Augmented Reality PoC on Cloudlets

Goal

Create an augmented reality app for Android smartphones that convincingly demonstrates the value of low-latency cloud services, and hence the need for cloudlets.

This PoC will be created entirely by the CMU team and released open source. The app that runs on the mobile device will be released to the Google Play store. Anyone can install it on his or her own Android device. The back-end server (with computer vision processing) will run on OpenStack++ and will also be available for download.

Requirements

1. Easily appreciated by any audience, without specialized knowledge.

2. Runs on standard Android OS/hardware. Google Nexus 6 smartphone is primary target; any other Android devices such as tablets are secondary, lower-priority targets.

3. Easily portable and demoable anywhere in the world. No content customization needed (e.g. no need to create new recognizers for objects, no need to enter names of buildings, etc.). Minimal software reconfiguration needed to give demo at a new site.

4. Wi-Fi based cloudlets are primary focus. However, demo also works with 4G LTE cloudlet at CMU, and can be given on other similar setups anywhere. Some day, when MEC hardware becomes available, this demo can be ported to run on it.

5. Primary focus is demo to individuals using smartphone. Secondary focus is giving demo on an Android tablet to a small group. Also possible to give demo to room-size audiences using a projector (e.g. Android screen mirroring) with some possible loss of crispness due to additional screencast and projection latency.

6. The cloudlet should run OpenStack++. However, there is no intent in this PoC to demonstrate cloudlet discovery, rapid provisioning or VM handoff. A later extension of this PoC (or a completely separate PoC) may focus on those capabilities. If it helps to speed up the implementation, the cloudlet may optionally run the Gabriel software stack on top of OpenStack++.

Description

Via a configuration screen, the Android smartphone can be connected to a cloudlet at a particular IP address, or to an AWS cloud site (AWS US-East, AWS US-West, etc.). When you switch from configuration screen to demo screen, the outward facing camera on the smartphone continuously streams video to the back-end server running at cloudlet. Computer vision software on the cloudlet analyzes each frame to find people. It then performs some funny transformation of each person it finds. For example, it may create a “skeleton view” of that person. Or it may put funny clothes on that person. Or show them naked. Exact details to be figured out by CMU team and may involve extensive experimentation. The computer vision for recognition and the transformation of the image are done extremely fast. The transformed input frame becomes one frame

of an output video stream that is sent back to the mobile device. As the user moves the mobile device, the people shown on the screen are "instantaneously" transformed into skeletons or some other funny representation. The tracking speed is excellent. For example, if a person waves his hand, the corresponding skeleton waves its hand without any perceptible delay.

Using simple touches on the screen, the demo can be toggled between 3 states:

a) Show the camera view directly (no transmission to cloudlet, no transformation of image); this is the fastest possible speed and crispness, and is at the raw speed of this mobile device. This is what you normally see on your smartphone when you run the camera app.

b) Show the cloudlet output video stream with zero processing on the cloudlet (no recognition, no transformation). This shows the reduction in crispness and increase in latency due to transmission to cloudlet and back.

c) Show the full demo (i.e., process and transform each frame). This will have the lowest crispness, but hopefully still so good that people are impressed.

Simple metrics such as output frames per second and per-frame delay can quantify a), b) and c). With a powerful enough cloudlet and adequate upstream and downstream bandwidth, the frames per second should be the maximum sustainable on the device. The per-frame delay will be determined by the end-to-end latency of the setup. Using the skeleton example, the per-frame delay will determine the lag between when a person raises their hand and when the image on the smartphone raises its hand. The frame rate will determine the smoothness of motion in the image. Audience can easily compare frame delay and smoothness in states a) and c). The differences in these metrics between states a) and c) are the key performance indicators of this demo.

Note: the terms "upstream video" and "downstream video" should be loosely interpreted here. We deliberately leave open and unspecified exactly what gets transmitted upstream, what gets transmitted downstream, and how much processing gets done on the mobile device in either direction. Only experimentation with an actual implementation will determine the optimal choices for each of these design choices. Our preliminary experiments with video suggest that encoding a sequence of frames into an MP4 video on the cloudlet (to generate the downstream video) may incur too high latency. Experimentation will be needed to pick the best data representation for upstream and downstream data transmission.

Timeline

- **Phase 1 (Oct – Dec, 2015):** *Conceptualization and Planning*

  Do the initial conceptualization and refinement of the PoC. Convert the above description into a more precise and complete description. Explore open source resources (especially computer vision and augmented reality) that can be leveraged. Storyboard and conceptually debug the application so that it is clear exactly what functionality needs to be implemented on the Android device and on the cloudlet.

- **Phase 2 (Jan 1 – May 30, 2016):** *App Implementation and Performance Tuning*

  Focusing on Wi-Fi initially and Intel-donated cloudlets and Nexus 6 smartphones, do the

actual implementation, testing, debugging, and refinement of the application. Do the performance optimizations necessary to make the user experience on cloudlets as smooth and crisp as possible. Add the controls necessary to dynamically switch between cloudlet and AWS cloud. Show the difference in user experience between cloudlet and cloud, including difference between AWS East, West, etc. Show initially on Wi-Fi, but also show on 4G LTE cloudlet at CMU. At the end of this period, the app and cloudlet software are "done" in terms of functionality, performance and configurability.

- **Phase 3 (Jun 1 – Aug 31, 2016):** *Modifications for Portability and Showmanship*

  Make the app available in the Google Play Store. Make it easily viewable by a large audience using Android screencast. Experiment to discover the smallest stand-alone cloudlet that is powerful enough to give good performance. Create a portable demo specification of hardware (cloudlet and mobile device) plus open source software that can be downloaded and easily installed so that any of the OpenEdge team members can create such a demo. Provide training via Webex to OpenEdge team members to give good demos.