Collage of Patterns

SK Semwal D Carroll
Department of Computer Science, University of Colorado, Colorado Springs, USA, 80933
semwal@cs.uccs.edu | *thatsashame@pcisys.net*

Abstract

The work reported in this paper is in the area of camera based Augmented Reality. The paper extends ARToolKits's usage for applications where the one camera can see multiple patterns at the same instant of time. The placement of these patterns can then be changed to convey different information. Thus, several messages can be created by simply changing the relationship among the patterns. As the same set of patterns are used to create a large amount of information, hence the term **collage-of-patterns**. This collage-of-patterns can convey precise information over highly distributed wearable systems. As the message is directly related to the patterns used and their spatial relationship with each other in the collage, the intent could be only be deciphered by the sender and recipient which provides extra security which is always a concern. A larger number of messages can be created with small number of patterns. In future, the same collage-of-patterns can be mapped to several native languages, this creating a means for sharing information among diversified different language. The paper describes the motivation behind our research, discusses our implementation showing feasibility of our idea, discussed the main results, and identifying future research directions.
.

Key words: ARToolKit, multiple patterns for indexing

1. Introduction

Our goal is to be able to provide consistent and precise information using set of patterns using the ARToolKit. The main idea of the proposal is to use existing patterns to create a collage-of-patterns which can convey different messages based upon the placement and relationship of these patterns. This collage-of-patterns needs to be recognized by one camera mounted on a wearable system which can be deployed with the participants. Thus wearable systems can either at the same place in close-proximity or can be separated by miles. The goal is to be able to convey a variety of messages using the collage, and a simple (cheaper) and robust system. In this paper, we show that such an experiment is possible using the ARToolKit [1]. In our experiments, we have found that the ARToolKit can provide robust results by recognizing several arrangements of simple patterns arrangements which are automatically detected [1]. Details of ARToolKit and wellknown MagicBook experiments are described in [1]. Several papers and associated results for ARToolKit and its comparison to other systems have been described elsewhere, e.g. [1]. This paper instead would focus on using ARToolKit for our application. We have been able to make the toolkit work with both the SGI and PC based systems. Using the camera on a PC or SGI system, the toolkit programs can robustly recognize simple patterns, such as black/shaded rectangles or circles on white paper even at acute angles. We have extended this capability so that many different patterns can be simultaneously recognized by the ARToolKit. The shape of these patterns can be of varying sizes and still be recognized by the system for example camera can zoom in/out yet still work robustly. We also found that the performance of the system is affected as the light changes and with the quality of the camera used for experiments. We added more ambient light (table lamps projected on the walls of our laboratory room) and ARToolKit worked well with simple Sony handheld cameras.

As we tested ARToolkit further, we found out that the ARToolKit can detect the same pattern or marker for long periods of time even if the pattern is moved around. The pattern is quickly recognized as it enters the field of view of the camera either due to pan/ zoom, or removal of the obstructing object. Rather than simply recognizing patterns individually, we wanted to recognize a group of patterns. Further, when many of these patterns are present in a frame, their relative positions to one another are taken into account, resulting in many more combinations that can be used for various applications.

Ultimately, we are interested in annotating actual three dimensional terrain by simple arrangement of some landmark objects (patterns) so that an Augmented Reality system such as above would be able to recognize these arrangements automatically and deliver a precise message. The context of the message would be dependent upon the arrangement of patterns. Obvious extension to this idea if to be able to deliver same message in multiple languages based upon what the Augmented Reality systems can see. We have not implemented this idea, this is similar to Kanji character providing same meaning in both Chinese and Japanese languages. Thus obvious use of our research would be to share information among personnel from different countries. Our experiments provide a large number of possibilities by using only a few patterns in the

ARToolKit. A simple pattern recognition application would likely recognize one or more predefined non-complex patterns and act on them when recognized. For example, one might overlay a 3D object at the estimated position of the pattern, and when the pattern moves, the object on the display moves as well. In our program, rather than simply recognizing patterns individually, a group of patterns can be recognized as a single entity. Further, when many of these patterns are present in a frame, their relative positions to one another are taken into account, meaning many more combinations that can be used for various applications.

As mentioned earlier, our implementation is built upon the ARToolKit to handle simple pattern recognition. ARToolKit provides recognition of black and white patterns with a simple 2D object in a square patter with a black border. An ARToolKit program is used to train it to know when that pattern shows up in a frame grabbed from the camera. When a pattern is found to be in the frame, the toolkit calculates its position and orientation. These pieces of information are then used to calculate one pattern's relative position and orientation when compared to others.

We demonstrated a possible use of these techniques in our implementation. A series of patterns are presented to the camera, oriented and positioned in different ways. In the console window, at various points in time when the user wants to, words are presented based on which patterns are present and their relations to the other patterns.

2. Patterns Relations

We are given a pattern's center and its general direction (in the form of a rotation matrix) from ARToolKit when a pattern is recognized. This matrix can be used to create a local coordinate system for the pattern. By projecting the centers of all other markers in the scene onto the axes of the first pattern, (via dot products) we can determine how the other patterns are related to the first one. The signs of the projections determine whether the other patterns are to the right or to the left, or above or below.

If more than one pattern is present in a frame, one pattern's relation to another is defined by one of eight possible relations:

- Directly above
- Directly below
- Directly to the right
- Directly to the left
- Above and to the right
- Above and to the left
- Below and to the right
- Below and to the left

A pattern is directly right/left/above/below another when the other pattern's center is within the bounds (no further than one half pattern width from the center) of the pattern. This means one pattern can be slightly higher than another, yet still be considered directly right to the first pattern rather than to the right and above. If it's beyond the bounds of the first pattern, then it will not be considered as directly in that direction.

Note that if one pattern relates to another second pattern in one way, (say, the second is above and to the left of the first) this does not mean that the relation of the first pattern to the second is the opposite. (i.e., below and to the right) This is because the second pattern may be facing in a different direction than the first, and when related to the first pattern, only the center (not the direction) of the second pattern is taken into account. Take two people in a room, one standing on the floor, the other standing on the ceiling (bear with us here!), The person on the floor says that the person on the ceiling is above them, while the person on the ceiling also says that the person on the floor is above them, since that person's up is different from the person on the floor's up. In the same way, two patterns may have the same relation from each pattern's perspective.

3. Relation Patterns permutation count

Since each pair of patterns can create up to eight different relations between them, we can estimate how many possible combinations exist given the number of patterns that the system wants to handle with just two patterns. Having two patterns creates eight different possible combinations if their local rotations remain fixed. If the second is allowed to revolve, there are eight combinations possible for each of the eight positions relative to the first, making 64 possibilities. To illustrate, take two patterns. You can easily revolve a second pattern around the first into the eight positions. Next, you can also revolve, at each of these positions, the second pattern. Since this pattern has its own local coordinate system, even though neither pattern changes positions, the first pattern will appear to change positions when compared to the second. Thus we have at least 64 combinations with only two patterns used. We can then swap the two patterns to create more possibilities.

Adding a third complicates things even more. To simplify, we can combine the second and third patterns into a single unit, pretending that these two patterns are actually one. This leaves us with the two pattern case above, with 64 cases. Now, consider this two pattern subunit, consisting of the second and third patterns. Again, since there are two patterns, we have 64 cases. It is possible to have each of these cases with each of the cases between the first pattern and the pattern “subunit.” Thus, we now have 64^2 different cases with only three patterns.

One case remains here: The first pattern to the third pattern, which wasn't taken into account previously. This complicates matters, as it appears dependencies crop up compared to the cases already handled, notably, the first to second and second to third. For example, align all three horizontally. If you try to move the third up or down, you change the relation with the second as well as the first. However, if you put the second above and to the right of the first, and then place the third below and to the right of the second, you have more freedom to move the third relative to the first. In short, finding such relationships is a difficult process that is dependent upon how the patterns are placed and their room for movement, as well as the pattern's width. However, it should be pointed out that with only three patterns we are able to create a large number of combinations, thus resulting in a large vocabulary for messages.

In the final analysis, a lower bound can be found to estimate the number of possible combinations. This is a power of 64, namely, 64^{n-1}, where n (greater than one) is the number of patterns that are used. Some of these combinations can lead to up to three more possibilities depending upon where and how they are stationed. Another way to create 64^{n-1} combinations is to use two patterns in a 3x3 block resulting in 64 combination, and use several such 3x3 blocks to create 64^{n-1} combinations.

Moreover, some possibilities are hard to use, especially if there are many patterns in a frame. Using the third dimension is possible, which would create 26 areas instead of just 8. However, we have found that using it with 3D is cumbersome and not intuitive. For most applications, one would work on a flat surface and manipulate the patterns that way. Also, when using ARToolKit, if one pattern occludes another even partially, the occluded pattern is no longer recognized. So, using it in such a manner would not advisable, though it would be easy to implement, as the application is provided the information necessary and it is just a simple extension of what was already done.

4. Attaching Words to Relational Pattern Structure

Let's say we want the word "Guttywwig" to appear whenever pattern A is above and to the right of pattern B and whenever pattern B is directly below pattern C. We have two conditions that need to be fulfilled in order to be able to show that word. Furthermore, each condition consists of two markers and a relation. (directly above/below, above and to the left, etc.) We can also attach any number of other conditions (say, pattern A has to be to the right and above pattern D, etc.) that need to be satisfied. Thus, a relational pattern would have to be able to have any number of conditions, and each condition would need a pair of patterns and a relation.

The coded structure of a relational pattern is done in much of the same way. One relational pattern consists of any number of pattern pairs (with a relation) that all have to be satisfied so that the entire pattern can be considered present. Each relational pattern has a linked list of each of these relational pattern pairs. When the program receives a frame, it finds out which patterns are present on it. Then, with the patterns it knows are in the frame, it calculates the relative positions of each pattern when compared to all other patterns. This information is stored in a table. Last, it goes through the list of relational patterns, seeing which conditional pattern pairs are satisfied. If even one in a relational pattern is not satisfied, the relational pattern is marked as out and processing continues onto the next one, meaning no processing is done for any of the other pairs.

4. Results

As a first example, we arranged four different patterns on a surface (Figure 1). When the user presses the space bar, the current frame is examined, analyzing the different relationships between the four patterns. If certain relationships are found to be in the frame, various words are printed to standard output. In the first figure, the four patterns appear with their local axes drawn over them.

Figure 1: Four patterns create associated word From left to right, the numbers corresponding to each pattern in the figure are 0, 1, 3, 2

The relationships are specified by the programmer in code similar to this statement:

```
if(!(lp.addPositionMarker(0, 1, MRK_HIGHER,
"Lamp") &&
 lp.addPositionMarker(1, 2, MRK_SAME_VERT,
"Saggy") &&
 lp.addPositionMarker(2, 3, MRK_RIGHT, "Corporeal")
&&
```

```
 lp.addPositionMarker(0, 1, MRK_LOWER, "Wet") &&
 lp.addPositionMarker(1, 2, MRK_SAME_HORIZ,
"Messerschmidt") &&
 lp.addPositionMarker(2, 3, MRK_LEFT, "Lincoln")))
```

The first call to addPositionMarker tells it to print "Lamp" when pattern 0 is higher than pattern 1, the second tells it to print "Saggy" when pattern 1 is about the same level vertically with pattern 2, and so on. The first figure specifies the first three calls in the above statement, meaning the words "Lamp Saggy Corporeal" appear.

After moving the patterns around a bit, we can get a second example (Figure 2):

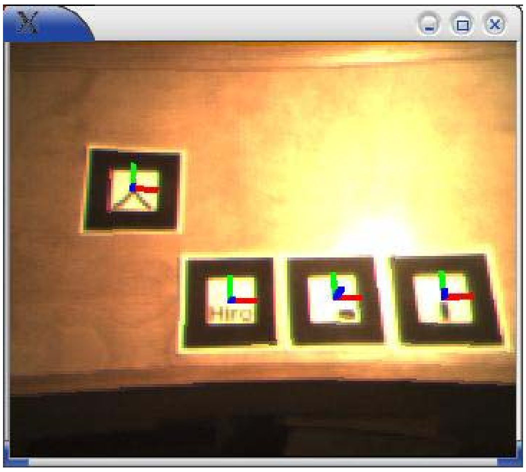

Figure 2: Produces "Wet Messerschmidt Lincoln.

This one produces the words "Wet Messerschmidt Lincoln." Other combinations can be easily created as well.

5. Conclusion

Instead of just using each pattern as one unit, we can combine two or more patterns and place them at different positions relative to each other, each relationship representing something different. For example, pattern A can be above pattern B, and this represents one thing. Pattern A can also be below B, to the right of B, to the left of B... etc. With two patterns, we have just created many more possibilities than just using each pattern as a single unit.

In our example application, four patterns are arranged in various positions and orientations on a surface. The user then requests that the patterns and their relationships to each be analyzed. If certain relationships are found, it prints out a word. Rearranging the patterns will produce different words and outcomes.

What makes this a powerful tool is that only three patterns can be used to represent 64^2 different things. Both augmented reality and pattern recognition are costly both computation-wise as well as being intense in memory. Having a different pattern for each different scenario means comparing each frame to each pattern, clearly not a quick procedure. Here, we gather quick and useful information of a frame through the utility ARToolKit, using only a handful of patterns. Then, it analyzes the relative positions of each pattern compared to the other patterns, and does something specified by the programmer when certain relationships are found. This can be handled in a timely manner, since few patterns are kept in memory to be compared to each camera frame, making this ideal for real-time applications. Being able to generate a large number of combinations in a small space has several applications in virtual environments specifically in navigation and wayfinding [2]. Our research is directly applicable to a) in accessing large database; b) changing the context and meaning of a situation with subtle positional changes of patterns; and c) physical and network security applications.

6. References

1. I Poupyrev, DS Tan, M Billinghurst, H Kato, H Regenbrecht, N Tetsutani, Developing a generic Augmented Reality Interface. Pp 44-50, IEEE Computer March 2002

2. Rudolph P. Darken and Terry Allard. Spatial Orientation and Wayfinding in Large-Scale Virtual Spaces: An Introduction. Presence Teleoperators and Virtual Environments, 7(2), pp. 101-107 (1999).

7. Acknowledgements

We wanted to thank Dr. Bill Ayen and NISSC for supporting the seed grant. Dave Lohman was critical assistance in dealing with SGI issues of our project. We also gratefully acknowledge SGI's Richard Grossen; Shane Kopeland of James River Technical; Gwen Pech and Jim Bryant at the BP Center of Visualization for a variety of help on this project. Special thanks are due to Dr. Terry Boult in providing several cameras for our impmenetations.

This research is partially funded from the research sponsored by the Air Force Research Laboratory, under agreement number F49620-03-1-0207. The US government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory of the U.S. Government