Using Augmented-Reality on Planar Surfaces for Previewing Decor Changes

Nima Soltani, and Mehmet Yilmaz

Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA

It is difficult for consumers to judge whether a painting or poster would fit well in their rooms without actually mounting the painting on the wall. A solution is proposed to allow users to their DROID mobile phones to preview what the painting looks like on the wall using the phone's camera and its screen. The system is a marker-based, augmented-reality system that can place paintings on walls in real-time. It gives the user an accurate estimate of how the painting would fit in terms of size on the wall, as well as an approximate estimate of how the colors on the painting would look on the wall. It can construct both homographies and affine transforms to map the painting onto the wall.

***Index Terms*—augmented reality, fiducial markers, pose estimation, virtual reality**

I. Introduction

WHEN CONSUMERS look to purchase paintings or posters, an important consideration is whether the painting would go well with their current décor. A customer's choice of a painting would depend on criteria such as its color, size and content, in comparison to their room. The current method is to take a picture of the room to the store to get a good idea of what would go well or to take the painting home to see how well it fits in the room. This project aims to address this uncertainty, giving more confidence to consumers when shopping for paintings while making the process of selling paintings a more efficient one for vendors.

Using the DROID smartphone's image-processing capabilities, this project aims to superimpose the painting on the wall in real-time. The simulated painting is intended to be as realistic as possible. This means the orientation of the painting should be such that it fits the wall, and the size of the painting should be to scale. Also, the simulated painting should be adjusted so its colors appear correctly in the lighting conditions of the room.

A. *Prior and Related Work*

Augmented Reality, the overlaying of objects onto the real world, has been a popular area of research since handheld devices came out to market. Augmented Reality can be used for different applications such as navigation, indoor decoration, interactive games etc. In AR systems, it is essential to align the virtual objects properly with respect to the real world to create a perception that the two worlds coexist, which requires the camera pose and intrinsic parameters to be estimated accurately in every frame. Most of the vision-based techniques that tackle this problem can be classified as markerless and marker-based AR.

Markerless AR systems [2][3][4] rely on tracking natural features on the scene to compute the 2D homography which is further used to update the camera pose that is estimated initially using points whose real world coordinates are known. However, the disadvantage of feature-based methods is that they require a textured scene in order to have sufficient features to track the scene. Furthermore, they suffer from the drift effect, which means that the estimation slowly diverges from the actual solution.

In order to overcome the aforementioned problems, a model of the object that is being tracked can be used. Marker-based systems are a special case of this technique, which use fiducial markers with known real world coordinates to keep track of the camera pose. ARtoolkit [7] is a popular such method, which can give a real time performance on current PCs and relatively high frame rates on handheld devices. In ARtoolkit, binary rectangular markers with various patterns are employed. While the corners of the marker is used to compute the pose of the camera, the pattern on the marker helps to reduce the false positive detections and to distinguish between different markers that are put on the scene. Even though it is the most popular AR system, ARtoolkit ceases to give reliable results under less-controlled lighting conditions as it uses grayscale intensity threshold to localize the marker.

ArTag [5] is another marker-based technique which follows a similar line of method as its predecessor. However, it can detect the markers in a more robust way as it uses edge pixels for thresholding the image and replaces random patterns on the marker with binary which improves false positive detection and allows for a larger library of markers. Studierstube [6], a more recent framework, utilizes the advantages of both of the previous methods. While it uses binary data as marker pattern, its detection process is built on the ideas in ARtoolkit.

Not all the marker-based systems rely on corner detection and binary markers. Sticker et al. [8] perform a line tracking on the marker and use the orthogonality of the sides of the marker. Furthermore, they use a colored barcode on the marker as their pattern.

II. Design Decisions

Even though markerless AR systems are able to perform very accurately in certain settings and compute the pose on the planar surfaces without the need of any user interaction, the necessity of abundant features on the scene makes it unsuitable for our application, which basically tries to augment paintings on a background, which is most of the time a plain wall. In addition, we would require the application to run almost in real-time on a handheld device, which mandates the use of a computationally simple method. Marker-based systems simply provide good results under these two constraints. First of all, they can locate the marker by using simple techniques that don't require much computation. Moreover, they can almost always give reliable results when the marker is in the viewpoint of the camera.

However, the current state-of-the art markers have binary patterns on them mainly to distinguish several markers on the scene. Since for our application we always used a single marker, we constructed a new marker for our project. The marker we used can be seen below in Figure 1. It consists of four black rectangles evenly spaced about the page. The center of the page is a white area the same size as the black rectangles.

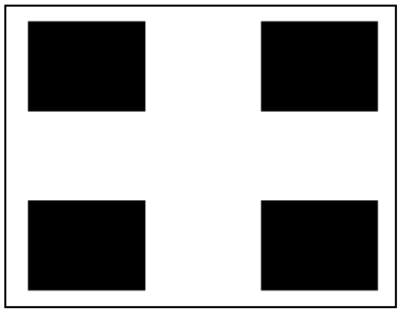
Figure 1. Marker used in system

A. System overview

The system can be generally classified into 5 separate stages: image capture, marker detection, transform detection, color correction and painting augmentation.

1) Image Capture

While ideally one would like to use a full-resolution image in the calculation for both aesthetic and precision reasons, doing so would slow the program down significantly. Therefore in order to be able to use the program in a more interactive, real-time fashion, the image is sub-sampled by 2 both vertically and horizontally. The camera on the DROID mobile phone gives the image in YUV420P format so it is necessary to convert this to RGB or grayscale to be able to do our image processing techniques.

2) Marker Detection

In order to take advantage of the structure of the marker, the first step taken is to threshold the grayscale image to obtain a binary image. While this step makes the application more sensitive to lighting conditions, it saves computation time over a gray-level edge thresholding like in ArTag. Moreover, the threshold can be made adaptive to increase its robustness [11]. Simple horizontal and vertical edge detection is applied to the binary image, which is further fed to a region labeling algorithm. The reason we computed regions on edge pixels in an untraditional way, is to save computation time as we get to process very few pixels.

After eliminating the outlines with extreme areas, we apply the corner detection method in [6] to detect the quadrilaterals. First, we estimate the centroid of the outline as the mean location of the pixels on the outline. Then we pick an arbitrary point on the outline and search for the most distant point to that. In a convex outline, this point has to be one of the corners. Then we draw a diagonal between the centroid and this corner and look for the points on each side that have the largest vertical distance to the diagonal. These two corners are used to construct more diagonals and we keep searching for other corners recursively. The procedure is explained in the figure below.

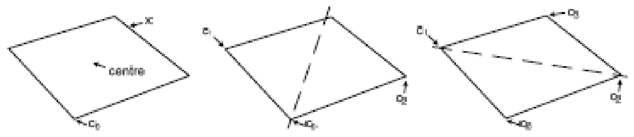

Fig.2 Detecting corners of a quadrilateral [6]

Outlines that don't have just four corners are eliminated and the remaining outlines are processed for further verification. First, we check the pixel value variance inside the quadrilaterals and keep only the ones with very low variance. Then we compare the lengths of parallel sides which we require to a have a difference less than a predefined threshold. In the last step, we keep four of the remaining outlines that have a very similar area. Thus if we lose track of one of the quads on the marker, we assume that the detection process has failed.

3) Transform Detection

After locating the corners of the marker, there are 16 points for which mappings can be defined. The Direct Linear Transformation (DLT) algorithm (Hartley, 2004) can be used to find the mapping from the source plane to the plane of the wall, which uses the corners on the image $\boldsymbol{x_i} = (x_i, y_i, 1)$ and their matches in the marker domain $(x_i^{'}, y_i^{'}, 1)$ to build the $2n\,x9$ matrix below, whose nullspace defines the transformation among the domains. The transformation either can be restricted to be affine or can be computed to its most general form, namely homography.

$$A = \begin{bmatrix} \boldsymbol{0^T} & -\boldsymbol{x}_i^T & y_{\boldsymbol{i}}^{'}\boldsymbol{x_i^T} \\ \boldsymbol{x}_i^T & \boldsymbol{0^T} & -x_{\boldsymbol{i}}^{'}\boldsymbol{x_i^T} \end{bmatrix}$$

Affine transformations can only capture scaling, translation and rotation, and has only 6 degrees of freedom. Thus affine requires only three point correspondences, resulting in 6 variables. The DLT method, however, uses a singular value decomposition calculation allowing us to use all 16 points to give a more accurate estimate of the coefficients.

Another transformation that was implemented was a homography. A homography can capture the same effects as an affine, but it can also capture perspective transformations as well. There are eight degrees of freedom in this case, which means that only four points are necessary, but again we use all

16 points in the DLT to get a more robust estimation of the homography.

In reality, for both affine transformations and homographies, there will be relatively good and bad corner estimates. Since SVD will not be able distinguish between them, it is possible for an inaccurate corner to badly skew the coefficients. In order to minimize the effects of this, the DLT is computed iteratively. First the DLT is computed using all 16 points, and then the estimated transformation is used for the reprojection of the virtual coordinates on the image. The error between the estimated corners and the image corners is then obtained and the 12 points with the best estimates are used to compute a new homography or affine transformation. This process is intended to selectively remove outliers, assuming that most corners in the image are reasonably accurate in their location.

Additionally, in order to allow the user to move the painting around on the wall without moving the marker by hand, the code can virtually move the painting with respect to the center of the marker in the source plane, so the picture is consistently placed in the intended position despite the relocation of the camera around in the room.

4) Color Correction

In most lighting conditions the colors of the room will not span the whole [0,255] range, in which case, color matching algorithms that depend on the colors of the whole scene will not work. Instead, it is possible to exploit the colors on the marker itself as upper and lower limits: use the white portions of the marker as a measure of the maximum color value and the black portions of the marker as a measure of the minimum color value. Afterwards the each color used in the image can be component-wise transformed via an affine transformation to lie between the minimum and maximum colors.

Choosing the good values for estimates of the minimum and maximum RGB components in the marker is important, and as such it is not sufficient to use a single detected minimum or maximum value as the maximum, because the image becomes very sensitive to noise, which is especially prevalent in poorer lighting conditions. The method developed for this project instead takes a 3×3 region centered on the marker (white region) and takes the averages of the three largest values for the RGB channels (independently) and uses these RGB values as the maximum RGB values. Similarly for the minimum RGB values the 3×3 region is chosen in the black regions and the averages of the 3 lowest RGB values are used as the minimum RGB values.

5) Painting Augmentation

The final stage is to actually apply the homography or affine transforms and do color correction. The resulting image is finally drawn on top of the rest of the scene uaing the GL library in Android and the process is complete.

III. Results

In this section the performance of the system on the DROID mobile phone will be discussed in terms of speed, and accuracy. Finally some sample results of the system are shown, both for affine transforms and homographies.

A. Performance

The overall frame rate of the system ended up being 2-3 frames per second, which was much lower than was anticipated. There were many factors that slowed down the performance, most of which were due to limitations of the mobile platform itself.

The image itself needed to be converted to RGB or grayscale, which is a slow process in itself. Furthermore, transforming the virtual painting and augmenting it to the picture consumes considerable computation time. Also, the code in its current state searches the image whole image for the marker without considering the previous position obtained. We believe that reducing the search area using the previous position information can lead to a slight increase in the frame rate.

Nonetheless, we made several improvements to speed up the process. For instance, in the region labeling step, the regions that were labeled were the edges as opposed to the binary data; also, by ignoring regions that were too small or too large, plenty of extraneous computations were avoided. However, due to lack of time we couldn’t spend much time on the optimization of the method and the code.

B. Accuracy

For the most part, the obtained results had a high-level of accuracy, in that the painting actually seemed like it was on the wall. Examples of the output using affine transformation can be seen in Figure 2 and sample results for a homography can be seen below in Figure 3.

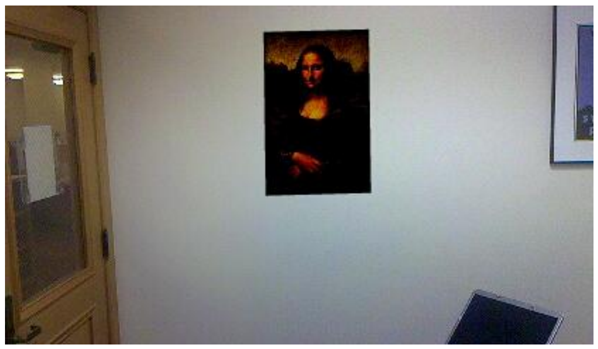

Figure 2. Sample output from affine transform.

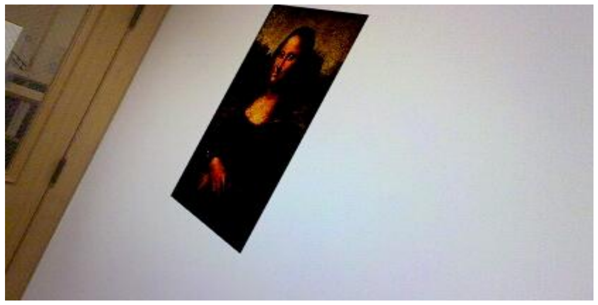

Figure 3. Sample output from homography.

The accuracy, however, did suffer some drawbacks, especially as the phone got farther and farther away from the wall. Increasing the distance to the wall caused problems because the resolution of the image was too low, and thus side lengths became on the order of 10 or fewer pixels. In these cases, misestimating the location of the corner has a much more significant impact on the homography/affine transform calculations than in the case where the side lengths are very large. To have a better transform calculation, the location of the corner would have to be estimated to subpixel accuracy. The Harris detector was investigated to try to achieve this requirement but it did not prove to be any more accurate than the method already implemented.

In order to mitigate the effect of this problem, the DLT is performed on 16 points when in fact only 3 or 4 are required (3 for affine and 4 for homography). The redundancy should help the DLT get a better estimate of the transform matrix or homography, but in order to get an even better estimate of the transformation, we find the error in projecting the marker from source plane onto the plane of the wall, and use DLT again to calculate a new estimate of the transformation, but using the 12 points with least error in the previous mapping. This helps improve the accuracy.

The problem mainly affects the homography calculations, and it causes the painting to have a bit of jitter between frames. The affine transformation is not affected because it assumes the wall is vertical as well as the camera.

IV. Future Directions

Even though in our work, we only computed the 2D affine and projective transformations of the marker, the method can easily be extended for 3D camera pose estimation and the overlaying of 3D virtual objects. We already computed the calibration parameters, and the pose estimation can be done using a slightly modified version of the DLT algorithm [1], which we are currently working on. Moreover, we can implement a Kalman filter of the camera pose matrix, which would help us to estimate the location of the marker in the next frame and reduces the search area to a neighborhood of the estimated position, which would hopefully boost current frame rate. In addition, the built-on sensors of Droid like e-compass and the accelerometer can be employed to assist the camera pose estimation. In the literature, there are many papers [9][10] which report increased accuracy by using such hybrid methods. Sensor information and the visual marker location can be merged together after by means of nonlinear filters like EKF and Particle filters after computing the necessary calibrations.

V. Conclusion

The proposed system met the needs for this application: it provided a good estimate of what the painting would look like if it were mounted on the wall, both in terms of size and color. The solution is simple enough to be easily used by consumers (who of course have DROID phones), as its only extra requirement is that a sheet of paper with a marker printed on it be taped to the wall. The frame rate achieved was lower than intended, but it is still a real-time application. Perhaps more experienced Android developers would have been able to come up with higher frame rates by using better data structures to store the data. The accuracy could be improved by trying other methods of determining corners to sub-pixel accuracy.

Appendix

Nima:

- modified the EE368Viewfinder project to get frames of data to the application
- integrated the linear algebra library, ojAlgo, into the project and found ways of using the touchscreen on the DROID
- implemented the DLT methods to obtain homography/affine transform
- applied the transform to image and displayed it on screen
- implemented color matching method

Mehmet:

- found and read many papers to get a good understanding of problem, to use the DLT to get homography/affine transform
- implemented region labeling
- implemented quadrilateral detection
- researched increasing accuracy of detected corners

Nima and Mehmet:

- read papers
- debugged
- wrote proposal/poster/report

References

[1] Hartley, R. (2004). *Multiple View Geometry in Computer Vision.* West Nyack, NY, USA: Cambridge University Press.

[2] V. Ferrari, T. Tuytelaars and L. Van Gool, "Markerless Augmented Reality with A Real-Time Affine Region Tracker," *Proc. IEEE and ACM Int"l Symp. Augmented Reality,* vol. I, IEEE CS Press, Los Alamitos, Calif., 2001, pp. 87-96.

[3] Simon, G.; Berger, M.-O.; , "Pose estimation for planar structures," *Computer Graphics and Applications, IEEE* , vol.22, no.6, pp. 46-53, Nov/Dec 2002

[4] Vacchetti, L.; Lepetit, V.; Fua, P.; , "Combining edge and texture information for real-time accurate 3D camera tracking," *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on* , vol., no., pp. 48- 56, 2-5 Nov. 2004

[5] Fiala, M.; "ARTag, a fiducial marker system using digital techniques," Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on , vol.2, no., pp. 590- 596 vol. 2, 20-25 June 2005

[6] Schmalstieg, Dieter; Wagner, Daniel; , "Experiences with Handheld Augmented Reality," Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on , vol., no., pp.3-18, 13-16 Nov. 2007

[7] Kato, H.; Billinghurst, M.; , "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on , vol., no., pp.85-94, 1999

[8] D. Stricker, G. Klinker, D. Reiners, “A Fast and Robust Line-based Optical Tracker for Augmented Reality Applications”, Proc. 1rst International Workshop on Augmented Reality (IWAR'98), San Francisco, Nov. 1998, pp. 31-46.

[9] Satoh, K.; Anabuki, M.; Yamamoto, H.; Tamura, H.; , "A hybrid registration method for outdoor augmented reality," *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on* , vol., no., pp.67-76, 2001

[10] Ribo, M.; Lang, P.; Ganster, H.; Brandner, M.; Stock, C.; Pinz, A.; , "Hybrid tracking for outdoor augmented reality applications," *Computer Graphics and Applications, IEEE* , vol.22, no.6, pp. 54- 63, Nov/Dec 2002

[11] Pintaric, T.; , "An adaptive thresholding algorithm for the augmented reality toolkit," *Augmented Reality Toolkit Workshop, 2003. IEEE International* , vol., no., pp. 71, 7 Oct. 2003