# GroZi

## Grocery Shopping Assistant for the Blind

**Faculty Advisor: Serge Belongie**
**Qualcomm Consultant: Jeffrey Su**
**Robert Tran**
**Charles Taira**

### Motivation

The GroZi project aims to develop computer vision-based technology to assist the blind with grocery shopping. Currently, blind patrons require a clerk to escort them around the store and pull items from the shelves. This can lead to a number of problems: for example, a blind shopper may intend to purchase Cheerios cereal, but instead the clerk may accidentally select Honey Nut Cheerios. While this example is trivial, it extends to other, more serious situations, such as diabetics or other customers with serious dietary constraints. With the advent of advanced computer vision and pattern recognition, we can extend this new assistive technology to help blind users identify items on their own in the grocery store. The GroZi project hopes to provide this independence for blind shoppers.

### Past work

Multitouch Prototype

We have developed a multi-touch solution that met our basic goals which are to identify the product and produce an auditory cue to guide the user in grabbing the product. However, we later learned that this solution was very limited in portability and viability, as the system required 2 cameras and a multi-touch surface to be mounted on the shopping cart, which was difficult to calibrate. These drawbacks among others called for a different solution.

Grozi 2.0

We recognized the hardware complications with the previous solution and decided to pursue a pure software approach. We purchased an Android tablet to circumvent the hardware issues from the previous solution and developed an application instead. The Grozi application offered the same functionality of the multi-touch solution. It was a more portable (can be downloaded from the Android market), less error-prone, and overall a much cleaner solution. Though it addressed the basic requirements, the application was not very intuitive or user-friendly. To use the application, the shopper had to use one hand for holding the tablet and another for navigating the tablet to identify the product. We had demoed the application to our stakeholder who was able to successfully grab the correct item off the shelf, but only through repeated instruction. From the demonstration, we realized the need for a

feature to identify products automatically.

**Proposed Solution**

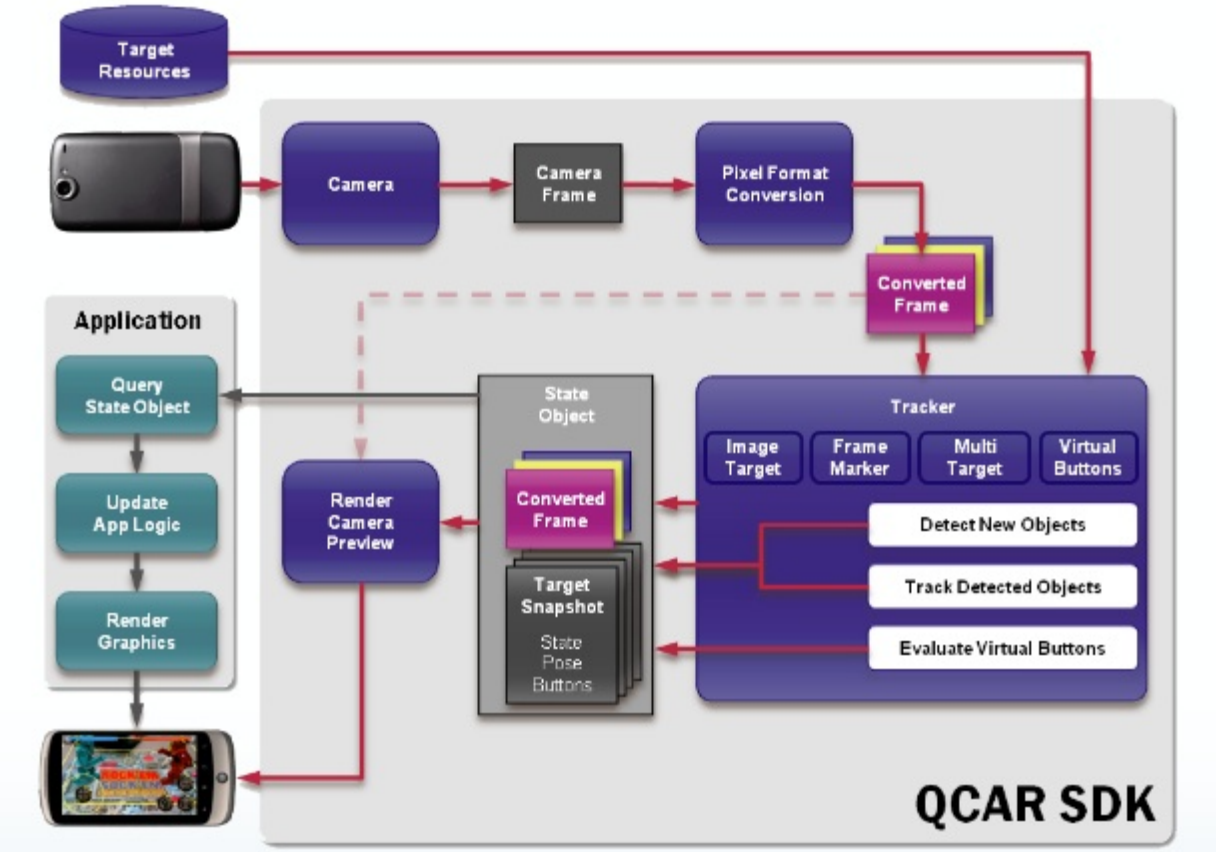
We needed to address the user-experience concern of our existing GroZi application of requiring both hands to navigate the application. We decided to add an "auto-detection" feature, which would essentially play the appropriate sound upon detection of the grocery product as opposed to playing the sound only when the user to touches the tablet screen to find the product. We believe employing the auto-detection feature will offer more flexibility to the blind user, freeing one hand to grab the product.

**Implementation**

We have developed the GroZi application using the Qualcomm Augmented Reality SDK. Qualcomm's AR SDK provides computer vision technology to align graphics with printed surfaces and simple 3D objects, support for multiple development tools, and royalty-free development and distribution. A QCAR SDK-based AR application is composed of the following core components:

1. *Camera* - Ensures that every preview frame is captured and passed efficiently to the tracker.
2. *Image Converter*- The pixel format converter converts between the camera format to a format suitable for OpenGL ES rendering and for tracking.
3. *Tracker*- Contains the computer vision algorithms that detect and track real world objects in camera video frames. Based on the camera image, different algorithms take care of detecting new targets or markers, and evaluating virtual buttons. The results are stored in a state object that is used by the video background renderer and can be accessed from application code.
4. *Video Background Renderer*- Renders the camera image stored in the state object. The performance of the background video rendering is optimized for specific devices.
5. *Application Code*- Application developers must initialize all above the components and perform following key steps in the application code:
    a. Query the state object for newly detected targets, markers or updated states of these elements
    b. Update the application logic with the new input data
    c. Render the augmented graphics overlay

6. *Target Resources*- Created using the on-line Target Management System. The downloaded assets contain an XML configuration file - config.xml - that allows the developer to configure certain trackable features and a binary file that contains the trackable database. These are compiled by the application developer into the app installer package and used at run-time by the QCAR SDK.




*Data flow diagram of the QCAR SDK in an application environment*

Auto-detection feature

To have the application play a sound upon detection, we needed a way to determine the correct sound to play and create an asynchronous thread to play the sound.

To obtain the name of the correct sound file, we modified the native image renderer function to return a JString object containing the name of the detected image. Upon detection of a recognized image, the ImageTargetsRenderer object will invoke the native renderer function which will return the name of the image in string form.

Native function header:

```
JNIEXPORT jstring JNICALL
Java_com_qualcomm_QCARSamples_ImageTargets_ImageTargetsRenderer_renderFrame(J
NIEnv * env, jobject obj)
```

Next, we create an asynchronous thread to play the sound whenever the renderer returns a nonempty string. We created an ImageThread class that extended Android's AsyncTask class.

```
public class ImageThread extends AsyncTask<ImageTargetsRenderer, Void, Void>
	{
	protected Void doInBackground(ImageTargetsRenderer... renderer) {
	while(delay()){
		if(!renderer[0].detectedImage.equals("")){
				mSoundManager.playSound(renderer[0].detectedImage)
		}
				renderer[0].detectedImage = "";

			}
			return null;
		}

		protected void onProgressUpdate() {
		}

		protected void onPostExecute() {
		}

		public boolean delay(){
			try {
				Thread.sleep(3000);
				return true;
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
			return false;
		}
	}
```

* Note: We have created a delay function to only poll the renderer for detected image each 3000ns interval (after many trials, 3000ns seemed to be the best delay time). This will lessen the processing load by not checking and returning values and playing sounds constantly.

## Evaluation and Future Direction

GroZi is currently a functional Android application that allows the user to identify products before him/her. But what if there are multiple products before the individual? How will the blind user be able to pick up the right product? To solve this challenge, development of a navigation feature would prove useful. By dividing the screen captured by the camera into four quadrants, implementing voice navigation for the user's hand is possible. If the targeted product is on the left or right side of the screen then an automated "left" or "right" voice would help navigate the user's hand. Similarly, an "up" or "down" functionality would serve useful for the blind user.

Even with the functionality of this GroZi prototype, there are still questions to be addressed for the future of the project. One of the main issues is scalability. Currently, the application identifies products that are in source library. But what if the user is looking for other products that are sold at this grocery store that are not in the library? For this product to create true value for its blind users, it must be able to support thousands of products without slowing down performance.

## Bibliography

1) GroZi Project
http://grozi.calit2.net/
http://globalties.ucsd.edu/grozi.html

2) Qualcomm AR SDK
https://developer.qualcomm.com/develop/mobile-technologies/augmented-reality

3) Android SDK/NDK
http://developer.android.com/sdk/index.html
http://developer.android.com/sdk/ndk/index.html