Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Active visual features based on events to guide robot manipulators in tracking tasks

P. Gil*, G.J. García, C.M. Mateo*, F. Torres****

**Computer Science Research Institute, University of Alicante, San Vicente del Raspeig, Spain (Tel: +34-965-903400; e-mail: {pablo.gil,cm.mateo}@ua.es)*
*** Physics, Systems Engineering and Signal Theory Department, University of Alicante, San Vicente del Raspeig, Spain (Tel: +34-965-903400; e-mail:{gjgg, fernando.torres}@ua.es)*

Abstract: Traditional visual servoing systems do not deal with the topic of moving objects tracking. When these systems are employed to track a moving object, depending on the object velocity, visual features can go out of the image, causing the fail of the tracking task. This occurs specially when the object and the robot are both stopped and then the object starts the movement. In this work, we have employed a retina camera based on Address Event Representation (AER) in order to use events as input in the visual servoing system. The events launched by the camera indicate a pixel movement. Event visual information is processed only at the moment it occurs, reducing the response time of visual servoing systems when they are used to track moving objects.

Keywords: AER, asynchronous vision sensor, visual servoing, robot vision systems, spike events, image motion.

1. INTRODUCTION

One of the most valuable advantages of an image-based visual servoing scheme is its robustness to calibration errors, Espiau (1994). These systems have been proved to be robust to great errors in the camera intrinsic parameters estimation, but also to calibration errors between camera and robot 3D Cartesian space, Malis et al. (2010). Thus, these systems have been widely used in the literature for guiding a robot manipulator in unstructured workspaces, Chaumette et al. (2006). Nevertheless, there is a main parameter of these systems, which is often simplified: visual features. This simplification detracts image-based visual servoing from dealing with unstructured environments.

In general, in classical visual servoing systems, features can be segments, points of edge, etc. based on gradient operations, Marchand et al. (2005). These features identify keypoints of the object in the scene and allow us to guide robots and generate trajectories between an initial pose and a desired pose. The object is usually a pattern with fiducial marks, Fiala (2010). Some efforts have been made to deal with this problem in the literature. By the end of 20th century, Janabi-Sharifi et al. (1997), proposed an automatic visual features selection method to increase the possibilities of the system. More recently, works like, Chaumette (2004) or, Kragic et al. (2001) tried to obtain different visual features like image moments or cue integration, which could be obtained directly from the object to be tracked. Nowadays, the topic continues being studied in the literature, Gratal et al. (2012). In this last work, Gratal et al. described a visual servoing scheme to track unknown objects by using a 3D model tracking based on virtual visual servoing.

In this paper, we implement a classical image-based visual servoing system behaviour using Address Event Representation (AER) technology, Lichsteiner et al. (2008). In this case, we have changed the way in which the visual features are obtained. The used camera sensor is a retina camera DVS128, and its behaviour is very different to classical cameras based on CMOS or CCD sensor. With DVS128, features are not obtained from gradients in an image. They are computed from pixel-event. Then, a clustering process is carried out to obtain active visual features from pixel-events. The data stream is reduced by using the event-based control theory, Aström (2008). An event is something that occurs requiring some response. The basic idea is to communicate, compute, or control only when something significant occurs in the system. Event-based control has been applied to many fields. Sanchez et al. (2009) used events to control the level of a water tank. García et al (2013) proposed a visual controller based on events to avoid loss of features in the image.

In addition, in indirect visual servoing systems, the joint pose of the robot is not controlled. A classical image-based visual servoing scheme controls the robot end-effector's movement from the computation of the visual difference between any position of the features and the final one. Traditionally, visual servoing systems do not deal with moving objects, and only the robot is moved in the scene to achieve the desired pose.

Visual servoing was born as a positioning technique, Espiau et al. (1992). An eye-in-hand technique is referred as a specific camera configuration where the camera is mounted at the robot's end-effector. The knowledge of the fixed camera pose with respect to the robot's end-effector allows us to talk about the end-effector velocity instead of the camera velocity without loss of generality. The classical image-based visual servoing control law computes the camera velocity to minimize the error between the current position and the desired one, Chaumette et al. (2006). When the reference object (from which visual features are computed) is moving in the scene, the controller must take this movement into account. A new term is added to the control law. This new term must be estimated. One of the most used techniques to estimate this term is a Kalman filter, as it can be seen in a recent work of Liu et al. (2012). In early works, such as Cretual et al. (2001), authors proposed a method to do visual servoing based on movement. Firstly, they used geometric features retrieved by integration motion without changing the classical control laws. Secondly, they used a motion field to model features in image sequence. In those works, the amount of information to be processed was high due to the very dense map of features generated from motion field, if it is compared with classical visual systems that use only four features. To attach this problem and solve it, we have used a retina camera based on AER sensor. The pixels of an image are captured from AER sensor and they are only represented when those pixels have been affected by luminosity changes. If the pixel suffers changes, the pixel represents a keypoint of a moving object. The key is the small data volume generated by AER sensors in contrast to classic cameras which represents the information of all pixels in the sensor.

The main goal of this work is to design a visual servoing system to guide a robot manipulator by tracking a moving object without fiducial markers, reducing the complexity of high-level computer vision algorithms in order to work in a real-time scope. After an initial step where the object starts its movement and the robot is stopped, this proposed system switches to a classical image-based visual servoing scheme to perform the tracking.

The paper is structured as follows. Section 2 shows the basics of AER technology and an approach to select events in the frames. In Section 3, a strategy to determine how they can be used to obtain active visual features is presented along with the visual control system to track moving objects from these computed active visual features; Section 4 presents different experiments to validate the proposal; and finally, in Section 5 the main conclusions are discussed.

2. EVENT-FEATURES FOR ROBOT CONTROL

2.1 Event Representation System: AER technology

AER sensor generates digital images composed of 128 rows and 128 columns, so it has 16384 candidates to pixel-events. The value of each pixel is computed from the changes in luminosity as a brightness derivative in time. The changes in luminosity can be obtained when an object is moving in front of retina camera whenever the ambient light is not varying. In DVS128, two encoders (1 for rows and 1 for columns) are used to generate two packets composed of a code of 7 bits that will be transmitted. The code of each packet from an encoder represents the position of a pixel in a row and a column. Each packet from each encoder is sent using a parallel bus composed of 7 lines of communication to send all code simultaneously, a bit for each line. Nevertheless, AER technology in Dynamic Vision Sensors (DVS) only sends the code (the position of pixels) when the brightness of a pixel has changed in time. But also, the retina camera never transmits intensity of the pixels (value of brightness [0-255]), only the position of pixel which changes. The values of brightness cannot be obtained because they have been codified using Pulse Frequency Modulation (PFM).

AER works in three steps. Firstly, the system counts the signal spikes. These spikes are used to determine the luminosity changes when the movement occurs. Secondly, the spike frequency is saved in a table in memory. The position index of the table represents the pixel position, row and column. Thus, the number of luminosity changes is associated to a pixel. Thereby, the retina camera digitizes the number of luminosity changes and then the pixel represents the number of times that a brightness changes and not the brightness values. Each pixel represents a spatial derivative in time of the intensity obtained from the moving objects in scene. This fact implies that only the information of moving objects can be seen in the resulting image.

2.2 Classification and selection of events

AER works in real-time scope. It processes the information captured in a continuous way. The information flow cannot be stopped. The implementation requires processing or discarding the acquired information. Thus, the computed events, their classifications and the visual features computed from events can be automatically self-corrected over time. Fig. 1 shows the approach to estimate visual features from motion represented by pixel-events computed from AER.

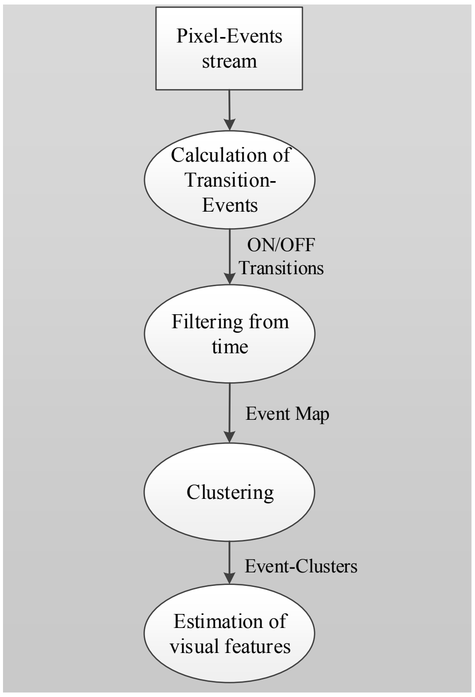

Fig. 1. Our method to classify events and estimate visual features

In Fig. 2, the dots are a series of pixel-events with a polarity either positive (labelled as 'ON') or negative (labelled as 'OFF'). The polarity is positive when there is a positive change in brightness and, thus, a negative change in the contrast is detected.

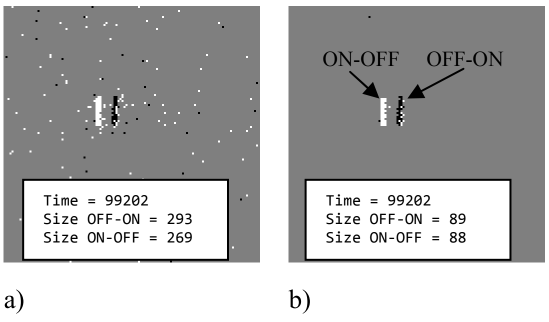

Fig. 2. a) Original AER image (without filter). b) Filtered image of a moving scene.

Each pixel-event is represented by a tuple *(t_k, p_k, x_k, y_k)* where the scalar t_k is the timestamp when the event is generated and it is measured in μsec. The value p_k is the polarity and it is '1' when is ON and '0' when is OFF. The coordinate x_k, y_k represents the position of the pixel-event in the frame. The index *k=1…n* where n is the number of pixel-events detected in the frame.

In each frame, pixel-events whose activation is decorrelated with the scene can appear. Sometimes, this variability can be defined as an unpredictable noise. This random noise occurs due to the silicon substrate characteristics of the circuits of the hardware device, DVS128. This noise is attenuated using a median filter.

In our approach, an idea similar to Censi, et al. (2013) has been used. Thus, we look for both ON-OFF and OFF-ON transitions from a frame sequence. Each transition is modelled as *(τ_t, Δt, Δp, x_k, y_k)* where τ_t is the instant of time at which a transition is detected, the time interval $\Delta t=t_k-t_{k-j}$ is the transition time and it represents the elapsed time to generate a transition from the generation of a pixel-event until the instant when this pixel-event changes its polarity and Δp identifies the type of two transitions. The transition history represents the state of a particular pixel-event over time (Fig. 3).

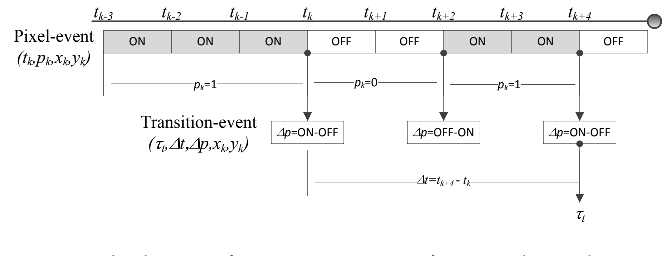

Fig. 3.Calculation of transition-events from each pixel-events stream

In order to classify events, our method uses the event transition and its associated time. Thus, a high transition time indicates a pixel in the frame with slow movements. Further a low transition time indicates that the pixel is moving quickly. Then, our method filters each frame to show only the transitions selected according to the speed of movement represented in a frame sequence. Hence, we can easily find active regions, which represent movement in a frame from a real dynamic scene.

$$e_k(\tau_t, \Delta t, \Delta p, x_k, y_k) = \begin{cases} 1, & \tau_1 < \tau_t < \tau_2 \\ 0 & otherwise \end{cases} \quad (1)$$

In our case, two thresholds are used to detect the type of movement. Using (1), these two thresholds allow us to filter high speed if the transition time is low, and slow speed if the transition time is high or a speed ranging between both thresholds. Thereby, for transition *(t_k, Δt, Δp, x_k, y_k)* the event is labelled according to (1).

The experiment in Fig. 4 and 5 consists of an observation of two objects moving at a constant lineal velocity. The surface of the objects is homogeneous (they have the same texture and colour) but they are moved at different lineal velocities and different trajectories in a plane. The background is a black surface and the objects are two autonomous moving robots programmed from open code. In this experiment, the detection of the objects is computed from different known speeds obtained by driving the DC motor that moves the moving robots.

Firstly, Fig. 4 shows the ability of the proposed filter to analyse the quantity of movement in real-time using the continuous train of pixel-events from the camera. Fig. 4a shows the scene without filtering. Fig. 4b and Fig. 4c show the detection of pixel-events, which represent the surfaces of each mobile robot at high velocity and at low velocity, respectively. The objects are computed by accumulating pixel-events for a specific duration. This duration is dependent on the movement velocity. The fast movement provides the best estimation of the object position due to a dense and accurate events map.

In order to cluster the event-image, a map based on 3-D array is computed. The map is used to save references among groups having similar timestamp and spatial closeness. Thereby, *C={C_1,…,C_m}* are the groups of events where *n* represents the number of them. The number of groups detected is dependent on the thresholds used in (1). So, they are dependent on the timestamp when they are detected. In our case, *m=3* because two thresholds were used. These clusters represent the background (objects without movement), foreground (objects which are moving very quickly in front of the camera) and other objects that are moving slowly far away from the camera. Each cluster, C_i, is defined as:

$$C_i(t_i, x_i, y_i) = \begin{cases} t_i = \sum_{e \in C_i} \tau_t / n_i \\ (x_i, y_i) = \sum_{e \in C_i} (x, y)_k / n_i \end{cases} \quad (2)$$

$$|\tau_t - t_i| < \varepsilon_{temporal} \quad (3)$$

$$\left|(e.x_k - C.x_i)^2 + (e.y_k - C.y_i)^2\right| < r_{dis\tan ce} \quad (4)$$

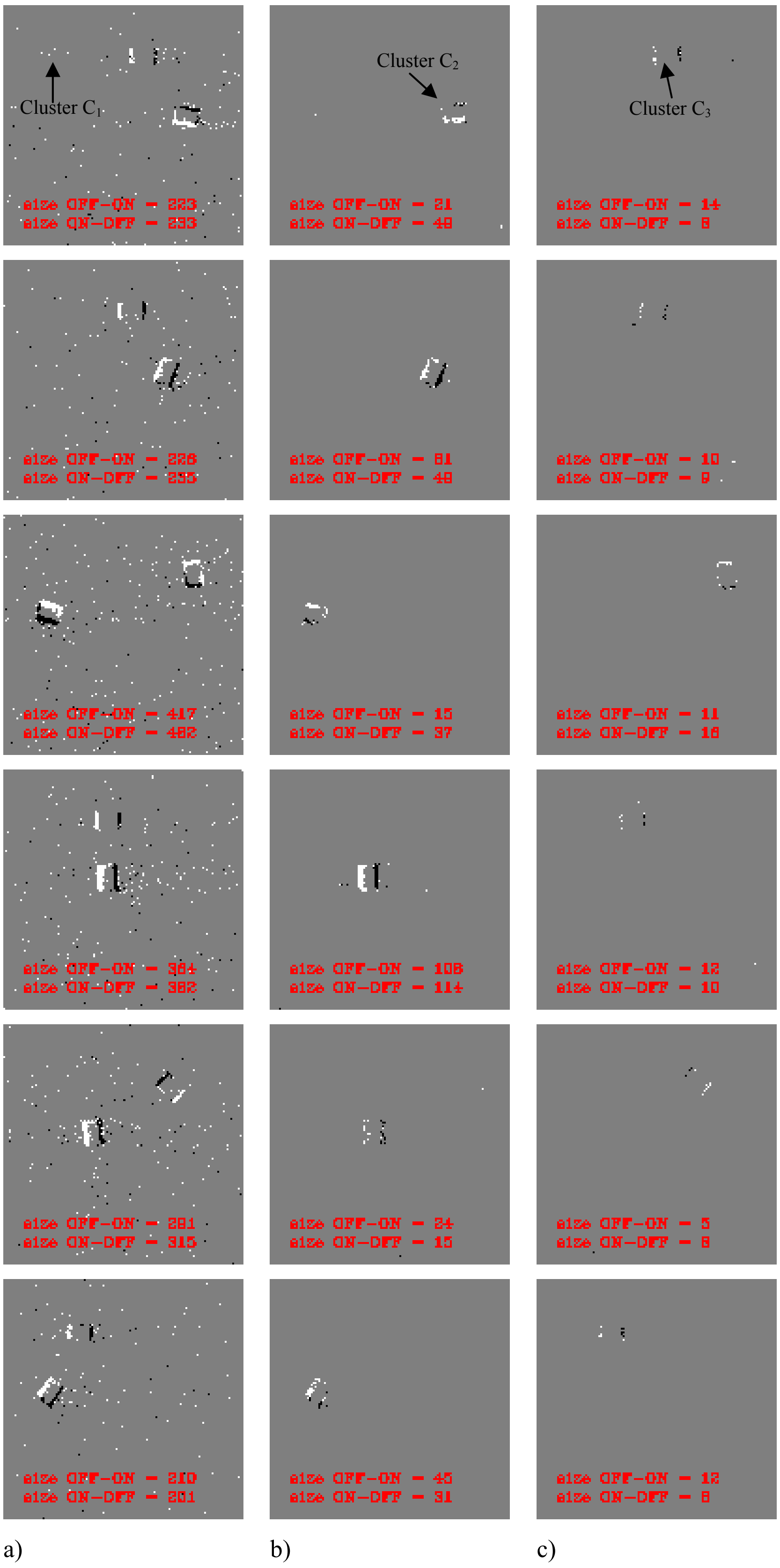

Fig. 4. Events selection from AER images. a) All events. b) High line velocity. c) Low line velocity

In order to obtain each cluster, a comparison between the event's timestamp, τ_t, with the average timestamp of the neighbouring (x_k+dx,y_k+dy) was done. Thus, an event is an item of the cluster supposing it satisfies (3) and (4). That is, if the average timestamp is less than a tolerance value, ε, and if the position of the event is in a neighbouring with radius, r pixels.

In the experiment shown in Fig. 4 and 5, we measure the ability for the filtering of movement from some objects moving at a constant but different velocity. The constraints on the timestamps of the events applied from thresholds allow us to be able to estimate the position of the objects at each instant of time.

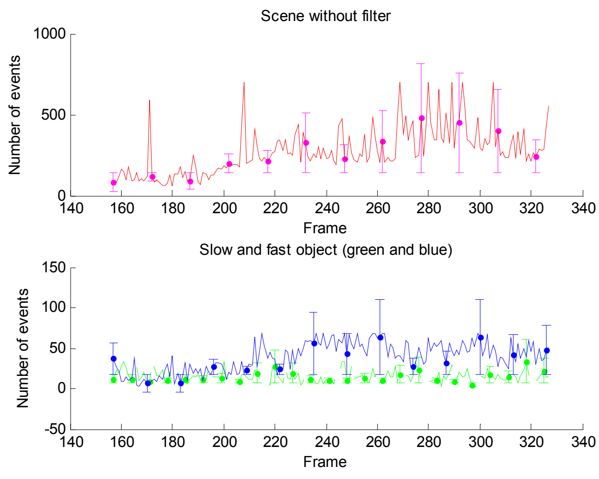

Fig. 5. Events integrated in each object shown in Fig. 4

Fig. 5 shows the number of events for the scene without filter and for each detected event-cluster for a sequence of frames. First and last frames have been erased to measure the number of events. In this case, the detected event-clusters represent the grouped pixel-events for each object with movement. In addition, the average and standard deviation have been drawn in the sequence of movement in order to estimate the observed errors. The position errors are due to variability in the number of events computed from two consecutive times, t_k and t_{k+1}. Although the objects are moved at constant speed, the number of events of its clusters is changed. Those take place because the sensor emits a noise impulse then spontaneous events occur, randomly. In addition, optic characteristics, lens mounted on the sensor, the mobile surface geometry for each object and how these surfaces reflect the light when the position and orientation changes determine this error, too.

3. VISUAL FEATURES AND TRACKING SYSTEM

3.1 Active visual features from selected events

The real-time processing to compute the events limits the quality and precision to obtain features to be used in visual servoing systems. In visual servoing systems, the features must be obtained with robustness, accurately and easily. This way, good result is guaranteed and a success guide of robots can be done using the extracted features. That is achieved because the visual features are computed from fiducial markers in the image. A disadvantage is the high-level image processing slowing down the behaviour of the visual servoing system based on image. To solve that, in this paper, the visual features are previously computed from selected events (Fig. 6). These events must satisfy several criteria among all events detected by the retina camera. The criteria are discussed below.

Each cluster represents two groups of survivor pixel-event transitions such as ON-OFF and OFF-ON. These two groups define a solid object. First step is the computation of the position of the cluster. The position is met by measuring the pixel-event distribution of the two groups. This objective is achieved by computing the median parameter in each case. The median provides better detection that the mean parameter because it measures the central tendency. The median is not influenced by noise in the cluster. It is very much robust. The noise in the cluster can be identified by few extreme values or bad definition in the contours of groups. The Cartesian coordinates of these two means define the visual features as two points in each frame.

In addition, the line segment used to connect these points determines the major axis of the object, and the angle of this axis defines the orientation of the object. In order to obtain two additional points as visual features, the middle point of the major axis is computed. Then a new line segment which is at a right angle with the major axis is computed from the intersection in that point. The two additional features are two virtual points located at the same distance from the middle point. Therefore, the visual points are the endpoints of the two axes (Fig. 6).

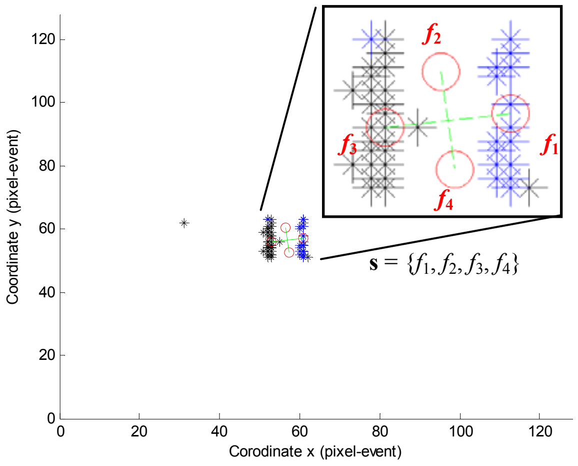

Fig. 6. Computation of Active Visual features. OFF-ON and ON-OFF events are represented by blue and black asterisks.

3.2 Visual servoing

Image-based visual servoing uses only visual features extracted from the acquired images, $\mathbf{s}$, to control the robot. Therefore, these controllers do not need neither a complete 3-D model of the scene nor a perfect camera calibration. The desired visual features, $\mathbf{s}_d$, are obtained from a desired final position of the robot in the scene. Image-based visual servoing controllers minimize the error between any robot position and the goal position by minimizing the error of the visual features computed from the images acquired at each robot position, $\mathbf{e} = (\mathbf{s} - \mathbf{s}_d)$. To minimize exponentially this error **e**, a proportional control law is used:

$$\dot{\mathbf{e}} = -\lambda \mathbf{e} \tag{5}$$

where λ is a proportional gain.

In a basic image-based visual servoing approach, the velocity of the camera, $\mathbf{v}_c$ is the command input for controlling the robot movements. To obtain the control law, the interaction matrix, $\mathbf{L}_s$, must be firstly presented. Chaumette et al. (2006) denoted this concept. Interaction matrix relates the variations of the visual features in the image with the variations of the poses of the camera in the 3D space, i.e. its velocity.

$$\dot{\mathbf{s}} = \mathbf{L}_s \mathbf{v}_c \tag{6}$$

From Equation (5) and Equation (6), the velocity of the camera to exponentially minimize the error in the image is obtained:

$$\mathbf{v}_c = -\lambda \hat{\mathbf{L}}_s^+ (\mathbf{s} - \mathbf{s}_d) \tag{7}$$

where $\hat{\mathbf{L}}_s^+$ is the pseudoinverse of an approximation of the interaction matrix. This camera velocity is then transformed to obtain the velocity to be applied to the end-effector of the robot. To do this, the constant relation between the camera and the end-effector is used in a camera-in-hand configuration (i.e., the camera is mounted at the end-effector of the robot).

In this work, we have developed a tracking system for objects moving in the robot's workspace. The tracking has been divided in three different phases (see Fig. 7).

A first stage permits to obtain an initial time, τ_t, that can be used to filter the object which must been tracked (see Section 2).

The second stage guides the robot manipulator in the tracking of the initial movement of the object (which may produce a fail of a classical visual servoing scheme by losing the visual features in the image). Active visual features (AVF) are used in the control law depicted in (7) in order to perform a quickly native tracking of the initial movement of the object. The proposal allows the system to guide the robot when the object radically changes its velocity under conditions where classical image-based visual servoing can lose its visual features.

Finally, when active visual features are lost, a classical image-based visual servoing continues the tracking of the moving object.

The first stage has been detailed in Section 2. For the second step, active visual features, **s**, are obtained as described in Section 3.1. The desired visual features, $\mathbf{s}_d$, are stored in an off-line step positioning the visual features in the middle of

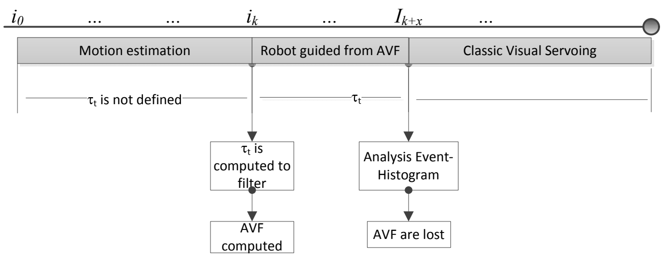

Fig. 7. Scheme of the tracking system proposed

the image. As visual features are obtained from the movement of the object, the set of desired visual features are obtained by moving the object but not the manipulator with the camera. For our experiments, the camera movements have been restricted to a plane parallel to a table where the object is moving. Thus, the movement establishes an area for the object and with this information the desired visual features are stored in the middle of the image plane. When the objects move through the scene, the camera launches the events and a new set of visual features, **s**, is computed. This feedback permits to obtain a new robot's end-effector velocity. The new velocity is restricted to the translational velocities in a parallel plane to the table (X, Y) and a rotational velocity for the Z axis.

In order to switch from first stage to the second one, a histogram as shown in Fig 8 is used. Histogram represents the number of events for each transition time in the image.

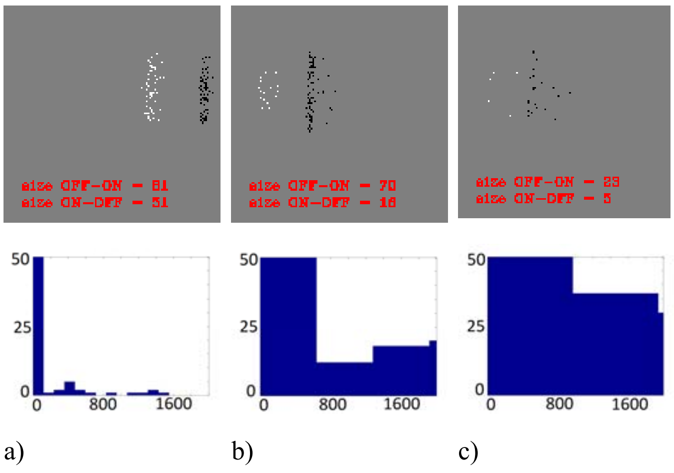

Fig. 8. Histograms show the quality of events. Each histogram represents elements number in y-axis and delta-time in x-axis.

When the mobile robot is moving and robot is stopped, the robot motion generates events whose associated transition is dominant in the event image (Fig. 8a). This time permits to choose, τ_t, to better filter the event image and to obtain the pixel-event belonging to the robot motion. Then AVF can be computed with more accuracy. Thereby, the mobile robot can be tracked by the robot manipulator. However, through the guided process, the manipulator changes the velocity to track the AVF because the features obtained are closer to the desired one at each iteration. Relative speed between manipulator and mobile robot is smaller so the histogram changes as shown in Fig. 8b. When the relative speed approaches to zero, the histogram begins to be homogenous (Fig. 8c) and consequently the AVF are lost. Once the AVF are lost, the system switches to an image-based visual servoing like in equation (7). Visual features are obtained from a conventional camera located at the robot's end-effector. The centre of gravity of the mobile robot is used to obtain the four points in the image, similarly to the technique employed in Section 3.1.

4. RESULTS AND DISCUSSION

The system architecture to test the proposal is based on a robot manipulator of 7 degrees of freedom (a Mitsubishi PA10). Two cameras have been attached at its end-effector: an AER camera (DVS128) and a conventional Gigabit Ethernet camera (Mikrotron MC1324). The first camera is employed to guide the manipulator in the second stage, where we have proposed an algorithm to track a moving object using the events provided by this kind of cameras. The second camera is used to guide the robot in the third phase, where the robot is guided with a classical image-based visual servoing scheme. A mobile robot developed by Goshield has been used to represent the moving object in the scene with different velocities. The test system architecture is depicted in Fig. 9.

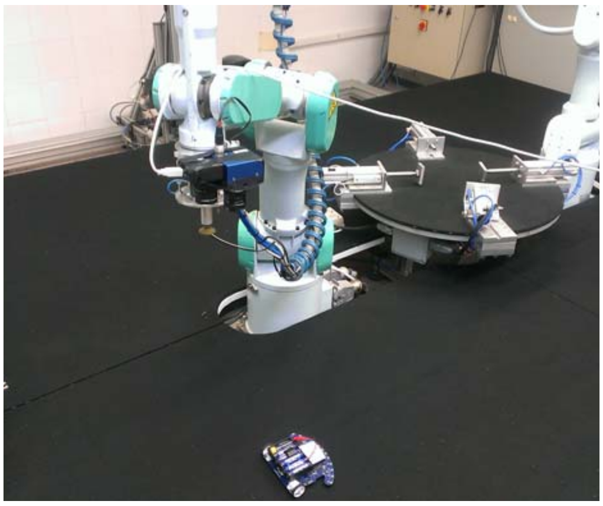

Fig. 9. Test system architecture

Fig. 10 represents an experiment where the mobile robot controlled crosses the manipulator's workspace. In order to have a better comprehension of the active visual features extraction, this figure shows a sequence obtained from the camera located at the manipulator in a parallel plane to the table. The manipulator has been stopped in order to better see the mobile robot movement through the image plane. This experiment shows the evolution of the active visual features through the image. This evolution can be then tracked with the proposed scheme described in Section 3.2.

5. CONCLUSIONS

This paper describes an event-based visual servoing system. Visual features for the robot guidance are obtained without fiducial marks in the scene. Traditional visual servoing use a pattern-object with marks to easily extract visual features using high-level computer vision techniques and to determine its position in the image. On the contrary, the approach shown in this paper uses the measurement of motion to detect the position of the object. Thus, a pattern-object is not required in the scene. In this work we have used a retina camera based on AER in order to compute the movements through events as input in the visual servoing system. This approach only uses event visual information and at the moment they occur for reducing the response time. The number of events depends on the latency, transition time and type of pixel-events (ON, OFF). In addition, the number of events depends on the speed of motion in the scene. In this paper, we have evaluated the use of this technology for tracking the initial motion of mobile robots from a camera mounted at the end of a robot manipulator.

The developments shown in this paper can be used directly to perform a pick-and-place task. Now, we are working on extending the second stage to completely control the robot manipulator through only the events camera. Information of the movement can be directly retrieved from the AER camera. This information can be employed directly in the control law where other works employ an estimate of it.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the Spanish Government and European FEDER funds (DPI2012-32390), the Valencia Regional Government (GV2012/102 and PROMETEO/2013/085) and the University of Alicante (GRE12-17).

REFERENCES

Aström, K. J. (2008). Event Based Control. In Astolfi, A. and Marconi, L. *Analysis and Design of Nonlinear Control Systems*, 127-147. Springer Berlin Heidelberg.

Censi, A., Strubel, J., Brandi, C., Delbruck, T. and Scaramuzza, D. (2013). Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor. *In Proceedings of IEEE/RSJ International Conference on Intelligent Robos and Systems (IROS).*

Chaumette, F. (2004). Image moments: a general and useful set of features for visual servoing. *IEEE Transactions on Robotics,* 20(4), 713–723.

Chaumette, F., and Hutchinson, S. (2006). Part I : Basic Approaches. *IEEE Robotics & Automation Magazine*, 13(4), 82–90.

Cretual, A. and Chaumette, F. (2001). Visual Servoing Based on Image Motion. *International Journal of Robotics Research,* 20 (11), 857-877.

Espiau, B., Chaumette, F., and Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3), 313–326.

Espiau, B. (1994) Effect of camera calibration errors on visual servoing in robotics. In Yoshikawa, L.T and Miyazaki, F, *Experimental Robotics III, LNCIS vol.* 200, 182–192. Springer Berlin Heidelberg.

Fiala, M. (2010). Designing Highly Reliable Fiducial Markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 32 (7), 1317-1324.

García, G.J., Pomares, J., Torres, F. and Gil, P., (2013). Event-based visual servoing. *In Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 307-314, SciTePress.

Gratal, X., Romero, J., Bohg, J., and Kragic, D. (2012). Visual servoing on unknown objects. *Mechatronics*, 22 (4), 423–435.

Janabi-Sharifi, F., and Wilson, W. J. (1997). Automatic selection of image features for visual servoing. *IEEE Transactions on Robotics and Automation,* 13(6), 890–903.

Kragic, D., and Christensen, H. I. (2001). Cue integration for visual servoing. *IEEE Transactions on Robotics and Automation*, 17(1), 18–27.

Lichtsteiner, P., Posch, C. and Delbruck, T. (2008). A 128x128 120dB 15μs Latency Asynchronous Temporal Constrast Vision Sensor. *IEEE Journal of Solid-State Circuits,* 43 (2), 566-576.

Liu, C., Huang, X., and Wang, M. (2012). Target Tracking for Visual Servoing Systems Based on an Adaptive Kalman Filter. *International Journal of Advanced Robotic Systems*, 9(149), 1-12.

Malis, E., Mezouar, Y. and Rives, P. (2010). Robustness of Image-Based Visual Servoing with a Calibrated Camera in the Presence of Uncertainties in the Three-Dimensional Structure. *IEEE Transactions on Robotics*, 26(1), 112–120.

Marchand, É. and Chaumette, F. (2005). Feature tracking for visual servoing purposes. *Robotics and Autonomous Systems*, 52(1), 53–70.

Sánchez, J., Guarnes, M. A. and Dormido, S., (2009). On the application of different event-based sampling strategies to the control of a simple industrial process, *Sensors*, 9, 6795-6818.

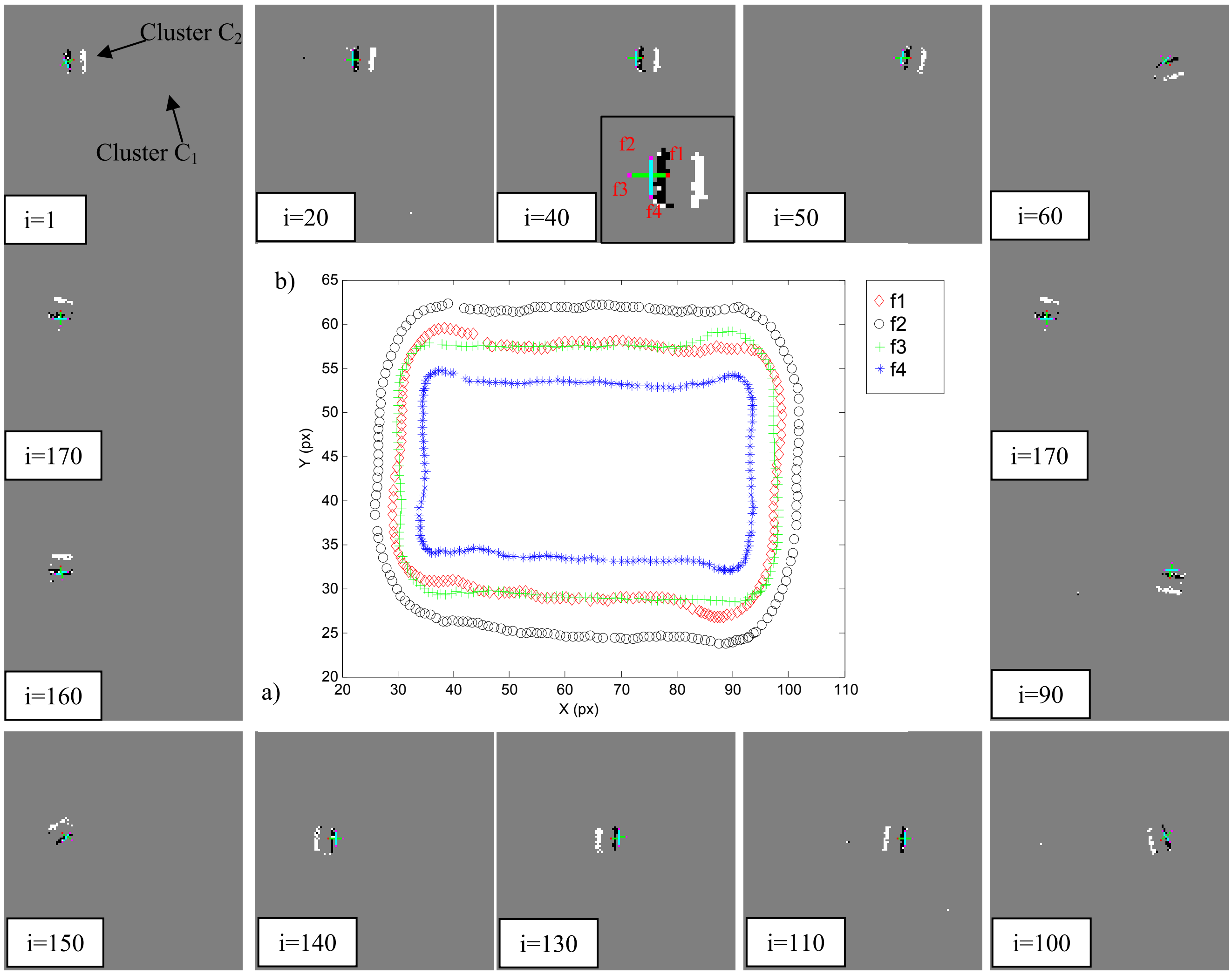

Fig. 10. a) Detection of Active Visual features from Event-Clusters (outside). b) Evolution of features for each frame (inside).