**High Performance Computing**

**Homework #8**

**Due #1: Thursday April 1 2013 by 12:00 PM (Noon)**

**Due #2: Thursday April 11 2013 by 12:00 PM (Noon)**

**Email-based help Cutoff: 12:00 PM on Wed, Mar 27 2013**

Total Maximum Points: 55

|  |
| --- |
| **Submission Instructions**  This homework assignment must be turned-in electronically via Niihka. Ensure your C++ source code is named *MUid*\_ImageSearch.cpp and MUid\_ImageSearch.h, where *MUid* is your Miami University unique ID. Ensure your comprehensive performance report document (required only for the second part of the homework submission) is named with the convention MUid\_ImageSearch.pdf. Upload these two files onto Niihka. |

|  |
| --- |
| **Objective** |
| The objective of this homework is to continue to gain experience with OpenMP to develop parallel applications by developing a reasonably straightforward image searching program. |

# Grading Rubric:

|  |  |
| --- | --- |
|  | This is an advanced course and consequently the expectations in this course are higher. Accordingly, the program submitted for this homework must pass necessary tests in order to qualify for earning a full score.  **NOTE: Program that do not compile, have methods longer than 25 lines, or just some skeleton code will be assigned zero score.** |

Scoring for this assignment will be determined as follows assuming your program compiles (and is not skeleton code):

* **Part #1 (Due March 28): 25 points**: allocated for overall implementation of the required data parallel program described in Part #1 of this document using good object oriented design techniques.
* **Part #2 (Due April 4): 20 Points:** allocated for extension to program from Part #1.
* **10 points**: allocated for comprehensive performance analysis report to be submitted at the end of Part #2.
* **-1 Points**: for each warning generated by g++ when compiling your C++ program.
* **NOTE:** Points will be deducted for violating stylistic qualities of the program such as: program follows formatting requirements (spacing, indentation, suitable variable names with appropriate upper/lowercase letters, etc). The program includes suitable comments at appropriate points in each method to elucidate flow of thought/logic in each method. Program strives to appropriately reuse as much code as possible.

|  |  |
| --- | --- |
|  | An intermediate solution will **not be** posted for this homework. Consequently plan your work and time accordingly. It may be best to read requirements for both parts of this homework and structure your solution appropriately so that it is extensible to the second part. |

# Background

This homework consists of two parts that collectively requires the development of an OpenMP-parallelized image searching program. The images being used in this homework are Portable Network Graphics (PNG) files (see: <http://en.wikipedia.org/wiki/Portable_Network_Graphics>) that stores each pixel in an image. In other words, the image is a matrix of pixels and each pixel consists of four 8-bit values corresponding to the Red-Green-Blue-Alpha values. There are several different ways to view PNG files on Linux. The common command that I use is:

|  |
| --- |
| $ eog MiamiMarcumCenter.png |

# Starter Code

You are supplied with PNG reading/writing class and several sample test images as part of this homework. Copy the necessary files from the shared folder on Red Hawk using the following command (don’t miss the period at the end):

|  |
| --- |
| $ cp –r /shared/raodm/csex43/homeworks/homework8 . |

In this homework, you are supplied with a PNG class (review the supplied PNG.h file) that facilitates reading and writing of PNG files that contain images in RGBA (Red-Green-Blue-Alpha) format. Note that the supplied PNG class requires the images to be in RGBA format otherwise it will not read such files (and will generate an exception). The pixels in an image may be obtained via call to the PNG::getBuffer() method. This method returns a “flat” buffer in which the pixels are stored in a row-major organization as illustrated in the figure below:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Conceptual organization of pixels in an image*. |  | *Row-major organization of pixel data in the buffer returned by PNG::getBuffer() method.* |

Each pixel consists of four 8-bit values stored in the order Red-Green-Blue-Alpha and they can be accessed as suggested below:

|  |
| --- |
| PNG img;  // Maybe some more code here...  **const** std::vector<**unsigned** **char**>& buffer = img.getBuffer();  **const** **int** index = getPixelIndex(10, 10, img.getWidth());  std::cout << "red = " << buffer[index]  << "green = " << buffer[index + 1]  << "blue = " << buffer[index + 2]  << "Alpha = " << buffer[index + 3] << std::endl; |

## Compiling:

The supplied PNG.cpp utilizes the system image processing libraries. Consequently, when compiling your program ensure you link with the library as shown below:

|  |
| --- |
| $ g++ -g –Wall –O3 –stdc++11 –fopenmp raodm\_ImageSearch.cpp PNG.cpp –o ImageSearch **-lpng** |

# Part #1: Parallelized Sub-Image Search

**Due: Thursday March 28 2013 by 12:00 PM (Noon)**

## Description:

This part of the homework requires development of an suitably parallelized OpenMP-based image search program that can identify and mark (with a red-box) occurrences of a given sub-image in a larger image as illustrated in the figure below:

|  |  |
| --- | --- |
| **The Larger image**: This is the image to search in (first command-line argument to the program) |  |
| **The sub-image**: This is the sub-image to search for (second command-line argument to program) in the given larger image. |  |
| **Resulting image**: This is an image generated by the program in which sub-images in a black-box  (the output image name is supplied as third command-line argument to program). **Note: You need to zoom-in to see the red-boxes around each one of the 50 stars.** |  |

## Search Options:

The search for sub-images must permit the following additional options:

1. Percentage pixel match: A section of the search image must be considered as a match to the given sub-image if sufficient number of pixels matches. This parameter specifies the percentage number of pixels that must match (based on second option below) in order to decide that a region of the image is a match to the given sub-image. This parameter is a value in the range 1 to 100 (corresponding to 0% to 100%). For example, if this parameter is 50, then a 50% pixel match is deemed sufficient to identify a section of the search image as a match to the search image. Note that once a pixel has been positively identified to be part of a sub-image, it should not be reconsidered again as part of subsequent searches.

Accordingly, with a 50% match the image search program should be able to identify stars (using the search sub-image shown earlier) and identify 50 stars on the following image (which has some partial stars showing (possibly because the flag was waving or the camera had some distortion in it):

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Source image (with some of the stars distorted) |  | Resulting image with the stars identified using a 50% match of pixels. |

This is an optional command-line argument to the program. This value may be specified as the fifth command-line argument to the program. If sufficient number of arguments is not supplied, then the default value for this option is 75 (corresponding to 75% match).

1. Pixel color tolerance: The three color values for each pixel (namely RGB) does not need to match exactly and can match approximately with a given tolerance. This option is an integer the range 1 to 255 that indicates the acceptable tolerance between pixel values in the larger search image versus the sub-image. For example, assume that a pixel has values (230, 0, 255) and the tolerance is set to 10 then the following pixels would be a match: (240, 0, 255), (220, 9, 250), (256, 9, 246) while the following pixels will not be a match: (200, 0, 255), (230, 20, 255), (200, 0, 200). In other words, close shades of the same color (other than “50 shades of grey”☺) are acceptable but not shades of other colors.

This is an optional command-line argument to the program. This value may be specified as the sixth command-line argument to the program. If sufficient number of arguments is not supplied, then the default value for this option is 32.

## Command-line Arguments to program:

The following 6 command-line arguments (the last three are optional) must be accepted and suitably processed by the program:

1. The first required command-line argument is the path to the large PNG file to be searched-in.
2. The second required command-line argument is the path to the PNG file that contains the sub-image to be searched for.
3. The third required command-line argument is the name of the output PNG file to which the resulting image is to be written.
4. The fourth optional command-line argument is a Boolean string ("true" or "false") to indicate if the search sub-image is a mask. If this parameter is true, then the sub-image is assumed to be a mask and will be appropriately handled in the second part of this homework. In the first part of this homework you may simply assume that this parameter is always false and ignore it if it is specified.
5. The fifth optional command-line argument is the desired percentage pixel match (as described earlier) to determine a match between the given sub-image and the search image. If this parameter is not specified then use 75% as the default value.
6. The sixth optional command-line argument is the pixel color tolerance (as described earlier). If this argument is not specified then use 32 as the default value.

## Part #1 Sample Outputs

Expected outputs from multiple independent runs of the completed program are shown below (resulting images with suitable red-boxes are not shown but are necessary for full score in this homework):

|  |
| --- |
| **$ ./ImageSearch MiamiMarcumCenter.png WindowPane.png result.png false 50 32**  **sub-image matched at: 576, 1790, 613, 1835**  **sub-image matched at: 582, 1994, 619, 2039**  **Number of matches: 2** |

|  |
| --- |
| $ ./ImageSearch MiamiMarcumCenter.png WindowPane.png result.png false 50 64  sub-image matched at: 564, 1019, 601, 1064  sub-image matched at: 566, 816, 603, 861  sub-image matched at: 567, 1224, 604, 1269  sub-image matched at: 568, 616, 605, 661  sub-image matched at: 574, 1790, 611, 1835  sub-image matched at: 580, 1994, 617, 2039  sub-image matched at: 589, 2196, 626, 2241  sub-image matched at: 604, 1018, 641, 1063  sub-image matched at: 606, 616, 643, 661  sub-image matched at: 606, 1223, 643, 1268  sub-image matched at: 614, 1789, 651, 1834  sub-image matched at: 619, 1993, 656, 2038  sub-image matched at: 627, 2195, 664, 2240  sub-image matched at: 798, 1208, 835, 1253  Number of matches: 14 |

|  |
| --- |
| **$ ./ImageSearch TestImage.png a.png result.png**  sub-image matched at: 49, 654, 62, 666  sub-image matched at: 49, 677, 62, 689  sub-image matched at: 49, 764, 62, 776  sub-image matched at: 120, 295, 133, 307  sub-image matched at: 144, 39, 157, 51  sub-image matched at: 144, 128, 157, 140  sub-image matched at: 144, 482, 157, 494  sub-image matched at: 144, 504, 157, 516  sub-image matched at: 144, 617, 157, 629  sub-image matched at: 144, 797, 157, 809  sub-image matched at: 168, 68, 181, 80  sub-image matched at: 202, 136, 215, 148  sub-image matched at: 202, 396, 215, 408  sub-image matched at: 202, 467, 215, 479  sub-image matched at: 250, 813, 263, 825  sub-image matched at: 298, 430, 311, 442  Number of matches: 16 |

|  |
| --- |
| $ **./ImageSearch Flag\_Unknown.png star.png result.png false 50 8**  sub-image matched at: 15, 34, 75, 97  sub-image matched at: 15, 166, 75, 229  sub-image matched at: 15, 299, 75, 362  sub-image matched at: 15, 432, 75, 495  sub-image matched at: 15, 564, 75, 627  sub-image matched at: 15, 697, 75, 760  sub-image matched at: 74, 99, 134, 162  sub-image matched at: 74, 232, 134, 295  sub-image matched at: 74, 364, 134, 427  sub-image matched at: 74, 497, 134, 560  sub-image matched at: 74, 630, 134, 693  sub-image matched at: 131, 33, 191, 96  sub-image matched at: 131, 166, 191, 229  sub-image matched at: 131, 298, 191, 361  sub-image matched at: 131, 431, 191, 494  sub-image matched at: 131, 564, 191, 627  sub-image matched at: 131, 696, 191, 759  sub-image matched at: 188, 99, 248, 162  sub-image matched at: 188, 232, 248, 295  sub-image matched at: 188, 365, 248, 428  sub-image matched at: 188, 497, 248, 560  sub-image matched at: 188, 630, 248, 693  sub-image matched at: 245, 33, 305, 96  sub-image matched at: 245, 166, 305, 229  sub-image matched at: 245, 298, 305, 361  sub-image matched at: 245, 431, 305, 494  sub-image matched at: 245, 564, 305, 627  sub-image matched at: 245, 696, 305, 759  sub-image matched at: 301, 101, 361, 164  sub-image matched at: 301, 234, 361, 297  sub-image matched at: 301, 366, 361, 429  sub-image matched at: 301, 499, 361, 562  sub-image matched at: 301, 632, 361, 695  sub-image matched at: 359, 33, 419, 96  sub-image matched at: 359, 166, 419, 229  sub-image matched at: 359, 299, 419, 362  sub-image matched at: 359, 431, 419, 494  sub-image matched at: 359, 564, 419, 627  sub-image matched at: 359, 697, 419, 760  sub-image matched at: 415, 101, 475, 164  sub-image matched at: 416, 232, 476, 295  sub-image matched at: 416, 365, 476, 428  sub-image matched at: 416, 498, 476, 561  sub-image matched at: 416, 630, 476, 693  sub-image matched at: 472, 34, 532, 97  sub-image matched at: 472, 167, 532, 230  sub-image matched at: 472, 299, 532, 362  sub-image matched at: 472, 432, 532, 495  sub-image matched at: 472, 565, 532, 628  sub-image matched at: 472, 697, 532, 760  Number of matches: 50 |

## Part #1 Turn-in

Once you have verified correct parallel operation of your OpenMP-based image search program, turn-in just your C++ source files to Niihka.

# Part #2: Parallelized mask-Image Search

**Due: Thursday April 4 2013 by 12:00 PM (Noon)**

|  |
| --- |
| star_mask.pngand.pngStarStrip_mask.png |
| Example Masks |

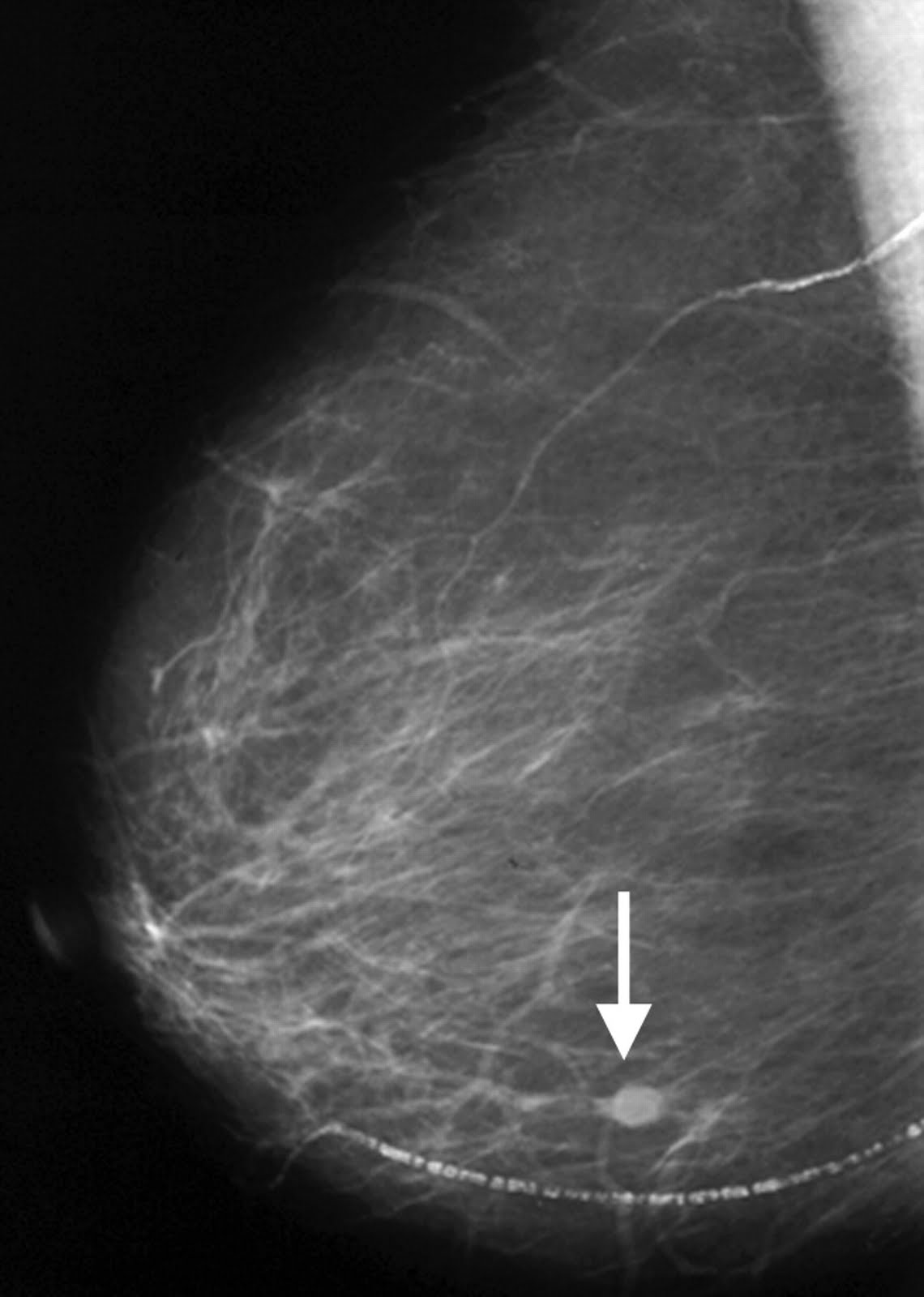
## Description:

This part of the homework requires extending the image search program developed in the previous part to also handle searching for matches using an image “mask”. An image mask is essentially a black-and-white image that essentially represents a pattern to search for. Examples of image masks are shown in the adjacent figure. The pixels in an image mask are interpreted as follows:

* Pixels that are black essentially define the “background” for the information we are searching.
* Pixels that are not-black (or shades of grey) is the information or meaningful pattern to be identified.

The objective of the search is to essentially distinguish the meaningful information immaterial of the (background and foreground) colors – the key distinguishing factor is sufficient contrast between the background and foreground (with black-and-white being the extreme contrast) as illustrated by the following examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Mask | Successful Match | Successful Match | Successful Match | Unsuccessful Match |

Searching using masks is a powerful strategy not only for color agnostic processing but is also a handy strategy for processing images and information where the contrast in regions of the images is important. For example, this strategy can be used to identify high risk areas with potential tumors from mammograms (see adjacent image), high-contrast CT scans, and MRI images.

## Searching using a Mask:

In the previous part of the homework, the pixels were matched based on their RGB color components. However, with a mask the key distinguishing factor is the contrast between background (identified by black pixels in the mask) versus the information (or foreground identified by white pixels in the mask). Consequently, the image search must proceed in the following manner:

1. Given a region of the image to search, first compute the average background- averaging the color of the pixels corresponding to the black pixels in the mask. For example, given the series of star images above, in the first successful case, all pixels corresponding to the light-blue color would be averaged (since the corresponding pixels in the mask are black).
2. Once the average background color has been determined, next recheck each pixel to verify if it matches the mask in the following manner:
   1. If the corresponding pixel in the mask is black, then the pixel should be “*same shade*§” of the average background.
   2. If the corresponding pixel in the mask is white, then the pixel should not be the “*same shade*§” as the average background.

* In this context, two colors are considered to be the “*same shade*” as determined by pixel color tolerance value (sixth optional command-line argument to the program).

1. The net matching pixels are computed by subtracting the number of differing pixels from the number of matching pixels. If the net number of matching pixels is more than the required percentage pixel match (fifth optional command-line argument to the program) then a match is found.

## Command-line Arguments to program:

The command-line arguments are exactly the same as in Part #1 with the following one difference:

* The fourth optional command-line argument is a Boolean string ("true" or "false") to indicate if the search sub-image is a regular image or a mask. If this parameter is true, then the sub-image is assumed to be a mask and will be appropriately handled. Otherwise (the argument is false) the sub-image is processed as described in Part #1 of this homework assignment.

## Graduate Students:

The program for graduate students I also expected to identify conditions that are 90°, 180°, and 270° rotations of the mask.

## Part #2 Sample Outputs:

Expected outputs from multiple independent runs of the completed program are shown below (resulting images with suitable red-boxes are not shown but are necessary for full score in this homework):

|  |
| --- |
| **$ ./ImageSearch TestImage.png and\_mask.png result.png true 75 16**  sub-image matched at: 73, 630, 85, 660  sub-image matched at: 120, 310, 132, 340  sub-image matched at: 202, 677, 214, 707  sub-image matched at: 226, 864, 238, 894  sub-image matched at: 274, 67, 286, 97  sub-image matched at: 396, 63, 408, 93  sub-image matched at: 444, 860, 456, 890  sub-image matched at: 468, 673, 480, 703  sub-image matched at: 550, 306, 562, 336  sub-image matched at: 597, 626, 609, 656  sub-image matched at: 67, 961, 97, 973  sub-image matched at: 310, 1115, 340, 1127  sub-image matched at: 630, 1162, 660, 1174  sub-image matched at: 677, 1033, 707, 1045  sub-image matched at: 864, 1009, 894, 1021  sub-image matched at: 44, 1429, 74, 1441  sub-image matched at: 231, 1405, 261, 1417  sub-image matched at: 278, 1276, 308, 1288  sub-image matched at: 598, 1323, 628, 1335  sub-image matched at: 841, 1477, 871, 1489  Number of matches: 20 |

|  |
| --- |
| **$ ./ImageSearch TestImage.png i\_mask.png result.png true**  sub-image matched at: 45, 243, 60, 248  sub-image matched at: 45, 249, 60, 254  sub-image matched at: 45, 377, 60, 382  sub-image matched at: 45, 383, 60, 388  sub-image matched at: 45, 528, 60, 533  sub-image matched at: 45, 545, 60, 550  sub-image matched at: 45, 750, 60, 755  …  sub-image matched at: 825, 1272, 830, 1287  sub-image matched at: 835, 1158, 840, 1173  sub-image matched at: 837, 1005, 842, 1020  sub-image matched at: 845, 1158, 850, 1173  sub-image matched at: 874, 1135, 879, 1150  sub-image matched at: 875, 1449, 880, 1464  sub-image matched at: 894, 1367, 899, 1382  Number of matches: 601 |

|  |
| --- |
| **$ ./ImageSearch Flag\_of\_the\_US.png star\_mask.png result.png true 50 32**  sub-image matched at: 20, 35, 80, 98  sub-image matched at: 20, 168, 80, 231  sub-image matched at: 20, 300, 80, 363  sub-image matched at: 20, 433, 80, 496  sub-image matched at: 20, 566, 80, 629  sub-image matched at: 20, 698, 80, 761  sub-image matched at: 77, 101, 137, 164  sub-image matched at: 77, 234, 137, 297  sub-image matched at: 77, 367, 137, 430  sub-image matched at: 77, 499, 137, 562  sub-image matched at: 77, 632, 137, 695  sub-image matched at: 133, 36, 193, 99  sub-image matched at: 133, 169, 193, 232  sub-image matched at: 133, 301, 193, 364  sub-image matched at: 133, 434, 193, 497  sub-image matched at: 133, 567, 193, 630  sub-image matched at: 133, 699, 193, 762  sub-image matched at: 190, 102, 250, 165  sub-image matched at: 190, 235, 250, 298  sub-image matched at: 190, 368, 250, 431  sub-image matched at: 190, 500, 250, 563  sub-image matched at: 190, 633, 250, 696  sub-image matched at: 247, 36, 307, 99  sub-image matched at: 247, 168, 307, 231  sub-image matched at: 247, 301, 307, 364  sub-image matched at: 247, 434, 307, 497  sub-image matched at: 247, 566, 307, 629  sub-image matched at: 247, 699, 307, 762  sub-image matched at: 304, 102, 364, 165  sub-image matched at: 304, 235, 364, 298  sub-image matched at: 304, 367, 364, 430  sub-image matched at: 304, 500, 364, 563  sub-image matched at: 304, 632, 364, 695  sub-image matched at: 361, 35, 421, 98  sub-image matched at: 361, 168, 421, 231  sub-image matched at: 361, 301, 421, 364  sub-image matched at: 361, 433, 421, 496  sub-image matched at: 361, 566, 421, 629  sub-image matched at: 361, 699, 421, 762  sub-image matched at: 418, 102, 478, 165  sub-image matched at: 418, 234, 478, 297  sub-image matched at: 418, 367, 478, 430  sub-image matched at: 418, 499, 478, 562  sub-image matched at: 418, 632, 478, 695  sub-image matched at: 475, 35, 535, 98  sub-image matched at: 475, 168, 535, 231  sub-image matched at: 475, 300, 535, 363  sub-image matched at: 475, 433, 535, 496  sub-image matched at: 475, 566, 535, 629  sub-image matched at: 475, 698, 535, 761  Number of matches: 50 |

|  |
| --- |
| **$ ./ImageSearch MiamiMarcumCenter.png WindowPane\_mask.png result.png true 50 64**  sub-image matched at: 567, 818, 601, 859  sub-image matched at: 567, 1021, 601, 1062  sub-image matched at: 568, 619, 602, 660  sub-image matched at: 568, 1226, 602, 1267  sub-image matched at: 578, 1791, 612, 1832  sub-image matched at: 582, 1996, 616, 2037  sub-image matched at: 590, 2198, 624, 2239  sub-image matched at: 605, 817, 639, 858  sub-image matched at: 605, 1020, 639, 1061  sub-image matched at: 606, 618, 640, 659  sub-image matched at: 607, 1225, 641, 1266  sub-image matched at: 616, 1791, 650, 1832  sub-image matched at: 620, 1995, 654, 2036  sub-image matched at: 627, 2197, 661, 2238  sub-image matched at: 816, 1209, 850, 1250  Number of matches: 15 |

## Performance Analysis

A section of your report must be dedicated to provide evidence (typically a performance profile is used) to support your rationale for parallelizing a specific section(s) of your program.

Once you have tested and verified correct operation of the parallel program, develop a report containing a comprehensive performance analysis (using at least 5 sample runs with 95% confidence intervals) of your program using 1 through 8 threads using three different images (namely: Flag\_Unknown.png with star\_mask.png, TestImage.png with and\_mask.png, MiamiMarcumCenter.png with window\_mask.png) and plot the three different sets of data using a suitable graphs that contrasts the theoretical vs. observed speedups. You may use the experimental documents from Exercise #9 and earlier homework as a guide to develop your final report.

## Part #2 Turn-in

Submit your C++ source file (MUid\_ImageSearch) and your report document (a PDF file) named wit the convention MUid\_ImageSearch.pdf that meet the requirements of this homework via Niihka. No credit will be given for submitting code that does not compile or just skeleton code. Verify that your program meets all the requirements as stated in the grading rubric. Ensure your C++ source files are named with the stipulated naming convention. Upload all the necessary C++ source files to onto Niihka. Do not submit zip/7zip/tar/gzip files. Upload each file independently.