Augmented Reality Using Projective Invariant Patterns

Lucas Teixeira, Manuel Loaiza, Alberto Raposo and Marcelo Gattass

Tecgraf - Computer Science Department
Pontifical Catholic University of Rio de Janeiro, Brazil
{lucas,manuel,abraposo,mgattass}@tecgraf.puc-rio.br

Abstract. This paper presents an algorithm for using projective invariant patterns in augmented reality applications. It is actually an adaptation of a previous algorithm for an optical tracking device, that works with infrared illumination and filtering. The present algorithm removes the necessity of working in a controlled environment, which would be inadequate for augmented reality applications. In order to compensate the excess of image noise caused by the absence of the infrared system, the proposed algorithm includes a fast binary decision tree in the process flow. We show that the algorithm achieves real time rates.

1 Introduction

Augmented Reality (AR) is a research area interested in applications that enrich the visual information of the real environment where the users interact. Computer vision algorithms are commonly used in AR applications to support the detection, extraction and identification of markers in the real scene and to continuously track these markers, allowing that users change their points of view.

The definition of what may be considered a marker has been changed as long as tracking algorithms evolve; patterns of points, planar patterns and specific characteristics of the tracked object may be considered markers. However, the main requirement for markers remains the same: They should have a format that can be easily identified. We propose an algorithm for a tracking pattern based on projective invariants and demonstrate its good performance in the identification and real time tracking of the pattern. The proposed algorithm is based on the algorithm presented in [1], which was developed to support an optical tracking system that works in illumination controlled environments, since it is based on infrared (IR) filtering. We propose changes in this algorithm to allow its use in non-controlled environments for the implementation of AR applications. The main adaptation in the algorithm is the use of a binary decision tree classificator to eliminate a significant part of false markers that appear in the image due to the absence of IR filtering.

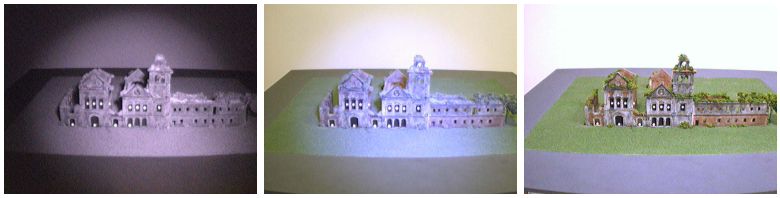
2 Related Work

One of the most common solutions for objects identification in AR is the use of ARToolkit-like planar patterns [2]. In addition to the identification of objects,

this kind of pattern also allows the camera calibration in relation to the pattern, enabling the easy insertion of virtual objects over or around the pattern. One of the drawbacks of this kind of pattern is that it is very "invasive", becoming very apparent in the scene.

A different and more restrictive solution are systems that use IR illumination together with cameras with IR pass filters (filters that allow the passage of high wavelengths only) to detect retro-reflexive patterns in the scene. This combination of IR light and filters was used in [1] to develop a technique for pattern identification and tracking. The problem of making AR with a system that uses IR filters is that the captured image looks like a night-vision goggles image, which is very different from the users' normal vision – Figure 1(a).

An attempt to use the above technique in an AR application was presented in [3]. In that work, the solution to obtain a more "realistic" image was to use lens without an IR filter, a weak IR illuminator to accentuate the retro-reflexive markers, and fluorescent ambient light (that doesn't emit IR). The work also presents a technique to project virtual objects using patterns of collinear points. This approach, however, presented two problems. The first one is that the image is still preceptively altered due to the absence of the IR cut filter (filter that doesn't allow the passage of high wavelengths), present in any regular camera. This difference is showed in Figures 1(b) and 1(c). The second problem is that the absence of an IR pass filter introduced a lot of noise in the image, what increased the time spent to find the patterns in the scene, restricting the velocity of camera movements to keep real time rates.



(a) IR pass filter (b) no filter + IR light (c) IR cut filter

Fig. 1. Same image with different illumination and filtering setups.

The present work proposes a solution for the above mentioned problems and explores the use of collinear patterns to support AR applications. Examples of applications where this proposal may be used are applications for adding or visualizing information over previously marked objects, where the main problem is the necessity of large patterns in the scene, falling again on the "visual invasion" problem of ARToolkit-like patterns.

3 Original Algorithm for Projective Invariant Patterns

In this section we explain the algorithm originally proposed in [1], upon which the algorithm proposed here is based. The original algorithm will be referred as APIP (Algorithm for Projective Invariant Patterns).

3.1 Tracking Algorithm

APIP was developed to support the process of creation, identification and tracking of specific patterns used in an optical tracking system. This kind of system works in a restrictive interaction ambient, defined by the properties of IR light and retro-reflexive materials used to accentuate the markers from the rest of the scene. APIP may be summarized as:

```
Input image of each captured frame
Image binarization
Extract circular areas based on contours found in the image
Group circular areas using a quadtree
For each quadtree branch
 Generate test patterns of 4 markers. These test patterns are
  generated by the combination of the circular areas
 For each generated pattern
    Execute the collinearity test
    If collinear
      Test the value of the projective invariant for this pattern
      Compare the value of the projective invariant of the
       candidate with the value of the wanted pattern
          If equal
             Save the set of markers that compose the pattern
If the pattern is recognized
 Create a bounding area around the markers so that in the next
  frame the markers may be extracted in this area
```

3.2 Main Techniques Used by APIP

Projective Invariants: The implementation of projective invariant properties has been discussed especially in the area of pattern recognition [4]. It is based on the invariant property of cross ratio, particularly in the case of perspective projection. The cross ratio property states that if we have 4 collinear points (A,B,C,D) we can define a cross ratio value based on these points distances according to the following relationship: $Crossratio(A,B,C,D) = \frac{\left(\frac{|AC|}{|BC|}\right)}{\left(\frac{|AD|}{|BD|}\right)}$.

Collinearity: Collinearity is a perspective projection invariant characteristic explored in the proposed tracking application. This characteristic, besides having

an unique identifier extracted from the projective invariant theory, may generate a specific format that can be used as a filter to discard several candidate groups of 4 points that don't fit this format.

Bounding Area: A second technique used to reduce computational costs is the generation of a bounding area around the pattern position in the current frame. Once the pattern is found and validated, the system creates a bounding area, used as a simple way to predict and restrict the area where a well recognized pattern may appear, based on information from frame t in frame t+1.

3.3 Limitations

APIP was developed to work in an environment where image noise problems are reduced due to IR filtering. This avoids the generation of false markers that would hamper its real time performance. The proposal of the present paper is to adapt the algorithm to be used in environments without the IR restrictions. The main problem with this new approach is that the number of false markers are increased due to the absence of the IR system (IR pass filter + IR illumination). We show that with some modifications in APIP we can quickly discard these false markers (or at least the majority of them), keeping the real time performance, necessary in AR applications.

4 Adapted APIP

In order to treat the problem of noise increase, we developed a marker classifier based on machine learning. This classifier is a binary decision tree (BDT) that is used to eliminate the majority of false markers. The BDT is necessary because the algorithm to find ellipses [5] only adjusts the best ellipse within a connected component with any format. In addition to the BDT, we introduce a restriction for the size of the pattern in order to help the optimization considering the specific case of using the projective invariants technique in AR. About the illumination, the system just assume that the light sources are static or only exist ambient illumination. Figure 2 shows an overview of the adapted APIP.

4.1 Binary Decision Tree Classification

BDT is a fast and broadly used form of classification [6]. The tree is generated by passing a list of examples for a generic algorithm of decision tree generation. Each example is a register of the universe about which we want to learn. It is formed by the attributes that describe this example in that universe and the class to which it belongs. The algorithms that need the correct class included in each example during the training are called supervised algorithms. The intermediary nodes of the tree have a reference for an attribute and two child nodes. Leaf nodes store only one of the possible classes in the universe. In order to use

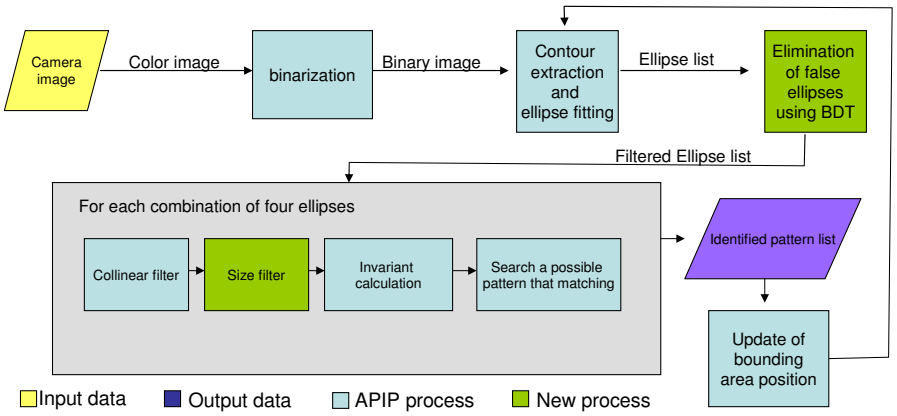


Fig. 2. Adapted APIP process flow.

the tree to classify a sample (i.e., an example without an associated class), we traverse it from the root and, at each intermediary node, we use the attribute associated to this node to decide the child to go. When we arrive at a leaf node, its associated class is the result of the classification. In this section we describe how to construct a BDT capable of eliminating the majority of false markers in a segmented image. Initially we explain how to choose the attributes to describe an elliptical marker. Then, we show how to generate the training examples (corpus) and suggest a technique to increase the robustness of the corpus. We then define the tree generation algorithm we use and explain how to use the tree, keeping it robust with the markers' scale variation.

Attribute Generation: APIP doesn't restrict the kind of marker, but to create a smaller and more robust BDT, we are going to be restricted here to elliptical markers. We can use circular or spherical markers in the scene, which will be viewed as ellipses in the image due to perspective distortion. In order to describe image segmented areas in discrete attributes, we may use complex methods that use adjacent angles, such as that proposed in [7]. These methods achieve good results, but it is shown that performance of attribute generation is expensive, on average 20 ms for frame. However, the mentioned methods are proposed to identify complex shapes like hand signs, and we are looking for ellipses that are much easier to define. In order to increase the performance we use an aligned patch of binary image with $\Delta \times \Delta$ around the centroid of the best ellipse that fits into each segmented area. Therefore, a sample has Δ^2 binary attributes that indicate whether each pixel of the patch is white or black.

Generation of the Training Examples: The set of training examples is generated by making a movie with IR cut lenses over the interest area containing

the patterns and the background where the AR application will take place. To reduce the size of the BDT, we use a reduced dataset, since we need the classification as fast as possible. However, we are accepting the possibility of having to make new trainings for different applications.

The movie must be recorded with smooth camera movements, so that the bounding area doesn't loose the pattern, what would demand reinitialization, which is a slow process. After that, the invariants' values are measured in some frames with different points of view that were manually chosen. With the measured values, we start the APIP process of extracting and classifying the points in each frame as marker or non-marker. Although this classification is 100% reliable, some segmented areas classified as non-marker by APIP may have a format very similar to an ellipse. Since we don't want that these cases confuse the BDT training, we use a stronger elliptical forms detection algorithm [8] to remove them from the training. We could have changed the class of these points instead of removing them, but we would be assuming the error of the ellipses algorithm, which may identify ellipses that are not very similar to those we want, and potentially augmenting the size of the BDT, unnecessarily. Finally, we generate a file with the attributes for the remaining points and their corresponding classes. This file is the input of the decision tree generation algorithm.

Oversampling: We showed that from the binarized image and the detected ellipses we generated the samples to be classified by the decision tree. In this process, each ellipse generates a single sample. In order to generate more training samples without having to capture more images, we use a sample extrapolation process. In [9] homographies were used to extrapolate samples of other perspectives to a patch, but this works well only in planar situations, which is insufficient for our system. Therefore, we use a more limited hypothesis, derived from the assertion that a patch around a centroid not aligned with x and y is equivalent to a camera rotation over the "roll" axis.

The Decision Tree Learning Algorithm: Rosten and Drummond [10] succeeded in using ID3 algorithm [11] to classify points with the knowledge of neighborhood color. Then we choose the C4.5 algorithm [12] (an extension of ID3) as the decision tree generator.

The learning algorithm generates a tree that can be used directly with the aligned patches of the points detected in real time, but in order to augment the robustness of the process regarding the scale, an additional process was added. The tree theoretically classifies the ellipses of the sizes it was trained. However, there are cases of ellipses larger than those used in the training, that happens when the user gets closer to the interest area. In such cases we use the ellipse's bounding box to calculate a factor to reduce the ellipse to a size a little smaller than the $\Delta \times \Delta$ patch. This factor is used to guarantee that the ellipse reaches a size compatible to the training and that its contour is well defined.

4.2 Pattern Maximum and Minimum Sizes

The excess of noise may generate false markers that pass through the previous tests. To reduce even more this possibility, we define maximum and minimum sizes for the pattern. We suppose that in an AR application it is normal the necessity of having space for a virtual object to be included in the scene, which allows the definition of a maximum size for the pattern. For the minimum size, we use the assertion that when the object is too far, the possibility of detecting the pattern is small. Since this filter is applied before the ordination of the points, we still don't have the organization of the pattern, but just a list of points. Therefore, we make these tests calculating the minimum and maximum x and y for the list of points, which define the bounding box of this group of points.

5 Results

Real time performance is an important requirement of AR applications. For this reason, the evaluation of the proposed algorithm is focused on this requirement. We assume that a real time AR application must be visualized at rates equal or greater than 30 fps, which means that the processing time cannot exceed 33 ms. The processing of an AR application includes not only the pattern identification and tracking, but also complimentary processes, such as the insertion of virtual objects in the scene. Therefore, we assume that the maximum execution time allowed for the proposed algorithm is 15 ms, i.e., half of the maximum processing time allowed. In the following we present an analysis of the individual costs in terms of time of the algorithm's key processes. At the end we can evaluate the limitations of parameters and conditions to accomplish the 15 ms restriction.

5.1 Image Loading and Markers Extraction

In this section we present some measured times for the processes of loading the image buffer, and the segmentation and extraction of elliptical markers in a synthetic image with the following characteristics: a base circular marker named of type 1, marker of type 2 (with the double size of type 1). Some results on these measures are presented in Table 1.

	1	10	20	30	40	50
Markers type 1	1.718	1.891	2.078	2.266	2.437	2.625
Markers type 2	1.719	1.953	2.219	2.468	2.703	2.953
Markers type 1+2	——	1.922	2.140	2.360	2.593	2.813

Table 1. Time (in milliseconds) spent to load and detect circular markers.

Based on the results presented in the table, we may calculate the average relative time to extract a marker as 0.02275 ms. The results also show that

depending on the number of markers and their sizes, the loading and extraction times vary, but the average time estimative indicates the maximum number of markers that may be treated at each frame.

5.2 Collinearity and Projective Invariants Tests

We also measured average times for collinearity tests and for the calculation and comparison of the projective invariant value for a collinear pattern, for collinearity test the mean time was 0.0000703 ms, and for projective invariant test with ordering pattern points: 0.0001600 ms.

These times were calculated as an estimative of the time to evaluate a single collinear pattern composed of 4 markers. As the total number of patterns to be analyzed is obtained by the number of combinations in 4 of the number of markers detected in the image, it is possible to define a formula to find the total cost (in time) and the maximum number of markers that can be present in an image: $C_4^n * (Collinearity\ test + Proj.\ invariant)$.

As may be seen from the above formula, the algorithm must give priority to the reduction of false markers passed to the process of pattern generation. The linear increment of the number n of detected markers causes an exponential increment in the combination of points to be tested as possible patterns. For this reason, the implementation of the BDT test was necessary to remove a large number of false markers.

5.3 Cross Validation

Results were obtained through 5-fold cross validation tests. The Datasets used were five training films of the same scene, recorded with smooth camera movements throughout the scene. This scene has the objects shown in Figures 3 and 1. Each video has around 16 seconds. We couldn't use randomly chosen groups of equal sizes because of the necessity of having different perspectives for the training. The Δ used was 15.

Classification results are described by four indicators: Number of true markers (TM), number of false markers (FM), number of true non-markers (TNM), number of false non-markers (FNM). The size of the pruned tree will referred by PTSize and the quality evaluators are described by:

$$Precision_{NM} = \frac{TNM}{TNM + FNM} \quad Recall_{NM} = \frac{TNM}{TNM + FM}$$

We use the BDT to eliminate non-marker ellipses. This is the reason to analyze its recall and precision. The main requirement of BDT training is that precision be very high, because when the BDT obtains false non-markers, projective invariants don't find the pattern. To achieve real time rates, recall needs to be high enough to remain a manageable number of false markers for projective invariants. Results in Table 2 indicates that these requirements are accomplished.

	PTSize	#frames	TM	FM	TNM	1-Precision$_{NM}$	Recall$_{NM}$
1	77	611	2424	20	85358	0.000199	0.999766
2	81	527	2099	9	76144	0.000276	0.999882
3	81	403	1601	11	47126	0.000297	0.999767
4	77	438	1715	37	49937	0.000400	0.999260
5	89	534	2122	14	68206	0.000337	0.999795

Table 2. 5-folt cross validation

5.4 Case Study

In this section, we test a real case using the BDT training described in the previous section. We recorded a video of 25 seconds with random movements to allow the capture of diverse views of the scenes showed in Figure 3. In the graphic presented in Figure 4, we show that all captured views contain more than 60 markers in the elipse list and after filtered by the BDT, they decrease to an average of 7 markers in the filtered elipse list. Black lines in the graphic indicate views where markers of the pattern were lost, invalidating the detection.

(a) (b) (c)

Fig. 3. Scene Objects.

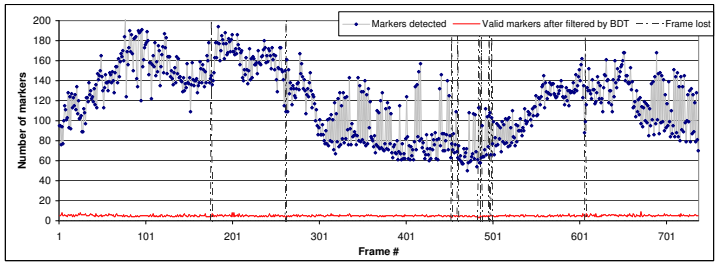


Fig. 4. Attribute generation.

For our experiments we used a modest 1.2Ghz tablet PC, and we obtained the following times: for the executions of the BDT we achieved less than 1 ms per fame analyzed for all frame samples. We also made synthetic tests to find the worst case of $\Delta = 15$ (225 attributes to test) and defined a sample of 400 detected markers per frame. The time needed to filter the sample was 2.4 ms. We concluded that, for the tested hardware, real time rates are achieved if the BDT, the collinearity filter and the size filter allow the passage of no more than 25 markers to be evaluated by the projective invariants.

6 Conclusion

We presented a new algorithm for the detection and tracking of projective invariant patterns, which are used as a support tool to develop AR applications. The advantages of the proposed algorithm is that it uses less "invasive" patterns when compared to planar ones, and allows the detection and tracking in non-controlled environments, differently from IR-based systems. We calculate the computational costs of all the processes that compose the algorithm and show that we can achieve real time rates. We expect to implement complementary algorithms to enhance the identification and tracking process, such as a camera calibration algorithm that uses the information about the markers that compose the patterns.

References

1. Loaiza, M., Raposo, A., Gattass, M.: A novel optical tracking algorithm for point-based projective invariant marker patterns. In: ISVC (1). (2007) 160–169
2. Kato, H., Billinghurst, M.: Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: IWAR. (1999) 85–94
3. Teixeira, L., Loaiza, M., Raposo, A., Gattass, M.: Hybrid system to tracking based on retro reflex spheres and tracked object features (in portuguese). In: X Symposium on Virtual and Augmented Reality - SVR. (2008) 28–35
4. Meer, P., Lenz, R., Ramakrishna, S.: Efficient invariant representations. International Journal of Computer Vision **26** (1998) 137–152
5. Fitzgibbon, A., Pilu, M., Fisher, R.: Direct least squares fitting of ellipses. IEEE Trans. Pattern Analysis and Machine Intelligence **21** (1999) 476–480
6. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 4–37
7. Cho, K., Dunn, S.: Learning shape classes. IEEE Transactions on Pattern Analysis and Machine Intelligence **16** (1994) 882–888
8. Aguado, A., Nixon, M.: A new hough transform mapping for ellipse detection (1995) Technical Report, 1995/6 Research Journal.
9. Ozuysal, M., Fua, P., Lepetit, V.: Fast keypoint recognition in ten lines of code. Computer Vision and Pattern Recognition. CVPR '07. (2007) 1–8
10. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European Conference on Computer Vision. Volume 1. (2006) 430–443
11. Quinlan, J.R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
12. Quinlan, R.J.: C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). Morgan Kaufmann (1993)