A Multiple Object Tracking System Applied to Insect Behavior

Diego Marcovecchio[†‡], Natalia Stefanazzi[♯], Claudio Delrieux[†], Ana Maguitman[‡], and Adriana Ferrero[♯]

†Laboratorio de Ciencias de las Imágenes (LCI)
Departamento de Ingeniería Eléctrica y de Computadoras (DIEC)

‡Grupo de Investigación en Administración de Conocimiento y Recuperación de Información - LIDIA
Departamento de Ciencias e Ingeniería de la Computación (DCIC)

♯Laboratorio de Zoología de Invertebrados II
Departamento de Biología, Bioquímica y Farmacia (DBBF)

Universidad Nacional del Sur (UNS)
Av. Alem 1253, (B8000CBP), Bahía Blanca, Argentina
Tel: (0291) 459-5135 / Fax: (0291) 459-5136

Abstract. Segmentation and tracking of multiple objects is an extensively researched field among Image Sciences. Multiple object tracking is, in general, a very hard problem due to the great number of potencial issues that might arise (such as the sudden movement of the tracked objects, changes on the appearence of either the tracked objects or the background scene, bad-quality frames, occlusion between an object and the scene or between multiple objects, or camera movement). Normally, object tracking is performed on applications that require the object locations to perform calculations later. This paper describes the research, design and development of a system created in order to track multiple insects on recorded videos.

Keywords: Video Processing, Object Tracking, Computer Vision.

1 Introduction

Given the increasing ease of access to multimedia-recording devices (off-the-shelf and ready to use devices like tablets, smartphones and small movie cameras), it is now possible to record high-quality videos in an almost effortless manner; this motivates the development of software applications that automatically process such information.

Being aware of the complexity of the generic multiple object tracking problem, some context-dependent conditions are usually assumed in order to find practical but flexible solutions. In this paper, we will describe the methods used to create an application that performs the segmentation and tracking of multiple objects (in particular, roaches running on Petri dishes) and will analyze several aspects on the path detected by each roach.

This work is the result of a joint-project between the *Laboratorio de Ciencias de las Imágenes* (IIIE-CONICET http://www.imaglabs.org), the *Grupo de Investigación en Administración de Conocimiento y Recuperación de Información* (http://ir.cs.uns.edu.ar), and the *Laboratorio de Zoología de Invertebrados II*, all of them from Universidad Nacional del Sur. The motivation for this work is that the first two groups develop an application to monitor the behavior of colonies of insects using the least invasive method possible, to test the effectiveness of several chemicals developed by the latter group. Another contribution and potential research direction is gaining insight for the development of new bioinspired algorithms based on ant-colonies techniques. The long-term and general goal is to develop an application that allows multiple detection and tracking of generic objects.

2 Related work

Multiple object tracking is a very complex task that requires the articulation of a pipeline with several sub-tasks to perform adequately. It is required to initialise proper regions of interest *(ROIs)*, to identify within them the desired targets, to perform a frame-by-frame following of the identified targets, to solve unexpected situations (like crossovers, superimpositions, and jerky movements) and to extract robust information regarding the individual trajectories of the targets.

Kim and Torralba proposed a system [10] that performs *ROI* detection very effectively. However, this system requires a set of different images to use as exemplar set, and in this work, we intend to perform *ROI* detection without training (i.e., we aim to detect our Region of Interest as soon as a video file is selected without any previous steps).

There are also some available programs that perform automatical or semi-assisted tagging of recognised actions in videos; for example, Takahashi's human action recognition in video surveillance systems [15], a traffic incident detection system [9], or an action-detection on tennis video recordings system [7]. However, these systems are not focused on tracking particular objects, and can only partially help to solve this problem.

Souded *et al.* presented an interesting feature-based particle filtering object-tracking system [13], but since it is focused on video-surveillance, it is designed to detect and track *every* moving object, while we are only interested in the insects inside a specific region of interest. Similarly, a more recent and robust system developed by Gao *et al.* [8] which also works with feature points based particle filtering was presented. However, the same problem remains: we are not interested in detecting *any* moving object, but only some insects inside a region. As we will see later, this approach can cause problems in our videos.

Agbinya and Rees used adaptive-color histogram-backprojection techniques to track multiple objects in surveillance and sports videos [1], but the system is not robust enough, and works with only short-lenght videos (around 10 seconds), while we need to track insects for approximately half an hour.

There is a known ant-tracking system created by Balch, Khan & Veloso[4] on the Carnegie Mellon University. Nevertheless, the application has several limitations and problems such as unresolved occlusion between the ants and the recipient walls, losing the track whenever two of the ants get too close together, splitting of the bounding boxes due to specular reflexes of the ants, and losing the track whenever some ant stops moving and stays in place for a long time (because they consider them to blend into the background).

Finally, in a recently performed study [12], which required tracking of several dozen ants, each ant had a tiny label attached to its back. While the tracking system works correctly, we are trying to develop a less invasive method by trying to track the path of each insect without interacting physically with them.

By using more advanced video processing techniques, feature-detection and with some improvements on the heuristics, in addition to assuming some conditions that are not necessarily more restrictive, but do allow us to narrow the problem, we were able to eliminate or reduce drastically most of the mentioned limitations in Balch, Khan & Veloso's system.

3 The Application

We designed and developed an application that processes videos performing the detection, tracking and statistical analysis of insects (in this case, cockroaches) running on Petri dishes. The program was created using the free computer vision library `OpenCV`, and the `Qt` Framework. Next, we will describe each subsystem separately.

3.1 Videos

The videos used were recorded by the researchers at the *Laboratorio de Zoología de Invertebrados II* in order to evaluate the repelent action of essential oils extracted from a native northern plant from Argentina. Paper discs of 18 cm diameter were divided on two halves; one was treated with 1mL of essential oil, and the other remained non-treated. The paper discs were then placed inside of Petri dishes, covered with 10-cm plastic rings treated with vaseline in order to prevent the escape of the cockroaches. The videos were recorded in closed rooms, with controled moisture and temperature conditions, during 30 minutes.

3.2 ROI detection

In order to detect the Petri dish, which is our *Region of Interest*, first we get a Gaussian-filtered version of the frame using the following 5×5 convolution kernel:

$$k = \frac{1}{159} \begin{pmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{pmatrix}$$

Next, a Canny edge detection algorithm[6] is applied on the blurred frame: following a procedure analogous to Sobel, a pair of convolution masks are applied:

$$G_{\mathrm{x}} = \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}$$

,

$$G_{\mathrm{y}} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{pmatrix}$$

and the gradient strenght G and direction θ are found with:

$$G = \sqrt[2]{{G_{\mathrm{x}}}^2 + {G_{\mathrm{y}}}^2}$$

$$\theta = arctan\left(\frac{G_{\mathrm{y}}}{G_{\mathrm{x}}}\right)$$

rounding θ to 0, 45, 90 or 135. After applying *Non-Maximum Supression*, and with only candidate edges remaining, a final thresholding with hysteresis step is performed using default lower and upper threshold values (that can be modified by the program user if necessary).

Once a binary image containing the best candidate edges is obtained, contour detection is performed by using the alternative version of the border-following algorithm proposed by Suzuki and Abe [14] (which follows only the outermost borders of a binary image); after this step, only the dominant points of the curve are stored by applying the Teh-Chin chain approximation algorithm [16]. Finally, we decide which of the contours detected corresponds to the Petri dish by selecting the biggest contour that has a round enough shape. Note that for efficiency reasons, in case of camera motion only semi-automatical *ROI* reposition is applied (i.e., the *ROI* is not repositioned automatically frame-by-frame; the program user needs to click for it). Fig. 1 shows an example of the application determing the *ROI*.

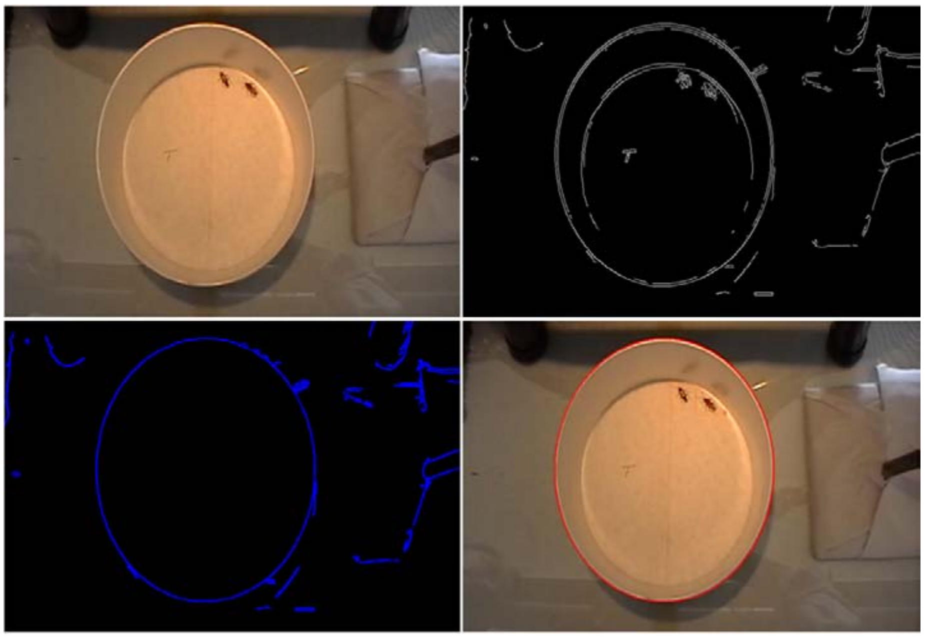
3.3 Segmentation

The application processes the video and initially recognizes the roaches by applying a *k-means* clustering algorithm [11] on the pixels inside the *ROI* that test positively in a comparison against a static color-characteristic centroid. Given the n positive pixels inside the *ROI*, k (number of roaches in the Petri dish) clusters are obtained by: first, randomly selecting k from the n pixels; second, associating each positive pixel with the closest of the selected k pixels, resulting in a Voronoi decomposition of the n pixels; third, the centroid of each of the k clusters becomes the new *mean*, and steps two and three are repeated until the variation epsilon $\varepsilon_{\mathrm{j_i}}$ in the iteration i of each centroid $\mathrm{P_j} = (\mathrm{X_j}, \mathrm{Y_j})$, with $j = 1..k$:

$$\varepsilon_{\mathrm{j_i}} = \sqrt{(X_{\mathrm{j_i}} - X_{\mathrm{j_{i-1}}})^2 + (Y_{\mathrm{j_i}} - Y_{\mathrm{j_{i-1}}})^2}$$

is small enough (in our case, $\varepsilon < 0.5$). As this is a heuristic algorithm, there is a chance it might not converge to the global optimum, depending on the

Figure 1 The ROI detection pipeline. From left to right and top to bottom: (a) the original video frame, (b) the Canny-Edge filtered image, (c) the detected contours on the Canny-Edge filtered image, and (d) the selected contour drawn on top of the original video frame.



initial clusters. Nevertheless, since the algorithm is fast enough (runs in polynomial smoothed complexity [2]), it is run multiple times with different starting conditions to check the correctness of the results.

Each insect (whose center is now defined by one of the k cluster centroids) is then trapped inside a bounding box. From now on, each of the roaches are identified frame-by-frame by using their bounding box, movement vector, and a path history that allows us to draw the trail of each insect.

Note that, due to this constraints, the system has two limitations at this moment: first of all, the number of insects needs to be known by the program user in order to apply the *k-means* algorithm. Second, since the color-characteristic centroid is initialized statically, the program would not be able to track white insects like *Myllocerus undatus Marshall* out-of-the-box. Nevertheless, it is important to notice that the system is flexible enough to allow easy user-settings to perform this operations in a semi-assisted way (i.e., to allow the user to select or specify the color-characteristic centroid, and then start to track the insects), and that with a few improvements all of this operations would be available automatically.

3.4 Tracking

In each frame, every bounding box surrounding an insect is analyzed pixel by pixel, applying for each one the comparison against the color-characteristic centroid once again. This way, the movement of the characteristic pixels inside each bounding box is detected. By applying erosion and dilation techniques in order to reduce video noise problems, each bounding box is adjusted according to the new mean position of the positive pixels found, and a new position is added to the trail history.

The occlusion problems between insects mentioned on Balch, Khan and Veloso's system are partially solved by different methods; first of all, whenever two bounding boxes have overlapping pixels, this pixels are ignored. In the previously mentioned system, whenever two ants were too close together, the bounding boxes started capturing pixels from the other ants, and finally collided completely. In our approach, by discarding the overl apping pixels, each bounding box remains tracking only one insect. However, it could occur that not enough pixels are detected due to a large overlapping area between boxes; in this case, the previously described clustering algorithm is reapplied using every positive pixel inside the *ROI*. The algorithm, as mentioned, returns a new set of k points that correspond to the center of each of the roaches, but since several pixels might have been discarded in the previous frames due to being located in a region with overlapping bounding boxes, the new k centroids will probably not match the registered k bounding boxes centers perfectly, and will need to be adjusted to the new positions of the insects.

In order to decide which bounding box corresponds to each insect, a probabilistic model is used: the movement vector of each of the colliding roaches is obtained by analyzing the currently registered position of the roach, and its position 10 frames before. A new hypothetical position is obtained by calculating the mean position between the currently registered one, and the projected position of the roach (which is the currently registered position adding the corresponding movement vector). Finally, each of the bounding boxes are assigned to the free detected centroid that is closest to the new hypothetical position. Notice that this is another of the system's limitations: certain conditions (for example, two roaches staying in the same place, together, for a long enough amount of time) might cause a bounding box swap.

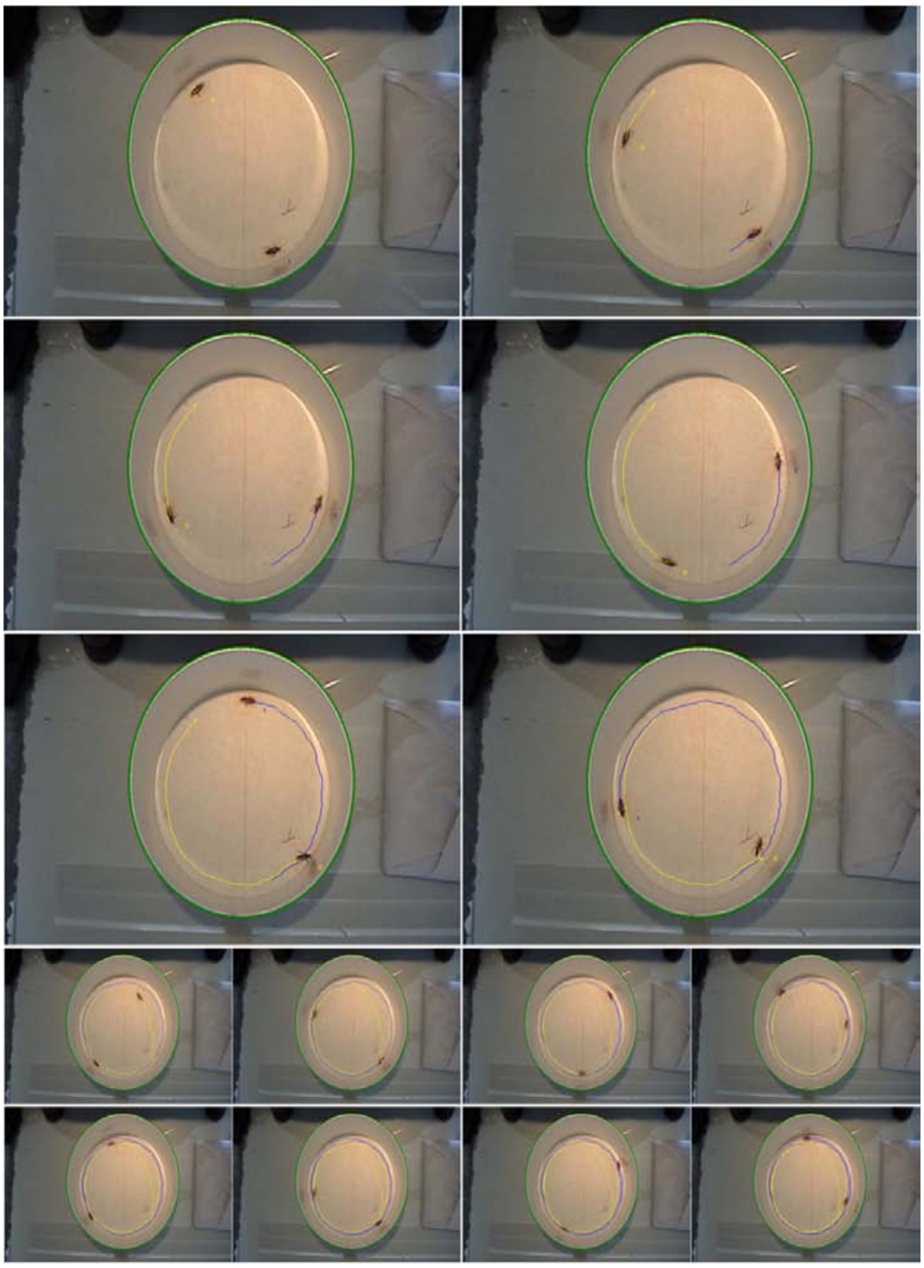
Another improvement in comparison to the ant-tracking system is that since we assume a constant amount of insects on the Petri dishes, and only check for positive pixels inside of the existing bounding boxes, these will never split into several ones. The ant-tracking system analyzed the difference between frames and placed a bounding box in each cluster of positive pixels, which caused new bounding boxes to appear in the specular reflexes on the Petri dish, and the system started tracking non-existent ants. The same problem occurs in the system developed by Gao *et al.*, because being a surveillance-oriented application, their system starts tracking every moving object. In our case, the bounding boxes simply keep following the actual insects.

Finally, the disappearing bounding boxes problem is solved, again, due to the fact that we adjust the position of the bounding boxes using the mean position of the characteristic pixels, instead of the difference between frames; in this way, the insects are not able to blend into the background. Fig. 2 shows the application tracking two insects for several frames.

3.5 Statistical analysis

Every piece of information gathered by the program is used to generate statistics that are later reported to the *Laboratorio de Zoología de Invertebrados II*; the most important is the time percentage spent on the treated and non-treated halves of the Petri dish, in order to determine if the developed insecticides are effective or not.

Figure 2 The application tracking two insects for several minutes. The trail left by each insect can be seen in a distinctive and unique color.



To accomplish this, each bounding box's center position is checked on every frame to see whether or not the insect has trespassed to the treated area of the Petri dish. The total number of trespassings can be compared to the total number of frames to obtain the time percentage spent on each half of the Petri dish. Notice that this is another of the system's limitations; tracking is not affected by camera movement, and ROI reposition can be applied semi-automatically, but the statistical analysis will not be perfect if the camera is moved. However, this is not a hard-to-implement feature and could be added in future versions of the application.

In addition, because a history of the trail of each insect is stored, and it is possible to obtain a timestamp for each of them (or the time-delta between each of them), it is also plausible to analyze the tortuosity of the path of each insect by either the straightness, sinuosity, or fractal index [5].

4 Conclusions

The system currently detects and tracks effectively in every normal condition presented on the videos, being able to generate percentual statistics about the time spent by each roach on treated and non-treated regions of the Petri dish with great effectiveness. Once the target video is selected, the program works in a fully-automatic way, except for the semi-assisted *ROI* reposition (in case it is needed).

The application has also shown robustness when abrupt changes on the lightness ocurred on the room where the videos were recorded, and is in general a great improvement compared to the previously known ant-tracking system. In addition, due to the relative simplicity of the tracking algorithm, the application works fast enough to track insects in real time in 1280 x 720 videos at 30 frames per second, which most feature-based and complex tracking systems have serious trouble with. Nevertheless, the system presents some limitations. The color of the insects is defined statically, and whenever two insects occupy the same space during a large amount of time, the application may confuse them and could potentially swap the bounding boxes. Similarly, abrupt camera motion requires a user response in order to explicitely ask for a *ROI* repositioning, and makes the statistical analysis less effective.

Currently, the system is being used on a daily basis to test the effectiveness of the essential oils over dozens of videos.

5 Future work

There are several features we would like to add to the system. First of all, it would be desirable to perform the segmentation of each insect without using a statically defined characteristic color. It would also be useful to add sanity checks in order to test if the insects are effectively trapped inside their bounding boxes, and if the *ROI* is correctly positioned at some time. *ROI* tracking to detect camera motion is another possibility. Dynamic detection of the treated and non-treated areas of the Petri dish would eliminate the camera motion constraints. Adding feature-based techniques to the tracking system would make the application even more robust. Roaches collisions could be resolved in a more complex and robust way (for example, an implementation of the Minimum Cost Bipartite Matching algorithm [3]). And finally, a different clustering algorithm could be applied to check how many insects are present in the video, instead of using this knowledge beforehand to apply k-means.

References

[1] Johnson I Agbinya and David Rees. Multi-object tracking in video. *Real-Time Imaging*, 5(5):295 – 304, 1999.

[2] David Arthur, Bodo Manthey, and Heiko Röglin. k-means has polynomial smoothed complexity. *CoRR*, abs/0904.1113, 2009.

[3] Hari Asuri, Michael B. Dillencourt, David Eppstein, George S. Lueker, and Mariko Molodowitch. Fast optimal parallel algorithms for maximal matching in sparse graphs. Technical Report 92-01, Univ. of California, Irvine, Dept. of Information and Computer Science, Irvine, CA, 92697-3425, USA, 1992.

[4] Tucker Balch, Zia Khan, and Manuela Veloso. Automatically tracking and analyzing the behavior of live insect colonies. In *Proceedings of the fifth international conference on Autonomous agents*, AGENTS '01, pages 521–528, New York, NY, USA, 2001. ACM.

[5] Simon Benhamou. How to reliably estimate the tortuosity of an animal's path:: straightness, sinuosity, or fractal dimension? *Journal of Theoretical Biology*, 229(2):209 – 220, 2004.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[7] Q. Huang G. Zhu, C. Xu. Player action recognition in broadcast tennis video with applications to semantic analysis of sports game. In *in Proc. ACM Multimedia, 2006*, pages 431–440, 2006.

[8] Tao Gao, Guo Li, Shiguo Lian, and Jun Zhang. Tracking video objects with feature points based particle filtering. *Multimedia Tools and Applications*, 58(1):1–21, 2012.

[9] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Incident detection at intersections utilizing hidden markov model. In *6th World Congress on Intelligent Transport Systems*, pages 1–10, 1999.

[10] Gunhee Kim and Antonio Torralba. Unsupervised Detection of Regions of Interest using Iterative Link Analysis. In *Annual Conference on Neural Information Processing Systems (NIPS 2009)*, 2009.

[11] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[12] Danielle P. Mersch, Alessandro Crespi, and Laurent Keller. Tracking Individuals Shows Spatial Fidelity Is a Key Regulator of Ant Social Organization. *Science*, 340(6136):1090–1093, May 2013.

[13] Malik Souded, Laurent Giulieri, and Francois Bremond. An Object Tracking in Particle Filtering and Data Association Framework, Using SIFT Features. In *International Conference on Imaging for Crime Detection and Prevention (ICDP)*, London, Royaume-Uni, November 2011.

[14] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985.

[15] M. Takahashi, M. Naemura, M. Fujii, and S. Satoh. Human action recognition in crowded surveillance video sequences by using features taken from key-point trajectories. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 9–16, 2011.

[16] C. H. Teh and R. T. Chin. On the detection of dominant points on digital curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(8):859–872, August 1989.