experimental Results*

For the experimentation, a real pizza dough was used. In order to evaluate the performance of the proposed method in *Pizza Making*, we individuated five different phases (see Figure 7) as follows:

a) Translation: the pizza is in the hands of the user who makes horizontal, vertical and circular movements;
b) Dynamic Background: the pizza is in one hand of the user who moves the other changing the background;
c) Manipulation: the user manipulates the pizza;
d) Extension: the user modifies the appearance of the pizza dough in order to reach its final shape;
e) Seasoning: seasoning toppings may occlude the pizza dough that is now ready to be baked.

We evaluated the performance of the DRASiW considering both its ability to track the shape of the pizza and to follow the position of the object in time. As shown in Figure 7, the MI of the network keeps track of the shape of the pizza during the interaction. In particular, the first two snapshots in figure show how the MI model of the pizza finely overlaps the target object during both pizza translation and passing from hand to hand. While in translation the pizza silhouette changes slightly, in the latter case the background suddenly changes and we can notice how well the MI model

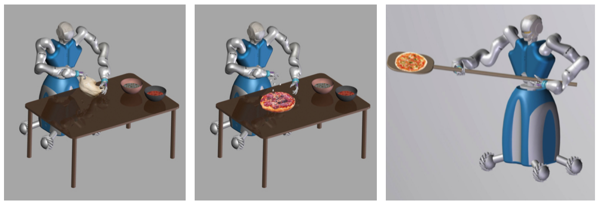

Fig. 5. Sketches of the RODYMAN robot manipulating a pizza.

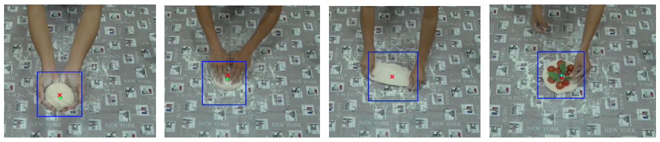

Fig. 6. Sketches of the main actions for making pizza.

succeeds to separate the changeable background from the target. In the third snapshot taken during manipulation, the target object is often and suddenly occluded by fingers: as we can see, the mental model quite well adapts to the frequent changes of the visible target, although some occluding parts (see right hand fingers in figure) are late to disappear in the model due to the chosen DRASiW's forgetting latency. This is not really a big issue since most of the MI model overlaps the target and in successive frames it will better adapt to the visible object, by gradually degrading the part of the model corresopnding to the occluded parts. In the fourth snapshot, we can see how, as soon as the user starts to enlarge the shape of the pizza dough, the MI model consequently adapts to the larger visible silhoutette. In case of hole generation during extension, holes will be initially covered my the mental model, although as soon as the forgetting mechanism starts having visible effects, corresponding holes will appear in the MI thus detecting those parts as background. In order to evaluate the tracking abilities, the system evaluates the center of mass on the MI (i.e., the red cross in Figure 7). Finally, we keep track of the position of the central retina classifier. Recall that, frame by frame, the position of the central classifier is updated according to the classifiers with the best response at the previous frame.

In Table I we reported the performance, obtained by adopting different segmentation filters, evaluated on the five subtasks. For each sequence, we evaluated common performance metrics for quantitative comparison. In particular, we considered the Tracking Error (TE), also referred to as central-pixel error, evaluated as the Euclidean distance (in pixels) between the MI center of mass, as generated by the DRASiW network, and the center of the real object from a Ground Truth (GT) evaluation (i.e., the green cross in Figure 7). Specifically, the GT is computed by visually evaluating, frame by frame, the center of mass and selecting the correspondent points over the videos (with resolution $854 \times 480$). Then, we analyzed the Success Rate (SR) that measures, in percentage, for how long a tracker is able to maintain the target object within its field of view (i.e, its retina) with respect to the entire duration of the video. In

| | | Gregor | | | Grabcut | | | Threshold | | | Kalman | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sub-task | # frames | TE (px) | TE≤45 (px) | SR(%) | TE (px) | TE≤45 (px) | SR(%) | TE (px) | TE≤45 (px) | SR(%) | TE (px) | TE≤45 (px) | SR(%) |
| Tran | 630 | $8 \pm 7$ | $8 \pm 7$ | 100 | $9 \pm 7$ | $7 \pm 7$ | 38 | $16 \pm 12$ | $10 \pm 12$ | 29 | $9 \pm 7$ | $6 \pm 6$ | 38 |
| Back | 151 | $5 \pm 3$ | $5 \pm 3$ | 100 | $24 \pm 6$ | $24 \pm 6$ | 100 | $25 \pm 39$ | $13 \pm 11$ | 90 | $5 \pm 4$ | $5 \pm 4$ | 100 |
| Man | 854 | $53 \pm 23$ | $5 \pm 3$ | 16 | $8 \pm 4$ | $8 \pm 4$ | 100 | $20 \pm 12$ | $15 \pm 9$ | 74 | $41 \pm 32$ | $10 \pm 7$ | 39 |
| Ext | 586 | $62 \pm 28$ | $26 \pm 7$ | 11 | $124 \pm 45$ | $11 \pm 9$ | 10 | $43 \pm 18$ | $9 \pm 10$ | 11 | $212 \pm 54$ | $17 \pm 13$ | 1 |
| Seas | 482 | $20 \pm 8$ | $19 \pm 8$ | 99 | $4 \pm 6$ | $3 \pm 5$ | 100 | $25 \pm 10$ | $25 \pm 10$ | 99 | $66 \pm 52$ | $5 \pm 7$ | 17 |

TABLE I

AVERAGE NUMBER OF FRAMES, TRACKING ERROR AND SUCCESS RATE MEASURES EVALUATED ON THE 5 SUB-TASKS.

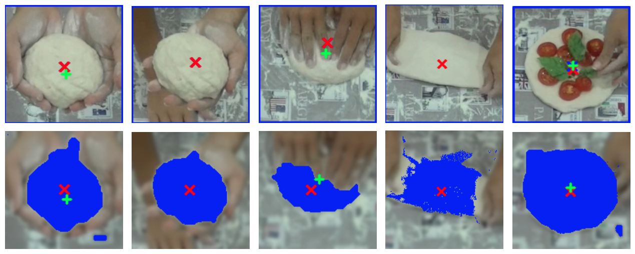

Fig. 7. Sketches of the DRASiW tracking results in a sequence (from left to right). The first row shows snapshots from the original video (one for each task), while figures on the second row are the relative DRASiW mental images. The green cross represents the Ground Truth (GT), while the red one identifies the mass center of the Mental Image (MI).

particular, we computed the success rate by considering the number of frames a system is able to track the target object, with a tracking error value of less than $k$ pixels, until the first failure. $k$ is computed as the sum between the half of the retina dimension ($35px$ in this setting) and the prediction window precision ($10px$). We also showed the TE until the first failure and we reported the dimensions of each video in terms of number of frames. For what concerns the frame rate, its average value on single tasks is about $10fps$. It still remains open the possibility to speedup the software by adopting some optimization and parallelization techniques. Our aim, here, is to identify the best segmentation solution for coping with the different issues shown in the considered 5 sequences. In case of translational movements, the best segmentation filter applied to the DRASiW is the Gregor filter. The DRASiW system endowed with the Gregor filter, in fact, is able to track the target object in all the directions with a very low TE ($8 \pm 7px$) and to reach the 100% of true classifications (SR) of the target object. The other filters do not achieve the same performance since they are not able to maintain the attentional retina focused on the object. Here, the main challenge is to separate the object from the background that is characterized by a colorful tablecloth with colors similar both to pizza dough and to the hands. As shown in Table I, the problem of changing background is well treated with all filters, which, once combined with the DRASiW tracker, are able to reach a very high SR (100% for Gregor, Grabcut and Kalman, and 90% for the Threshold filter) with an associated low TE (of about $5px$ with both Gregor and Kalman filters). In this setting, DRASiW classifiers are well trained on the static object pattern. However, Grabcut and Threshold filter are more sensitive in the segmentation process of a moving background.

Differently from previous cases, during the phase of manipulation of the pizza dough, the Gregor filter is not appropriate, producing a large TE and a very low SR. This is probably due to the great number of occlusions that occur during manipulation, which, by entering within the focus area of the Gregor filter box, changed the priority colors. The best result, during the manipulation process, is achieved by the GrabCut, both in terms of TE and SR. More similar to this task, in term of TE, it is the seasoning ($20px$ with the Gregor filter, $25px$ if using the Threshold filter, and $66px$ for the Kalman), where the best performance is achieved by the GrabCut. Here, the error is not due to the movements (in fact, during this phase, the position of pizza is fixed) but to the occlusions that can occur during the task. Furthermore, while in the manipulation phase, we evaluated quick occlusions of the pizza made by user's hands, during the seasoning there are permanent occlusions (for example, tomatoes and basil) that may cause a little modification in the mass center. Considering the manipulation, we note that when the pizza shape is fixed the system reaches a low tracking error of about $8px$, while in the case of extension, some occlusions occur (e.g., the human hands occlude the pizza during manipulation), the perceived 2D pizza shape quickly changes by moving the dough from the horizontal plane to the vertical one, and so the tracking error increases a little bit while the SR drastically decreases. Hence, the hardest problem to cope with, is the managing of the extension. This task involves, indeed, big movements of the mass center, causing bigger TE values with respect to the other sequences.

In Figure 8 we showed the trend of the Tracking Errors evaluated on the different phases of the *pizza making* task by using the different filters. By analyzing the plots it is possible to observe the different behaviors of the filters during the tracking process individuating frames of correct tracking and phases in which the target is lost.

## V. CONCLUSIONS AND FUTURE WORKS

The main contribution of this paper is to propose a methodology for object tracking in order to achieve both flexibility and robustness in tracking non-rigid deformable objects without prior-model of them. The proposed tracking is appropriate for a large variety of deformable objects with different color/texture patterns, being robust to partial occlusions, and drastic shape modification. The on–line training characteristic of the proposed DRASiW neural network

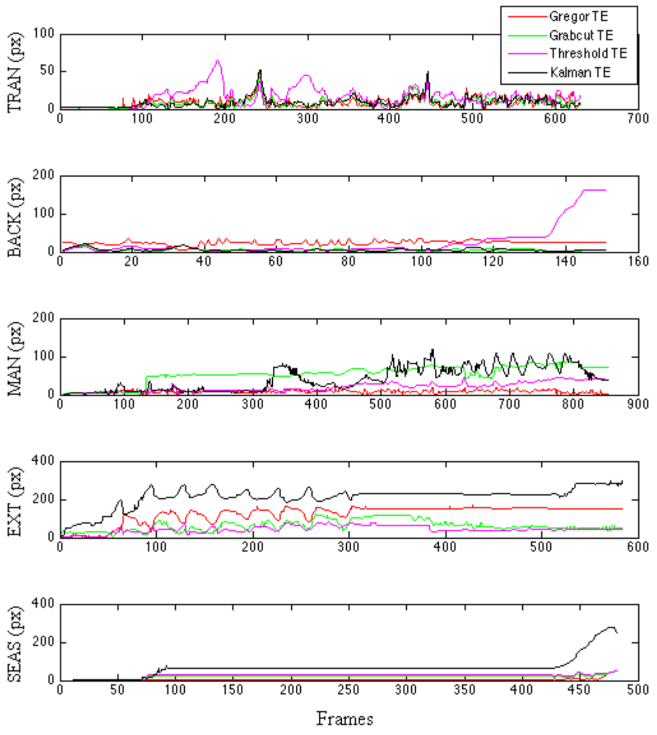

Fig. 8. Plots showing the trend of the Tracking Error obtained by using different filters on the five subsequences of the entire video.

model allows the robot to adapt in real–time to any new situations, such as, a new shape of the object and color and luminosity changes, and so on. The reinforcing behavior of the DRASiW system provides a mechanism that partially copes with occlusion. Moreover, the use of parallel classifiers trained on copies of the object silhouette, which are displaced along XY in the proximity of the target, allows tracking the movements of the objects in the space with an average error that is acceptable and comparable with the ground truth.

From the analysis of the experiments we can make the following concluding remarks. In general, the chosen filters seem to work orthogonally: apart from the dynamic background case in which the target is followed until the end of the video independently from the used filter, in the other cases one filter outperforms the others. For example, the translation case is well solved by the Gregor filter, while the Grabcut filter works better in the manipulation and seasoning cases. The extension case proved to be the killing situation for our tracker. Indeed, none of the filter+DRASiW combinations was able to follow the target for a long time. A motivation of this failure could be the impossibility, at the time being, to adapt and enlarge the bounding box size, which is the "focus window" of the DRASiW on the video scene. When the pizza dough is extended, it exceeds the box size, thus reducing the DRASiW possibility to model and then recognize its entire silhouette. By considering the tracking error measurements, we can say that in all cases, but the extension one, the error is limited, in particular, if we consider its average value over the time the target is followed by the tracker.

As many methods that deal with online learning of the object shape, also our approach cannot completely solve the problems of permanent occlusions. Moreover, if the system does not correctly track the object and starts to lose it, it will inevitably start learning the appearance of other objects in the attention window. So, when the mental model becomes sufficiently far from reality, it usually does not recover. Hence, as future work, in order to improve the performance of the proposed DRASiW–based visual servoing system, we plan to investigate the adoption of a dead reckoning strategy to anticipate the object current/next position by using its previously determined positions, and so, also to dynamically displace classifiers on salient and more probable areas, improving even more the frame rate, and so, real time tracking ability. Finally, the next step will be the extension of the proposed method from this 2D approach to 3D and the introduction of inferences about some characteristics of the object in order to fill the gap in manipulation issues.

## ACKNOWLEDGMENT

This work has been partially funded by the European Commission's 7th Framework Programmes as part of the project SAPHARI under grant 287513, and RODYMAN under ERC-grant agreement no. 320992.

## REFERENCES

[1] I. Aleksander, W. V. Thomas, and P. A. Bowden, "WISARD a radical step forward in image recognition," *Sensor Review*, vol. 4, pp. 120–124, 1984.

[2] N. Gordon, "A hybrid bootstrap filter for target tracking in clutter," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 33, no. 1, pp. 353–358, 1997.

[3] M. Pitt and N. Shephard, "Filtering via simulation: auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, p. 446, 1999.

[4] C. Nastar and N. Ayache, "Fast segmentation, tracking, and analysis of deformable objects," in *Proceedings of the Fourth International Conference on Computer Vision, 1993*, May 1993, pp. 275–279.

[5] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds." in *ICRA*. IEEE, 2013, pp. 1130–1137.

[6] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, IEEE Conf. on*, vol. 2, 2000, pp. 142–149.

[7] J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and M. Abidi, "Optical flow-based real-time object tracking using non-prior training active feature model," *Real-Time Imaging*, vol. 11, no. 3, pp. 204 – 218, 2005.

[8] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 33, no. 8, pp. 1619–1632, 2011.

[9] B. Willimon, I. D. Walker, and S. Birchfield, "3d non-rigid deformable surface estimation without feature correspondence," in *ICRA*. IEEE, 2013, pp. 646–651.

[10] I. Aleksander and H. Morton, *An introduction to neural computing*. London: Chapman & Hall, 1990.

[11] B. P. A. Grieco, P. M. V. Lima, M. De Gregorio, and F. M. G. França, "Producing pattern examples from "mental" images," *Neurocomputing*, vol. 73, no. 7-9, pp. 1057–1064, Mar. 2010.

[12] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, Aug. 2004.

[13] G. Welch and G. Bishop, "An introduction to the kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.

[14] Otsu, "A threshold selection method from gray-level histograms," *Systems, Man and Cybernetics, IEEE Trans. on*, vol. 9, no. 1, pp. 62–66, Jan 1979.