모바일 폰 기반의 사이버 자연사 박물관

홍성수*, 이르판 칸**
*,**호서대학교 컴퓨터공학과
[1]sshong@hoseo.edu, [2]manikhan@nate.com

An Efficient Method in Mobile E-health system for Large Images Processing

Sung-Soo HONG*, Irfan KHAN**
*,**Department of Computer Engineering, Hoseo University, Asan Campus, South Korea
sshong@hoseo.edu, [2]manikhan@nate.com

A B S T R A C T

These days rapid improvement in Mobile phones and their multimedia limits made them powerful enough to manage complicated tasks. Image processing related support for mobile devices is extremely comprehensive in wireless telemedicine. A basic challenge is how to get best quality of image with the limited screen size and resources of mobile phones. This paper deals with image processing features (capturing rendering and zooming in and out) of Mobile Media API and Advanced Multimedia Supplements (MMAPI and AMS) developed for Mobile Java Platform and customized algorithm is designed to keep all image task cost efficient by using minimum device resources and memory. This scenario is driven by the need for evaluation of a distant patient that cannot be moved to the expert.

Keywords: IP Multimedia Subsystem (IMS), Image Processing, Mobile E-Helath System, MMAPI and AMS,

1. Introduction

Mobile devices have become ubiquitous in the recent years which made mobile multimedia communication easy and faster. These days, mobile phones with high resolution built-in digital cameras are widely used all around the world, and these portable handheld digital devices provide a new way to visualize, store and transmit images. The aim of this paper is to put forward an image processing method to solve the problem of inefficiency of image processing in mobile devices because the image processing tasks become more complex than before. Therefore, traditional image processing methods have difficulty applying to resource constrained mobile devices. The above problem has been extensively explored

The Advanced Multimedia Supplements API (JSR 234) extends the Mobile Media API (JSR 135) with several controls simplifying usage of multimedia files and streams. Advanced Multimedia Supplements is intended to augment MMAPI by providing access to features that have become more common on mobile devices such as cameras, 3D audio files, audio radio, image encoding, and image post processing. JSR 234 allows application developers to access a device's multimedia capabilities, enabling applications that control multimedia. Other applications may incorporate multimedia, such as messaging applications that allows users to send pictures.

2. Related Work

Image resizing is a standard tool in many image processing applications. It works by uniformly resizing the image to a target size. Recently, there is a growing interest in image retargeting that seeks to change the size of the image while maintaining the important features intact, where these features can be either detected top-down or bottom-up. Top down methods use tools such as face detectors [Viola and Jones 2001] to detect important regions in the image, whereas bottom-up methods rely on visual saliency methods [Itti et al. 1999] to construct a visual saliency map of the image. Once the saliency map is constructed, cropping can be used to display the most important region of the image. Suh et al. [2003] proposed automatic thumbnail creation, based on either a saliency map or the output of a face detector. The large image is then cropped to capture the most salient region in the image. Similarly, Chen et al. [2003] considered the problem of adapting images to mobile devices. Santella et al. [2006] use eye tracking, in addition to composition rules to crop images intelligently. All these methods achieve impressive results, but rely on traditional image resizing and cropping operations to actually change the size of the image. Setlur et al. [2005] proposed an automatic, non-photorealistic algorithm for retargetting large images to small size displays. This is done by decomposing the image into a background layer and foreground objects. The retargeting algorithm segments an image into regions, identifies important regions, removes them, fills the resulting gaps, resize the remaining image, and re-insert the important region.

The use of seams for image editing is prevalent. Agarwala et al. [2004] describe an interactive Digital Photomontage system that finds perfect seams to combine

parts of a set of photographs into a single composite picture, using minimal user assistance. Jia et al. [2006] proposed Drag-and-Drop Pasting that extends the Poisson Image Editing technique [Perez et al. 2003] to compute an optimal boundary (i.e. seam) between the source and target images. Rother et al. [2006] developed AutoCollage, a program that automatically creates a collage image from a collection of images. This process requires, among other things, finding optimal boundaries, or seams, between many image fragments. None of the above methods discuss the problem of image retargeting. Changing the size of the image has been extensively studied in the field of image processing, where the goal is to generate a large image from a small one. This way, the original small texture image is quilted to form a much larger texture image. The scale and rotate functions are in the heart of the algorithm. Therefore, they must both fast and high quality. Our image algorithm let us use a shared buffer so that we waste no time allocating memory. For additional performance and efficiency, we chose multi-pass transforms.

3. Image Processing Method

MMAPI and AMMS (Advance Multimedia Supplements) is the key to mobile imaging in this e-health system, the quality of the snapshot image is a prominent factor. Image processing divides in different parts, capturing image(s) and managing quality (resolution) then rendering it with limited system resources and sizing it using according to screen of mobile phone explain as follow.

MMAPI allows Java ME application to capture still images, as well as play and record sound and video clips with better controls over audio, video and image processing abilities. Even though MMAPI's VideoControl can take a picture, but AMMS gives better control over the way the picture is taken and what is done with it.

3.1 Image Capturing Method

MMAPI allows Java ME application to capture still images, as well as play and record sound and video clips with better controls over audio, video and image processing abilities. Even though MMAPI's VideoControl can take a picture, but AMMS gives better control over the way the picture is taken and what is done with it. A Player is created using Manager.createPlayer("capture://video"). AMMS image encoding and Post Processing features are used for creating raw images and support for any kind of image format with better control over ImageEffectControl, ImageTransformControl, OverlayControl, etc.

```
//initializing the default camera
Player player = Manager.createPlayer("capture://video");

// check if camera rotate
if(CameraControl.ROTATE_LEFT==rotation ||
CameraControl.ROTATE_RIGHT==rotation)
    portrait = true;

//displays incoming video from camera onto screen
VideoControl videoControl = (VideoControl)
player.getControl("VideoControl");

// returns an array of bytes (image data) in the requested format
byte[] data = videoControl.getSnapshot(null);

// capture image
videoControl.getSnapshot("encoding=jpeg&width=2048&height=1536")

// setting focus control for image
if(focus.isAutoFocusSupported()) {
  focus.setFocus(FocusControl.AUTO);
  } else {
// Otherwise, Find out what was actually set.
int focusSet =focus.setFocus(Integer.MAX_VALUE);}
focusControl.setFocus(FocusControl.AUTO)
```

The instance of Player is created for capturing the live video with the help of built-in camera. Then get VideoControl for controlling the output of video, getSnapshot() method is called to get a snapshot of the displayed content. The "null" parameter is default capture format. Focus of the camera device is controlled with FocusControl interface and image is piled up in the form of a byte array and sent out to server through HTTP as a binary attachment for further transactions.

User can select encoding format (resolution) for still image(s) according to his/her need. Selected encoding will be passed as the parameter when taking the snapshot. Macro photography is a vital part of telemedicine as compared to resolution. Macro photography is a property of the lens to be able to focus on a subject just centimetres or even millimetres away

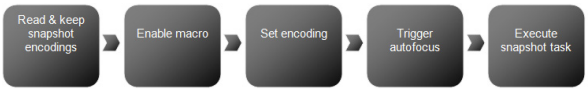

Figure3.1.3: Process of taking a snapshot.

3.2 Image Rendering Method

MIDP render the whole image by Image.createImage() which burns up large amount of memory, Java ME itself and other additional libraries need memory as well. Test on real device shows that rendering image with maximum resolution gives an OutOfMemoryError, simply due to the fact that Java VM couldn't allocate object due to no more memory. AMMS provides a way to create a thumbnail from the bytes stream of the saved photo by using MeidaProcessor & ImageTransformControl.

```
//read from original byte array stream and writing it to an empty
output stream
ImageTransformControl itc = (ImageTransformControl)
mp.getControl("javax.microedition.amms.control.imageeffect.ImageTransf
ormControl");
itc.setTargetSize(widthOfScreen,heightOfScreen,0)

//Implementation of an input & output stream that uses a byte array as
the source
mp.setInput(new ByteArrayInputStream(imageBytes),
MediaProcessor.UNKNOWN);
ByteArrayOutputStream baos = new ByteArrayOutputStream();
mp.setOutput(bos);

// Horizontal Resize
for (int x = 0; x < screenWidth; x++)
 {
  g.setClip(x, 0, 1, srcHeight);
  g.drawImage(src, x - (pos >> 16), 0, Graphics.LEFT |  Graphics.TOP);
```

```
  pos += ratio;
}

Image resizedImage = Image.createImage(screenWidth, screenHeight);
  g = resizedImage.getGraphics();
  ratio = (srcHeight << 16) / screenHeight;
  pos = ratio / 2;

//Vertical resize
for (int y = 0; y < screenHeight; y++)
 {
  g.setClip(0, y, screenWidth, 1);
  g.drawImage(tmp, 0, y - (pos >> 16), Graphics.LEFT | Graphics.TOP);
  pos += ratio;
 }
```

//Image transformations is done with MediaProcessor interface of type jpeg.

```
MediaProcessor mp = GlobalManager.createMediaProcessor("image/jpeg")
```

Memory consumption for rendering image depends on its resolution. On devices like mobile phone with limited system memory and memory allocated to JVM (java virtual machine) is challenging. Source rectangle (Captured image) of 2048*1536 pixels on the left side and target rectangle on right Nothing changes the source rectangle, the only thing changed is the target size. The target width and height is the height and weight of the mobile screen. As a result, the target image, or the resulting image after transformation is the original image resized into the size of the screen, rendered on the mobile phone.

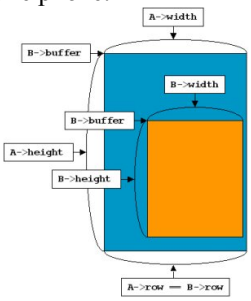

Figure3.2.1: Original image scales down to the width and height of the screen

3.3 Zooming and Panning Method

Rendering a picture with high resolution and fine focus on mobile phone with restricted resources is quite a challenge on mobile phone screen size. In mobile e-health, care taker takes a picture of infection, wound or etc and sends it to the doctor for his opinion, due to the small screen size of mobile phone it is hard to view details of image. Java ME does not offer such ready-made functionality. With the help of AMMS customized algorithm is designed for zooming in and out image task. Figure 3.3.1 shows the geometrical presentation of the zooming algorithm adopted in the mobile client.

// the object to read bytes of original image

```
MediaProcessor mp
mp=GlobalManager.createMediaProcessor("image/jpeg")
mp.setInput(src, MediaProcessor.UNKNOWN)
```

//start cutting a piece on x,y axis of size source width and height

```
ImageTranformControl.setSourceRect(x, y, sourceWidth, sourceHeight)
```

//Target Width and Height to be resized to

```
itc.setTargetSize(targetWidth,targetHeight,0)
```

//x & y coordinates on the source image to //start cut from

```
itc.setSourceRect(x,y,sourceWidth,sourceHeight)
```

//return the resulting object

```
Image.createImage(imageData,imageOffset,imageLength)
```

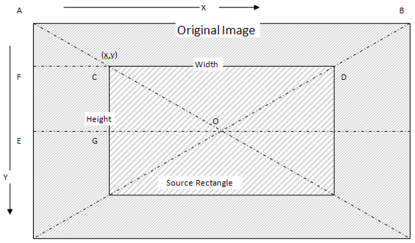

Figure3.3.1: Geometrical presentation of zooming

Rendering algorithm 3.2 squeezes a large image according to screen size of mobile phone with the help of AMMS. Zooming is to take a part of image and display it on the screen, more details of the image will expose. Algorithm 3.3 Cut a part of the original image and resizes it into a new image. The coordinate of left-top corner of original image is (0,0) the X-axis is pointing from left to right, and the Y-axis is pointing from up to down. The outer box stands for the original image of size 2048*1536, whereas the inner box is the sub-part of the image to be taken from the original image. The rule is that every zoom of original image shifts the point (x, y) away from (0, 0) along the diagonal by a fix step. The distance between (0, 0) and (x, y) is called zmLnInDgnl, and the fix step is named zmStp. Every zoom produces new zmLnInDgnl which equals previous zmLnInDgnl plus zmStp, then:

$$x = \frac{originWidth \times zmLnInDgnl}{originDiagonal} \qquad y = \frac{originHeight \times zmLnInDgnl}{originDiagonal}$$

$$height = \frac{(halfDiagonal - zmLnInDgnl) \times originHeight}{halfDiagonal}$$

$$width = \frac{(halfDiagonal - zmLnInDgnl) \times originWidth}{halfDiagonal}$$

Therefore the sub-image is extracted from the original image starting from (x, y) with calculated width and height, meanwhile preserving the same ratio as the original. AMMS is used to stretch the sub-image to fit the screen.

4. Proposed Image Processing Method

Taking snapshots and rendering the pictures are two important functions in the client, Figure4.1 shows the time spent in both processes. The larger solution an image has, the longer it requires to process. It can be seen as taking a snapshot is heavy work, a snapshot with the lowest resolution 640*480 takes less time than a snapshot of resolution 2028*1536. In contrast, rendering those pictures are very quick. The size of the snapshot image is important to consider when saving it to persistence storage

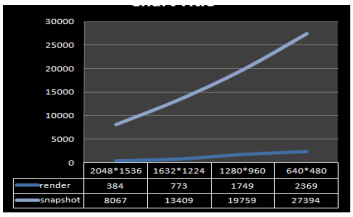

Figure4.1 Delay in taking snapshots and rendering.

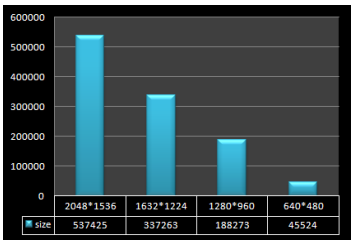

Figure4.2 Sizes of snapshots with different resolutions.

Above figures Shows the images obtained via MMAPI are not big, the image with the lowest resolution 640*480 is 46 KB on average, whereas size of the largest image is about half a megabytes. Though the current system saves those images into a dedicated folder in the server side file system, it is still feasible to store them into the database, because their sizes are not significant.

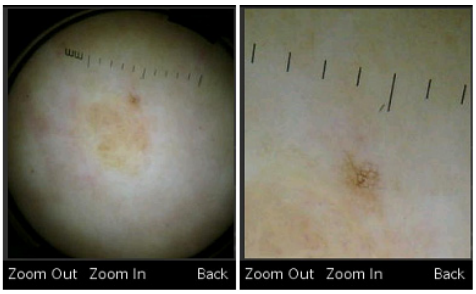

Figure4.3: zoomed image of a patients infection

5. Conclusions & Future Work

This conclusion of this project is a system prototype for mobile TeleMedicine image processing prototype system. The system proves the concept of building such application over IP Multimedia Subsystem, with particular focus on mobile imaging can be useful for people living in such area where medical services are not available.

Following improvements of the mobile telemedicine system are the next step to deal with. Java ME client performance and resource handling on data access layer. JSON (JavaScript Object Notation) could be a substitute to the XML format. Because of very compact data structure, this can help with better transmission time and its lightweight features for mobile applications. Authentication, authorization and access control should be importance in the system, in particular the web interface. Furthermore, other IMS enabled services to be explored, such as push-to-talk (PTT) and multimedia telephony (MMTel), using video conference for better interaction with experts.

6. References

[1] The IMS: IP Multimedia Concepts and Services, 3rd Edition (Miikka Poikselka, Georg Mayer)2009

[2]Rebecca Chen, Elisa Su, Victor Shen, and Yihong Wang. Introduction to IP Multimedia Subsystem (IMS). IBM developerWorks.

[3]Mobile Java Communication Framework | Ericcson Lab Developer http://developer.labs.ericsson.net/apis/mjcf.

[4]Eckerson, Wayne W. "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications." Open Information Systems 10, 1 (January 1995): 3(20).

[5] Developer's guidelines Java Platform, Micro Edition, CLDC – MIDP 2.0

[6]Lightweight UI Toolkit Developer's Guide, July 2008.

[7]JSR-234: Advanced Multimedia Supplements 1.1, Maintenance Release

[8]G Bonnet, Y Shen. Next generation telecommunication services based on SIP.

[9]JSR 139: Connected Limited Device Configuration 1.1, Maintenance Release

[10]Vijay Arvind Balasubramaniyan Lei Kong and Mustaque Ahamad. A lightweight scheme for securely and reliably locating sip users. In Proc. IEEE/IFIP Network Operations & Management Symposium, 2006.

[11]3GPP. Security aspects of early IP multimedia subsystem (IMS).