Vision-based localization for mobile robots using a set of known views

Pablo Frank Bolton, Montserrat Alvarado, Wendy Aguilar, Yann Frauel

Departamento de Ciencias de la Computación
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
frankpablo@gmail.com

A robot localization scheme is presented in which a mobile robot finds its position within a known environment through image comparison. The images being compared are those taken by the robot throughout its reconnaissance trip and those stored in an image database that contains views taken from strategic positions within the environment, and that also contain position and orientation information. Image comparison is carried out using a scale-dependent keypoint-matching technique based on SIFT features, followed by a graph-based outlier elimination technique known as Graph Transformation Matching. Two techniques for position and orientation estimation are tested (epipolar geometry and clustering), followed by a probabilistic approach to position tracking (Monte Carlo localization).

1 Introduction

Robot localization techniques consider a wide variety of perception models, which are the techniques they use to obtain information from their surroundings and interpret the data. Some proposals base the sensing of the environment on sonar or laser readings [1,2]; others pinpoint a robot's location by triangulating the intensity of signals being broadcast for that purpose [3,4]; there are even those that shun the use of a perception system and decide to keep track of the robot position through odometry error minimization, which of course, becomes terribly inaccurate over time.

Robot vision is one of the richest – yet complex – perception models. Several proposals of vision-based localization have been made using various approaches [5-11]. Among these, some techniques use a collection of training images to help the robot in finding its current position [8-11]. .

In this work, a vision based localization system is described, in which a robot navigates through a known environment based on image comparison and position estimation. A robot is required to detect its position and then track its own movements within a defined space of which an image database was previously gathered. This image database contains several key positions, or nodes, separated by roughly 90 cm, from which eight different views were collected, spanning 360 degrees in 45 degree steps. Figure 1 is a sketch of the test area. Robot localization comes in two flavors: global localization (in which a robot's initial position is not known) and position tracking (in which the starting position is known, as well as the odometry of its movements). In this work, the two localization problems are solved as described in the next sections.

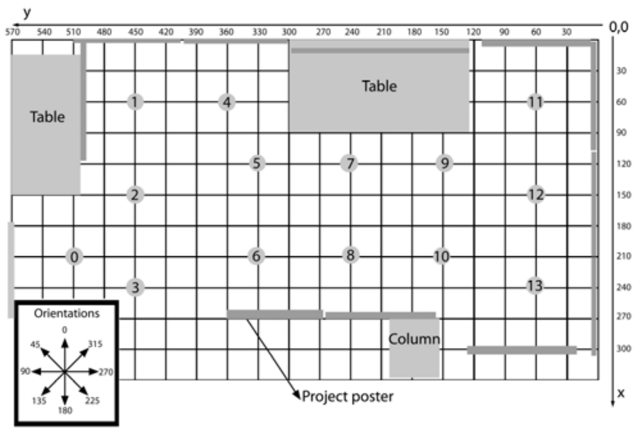

Figure 1: Test environment. The test area is a lab section surrounded by project posters. In the sketch, circles with numbers represent the viewpoint nodes, totaling 14. Each square represents 30 cm.

As described earlier, the image database is composed of eight views (45 degrees apart) for each of the 14 nodes, which gives us a total of 112 images. Figures 2 and 3 are of a sample series of images, taken from node 5.

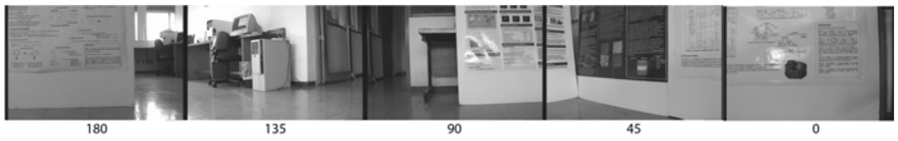

Figure 2: Views of orientations 180, 135, 90, 45 and 0 degrees**.**

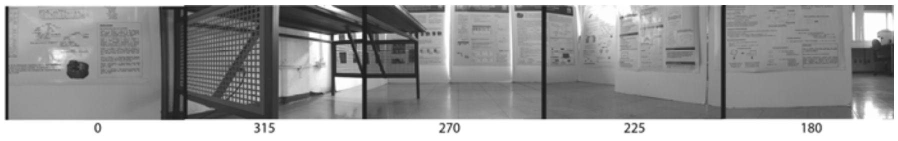

Figure 3: Views of orientations 0, 315, 270, 225 and 180 degrees.

The outline of the process is described in Figure 4. In the following sections, each element of this outline will be described.

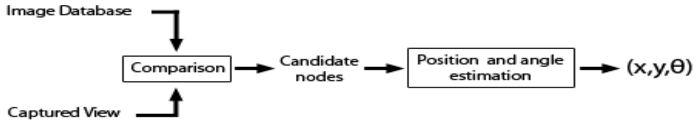

Figure 4: Process outline.

2 Image comparison

The first step in determining the robot's position is gaining information on which nodes are similar to the captured view (which is what the robot is “seeing” in the unknown actual position). This step constitutes a considerable contribution, for image similarity also entails certain restrictions on the robot's probable position. In other words, if the robot's actual view is very similar to some node's viewpoint, its position and orientation are also related to the actual robot's position. In the next section, this relation and the estimation of the actual position and orientation is discussed. First, similarity between images must be defined and determined through image processing.

What will be compared in the images? There are many alternatives to solve this question, which is a significant area of investigation in and of itself. For this work, David Lowe's SIFT (Scale-Invariant-Feature-Transform) [12] feature extractor was selected, which identifies points of interest or keypoints within an image depending on each pixel's intensity derivatives in the scale-space [13] of that particular image. A descriptor (which is a series of numbers that describe each keypoint's information) is assigned to each point. Each descriptor contains a 128 feature vector that contains information of the keypoint's region orientations. Each keypoint is defined by a pair of coordinates (position in the 2D image matrix), a scale (level at which the keypoint is found inside the scale-space decomposition), an orientation (obtained from the intensity gradient of that particular location), and the 128 feature vector describing the keypoint's surroundings.

In his work [12, 23], Lowe presents an approximate nearest-neighbor method for comparing two sets of keypoints. This method is called BBF (Best-Bin First). One important drawback of this technique is that there is no spatial information taken into consideration (other than region similarity) for obtaining the nearest match. This means that if a keypoint's region descriptor (the 128 feature vector) and its orientation and scale closely match another keypoint's data, these are a candidate match. Notice that this may happen for two completely unrelated points that share local similarity but that are really at very different 3D locations. These erroneous matches are called outliers. A technique called GTM (Graph-Transformation-Matching) [14] was implemented to reduce these types of outliers in the final matching.

GTM has a simple but effective conducting principle: iteratively eliminate correspondences that disrupt the neighborhood relationships. In order to do so, it constructs a K-nearest-neighbor (K-NN) graph for each view image (based on the space coordinates of detected feature points), and during each iteration it a) removes the vertex (match) which most disrupt the similarity structure on both graphs, b) reconstructs the corresponding K-NN graphs and repeats this process until both graphs are identical, which means outliers have been removed.

The entire process is shown through images in Figure 5. In (a), a couple of images are processed through SIFT. Their keypoints are obtained and matched using BBF. Several matches are outliers at this stage. In (b) the maximum common neighborhood graphs are obtained for each image, and in (c) these are used to select the keypoints that will remain as correct matches.

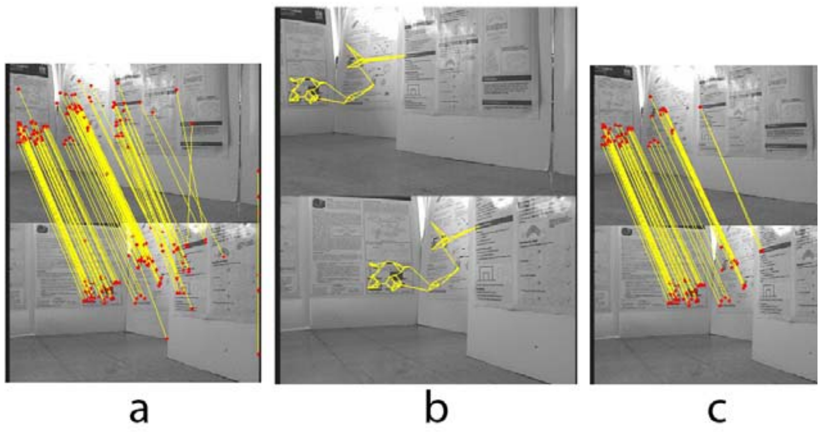

Figure 5: (a) Keypoint detection and matching with BBF. (b) Result of Graph Transformation Matching for images 1 and 2. (c) Matches after GTM.

This process may be implemented for the comparison of the actual robot view with the entire 112 images of the database. As mentioned before, every keypoint was detected in relation to a particular scale. Low-value scales are related to high frequency changes in the image. Therefore, it is at these scales that most small details or even noise will be detected. This scale range is also where most of the keypoints will be detected so, to optimize the image comparison process, a dynamic-scale match range is defined. This means that a first matching approximation will take into consideration only medium and high scaled keypoints (medium and low frequencies). If this first comparison renders enough matches to warrant further analysis, the matching will include the low scale (high frequency) information.

After this step, a list of relevant nodes (those with more matches than a certain threshold value) will constitute the base information for the position estimation phase.

3 Position and orientation estimation

At this point, the only information gathered in relation to the actual robot orientation and position is a list of node views that are similar to the current view. As mentioned before, these views have a pair of position coordinates (in relation with a predefined origin) and an orientation coordinate (spanning 360 degrees through 45 degree rotations). Together, these three coordinates form the State Coordinate of the view. Two different techniques were implemented to obtain the state coordinate of the robot in relation to the candidate nodes. The first technique is based on motion estimation using epipolar geometry [15]. The second method is based on clustering and is called QTc (Quality Threshold clustering) [16].

3.1 Epipolar geometry

This technique attempts to recover the spatial information through projective data. This means that it can recover the relation between two separate projections of the same (or very similar) view. In this case, the scene projections are the 2D images of the related views. Figure 6 presents the basis for this technique, which is called epipolar geometry.

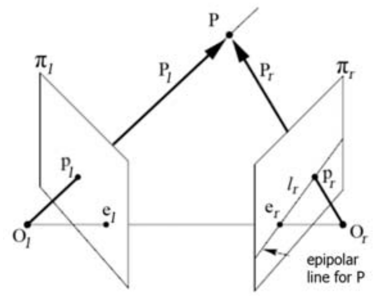

Figure 6: Epipolar Geometry

The point *P* in Figure 6 represents a point in the 3D space, while the points p_l and p_r are its projections on the left and right image planes respectively. The geometry forces a tight relation between points in one image plane, and lines in the other. This relation is called the epipolar constraint, and states that *corresponding points must be on conjugated epipolar lines*[15]. This means that a projected point p_i in the left image plane may be related exactly to a line in the right image plane. This line, called the right epipolar line, connects the projected point on the right plane, p_d, with the right epipole (which is the projection of the left optic center on the right image plane). This principle also applies for points in the right plane to lines in the left.

The algebraic expression of this restriction is $l_r = Ep_l$, where *E* represents a transformation matrix from a point to a line. This transformation matrix is known as the Essential matrix, and contains all the information necessary to reconstruct the epipolar geometry of the scene, from which motion estimation may be carried out. It is, therefore, of vital importance to obtain this matrix to be able to estimate the relation of one perspective to another. There are several ways of doing this[17-19]. One such technique is called the normalized-eight point algorithm [20,21], which uses a series of matches between projections to estimate *E*. It is called “normalized” because it normalizes pixel coordinates to reduce unwanted numerical instability.

The next step is to extract rotation and translation information between the projections. This process is shown in [15]. Once obtained, The rotation and translation matrices **R** and **T** respectively, contain information of how one camera system may be moved to arrive at the other. If motion between viewpoints is known, and the state coordinate of one viewpoint is known (one of the candidates obtained through image comparison), then the other viewpoint's state coordinate may be obtained (the actual robot position and orientation). It is very important to notice that the resulting solution for **T** is up to a scale factor. Even after solving the sign ambiguity for **T** [15], it only indicates direction of translation and not actual distance.

3.2 Quality Threshold clustering

Clustering is a data classification and ordering technique with many applications in the areas of data mining, pattern recognition, image analysis and many more. The basic principle is to decide classification approach and then, depending on the nature of the data, generate groups or generalized representations. This may serve the purpose of grouping elements under defining characteristics, or finding boundaries between element clusters. This last interpretation may have, if thus treated, a spatial connotation that is relevant to this study. If boundaries are defined, then areas are also defined.

Starting from the candidate nodes extracted in the comparison phase, a clustering algorithm may extract a definite area of concentration, both for position and for orientation, therefore generating a simple but accurate state estimation. There are many types of clustering methods. Generally they are divided into hierarchical clustering, partitional clustering and alternative methods. Out of all these techniques, the nature of this work requires a clustering technique that dynamically selects the number of clusters and generates cluster centroids that represent them. One technique that accomplishes this is called QTc (Quality Threshold clustering) [16].

Originally used in genomics, QTc is a partitional clustering scheme that derives from the popular *k*-means algorithm. It trades processing speed for the freedom of not having to define the number of clusters *a priori*. QTc works as follows:

1. A maximum cluster radius is defined.
2. Construct a candidate cluster for each point. This is done by including the next closest neighbor until the maximum distance is exceeded.
3. The cluster with the most elements is maintained and represented by a cluster centroid (removing all cluster members).
4. Recurse with the remaining points until no cluster may be detected.

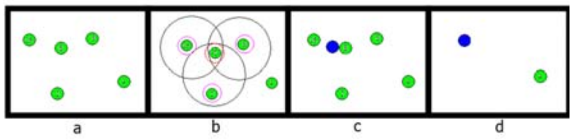

Figure 7: QTc applied to 5 points. Circles represent maximum cluster radius.

An example of this process is shown in Figure 7.

In Figure 7, (a) shows the initial set of points, the numbers inside the nodes represent the node's weight (relevant in the definition of the centroid) . (b) shows the maximum cluster (which is the one tied to the node in its center). (c) shows the location of the cluster centroid, which is the center of mass of the total node weights. This is obtained by the weighted average of the node positions within the maximum cluster. The values used to define the relevance of each node's position are, in this case, the amount of matches registered between that particular view and the robot's actual view (these were obtained in the comparison stage). (d) shows the final cluster centroids. Note that each final centroid is too far away to the other to unite in a sole cluster.

The example shown above represents the cluster estimation of position. Nevertheless, the same logic may be applied to the orientations. The final orientation estimate would be the weighted average of the candidate orientations:

$$\hat{\Theta} = \sum_i W_i O_i$$

where O_i represents the orientation of candidate i, and W_i represents the weight of candidate i (number of matches of candidate i / total number of matches). Say we have 4 candidate viewpoints. If one viewpoint has an orientation coordinate of 45 degrees (and a weight of 20), and three viewpoints have orientation coordinates of 90 degrees (with weights of 10, 20, and 30 respectively), then the final orientation would be:

$$\hat{\Theta} = 45 \times \left(\frac{20}{80}\right) + 90 \times \left(\frac{10}{80}\right) + 90 \times \left(\frac{20}{80}\right) + 90 \times \left(\frac{30}{80}\right) = 78.75^o$$

The results of implementing the comparison and position estimation stages are discussed in the *Results* section.

4 Monte Carlo Localization

Monte Carlo localization, or MCL [22], is a type of particle filter applied to robot localization. This technique employs a series of particles that represent the state-space, which for this particular case is the area of possible robot positions. If global estimation is being attempted with no information on the robot's initial position, then the robot's position is represented by an uniform probability distribution over all possible particles. An *importance function* is used to assign weights to each particle, and as the process continues, only particles that have weights over a certain value are maintained. In the localization scenario, weights are assigned in relation to a perceived signal. In [1], the perceived signal is a sonar reading of the robot's environment. Weights are assigned to particles depending on the probability that such a reading might have been registered from each particular particle's position (node). For this work, the perceived signal is a captured image, and the assigned weight is the amount of matches found between the actual view and each particle’s viewpoints. This weight is normalized over the entire amount of matches so as to accurately represent the probability of being the “most similar” node. In traditional MCL, the amount of particles is maintained throughout the operation. However, this is not the case in this study. As explained in the next section, when doing position tracking, only particles (nodes) close to estimated positions are incorporated.

The sequence of steps needed to pinpoint the best particle is as follows:

1. Robot motion: the robot moves and generates a series of candidate nodes, depending on the last stage's estimated position.
2. Sensor reading: after a reading, probability weights are assigned to particles. Those with lower weights than a certain threshold are eliminated.
3. The sequence is repeated until only one particle is being propagated.

This process is used to narrow down the possible robot's position during position tracking.

5 Results

Tests carried out on diverse settings proved that an indoor setting with medium to high amount of detectable characteristics is a good benchmark test. This is explained further down, but is essentially a case of “more is better” in the case of using matches to obtain projection relations and/or node weights.

In relation to the comparison stage, SIFT, BBF and GTM accomplish the task with minor precision errors. It is important to mention that certain outlier arrangements are undetectable by GTM and are therefore contributing factors in erroneous position estimation. This is especially true if the epipolar geometry method for state estimation is used. This is because motion estimation based on the eight point algorithm is highly sensitive to outliers. The effect of outliers is adding false information about corresponding projections, hence causing irreparable damage on the resulting **R** and **T** matrices. As mentioned above, the effect may be less serious if the outlier match represents a very small amount of the total information regarding the geometry of the space. This means that if there are a lot of correct matches, one or two incorrect matches may not completely thwart the estimation attempt.

When using epipolar geometry, the minimum amount of matches (in average case scenarios) needed to reconstruct the 3D geometry is – as the name suggests – eight correct matches. There are degenerate configurations and other circumstances in which this does not hold, but, on average, eight points is enough for estimating **R** and **T.**

In practice, due to the inability to eliminate all outliers, many more matches are needed to ensure a good estimation. For this reason, a “feature-rich environment” is advisable for augmenting estimation efficiency. In the series of experiments carried out in the above mentioned scenario, epipolar geometry proved too sensitive to noise (outliers or differences in the exact nature of the observed scene), and therefore not as precise as QTc. Orientation estimation accuracy was calculated as follows:

$$\Theta_{error} = \frac{\left(\left|\Theta_{real} - \Theta_{estimate}\right|\right)*100}{360}$$

On average, in the case of epipolar geometry orientation estimation, Θ_{error} is of about 3.5 %. This may seem low, but this means that an orientation estimation might be off ± 13degrees. Another problem with epipolar geometry is that the resulting translation is precise only up to a scale factor, which means it indicates direction but not distance from one view to the other. One solution to this problem is to combine translation information from different reference nodes. This would generate an intersection space where the actual robot position could be found. This, however, is also extremely sensitive to error and certain degenerate viewpoint configurations such as parallel views (which is when a pure translation is carried out solely on the optical axis).

The alternative state estimation method, QTc, turned out to render more accurate results for orientation coordinates as well as very good first order approximations for the position of the robot. Θ_{error} was of about 2.5% on average (about ± 9 degrees). The actual position of the robot was determined as the resulting centroid of nodes (if only one remained) or the average of the resulting centroids (if more than one centroid was brought forth by QTc). These “global localization” estimates where, on average, only 46 cm off.

The next step in state estimation is to move the robot and generate a set of probably-close nodes. This is done by moving the estimated position and adding, as node candidates, those nodes that are within a certain radius. This is done for all estimated positions (centroids) remaining from the last stage. The comparison-localization stages are repeated and a new state coordinate calculated. In practice, QTc improves position estimation only if a denser particle grid is incorporated. For time optimization, global localization may be carried out based on a medium density particle grid, followed by position tracking based on a local subset of a high density particle grid.

6 Conclusions

We proposed a localization technique based on the comparison of a current view against a database of known images. The comparison uses the SIFT features as a similarity metrics. The information from several views is integrated using either epipolar geometry or a clustering method. Temporal progression is taken into account with a variant of Monte Carlo localization. The proposed technique proved effective in solving the two basic localization problems. If the space is represented by a grid of particles, the estimated position and orientation varies depending on the grid's density. Epipolar geometry based on the normalized eight point algorithm proved too sensitive for position estimation, but may still be used for fine orientation adjustments once a general state coordinate is calculated through alternative methods (clustering in this case). Another factor that will contribute to time optimization is the scale at which images are compared, which is a factor that is also environment-dependent.

Acknowledgments

The authors acknowledge financial support from the PAPIIT program of Universidad Nacional Autónoma de México, under grant IN104408.

References

1. Burgard, W., Fox, D., Henning, D.: Fast Grid-Based Position Tracking for Mobile Robots. In Proceedings of the 21st Annual German Conference on Artificial Intelligence. London, UK, Springer-Verlag (1997) 289--300
2. Zhao, F.-J., Guo, H.-J., Abe, K.: A mobile robot localization using ultrasonic sensors in indoor environment. In Proceeding of the 6th IEEE International Workshop on Robot and Human Communication. (1997) 52–57

3. Detweiler, C., Leonard, J., Rus, D., Teller, S.: Passive Mobile Robot Localization within a Fixed Beacon Field. In Proceedings of the International Workshop on the Algorithmic Foundations of Robotics, New York, New York (2006)
4. Sewan, K., Younggie, K.: Robot localization using ultrasonic sensors. In Proceedings of Intelligent Robots and Systems 2004. (2004) 3762 - 3766
5. Elinas, P., Little, J.J.: σMCL: Monte-Carlo localization for mobile robots with stereo vision. In Proceedings of Robotics: Science and Systems. Cambridge, MA, USA(2005)373-380.
6. Beardsley, P.A., Zisserman, A., Murray, D. W.: Navigation using affine structure from motion. In Proceedings of the 3rd European Conference on Computer Vision. Volume 801 of Lecture Notes in Computer Science . Berlin, Springer (1994) 85-96
7. Robert, L., Zeller, C., Faugeras, O., Hebert, M.: Applications of non-metric vision to some visually guided robotics tasks. Technical Report 2584, INRIA. Sophia-Antipolis, France (1995)
8. Wolf, J., Burgard,W., Burkhardt, H.: Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In Proceedings of the IEEE International Conference on Robotics & Automation (2002)
9. Sim, R., Dudek, G.: Comparing image-based localization methods. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (2003) 1560 – 1562.
10. Menegatti, E. P. E., Zoccarato, M., Ishiguro, H,: Image-based Monte-Carlo localisation with omnidirectional images. Robotics and Autonomous Systems **48** (2004) 17–30
11. Ulrich, I., Nourbakhsh, I.: Appearance-based place recognition for topological localization. In Proceedings of the IEEE International Conference on Robotics & Automation (2002) 1023–1029
12. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal on Computer Vision **60** (2004) 91-110
13. Lindeberg, T.: Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics **21** (1994) 224-270
14. Aguilar, W., Frauel, Y., Escolano, F., Martinez-Perez, M.E., Espinoza-Romero, A., Lozano, M.A.: A Robust Graph Transformation Matching for Non-rigid Registration. Image and Vision Computing (2008) doi:10.1016/j.imavis.2008.05.004
15. Trucco, E., Verri, A.: Introductory techniques for 3-D computer vision. Prentice-Hall (1998)
16. Heyer, L.J., Kruglyak, S., Yooseph, S.: Exploring expression data: identification and analysis of coexpressed genes. Genome Research **9** (1999) 1106-1115
17. Zhang, Z.: Determining the Epipolar Geometry and its Uncertainty: A Review. International Journal of Computer Vision **27** (1998) 161-195
18. Luong, Q. T., Deriche, R., Faugeras, O., Papadopoulo, T.: On determining the fundamental matrix : analysis of different methods and experimental results. Tehcnical report RR-1894. Sophia-Antipolis, France, INRIA (1993)
19. Hartley, R. I., Zisserman, A.: Multiple View Geometry in Computer Vision, Second Edition. Cambridge University Press (2004)
20. Longuet-Higgins H.C.: A computer algorithm for reconstructing a scene from two projections. Nature **293** (1981) 133-135
21. Hartley, R. I.,: In defence of the 8-point algorithm. Proceedings of the Fifth International Conference on Computer Vision. IEEE Computer Society (1995) 1064
22. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte Carlo localization: Efficient position estimation for mobile robots. In Proceedings of the National Conference on Artificial Intelligence (1999) 343-349
23. J. Beis and D.G Lowe. Shape indexing using approximate nearest-neighbour search in highdimensional spaces. In Proc. IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, pages 1000-1006, 1997.