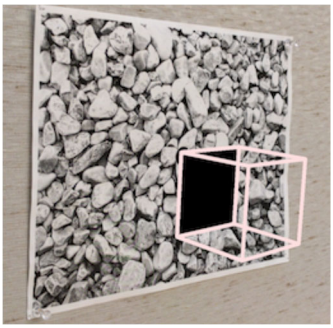
EEE598: Mobile Systems Architecture Fall 2018
**Project assignment 2**: ASUforia Augmented Reality Framework
Official Release Date: *2018 October 19*
Due Date: *11:59:59 PM, Monday, 2018 November 26*
Send any requests for clarifications or errata to likamwa@asu.edu

## ***Learning Objective:***

*Build library for an augmented reality framework*

Augmented reality combines the virtual world with physical environments by looking for correspondences and placing objects in view. This relies on *pose estimation,* understanding where the camera is with respect to a visual marker. After studying computer vision for a while, you've found that it's quite feasible to do pose estimation with vision libraries.



Indeed, the marker-based pose estimation procedure is simple:
1) Identify feature points and descriptors in a reference image
2) Capture camera frame
3) Identify feature points and descriptors in the camera frame
4) Match camera frame features with reference features
5) Perform pose estimation with Perspective-n-Point solver
6) Use estimated pose to draw on the camera frame
7) Repeat Steps 2-6

However, simply building your own app is not enough for you. You want to enable developers to simply and easily employ augmented reality for their own apps. Thus, you have set out to build a library for a marker-based augmented reality framework.

**Honor code**

Thoroughly read and acknowledge this affidavit BEFORE you open the assignment.

- My team will work independently on this project.
- I will **NOT** share or discuss *source code* with other teams. (I am allowed to discuss concepts)
- I will **NOT** copy/paste code from any sources.
    - (This includes Google/StackOverflow/OpenCV! Just retype things!)
- I will **NOT** debug others' code.
- I will **NOT** receive help debugging my own code.
- I will **NOT** copy/paste any information from other sources into my documentation.
    - Any referenced information will be sufficiently paraphrased and cited.

**Github**

Use Github to maintain a *private* repository of your code. Github education pack allows unlimited private repositories. Name your git repository "eee598-asuforia-[lastname1]-[lastname2]-[lastname3]", e.g., "eee598-asuforia-likamwa-kodukula". Add "roblkw-asu" and "kodukulav" as collaborators. Commit at least once a week.

**Specifications:**
To complete your assignment, you will need to provide::

(i) your ASUforia library (asuforia.java),

(ii) a Main Activity that uses the library (MainActivity.java), and

(iii) a PoseListener interface to connect the two through an onPose() callback. Here, we list the different functions to constitute the deliverables.

**PoseListener interface** (not class! Define the PoseListener interface in asuforia.java or poselistener.java. Implement the interface in MainActivity.java, e.g., with an anonymous inner class. For more tips, see: https://www.geeksforgeeks.org/anonymous-inner-class-java/)

- void onPose(…)
    - Callback that gets called when pose is estimated.
    - You get to decide what the parameters should be!

**Application Library**
This library will need to set up a Camera2 Device to capture frames and feed them to the pose estimation. The library will also interact with the library user's PoseListener, feeding coordinates and images so the user.

- nativePoseEstimation(…) [Defined in native .cpp file]
    - Uses OpenCV to determine pose.
    - Compare points against reference points
    - Return rotation vector and translation vector
- Asuforia Constructor Asuforia(PoseListener listener, Image refImage, Surface cameraSurface)
    - Stores PoseListener argument to act as callback interface.
    - Generates reference points from reference image.
        - Save the reference points for estimation usage
- startEstimation(…) : Starts the estimation process
    - Start camera (Camera2) in preview mode, sending YUV420_888 to onImageAvailable()
- onImageAvailable(…) implementation : Callback for Camera2 frames
    - Call nativePoseEstimation() C++ function
    - Call the PoseListener's function with the rotation and translation vector.
- endEstimation(…) : Ends the estimation process
    - Stops camera

MainActivity

- onCreate(…)
    - Implement a PoseListener with onPose function.
        - Paint a user interface surface with the camera image with a cube as an overlay on the marker in the image.
        - Can invoke OpenCV to paint the image with Java or Native code.
    - Create an Asuforia object
- Start pose estimation when the application comes to the foreground
- End pose estimation when the application leaves the foreground

Extra Credit:

- Connect Unity Game Engine with your ASUforia library, using the pose estimation to move the virtual camera.

Tips and Resources:
**DO NOT COPY/PASTE ANY CODE. RETYPE EVERYTHING.**
**DEEPLY UNDERSTAND HOW FUNCTIONS CONNECT THROUGH PARAMETERS**

C++ on Android
https://developer.android.com/ndk/index.html
https://developer.android.com/ndk/guides/prebuilts.html
https://developer.android.com/ndk/guides/cpp-support.html
https://developer.android.com/ndk/guides/stable_apis.html

Setting up Camera2 and onImageAvailable:
https://developer.android.com/samples/Camera2Raw/src/com.example.android.camera2raw/Camera2RawFragment.html

OpenCV 3.3 Android Pack: https://opencv.org/releases.html
Tutorial to link to OpenCV headers and libraries with .mk files:
https://medium.com/@siantlords/opencv-and-android-ndk-integration-in-android-studio-883a810189e2

OpenCV Pose Estimation Tutorial:
https://docs.opencv.org/master/dc/d2c/tutorial_real_time_pose.html
(Note: For your library, do not save/load model. Instead, compute reference points and descriptors from an input image).

**Documentation**

- Goals
  - State the high-level objectives you sought to achieve.
  - Restate the assignment description in your own way.
- Design/Implementation
  - Describe your application components, including any activities or classes
  - Describe interfaces between application components.
    - For example, explain the sequential flow of Camera2/ImageReader/Java-Native-Interface/PoseListener. How do these connect with one another? What are the relevant arguments for each of the functions?
- Benchmark Evaluation
  - Charts and/or tables
  - Speculate on reasons for performance differences
    - Image size, # features, clock speed
- Describe challenges encountered along the way
  - Include any missteps or misunderstandings.
- Include screenshots!

**Assignment Deliverables:**

1) Source code
    a. Package your app into an .apk
    b. Zip your Android project folder, including your source code.
2) Documentation
    a. Include your documentation as a pdf.
    b. Include a video of the app working.
    c. Include screenshots of the app working.
3) Declaration
    a. In a text file, type out (don't copy/paste) the honor code verbatim. Include a statement declaring that you and your team fully abides by the honor code.
4) Package all the above files in to a zip. Use the filename "Lastname1-lastname2.zip", e.g., "Hu-LiKamWa.zip". Submit the file via Blackboard.
5) Git
    a. Invite Prof. LiKamWa and TA Jinhan Hu as collaborators to your Git repository.

**Full Deliverable Rubric**

- 15%**:** PoseListener
    - Interface defined
    - Interface implemented in MainActivity
    - File-level and function-level comments
- 15%: Camera Image Handling
    - Camera image captured
    - Grayscale image passed to OpenCV
    - File-level and function-level comments
- 15%: Augmentation
    - Screen displays camera image
    - Screen displays augmentation
    - File-level and function-level comments
- 15%: ASUforia library Infrastructure
    - Constructor properly defined
    - Start/End works correctly
    - File-level and function-level comments
- 15% (Extra Credit): Unity Game Engine integration
    - Pose estimation moves the Unity camera to view a sphere on the marker.
    - File-level and function-level comments
- 10%: Performance measurements
    - +5% Extra Credit: Compare stock OpenCV against OpenCV4Tegra
- 20%: Thorough Documentation
- 10%: Consistent git commits to the repository.