LEGO AR WORLD

Nimay Kumar, Srinjoy Majumdar, Christopher Mao

ABSTRACT

Mixed reality is a new technology that has great potential. Unlike its predecessor, augmented reality, mixed reality anchors virtual objects to the real world and allows users to interact with them. We chose to use the techniques we learned in class to create a working iOS app that helps users to build LEGO structures by guiding their progress in real life with augmented reality models. The three stages of our app development were feature detection, image fitting and position matching, and 3D augmented reality. After testing MATLAB, OpenCV in Python, and Apple's ARKit, we realized that ARKit offered the best software development tools as well as effective image processing tools such as binary descriptors.

1. INTRODUCTION

In this paper, we will discuss the development of our app, Lego AR World, that we have developed for our Image Processing Project. This game helps users to build complex LEGO structures by guiding their progress in real life with augmented reality models. As the user builds up their LEGO structures in real life using bricks, they will be able to point their phone at their structure and get the full structure superimposed virtually. We will first explain our methods and then go over our results. From developing Lego AR World over the past two months, we have learned a lot about feature detection and optimizations in the video domain. We made the jump from Matlab to OpenCV in Python to Apple's ARKit. Below, we will highlight our key algorithms and optimizations. Then we'll go over our results and conclusions.

2. DEFINITION AND IMPLEMENTATION OF MIXED REALITY

Virtual reality immerses users in a fully artificial digital environment. Meanwhile, AR (augmented reality) overlays virtual objects in the real-world environment. What we have accomplished, on the other hand, is MR (mixed reality). This cutting-edge technology not only overlays but also anchors virtual objects to the real world and allows the user to interact with the virtual objects. While there are countless applications, we will be displaying some of the technology's potential through LEGOs.

3. METHODS

Since the project was very ambitious, we approached it in three stages: keypoint detection, image fitting, and 3D augmented reality.

3.1 Stage 1

In this stage, we created a prototype iOS application which is capable of recognizing the position of Lego Bricks in a picture. To do this we tried to use a SIFT detector to detect keypoints on the bricks. SIFT filters the image using several Difference of Gaussian (DOG) filters to detect keypoints in the image for an object. It then matches the keypoints from the model to the actual image to determine the transformation of the brick.

We first started working with MATLAB. Soon after, however, we discovered OpenCV (Open Source Computer Vision Library) in Python. This library included a great range of existing algorithms and operations.

3.1.1 SIFT

The first algorithm that we used from OpenCV was SIFT (Scale Invariant Feature transform). SIFT is used for visual searching and object matching. It can find objects that have been rotated and scaled [1]. It's a method for extracting distinctive invariant features from images that can be used to perform matching between different views of an object/scene. The features extracted are invariant to image scale and rotation, and they provide robust matching regardless of change in 3D viewpoint, addition of noise, and change in illumination. Features

are highly distinctive such that a single feature is very likely to be correctly matched against a large database of features from many images. SIFT object recognition works by matching the extracted individual features to a database of features from known objects "using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and then performing verification via least-squares solution." This allows robust object identification even among clutter and occlusion while achieving near real-time performance.

There are four steps. Step one is scale space extrema detection, which involves identifying potential interest points that are invariant to scale and orientation using a difference of Gaussians. Step two is keypoint localization, in which we perform a detailed fit to nearby data of each keypoint for location, scale, and curvature. During this step, some initial keypoints are rejected. Step three is orientation assignment. Orientations are assigned to each keypoint location based on image gradient direction. In step four, keypoint descriptor, we compute a descriptor from the local image gradients around each keypoint.

We found that an issue with SIFT was that feature extraction became a bottleneck since the code took a while to run, which is an issue in real-time video processing since the program needs to continuously run.

3.1.2 SURF

To fix this issue, we considered using SURF (Speeded up Robust Feature) [2]. SURF contains three steps: feature extraction, feature description, and feature matching. Similar to SIFT, SURF is useful for detecting and matching features since they're invariant to scale, rotation, translation, illumination, and blur. While SIFT is better in different scale images, SURF is better in rotation invariance, blur, and warp transform. Moreover, SURF is slightly faster than SIFT since it uses integral image-based box filtering.

3.1.3 ORB

In the end, we decided to use a binary descriptor, called ORB (Oriented FAST and Rotated BRIEF). ORB is a good alternative to SIFT and SURF since it involves a less computationally intensive keypoint detection and descriptor than SIFT, although it's less robust than SIFT [3]. SIFT imposes a large computational burden, especially for real-time systems and low-power devices. ORB builds on the FAST keypoint detector and the BRIEF descriptor, both of which have notably low cost. This allows ORB to be two orders of magnitude faster than SIFT. Upon implementing this feature detection method, we achieved a 2x faster runtime on feature matching. At this point, we were successfully able to track the position of LEGO bricks.

Figure 1.

Feature Detection Method	**Frames Per Second (FPS)**
SIFT	4 - 5
ORB	9 - 10

3.2 Stage 2

In this stage we built upon Stage 1 by placing a virtual model of a LEGO structure, built using LEGO Digital Designer and LeoCAD on top of the physical brick detected. We transformed the model of the full structure the user is trying to build according to the transformation we got from the ORB detector in Stage 1. After transforming the model, we projected the 3D model onto the 2D image so that we could simulate the 3D structure being on top of the brick in the image.

3.2.1 FLANN Point Matching

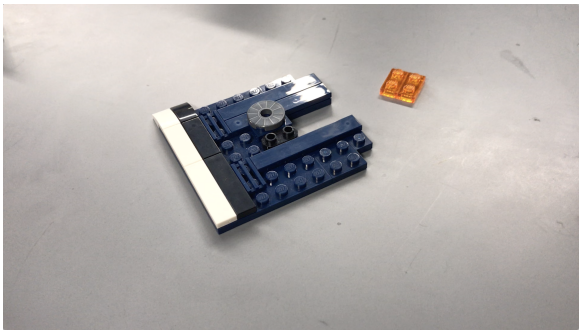
We used FLANN (Fast Library of Approximate Nearest Neighbors) for detecting the homography and matching keypoints we found between the two images. FLANN consists of algorithms optimized for fast nearest neighbor search and high-dimensional features. Compared to matching keypoints using a brute force method, we found significant speed improvements with this approach.

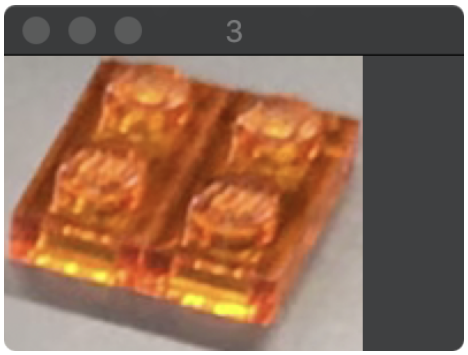
FBM (FlannBasedMatcher) was used to train a nearest-neighbor-index on a train descriptor collection. It uses its nearest search methods to find the best matches, making it an order of magnitude faster than the second best matcher.

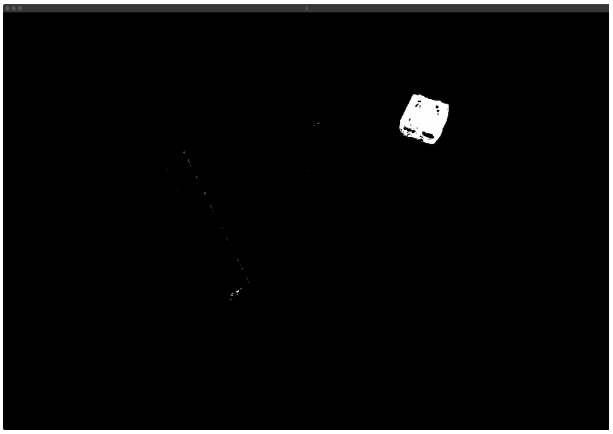
3.2.2 Color Filtering and Image Cropping

The final optimization in stage 2 was HSV color filtering and subsequent image cropping on video frames. We recognized that the LEGO bricks were typically in a small portion of most of the video frames, so we didn't need to find and match keypoints across the whole image. Instead, we filtered the image based on the color of the brick we were looking for. This provided us with a binary image. We then found the largest blob in the binary image and cropped the image to only encompass that blob. This vastly improved the speed in the processing pipeline and allowed us to finally run our app in real-time.

Figure 2.







3.3 Stage 3

In the final stage of our project, we applied the work from Stage 1 and 2 to an AR iOS application. After detecting the position of the user's physical brick, we superimposed the model of the structure onto the AR scene. We used Apple's ARKit to keep track of the 3D AR scene, so we didn't need to project the model onto the image like in Stage 2.

ARKit is a set of software development tools that enable developers to build AR apps for iOS [4]. It uses a binary descriptor similar to ORB, except that it processes images every 5 frames and motion senses each frame. This leads to essentially no lag in the program. To detect the object from all angles, we used an iPhone to scan all around, picking up many features that can be used for detection. Unlike other ARKit developers, we do plane detection via

ARPointCloud, which detects feature points and then clusters rather than just using models. These cluster points can be used to detect surfaces and planes such as tables and walls. Essentially, we place a model on a play, and then coordinates on the model.

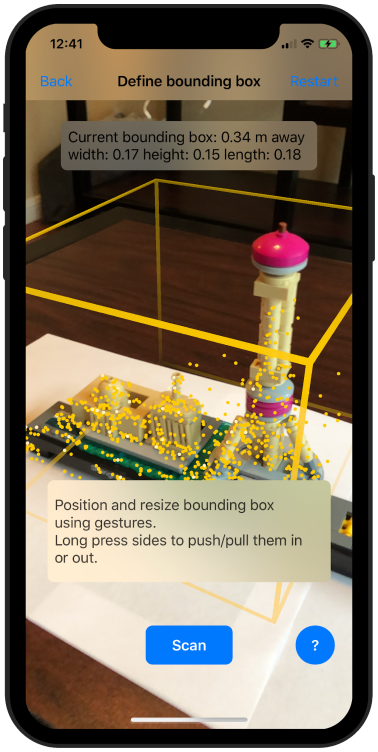
4. RESULTS

With the three stages complete, we produced a working iOS app. The change towards binary detectors and optimization using HSV color filtering greatly increased the speed of our app. These methods allow us to do real-time augmented reality as well as smooth tracking.

5. CONCLUSION

In conclusion, we learned about different descriptors and methods for image processing. We have also all become familiar with coding in MATLAB, Python, and Swift. We can use OpenCV to detect objects in images and ARKit to develop AR apps.

Figure 3.



6. REFERENCES

[1] DG Lowe, “Distinctive Image Features from Scale-Invariant Keypoints.” International Journal of Computer Vision, 2004 [Online].

[2] D. Mistry, A. Banerjee, “Comparison of Feature Detection and Matching Approaches: SIFT and SURF.” GRD Journals, vol. 2, iss. 4, Mar. 2017 [Online].

[3] Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011 [Online].

[4] J. Kim, H. Jun, “Implementation of image processing and augmented reality programs for smart mobile device.” International Forum on Strategic Technology, 2011 [Online].