Smart Beholder: An Extensible Smart Lens Platform

Chun-Ying Huang[1], Ching-Ling Fan[2], Chih-Fan Hsu[3], Hsin-Yu Chang[2], Tsung-Han Tsai[3], Kuan-Ta Chen[3], and Cheng-Hsin Hsu[2]

[1]Department of Computer Science, National Chiao Tung University
[2]Department of Computer Science, National Tsing Hua University
[3]Institute of Information Science, Academia Sinica

ABSTRACT

Smart Lenses refer to detachable, orientable and zoomable lenses that stream live videos over wireless networks to heterogeneous computing devices, including tablets and smartphones. Various novel applications are made possible by smart lenses, including mobile photography, smart surveillance cameras, and Unmanned Aerial Vehicle (UAV) cameras. However, to our best knowledge, existing smart lenses are closed and proprietary, and thus we initiate an open-source project called Smart Beholder for end-to-end solutions of smart lenses. The code and documents of Smart Beholder can be found at our website http://www.smartbeholder.org. Our Smart Beholder platform are useful to researchers for fast prototyping, developers for rapid development, and amateurs for hobbies. We have implemented Smart Beholder server (camera) using a popular embedded Linux platform, called Raspberry Pi. We have also realized Smart Beholder client (controller) on various OS's, including Android. Our experimental results show the practicality and efficiency of our proposed Smart Beholder: we outperform commercial products in the market in terms of both objective and subjective metrics. We believe the release of Smart Beholder will stimulate future studies on novel multimedia applications enabled by smart lenses.

Keywords: Streaming; wireless networks; camera; optimization; smartphones; mobile devices; quality of experience

1. INTRODUCTION

Miniature sensors and actuators are getting increasingly popular, and users expect to access intelligent information gathering systems, such as *smart lenses*, from their tablets, smartphones, and smart watches. Smart lenses refer to detachable, orientable, and zoomable lenses that stream live videos to heterogeneous computing devices over wireless networks. Users exercise these computing devices to control smart lenses for: (i) adjusting various settings, such as orientation and optical/digital zoom, (ii) previewing the current view, and (iii) capturing photos or videos. Through the separation of optical lenses and computing devices, smart lenses offer higher flexibility and enable novel multimedia applications that were not possible before. These applications include, but are not limited to:

- **Mobile photography.** Although smartphones are widely used by casual photographers, smartphone cameras often adopt small sensors without optical zoom supports due to limited spaces. This in turn leads to inferior photo/video quality and limited camera features [1]. In contrast, regular digital cameras give users full control over aperture, white balance, shutter speed, and so on. Smart lenses [2,5,13] are not limited by the small form factor of smartphones, and may provide similar features/quality as digital cameras.
- **Smart surveillance cameras.** Governments and organizations, such as police departments, may deploy surveillance cameras for real-time, high-quality, and interactive photo/video gathering, so as to improve urban safety, such as identifying fights, riots, protests, demonstrations, fires, chemical leaks, and stampedes. Amateurs may also deploy surveillance cameras at home [11], e.g., to check on their pets when traveling.
- **UAV (Unmanned Aerial Vehicle) cameras.** Increasingly more hobbyists and journalists adopt UAVs to capture aerial photos/videos, which are made possible because of UAVs' 3D mobility. Several UAV manufacturers [4,14], provide different mounts to turn smart lenses [5,7] and thermal imaging cameras [6], into UAV cameras.

Optimizing smart lenses for good user experience in these and future applications is not an easy task, because smart lenses are connected to computing devices via wireless networks, which are vulnerable to interference, channel fading, and shadowing. In addition, users have two expectations: (i) low interaction delay and (ii) high graphics quality, which are inherently contradicting to each other. Last, smart lenses are typically implemented on resource-scarce embedded systems, which further complicate the design, development, and implementation of smart lenses for diverse applications. Fortunately, researchers and developers have plenty of ideas on addressing these challenges. However, existing smart lenses products [2, 5, 7, 11, 13] are closed and proprietary, preventing researchers and developers from fast prototyping so as to validate their ideas.

To resolve the problem, we present *Smart Beholder*, which is an open-source end-to-end platform for smart lenses. The

MM '16, October 15-19, 2016, Amsterdam, Netherlands
© 2016 ACM. ISBN 978-1-4503-3603-1/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2964284.2973793

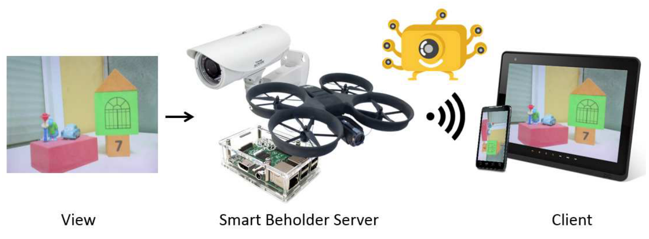

Figure 1: Sample usage scenarios of Smart Beholder.

code and documents of Smart Beholder are publicly available at http://www.smartbeholder.org, and detailed system designs are given in Huang et al. [8]. Figure 1 shows typical usage scenarios of Smart Beholder, which consists of a server and a client. Smart Beholder server is an embedded system equipped with a lens and a wireless interface, while Smart Beholder client runs on a computing devices such as a tablet or a smartphone. The server captures views and streams them in real-time to the client via wireless networks. Smart Beholder is designed with three goals: (i) cost effectiveness, (ii) low interaction latency, and (iii) high preview quality. These goals are achieved by: (i) choosing a just-powerful-enough embedded system (Raspberry Pi [12]), (ii) minimizing latency in all software components, and (iii) dynamically adapting the video coding parameters.

2. SYSTEM DESIGN

Figure 2 gives the high-level design of Smart Beholder. The server consists of network interface and video streamer. The network interface includes: (i) AP (Access point) service that allows clients to connect to the server, and (ii) the DHCP server that assigns clients IP addresses. The networks between Smart Beholder server and client can be single-hop WiFi networks (e.g., for mobile photography and UAV cameras) or multi-hop wired and wireless networks (e.g., for smart surveillance cameras). The video streamer captures videos using camera capturer, encodes videos using encoder module, and streams videos to the clients through the RTSP (Real-Time Streaming Protocol)/RTP (Real-Time Protocol) server. The Smart Beholder client consists of UI (User Interface) and video streamer. The UI renders received videos via viewfinder and takes user inputs using camera controller. The user inputs are interpreted as user commands and transferred from controller client to controller server. The user commands may include taking photos, recording videos, and configuring effect settings, such as white balancing, exposure, and sensitivity. In addition, the RTSP/RTP client periodically measures the available bandwidth and sends the results to server for dynamic parameter selections of preview videos.

3. INTENDED AUDIENCE

Smart Beholder is designed for three kinds of intended audience. First, researchers may use Smart Beholder platform to analyze and evaluate their new ideas without implementing everything from scratch. That is, Smart Beholder enables fast prototyping on smart lens based applications for academic exploration. Second, developers may use Smart Beholder to develop new products with high-quality user experience. The developers only need to adjust some parameters and add/delete components to improve the performance or offer new functionalities. This promotes rapid development of smart lens based applications. Third, amateurs may also build our platform and customize it for fun. This may encourage them to try innovative smart lens based applications. Smart Beholder works for researchers, developers, and amateurs because it is extensible, portable, configurable, and open. We envision that the release of Smart Beholder will stimulate more projects on new smart lens based applications. For example, in Section 8, we demonstrate how to use Smart Beholder to build a panoramic camera.

4. SOURCE TREE STRUCTURE

Smart Beholder is released with two types of software packs: `all-in-one` and pre-compiled binary packs. In addition, the source codes are available on our project website at http://www.smartbeholder.org. The `all-in-one` pack includes Smart Beholder source codes, third-party library source code, pre-compiled binaries, while a pre-compiled binary pack includes only binaries for running the software on Raspberry Pi [12]. Users may extend Smart Beholder to support other hardware platforms running embedded Linux. There are five subdirectories in the `all-in-one` pack; their descriptions are given in the following.

- `bin/`: Pre-compiled run-time files, including the executables, modules, and configurations. The current pre-compiled files are built on Raspbian Linux.
- `codes/`: Source codes for Smart Beholder, which is divided into three parts: libcore, module, and server.
- `deps.posix/`: Dependencies for POSIX platforms, including libraries and headers. Smart Beholder currently supports Raspberry Pi, which runs a POSIX-compatible Linux. This subdirectory is empty by default, as the files in this subdirectory are built from the sources in `deps.src`.
- `deps.src/`: Source files of the dependent third-party packages. You will need these files to build dependencies for POSIX platforms.
- `scripts/`: Sample setup scripts, e.g., to turn a Raspberry Pi into a WiFi AP.

5. COMPILATION AND SETUP

Smart Beholder can be installed by uncompressing the software packs, with or without the source codes. If `all-in-one` software pack is downloaded, the complete Smart Beholder system can be compiled from scratch. Smart Beholder is a platform dependent software, and the compilation and setup instructions are written for Raspberry Pi [12].

Users attach an official camera module to a Raspberry Pi, and install Raspbian Wheezy Linux To build Smart Beholder, users first make sure that `g++`, `pkg-config`, `libX11`, `libXext`, `libXtst`, `libfreetype6`, `libgl1-mesa`, `libglu1-mesa`, `libpulse`, `libasound2`, `libmp3lame`, `libopus`, `libogg`, `libvorbis`, `libtheora`, `libvpx`, `libx264`, and `libxvidcore` are installed. Users must install both binaries and development files for the above packages. Next, the following instructions are recommended:

1. Edit the `env-setup` script and point `GADEPS` to the absolute path of `beholder/deps.posix/`.
2. Load environment variables from `env-setup` by using `.` or `source` commands.

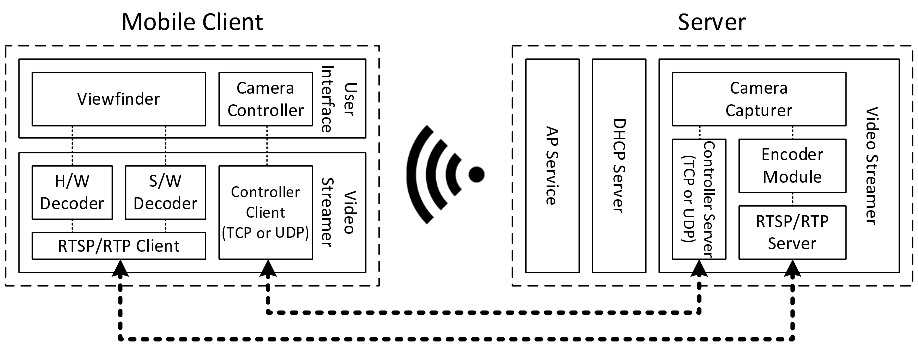

Figure 2: The design of Smart Beholder.

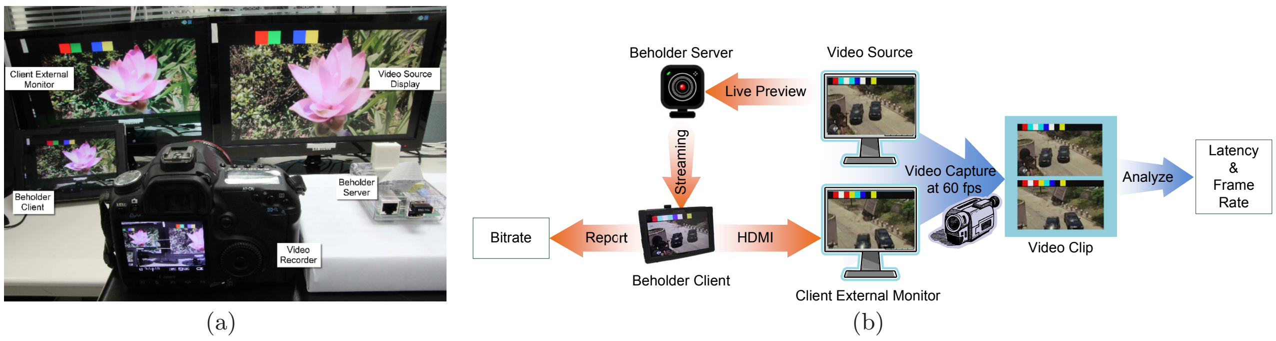

Figure 3: Experimental setup: (a) testbed and (b) procedure.

3. Build dependencies using `make -f Makefile.raspbian` under `deps.src/`. All third-party libraries will be then installed in `deps.posix/`.
4. Build Smart Beholder using `make all` under `codes/`.
5. Install Smart Beholder using `make install` under `codes/`. All binaries and configurations will be copied to `bin/`.

Users next configure Raspberry Pi into a WiFi AP using `hostapd` and `dhcpd` tools. Users may modify our sample configurations `scripts/hostapd.conf` and `scripts/dhcpd.conf` for this purpose. However, `hostapd` does not work with all wireless modules. In particular, a Linux wireless driver must support `nl80211` interface to work properly with `hostapd`, and thus users have to carefully choose WiFi adaptors.

6. EXECUTION GUIDE

This section explains how to run Smart Beholder server and client. Before running binaries, please ensure that the full path to `deps.posix/lib` has been added to the system-wide `ld.so` search path. Smart Beholder can be launched using the following command: `server-rpi config/server.rpi.conf`. If a Raspberry Pi is configured as an AP using our `hostapd.conf` and `dhcpd.conf`, a wireless device can associate to the AP's SSID `smartb` using password `smartbeholder`. On a PC or a Mac client, please use the following command: `client config/client.abs.conf rtsp://192.168.11.254:8554/beholder` to connect to Smart Beholder. On an Android device, the client can connect to `rtsp://192.168.11.254:8554/beholder`. Alternatively, users can use an RTSP video player to connect to the same URL to watch the live video captured by Smart Beholder. By default, Smart Beholder uses TCP port 8554 for RTSP streams and TCP/UDP port 8555 for control messages. Once connected, a client can send a mouse click or screen touch event to Smart Beholder for taking a camera shot. The captured photos are by default stored in `/tmp` directory. The photo directory can be changed using the `photo_dir` option in Smart Beholder server configuration.

7. PERFORMANCE EVALUATIONS

We conduct real experiments to evaluate and compare our proposed Smart Beholder against two products: Altek Cubic [2] and SONY QX100 [5]. In our experiments, we consider the following performance metrics: bitrate, latency, frame rate, and preview quality in MOS (Mean Opinion Score). Figure 3(a) shows our testbed setup. The video source is played on the right monitor, which is captured by the Smart Beholder server. The server then streams the preview video to the client running on the tablet on the left via a WiFi network. The tablet is connected to an external monitor, and we use a Canon EOS 600D camera to capture the two monitors at 60 fps (frame-per-second), and report the objective experiments results (see Figure 3(b)). For subjective evaluations, we recruit 52 subjects and perform 117 sessions that totaled 14,410 comparison rounds.

Sample experiment results are given in Figure 4 due to the space limitations. Figure 4(a) shows that the bitrate of Smart Beholder is 3 Mbps, which is half of other commercial products. Figures 4(c) and 4(b) reveal that the Smart Beholder achieves slightly higher frame rate as well as has higher interactivity since the latency of Smart Beholder is

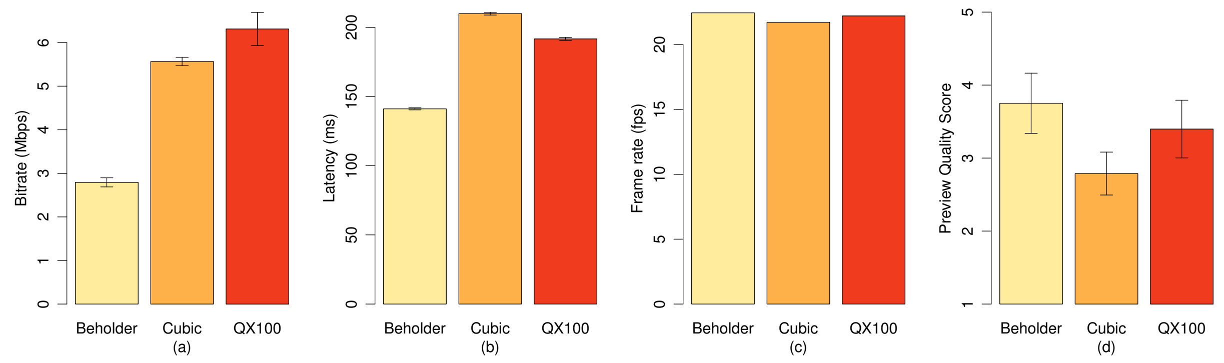

Figure 4: Sample performance comparisons among smart lenses: (a) bitrate, (b) latency, (c) frame rate, and (d) preview quality.

lower than that of others by at least 50 ms. The subjective evaluations of Smart Beholder on the preview quality also achieve higher scores than others as reported in Figure 4(d). In summary, our proposed Smart Beholder achieves higher preview quality without incurring network overhead. More details are provided in Huang et al. [8].

8. PANORAMIC SMART LENS: A USE CASE OF SMART BEHOLDER

There are increasingly more novel cameras in the market, such as panoramic [16] or 360° [9, 15] cameras. We build a panoramic smart lens using Smart Beholder, which works as follows. It continuously captures multiple images with multiple cameras pointing at different orientations, stitches these images, and then passes the resulting images to the encoder module. In this way, users see panoramic video on their Android devices.

In particular, we equip Raspberry Pi board with a multi-camera adapter board [10], which supports up to 4 cameras. While streaming, we program the GPIO pins to switch among cameras for images from different cameras, in a round-robin fashion. The resulting videos are then stitched using OpenCV [3] running on the Raspberry Pi. The resulting videos are then handled by a software-based video source. This video source iteratively feeds a sequence stitched images into a ring-buffer based pipe, which is shared with the encoder module. The encoder module then reads each frame from the pipe, performs the encoding task, and sends the encoded frames to a client via the RTSP and RTP protocols. The encoder module is put into sleep if no frame is available in the pipe, and is waked up right after a frame is pushed into the pipe. The video source and the encoder module are launched in different threads and the use of pipe increases the parallelism of the running threads.

Because Smart Beholder is designed and realized with high extensibility in mind, implementing the panoramic video source module (`vsource-stimage`) becomes relatively easy. In fact, we have only created 3 files with about 500 non-comment lines, before getting a working panoramic camera, which is illustrated in Figure 5.

9. CONCLUSION

We presented Smart Beholder, an open-source smart lens platform, which is available at http://www.smartbeholder.org. Smart Beholder enables various new multimedia applications, such as mobile photography, smart surveillance cameras, and UAV cameras. We designed Smart Beholder for three goals: (i) cost effectiveness, (ii) low interaction latency, and (iii) high preview quality. Our evaluations reveal that Smart Beholder indeed achieves the three goals by: (i) using a just-powerful-enough embedded system, (ii) reducing latency in individual software components, and (iii) adapting video coding parameters based on measured wireless network conditions. Smart Beholder can be leveraged by researchers for fast prototyping, developers for rapid development, and amateurs for fun. We envision that the release of Smart Beholder will stimulate more projects related to smart lenses and result in many new smart lens based applications. For example, we used panoramic camera as a sample application to demonstrate that Smart Beholder is highly extensible.

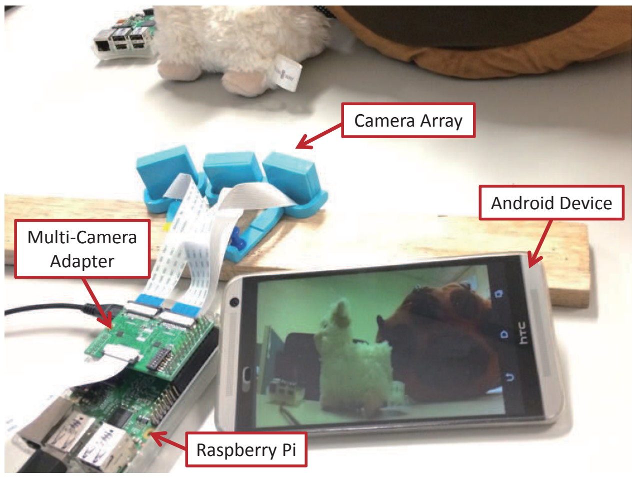

Figure 5: Our panoramic smart lens built on Smart Beholder.

10. ACKNOWLEDGEMENTS

This work was partially supported by the Ministry of Science and Technology of Taiwan under the grants: 103-2221-E-001-023-MY2, 103-2221-E-009-230-MY2, 102-2221-E-007-062-MY3.

11. REFERENCES

[1] 5 areas where cameras still beat smartphones if you want great photo quality. http://tinyurl.com/n8f5w8d.

[2] Altek Cubic: Perfect to selfies. http://www.altek.com.tw/cubic/.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Dragonfly innovative uav aircraft & aerial video systems. http://www.draganfly.com/industrial/products.php.

[5] DSC-QX100 lens-style camera with 1.0-type sensor. http://www.sony.co.uk/electronics/cyber-shot-compact-cameras/dsc-qx100.

[6] FLIR Thermal Imaging. http://www.flir.co.uk/cs/display/?id=41702.

[7] GoPro 3+ high-performance life capture. http://shop.gopro.com/cameras/hero3plus-silver/CHDHN-302-master.html.

[8] C. Huang, C. Hsu, T. Tsai, C. Fan, C. Hsu, and K. Chen. Smart beholder: An open-source smart lens for mobile photography. In *Proc. of ACM Multimedia'15*, Brisbane, Australia, 2015.

[9] LUNA 360 Camera - luna 360 vr camera. http://luna.camera.

[10] Multi Camera Adapter Module for Raspberry Pi. http://www.arducam.com/multi-camera-adapter-module-raspberry-pi.

[11] Petcube remote wireless pet camera. https://petcube.com/.

[12] Raspberry Pi. http://www.raspberrypi.org/.

[13] RE Camera a remarkable little camera. http://www.htc.com/us/re/re-camera/.

[14] Skybotix uav navigation solutions. http://www.skybotix.com/.

[15] The Bublcam - experience true 360 spherical technology. https://www.bublcam.com/.

[16] V360 - capture everything the way you play in full 360. http://www.vsnmobil.com/products/v360.