April 11

ECE 1778H

iMOSAIC

Final Report

- Anthony Soung Yee (995092741) - Apper
- M. Hamed Zahedi (992255812) - Programmer

Word Count = 2441 Words
Penalty = 0
Apper Word Count = 335 Words
Penalty = 0

1 Introduction

Our project is a mobile device Application (App) for visualizing and interactively exploring a large set of images, specified by the user. We employ an image processing technique called "image mosaicing", where a "primary" image is constructed from a large set of "tile" images. More specifically, we use an algorithm for creating an image collage or mosaic, where subsections (or even individual pixels) of a primary image are composed of other "tile" images, as shown in Figure 1.

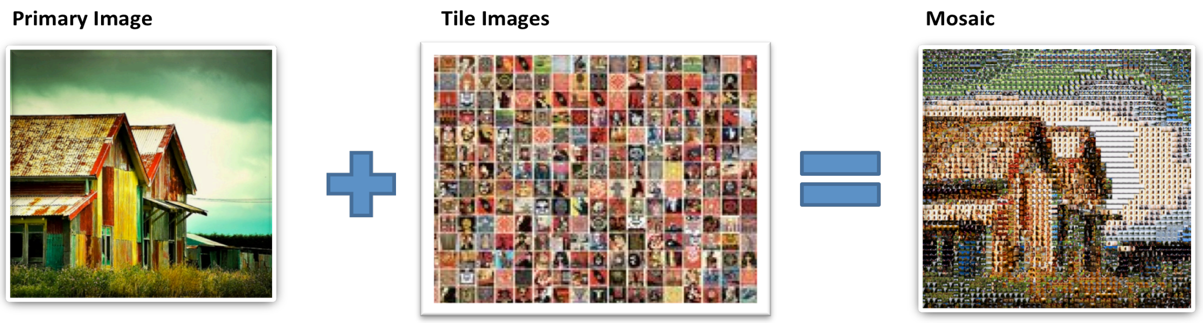


Figure 1 - Diagram illustrating the formation of an image mosaic from a primary and set of tile images

This approach is often used in multimedia applications, as a means of displaying a large number of images simultaneously, presented in a structured manner in the form of the primary image. This contrasts with the layout of many Web sites such as Google Images, which show large image collections (i.e. search results) in a grid form, but whose ensemble has no overall structure. With the advent of cell phone photography, we intend to bring this mosaicing functionality to a mobile device, to provide a fun and entertaining way to display large sets of photographs that users may have on their devices.

2 Overall design

The project requirement was to develop an application allowing users to generate an artistic mosaic representation of a primary image given a set of tile images – and further allow the user to "explore" the generated mosaic. A governing requirement was to reduce time to initial functioning prototype and allow incremental refinements. As such the application was designed to have the minimum working feature set for each functional block– while allowing for additional – non-critical functionality to be added later. A block diagram of major building blocks is depicted in Figure 2, below.

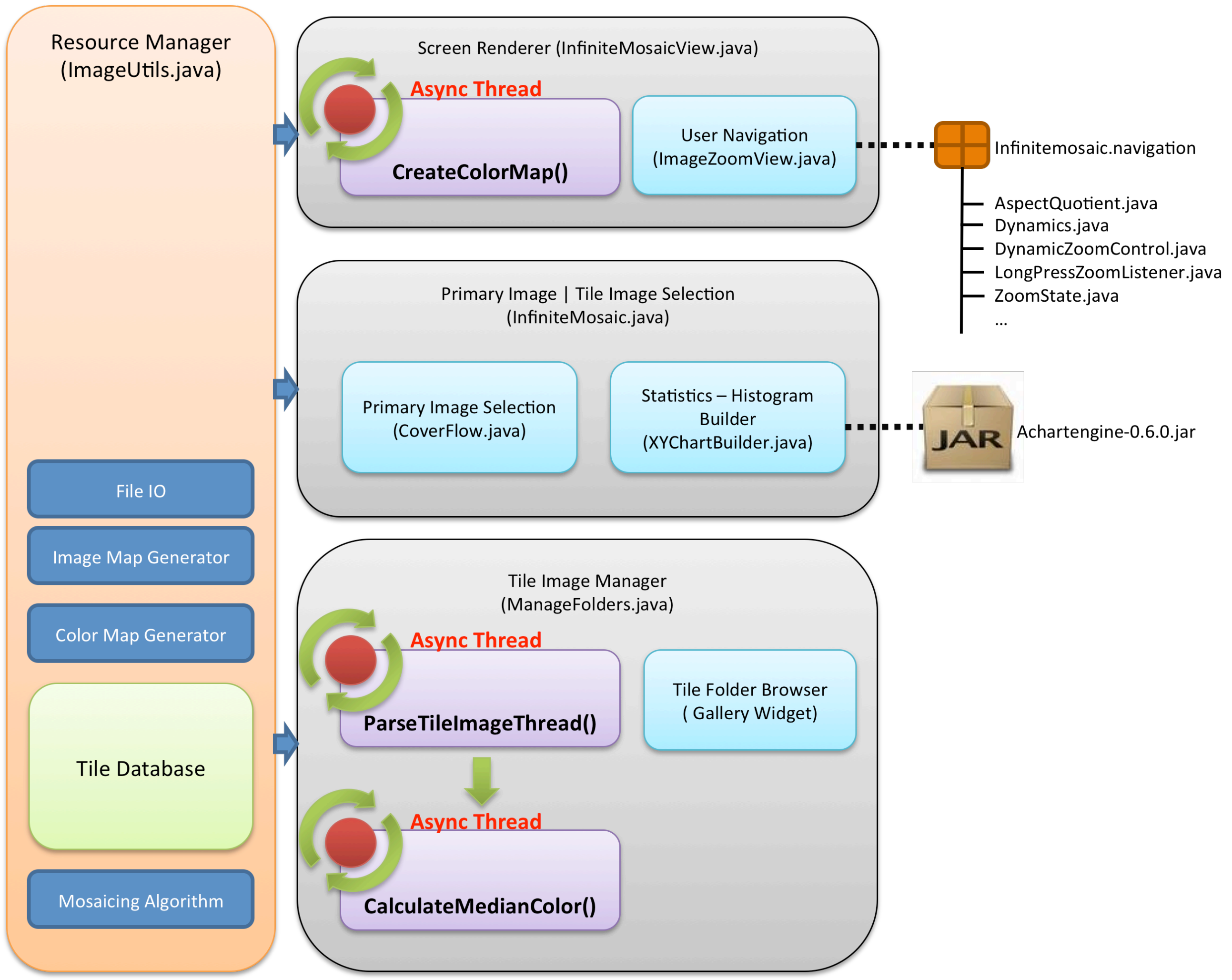

Resource Manager (ImageUtils.java)
File IO
Image Map Generator
Color Map Generator
Tile Database
Mosaicing Algorithm
Screen Renderer (InfiniteMosaicView.java)
Async Thread
CreateColorMap()
User Navigation (ImageZoomView.java)
Infinitemosaic.navigation
AspectQuotient.java
Dynamics.java
DynamicZoomControl.java
LongPressZoomListener.java
ZoomState.java
...
Primary Image | Tile Image Selection (InfiniteMosaic.java)
Primary Image Selection (CoverFlow.java)
Statistics – Histogram Builder (XYChartBuilder.java)
JAR
Achartengine-0.6.0.jar
Tile Image Manager (ManageFolders.java)
Async Thread
ParseTileImageThread()
Tile Folder Browser (Gallery Widget)
Async Thread
CalculateMedianColor()

Figure 2 - Block Diagram of high level layout showing major blocks

2.1 Activity Flow and layout

The activity screens that solicit user input and lead the user towards generating a mosaic are outlined in Figure 3 below. The layout and flow was developed specifically with Human Factors in mind. Buttons and graphics were designed in Adobe Photoshop to be pixel accurate to a display resolution of 320x480. Significant additional programming and testing effort would be required to create such a comprehensive layout that is portable across all screen resolutions.

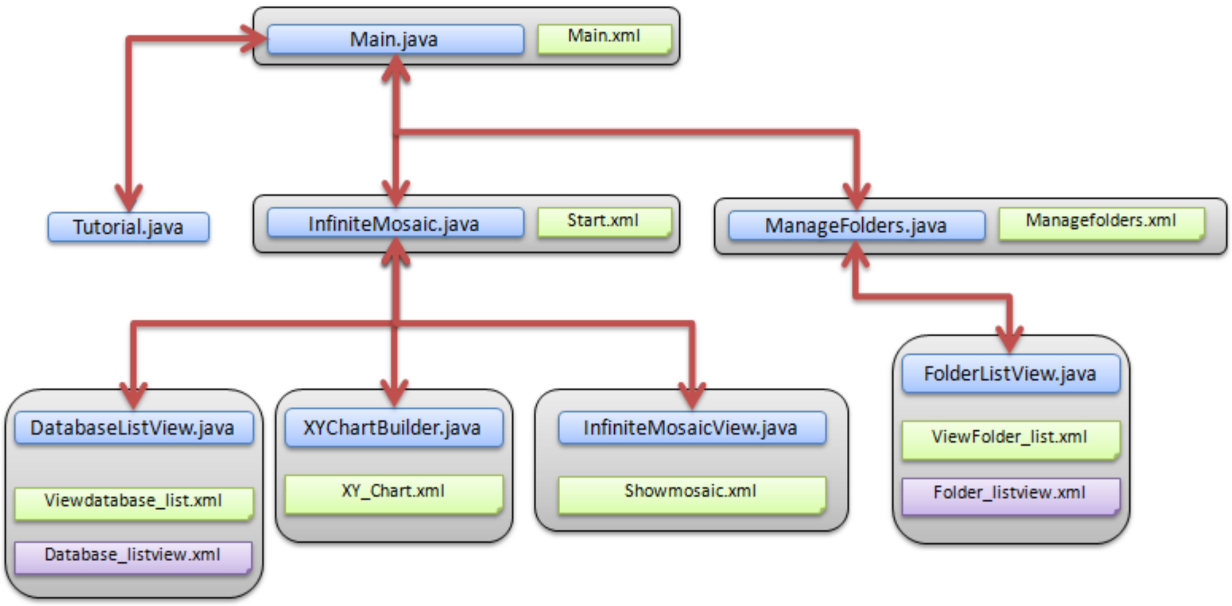


Figure 3 - Activity Flow Diagram for iMosaic showing layout elements

2.2 Algorithm

Two major algorithms are implemented in this project – the median color algorithm and the mosaicing algorithm. Given a bitmap in RGB colour space, the median color algorithm determines a single representative color for all the pixels in that bitmap. The algorithm is implemented in 2 versions outlined in Figure 4 below. In version 1, the algorithm is used to generate a colour map by downsampling a primary image. In version 2, the algorithm is used to determine a single representative colour from an entire bitmap. Both are implemented in Asyncronous threads on Android.

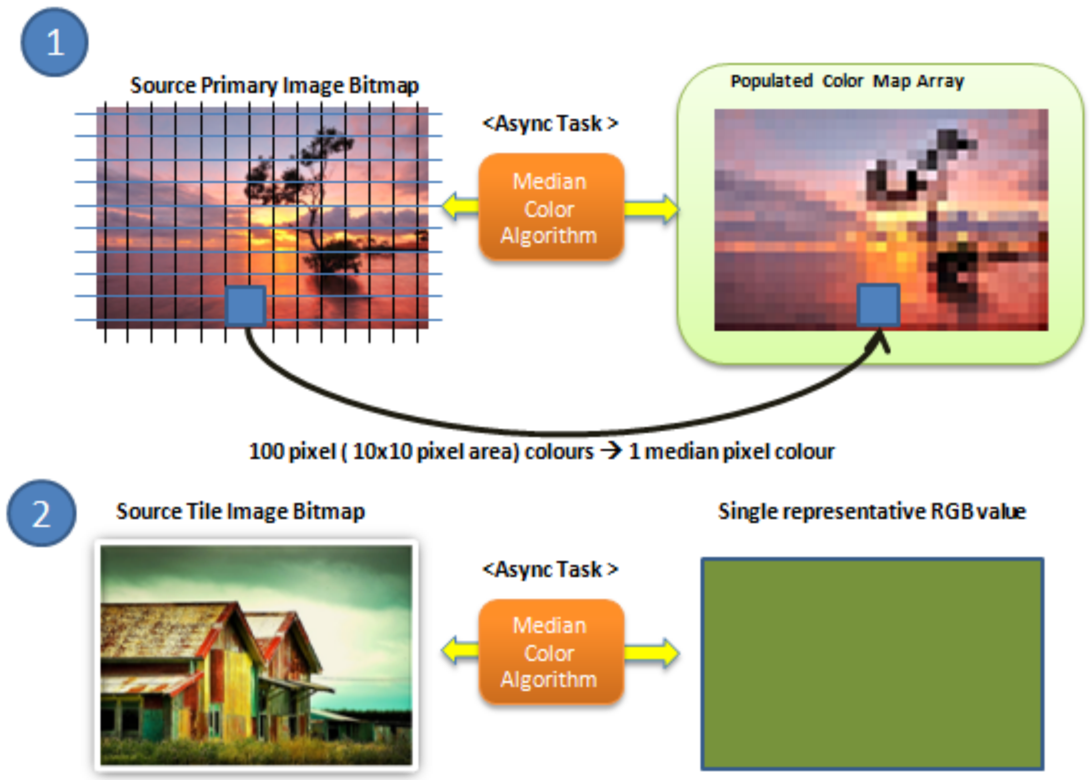

1
Source Primary Image Bitmap
<Async Task >
Median Color Algorithm
Populated Color Map Array
100 pixel (10x10 pixel area) colours → 1 median pixel colour
2
Source Tile Image Bitmap
<Async Task >
Median Color Algorithm
Single representative RGB value

Figure 4 - Median Colour Algorithm implementations in iMosaic

The second algorithm used in iMosaic app, is the mosaic generation algorithm. This algorithm takes as input, a color map array (as shown in Figure 4) and a list of tile images. For each colour in the colour map array, the mosaic generation algorithm determines the best fit "tile" image to represent that colour. The *imageInfo* class is used as the backing data structure for this operation. The X,Y co-ordinates, and the path file path are stored for each matching file. This data structure is then used to generate the mosaic bitmap.

2.3 Database Generation and loading (.MZK files)

When parsing a folder of image tiles, iMosaic automatically generates a database containing the results of the parsing operation. Each tile database represents a folder on the SD card. The name of the database is the name of the image folder with .MZK extension. An MZK file is an ASCII file containing the path to a file and its associated median colour. The generation and storage of mosaic databases is depicted in Figure 5 below. Since the database persists between sessions, the user is not required to color information this information every time. *imageUtils* class contains the necessary file i/o functionality to load and store this database as required. Currently iMosaic requires all tile image folders to be located at /mnt/sdcard/ and have the "pics_" prefix to be recognized as a valid image folder by iMosaic app.

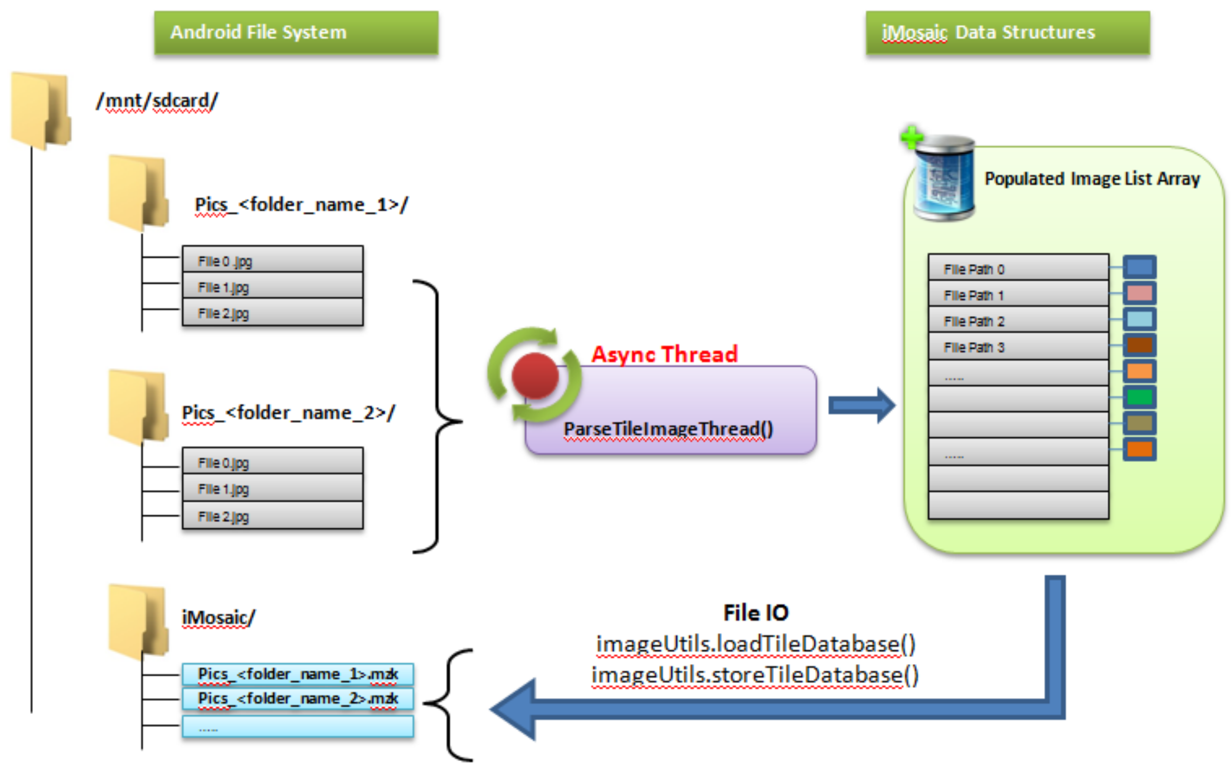


Figure 5 - Folder Structure used by iMosaic App - also showing Database generation thread

2.4 Performance

Figure 6 shows time versus number of images for tile database generation. As expected, the trend is linear because every pixel in every image requires processing. Time is directly proportional to the total number of pixels being processed. I/O latency (SD card access) was not characterized.

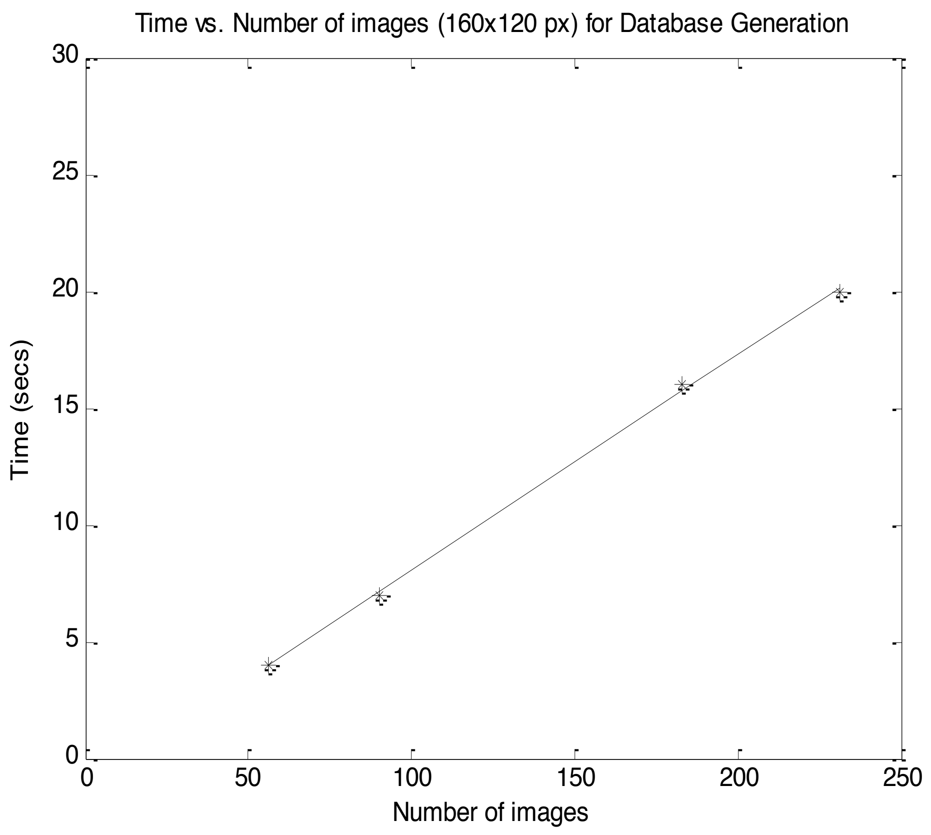
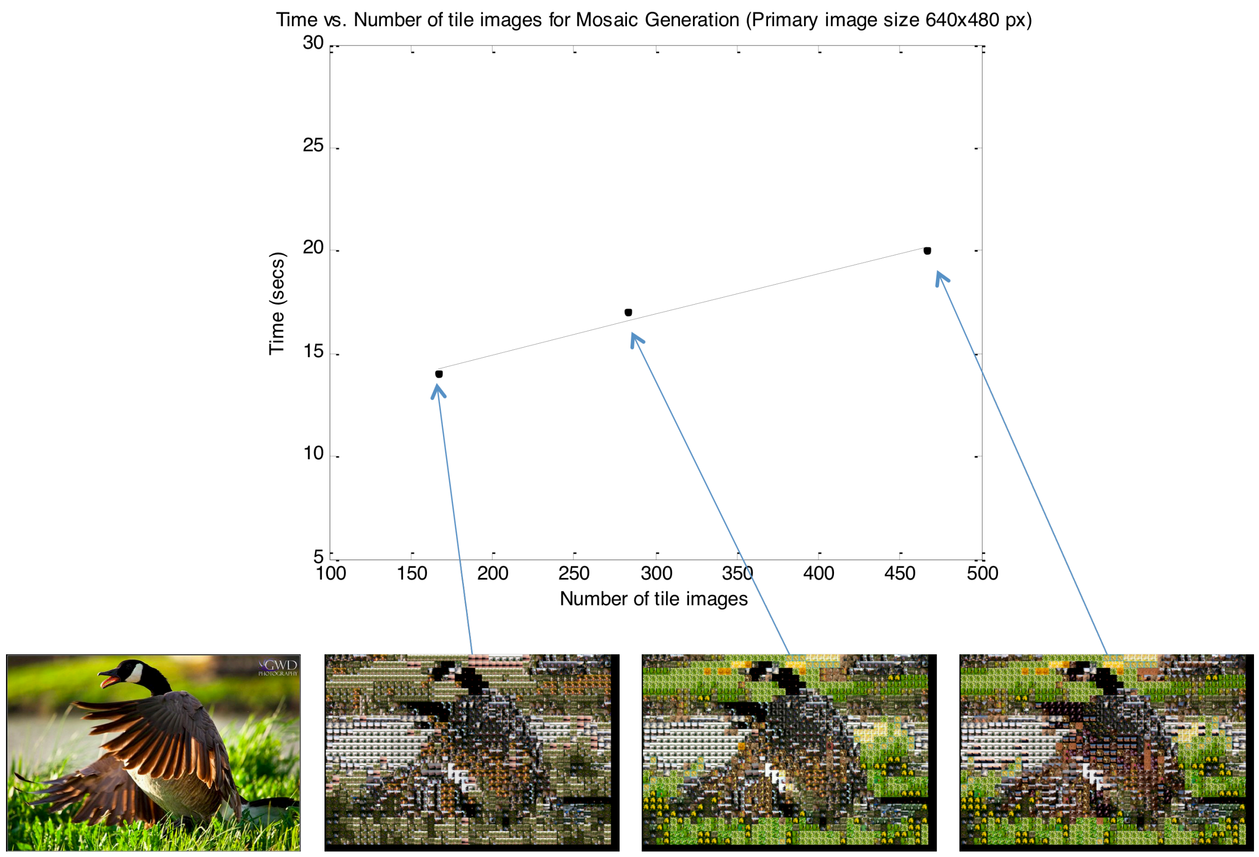


Figure 6 - Time Versus Number of tile images for Database generation as measured on LG Optimus 1 P500H @ 700MHz

Figure 7 shows the time versus number of images for mosaic generation. It also indicates the progression of quality of the mosaic with increasing image tiles. This is also a brute-force algorithm with no performance optimizations and no post-processing. A variety of modifications are available for increasing the performance of the mosaic generation. This algorithm is a worthwhile candidate for future improvement.



Original image   Mosaic w/167 images   Mosaic w/283 images   Mosaic w/466 images

Figure 7 - Time Versus Number of tile images for mosaic generation as measured on LG Optimus 1 P500H @ 700MHz

For a typical 640x480 image using 466 image tiles, mosaic generation took 19 seconds. Using a 10x10 tile width, this translates to (64x48 primary image tiles) X (466 tile images) / 19 seconds =~75000 image comparison operations per seconds on the LG Optimus 1 (P500H) @ 700Mhz

The algorithm is as follows:

$$\min\{\, d = \sqrt{A^2 + B^2 + C^2} \,\}$$

Where,

$A = R_t - R_i$ Difference between **RED** values of Primary and image (t) and tile “pixel” (i)

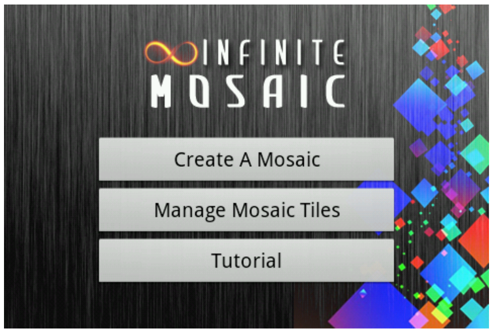
$B = G_t - G_i$ Difference between **GREEN** values of Primary and image (t) and tile “pixel” (i)

$C = B_t - B_i$ Difference between **BLUE** values of Primary and image (t) and tile “pixel” (i)

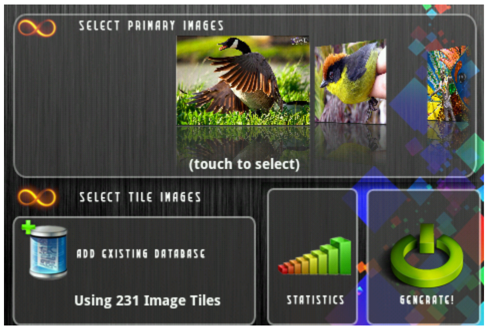
$d = \text{absolute difference between tile colour, i and primary image colour, t}$

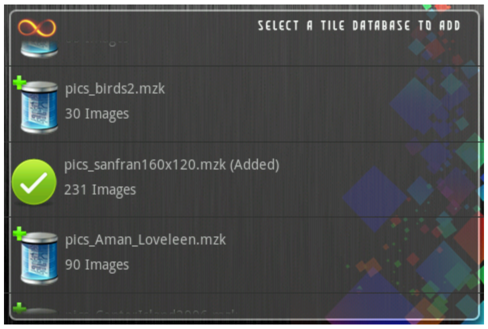
3 Functionality

The following section presents the functionality of the App, through the User Interface (UI).

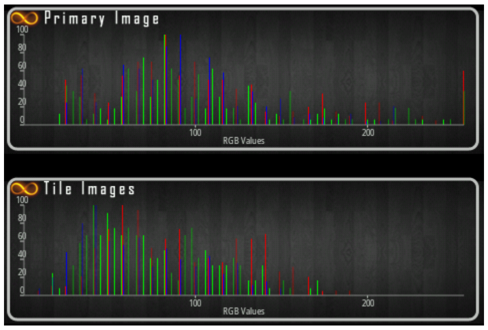


Main screen
This screen provides the user with three options, representing three major functions of the App. First, the 'Create mosaic' button brings the user to another screen to create an image mosaic. Second, the 'Manage Databases' button brings the user to another screen for adding and deleting databases of tile images, in order to create image mosaics. Third, the Tutorial button brings up a dialog to provide a set of instructions on the functions of the App. (This function was not completed due to time constraints)





pics_birds2.mzk
30 Images
pics_sanfran160x120.mzk (Added)
231 Images
pics_Aman_Loveleen.mzk
90 Images

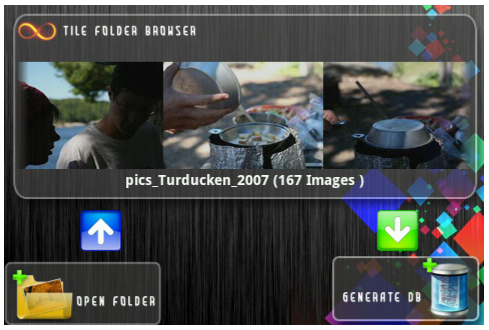


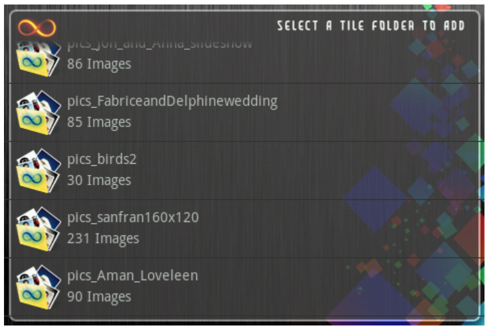
Create Mosaic screen
This screen outlines a series of steps needed to create a mosaic. Near the top of the screen, the user selects the Primary image using a Gallery widget, by scrolling left and right through the images. These primary images are found in the 'primary_image' folder on the SD card.

Next, the user selects the sets of tile images from which to construct the mosaic, using a custom list view. An indicator displays the number of images that are currently selected. Finally, the 'Generate!' button initiates the call to create the mosaic.

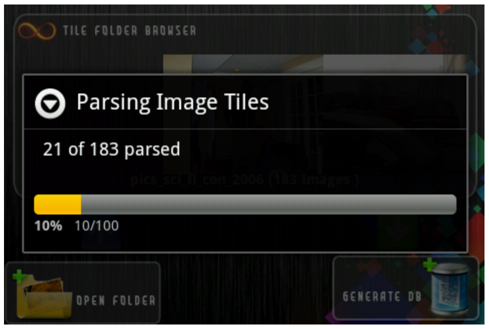
Finally, a histogram view shows the distribution of red, green, and blue pixels in the Primary image, as well as the set of Tile images selected.

Current limitation: File size. In order to shorten the processing time in generating the mosaic, only small file sizes are supported (approximately 640X480). Also certain primary images crash the App, for unknown reasons. A set of working primary images are provided.





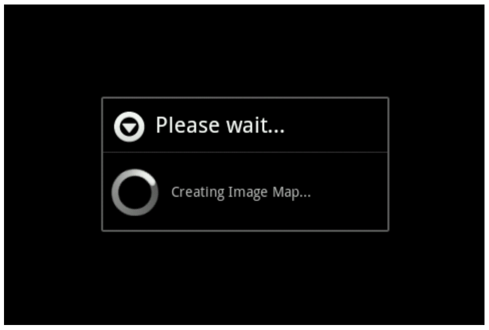
pics_Jon_and_Anna_slideshow
86 Images
pics_FabriceandDelphinewedding
85 Images
pics_birds2
30 Images
pics_sanfran160x120
231 Images
pics_Aman_Loveleen
90 Images



Parsing Image Tiles
21 of 183 parsed
10% 10/100

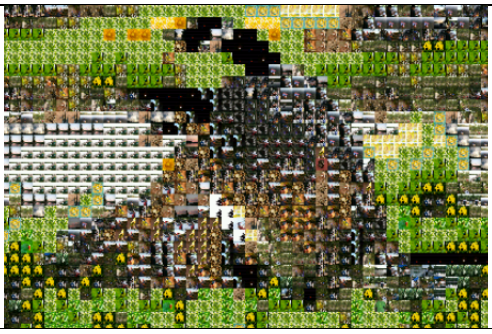
Manage Databases screen
This screen allows the user to select folders of images the SD card to create databases “tile sets”. These tile sets are used to create image mosaics. The user presses the ‘Open folder’ button to bring up a custom listview showing the available folders of images (folders with ‘pics_*’ as the name). The Gallery widget below displays the images from the folder, in order to be reviewed by the user. The user presses the ‘Generate DB’ button to start processing the images, packaging it into a .mzk file. A dialog popup displays the progress of creating the database.

Current limitation: File size. Only small tile images are supported, in order to shorten the processing time in creating a database. Images of size 160x120 are supplied.



Please wait...
Creating Image Map...

Mosaic display and navigation screen
Once the mosaic is generated and displayed on the screen, the user is able to pan and zoom using one finger gestures. To pan, simply drag the image mosaic in any direction . To zoom, hold press on any area of the screen for one second. The user receives vibration feedback, and the user can now gesture up to zoom in, and gesture down to zoom out. The navigation functionality also includes “elasticity” into the panning motion.

4 Lessons learned

This section highlights some important lessons learned over the course of the semester. Also included are remarks on what would have been done differently if this project would have been restarted.

4.1 Project management and communication

The group members learned a great deal about the importance of communication in delivering a project, from conception to completion. Given the time constraints, learning curve and scheduling issues, it was essential to be constantly communications our ideas to each other. One example of this occurred early on in the project, where it was found that the group members had different interpretations of the term mosaicing. As it turns out, there are two types of mosaicing algorithms, which were miscommunicated. However, once the problem was solved, we became aware of each other's use of vocabulary to describe different aspects of the project.

4.2 Display design

The Apper's field of research is in Human factors, which deals with human-machine interfaces. Although most Human factors practitioners are familiar with issues related to design principles for mobile devices, it was worthwhile getting hands-on experience in designing the user interface (UI). The Apper also had the opportunity to code the UI layouts in XML. This was very helpful in understanding some of the design choices that Android App developers must make when programming Apps. As well, it was interesting for the Apper to have to design for such a small screen size, which has implications for the layout of the App's screens.

4.3 Considerations for group size

Looking back at the amount of work expected to be accomplished during the time frame of the course, it may have been worthwhile to recruit a third group member. This would have had the added benefit of better optimised code, more functionality, etc. However, the group also learned about the importance of having passionate and dedicated group members. Given the time constraints and logistics of meeting throughout the semester, there were several opportunities for the collective efforts of the group to break down. However, because the group members shared mutual interests in the project succeeding, trust, as well as complementary skills, the working environment was amicable and productive.

5 Group member contributions

5.1 Hamed Zahedi (Programmer)

Hamed was responsible for the entirety of the software architecture and implementation. Hamed worked closely with Anthony on initial brainstorming of project ideas and directions, determining feasibility given the timeline, prioritizing and filtering the feature set to meet the project goals. Significant initial effort was spent researching initial feasibility by browsing online resources and training videos to minimize potential pitfalls during implementation.

During the collaboration phase, Individual high level project requirements (ability to navigate the mosaic, ability to select/view Image tiles etc.) were converted into low level software requirements and assigned a “difficulty” rating and a timeline feature sets were implemented in order of priority

Hamed was also responsible for developing the custom User Interface themes-including custom buttons, backgrounds, logos – all designed in Photoshop. This was an in iterative effort with significant input from Anthony on appearance and layout of individual views.

5.2 Anthony Soung Yee (Apper)

Anthony was responsible for conducting the research on mosaicing algorithms. Although his research work on mosaicing related more closely to stitching algorithms (as seen in panoramic photography), there were still some relevant techniques concerning optimisation that were useful for the current image mosaicing algorithms.

Anthony was also responsible for designing the user interface layout for the App. Using the principles of Human factors, he designed low fidelity prototypes (on paper and then in Powerpoint) with the goal of communicating the functions of the App easily to the user. In order to aid in the process of converting a prototype to an actual application, Anthony was responsible for implementing the individual screens in XML, as well as creating an App shell in Eclipse that links the screens together. It was useful for the Programmer to be able to see a skeleton of the final design before incorporating the project code.

6 Apper context

Although the form of mosaicing in the Apper's field strictly involves the Human factors in visual search using stitching algorithms as opposed to this form of mosaicing, there are several links between the two.

6.1 Relating the App to mosaicing in visual search tasks

There is a lot of research work investigating how people conduct visual search in data spaces such as image sets. A number of Human factors studies have been conducted for understanding the processes by which we conduct visual search in unstructured vs. structured data sets. For example, this formed the basis for the idea of a fish eye display. In a similar vein to my thesis topic, this technique of mosaicing may be more efficient for the human operator in finding a particular image within a large set, compared to conventional displays of search results. In other words, it could be argued that structuring images in a meaningful set may help the operator to find a target image in less time, or with fewer navigational gestures. Thus, although this collaging technique is not a core part of my thesis, the development of the mobile application may be worth conducting a formal study. This would be of greater interest to the HCI community, rather than the Human factors community.

6.2 Relating the App to Human factors

More generally, the field of Human factors has identified a host of issues that have important implications for the development of mobile applications. This includes accounting for navigational issues, using gestural interfaces and dealing with a limited screen size and resolution. One of the most important personal developments in this course was being given the chance to design and implement a UI for a mobile device. It highlighted many Human factors issues related to design, that could only be experienced by trying to create an App layout myself. This practical experience comes with at a low cost to an interface designer, and may be a very effective way to practice good UI design.

7 Future work

The following section describes some future work to enhance the current implementation of the App.

7.1 Social media integration

One relatively simple enhancement could be to incorporate social media functions such as sharing image mosaics. Moreover, it may be possible to have multiple users contribute to image mosaics, by posting pictures in a common location.

7.2 Incorporating online image search results

One of the functionalities that we wanted to include was using image databases such as Google image search or flickr as the tile images for the mosaic. It was imagined that you could enter a search query, and images obtained from the search results would be used to create a mosaic of related search images.

7.3 Optimisation and stability

One of the current limitations in the App is the processing time to create mosaics. Because of the time it takes to generate mosaics under the current implementation, the image sizes had to be reduced. With more time, it would be possible to further optimise the speed of the program.

7.4 Commercialisation

Given the slew of multimedia apps on the Android market, there is no doubt that this App could be commercialised. Because the App was intended for entertainment purposes with a requirement of only photographs, there may be high demand for the App. Given the growing number of images that users are storing on their phone, it is believed that a large user base would download this kind of App for displaying and sharing images to friends and family. However, we believe that there many enhancements and bug fixes must be addressed before commercialising. Furthermore, given the large number of Apps for displaying photos, it would be have to be priced competitively in order to attract customers. It is imagined that a free version could be offered alongside a paid version with enhanced features (online searching, for example).