Tecnológico de Monterrey

Programming languages

RecImage

By:

Manuel García Huerta A01701414
Francisco Palacios Arriaga A01364223

Index

Introduction

“The image recognition market is estimated to grow from USD 15.95 Billion in 2016 to USD 38.92 Billion by 2021, at a CAGR of 19.5% between 2016 and 2021.”

The technology advancements, that are currently being developed in the fields of machine learning and high bandwidth data services, are rapidly launching a new era as well as increasing necessity around the identification of objects in images, videos or even audio.
According to the report by MarketsandMarkets, the image recognition industry is divided in three main sectors: hardware, software and services. And each of them are currently being forced to create better and stronger products because of security issues around the globe.

As a matter of fact, in the United States, Homeland Security confirmed that for 2021 the top 20 airports in the states will implement facial recognition as an onboarding passing system for all international travellers. This type of technology is aiming to reduce the problems of boarding different types of clients, and assuring the best well being for each of them by identifying possible threats.This same story is repeated along different first world countries like United Kingdom, United Arab Emirates, Ireland and more.

Basically, all the advancements in the image processing field are due to the different algorithms like:

- Error diffusion
- Floyd–Steinberg dithering
- Ordered dithering
- Riemersma dithering
- Canny edge detector: detect a wide range of edges in images
- Generalised Hough transform
- Hough transform
- Marr–Hildreth algorithm
- SIFT (Scale-invariant feature transform)
- SURF (Speeded Up Robust Features)

As well as, all the devices that can output different media in order to process.

Hence, how can we provide an advantage in cutting edge world applications?, this type of algorithms and media involves a lot of time consuming tasks that a common computer would deliver after the information is needed, affecting to all the users involved, furthermore, to all the third parties that were protected or benefited from it.

As a solution, we propose to apply the concurrency paradigm to acquire the media into our machines in a faster way, reducing the input time for each image that is needed to process, and applying a basic image recognition technique parallelized, we can show the increase in performance output.

Concurrency and parallelization

Concurrency means executing multiple tasks at the same time but not necessarily simultaneously. In a concurrent application, two tasks can start, run, and complete in overlapping time periods.

By introducing this concept, we decided to take an approach into this programming paradigm because of the advantages at processing big amounts of information in less time than a linear execution. Java, being optimized to work with concurrency thanks to its libraries,

allowed us to acquire an image as input and parse it into a matrix of integers based on the rgb conversion.

With help of the ForkJoinPool class from Java, we are able to divide the whole image into smaller pieces and process each one concurrently, extracting the integer value of each pixel. By using this paradigm we are able to reduce this image processing to a half or a third of the time it would take with a sequential paradigm. After this conversion is done, we store the matrix in a file for other programs to consume it.

Parallelization

Parallelization means the breakdown of a problem into several pieces where each of them must not affect the result of the others, and after finished the complete solution can be done by joining the result of the breakdown. If a problem can be parallelized, we are able to program a computer to process every piece of the problem at the same time, reducing the task into into t / n, t being total time of the problem and n meaning the number of parts.

So, how can we take advantage of this approach nowadays? Easily, with a graphic processing unit (GPU). A GPU is meant to perform multiple math calculations in a faster way due to the existence of many small cores optimized for multitasking, in order to free the CPU from doing them. Today, GPU's are used mostly at rendering images, videos and in the gaming environment, but another great of employing them are in science.

By using Compute Unified Device Architecture (CUDA®) , a parallel computing platform and programming model developed by NVIDIA, we can interact with a GPU, developing many applications that would be benefited by the parallelization paradigm.

Implementation

The C program starts by validating the inputs given, if everything is correct it then proceeds to call a Java program that reads the original image from the path given, it concurrently processes the image by breaking it down in smaller pieces and accessing each pixel to get its Integer value. All this information is stored in an integer matrix with the same size as the image and it is then written on a file for other programs to consume it.

Once we acquire the images as processed rgb matrices with integers from the file created by the java program, we start the CUDA execution program.

At CUDA, the image recognition algorithm that we employ is based on the comparison of an image to find into an starting image, where we compare the first one pixel by pixel in every single part of the starting image in a parallelized way. We do this in two different rotations in order to be able to find the image even when rotated 180º degrees.

At runtime we iterate through all the starting image as a matrix, where thanks to CUDA we can create a thread for each cell in the starting matrix, an perform a search of every single pixel of the “tofind” matrix using as an starting point the cell that the thread has. The search that we made has a margin of error, which the user can modify to give a loose into an equality of two numbers.

The search is the result of the matches that each cell had when comparing them divided by the number of comparisons, for example: if we have a 10 x 10 matrix and we try to find a 2 x 2, the maximum number of matches for each cell of the first matrix will be 4, and the amount of comparisons is 4, giving a result of 1 if the 2 x 2 matrix exists on the 10 x 10.

A margin of error was implemented because of the following:

- Many image software programs affects the images, in actions like crops, tuning and creating images.
- As well, the equality between images can differ by a lot of factors.

Retaking the running of the program, after each thread finished we store the result of the lookup into a resulting matrix of the size of the starting one, and we find the maximum percentage of matching that we made with the time spent on the realization of the algorithm.

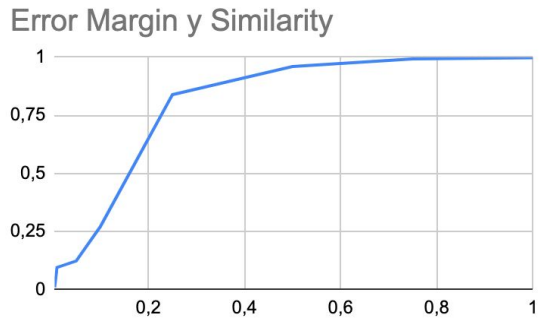
Results

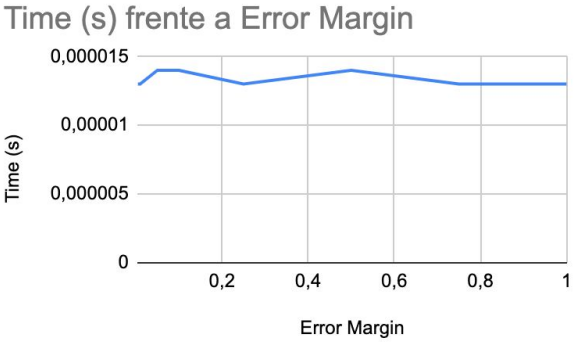
In order to prove the image matching algorithm and the different time that it took, we runned the program with several images of high quality definition (1920*1280) and multiple finding images, searching with a couple of margins of errors. The images used here are saved under the /images folder with the same names as stated in this document.

These are the results of finding an existing part of the image inside:

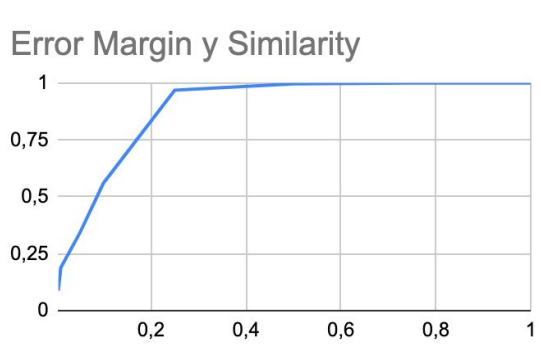
Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
bearOriginal	1280*1920	bearToFind	239*157
Error Margin	Similarity	Time (s)	
0	0,009354	0,000013	
0,001	0,094662	0,000013	
0,005	0,122591	0,000013	
0,01	0,269488	0,000013	
0,05	0,83885	0,000014	
0,1	0,959838	0,000014	
0,25	0,992911	0,000013	

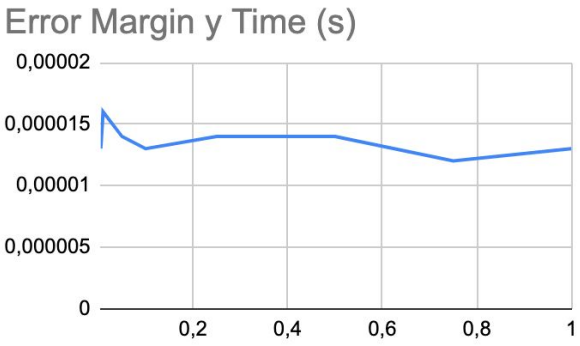
0,5	0,997975	0,000014	
0,75	0,998721	0,000013	
1	0,999174	0,000013	



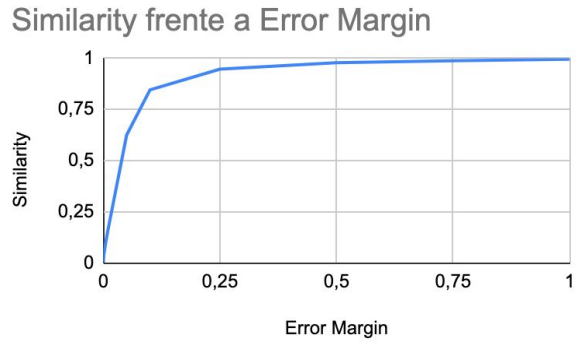


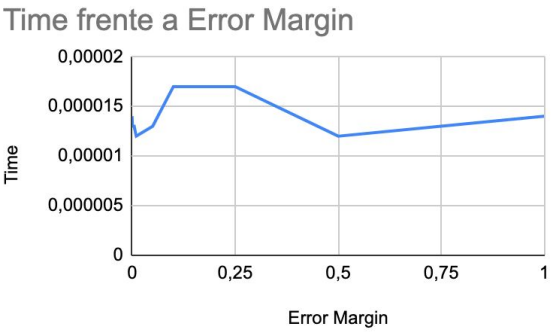
Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
knightOriginal	1920*1280	knightToFind	199*142
Error Margin	**Similarity**	**Time (s)**	
0	0,08886	0,000013	
0,001	0,189044	0,000016	
0,005	0,34185	0,000014	
0,01	0,560762	0,000013	
0,05	0,96889	0,000014	
0,1	0,996497	0,000014	
0,25	0,999929	0,000012	
0,5	1	0,000013	
0,75	1	0,000014	
1	1	0,000014	



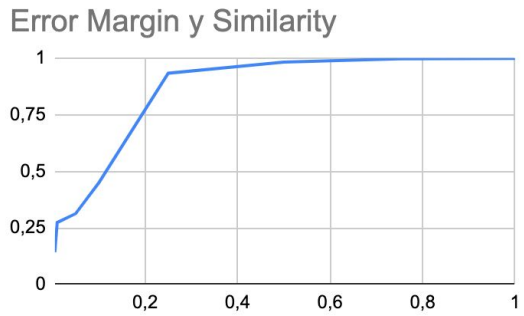


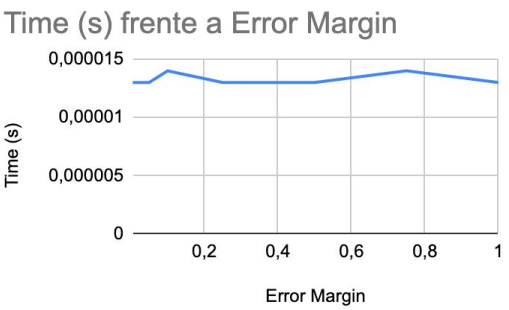
Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
spaceOriginal	1079*1920	spaceToFind	66*272
Error Margin	Similarity	Time (s)	
0	0,002507	0,000014	
0,001	0,037767	0,000013	
0,005	0,099822	0,000013	
0,01	0,162879	0,000012	
0,05	0,62433	0,000013	
0,1	0,84531	0,000017	
0,25	0,946803	0,000017	
0,5	0,97822	0,000012	
0,75	0,987355	0,000013	
1	0,995599	0,000014	



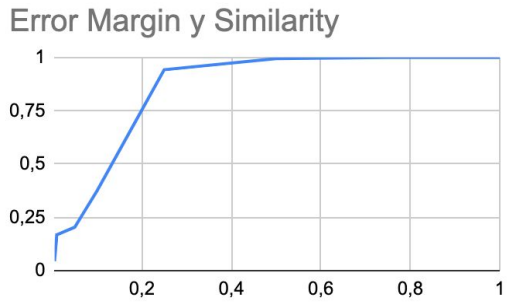


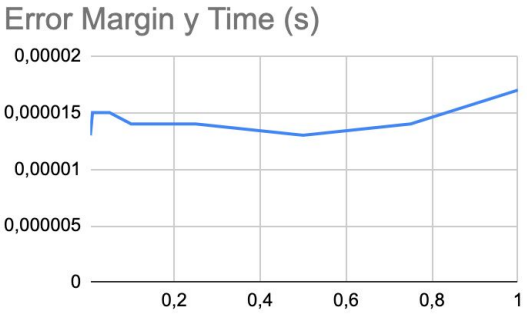
Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
coffeOriginal	1920*1280	coffeToFind	132*314
Error Margin	Similarity	Time (s)	
0	0,142106	0,000013	
0,001	0,273885	0,000013	
0,005	0,313115	0,000013	
0,01	0,448586	0,000013	
0,05	0,93331	0,000013	
0,1	0,982653	0,000014	
0,25	0,99655	0,000013	
0,5	0,99848	0,000013	
0,75	0,998987	0,000014	
1	0,999349	0,000013	





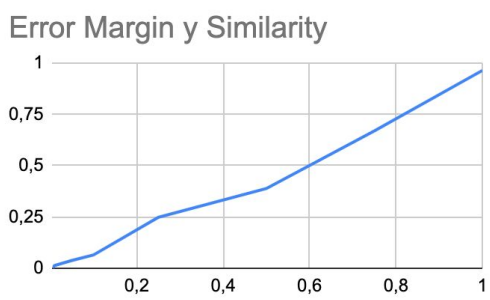
Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
personOriginal	1919*1280	personToFind	160*240
Error Margin	Similarity	Time (s)	
0	0,044948	0,000013	
0,001	0,168437	0,000015	
0,005	0,20401	0,000015	
0,01	0,374974	0,000014	
0,05	0,94268	0,000014	
0,1	0,995052	0,000013	
0,25	0,999974	0,000014	
0,5	1	0,000017	
0,75	1	0,000018	
1	1	0,000015	

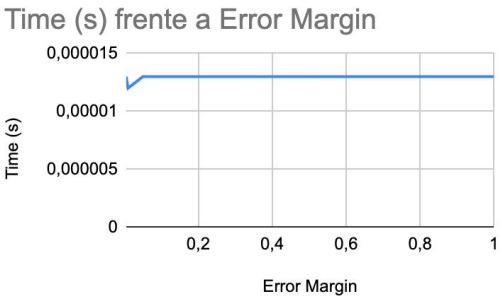




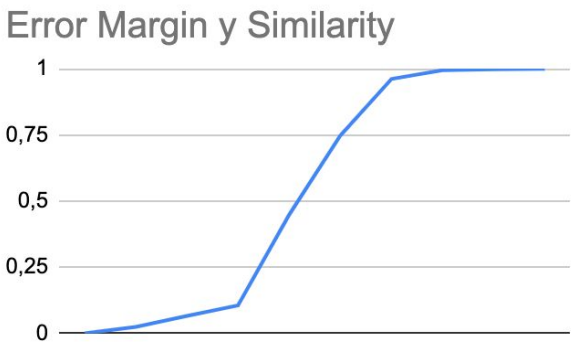
These are the results of trying to find an image part that does not belong to the starting image:

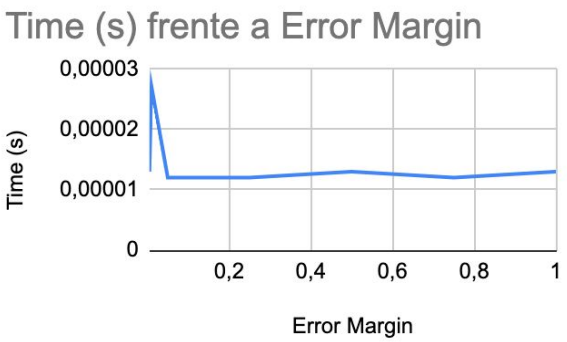
Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
bearOriginal	1280*1920	knightToFind	199*142
Error Margin	Similarity	Time (s)	
0	0,000531	0,000012	
0,001	0,013872	0,000029	
0,005	0,038679	0,000013	
0,01	0,065433	0,000012	
0,05	0,24867	0,000013	
0,1	0,388952	0,000013	
0,25	0,669616	0,000013	
0,5	0,963798	0,000013	
0,75	0,992781	0,000013	
1	1	0,000013	



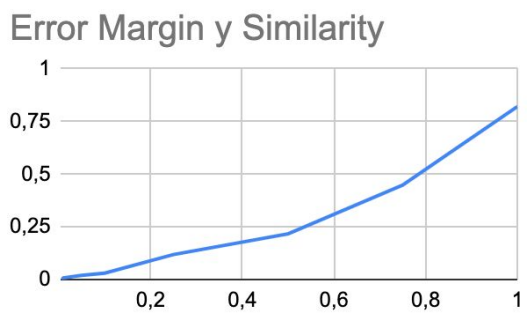


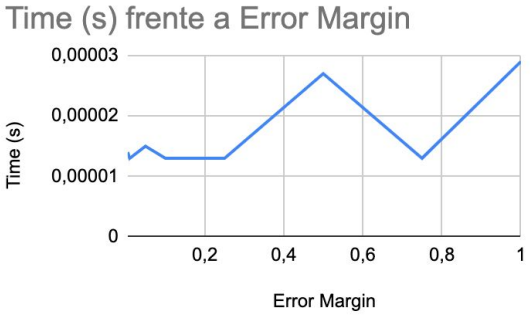
Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
knightOriginal	1920*1280	spaceToFind	66*272
Error Margin	Similarity	Time (s)	
0	0,00039	0,000012	
0,001	0,024176	0,000013	
0,005	0,065842	0,000013	
0,01	0,105949	0,000027	
0,05	0,44814	0,000012	
0,1	0,746936	0,000012	
0,25	0,961063	0,000012	
0,5	0,993817	0,000013	
0,75	0,997438	0,000012	
1	0,999276	0,000013	





Original Image Name	Original Image Size	ToFind Image Name	ToFind Image Size
coffeeOriginal	1920*1280	personToFind	160*240
Error Margin	Similarity	Time (s)	
0	0,000443	0,000027	
0,001	0,008359	0,000014	
0,005	0,020703	0,000014	
0,01	0,030703	0,000013	
0,05	0,11875	0,000015	
0,1	0,216667	0,000013	
0,25	0,448177	0,000013	
0,5	0,819974	0,000027	
0,75	0,997708	0,000013	
1	1	0,000029	





Conclusions

After running the tests with several images of different sizes, and collecting the data presented above, it is possible to state that the average time taken to look for similarity of a maximum size 300*300 image in a maximum size 1920*1280 image is 0.0000143625 s, having a minimum time of 0.000012 s and maximum of 0.000029 s.

This algorithm runs considerably fast, and could be scaled to be mounted over a camera that will repeatedly be searching for patterns. The test cases include searching for a human face and was proved as reliable.

We were able to find out that the program *Preview* on the iOS software changes the images when cropped, the image suffers from modification and while to the human eye is imperceptible, bitwise it does change. Since we are not able to determine the tool used to crop images, we decided to add a margin of error for the user to customize the restrictions of the program. With a higher error margin the similarity of the images tend to increase, as seen on the data above. It is also noticeable how for those images to find that were indeed a part of the original image, the similarity raised above 85% with a 5% error margin (due to the modification of the *Preview* program), however for those images to find that were not a part of the original image, the similarity raised above 85% at around a 25% to 50% error margin.

Setup

```
javac ImgIntoMatrix.java
nvcc mipro.cu -o mipro
gcc checkSimilarity.c -o checkSimilarity
```

Run

./checkSimilarity ./pathOriginalImage ./pathImgTiFind [0...1] [0,1]

References

[1]"CUDA Zone", NVIDIA Developer, 2019. [Online]. Available: https://developer.nvidia.com/cuda-zone. [Accessed: 22- Nov- 2019].

[2]S. Harding, "What Is a GPU? A Basic Definition of Graphics Cards", Tom's Hardware, 2019. [Online]. Available: https://www.tomshardware.com/reviews/gpu-graphics-card-definition,5742.html. [Accessed: 22- Nov- 2019].

[3]"Parallel Processing - an overview | ScienceDirect Topics", Sciencedirect.com, 2019. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/parallel-processing. [Accessed: 22- Nov- 2019].

[4]"Parallel Computing: Overview, Definitions, Examples and Explanations", Web.eecs.umich.edu, 2019. [Online]. Available: https://web.eecs.umich.edu/~qstout/parallel.html. [Accessed: 22- Nov- 2019].

[5]"Reading 17: Concurrency", Web.mit.edu, 2019. [Online]. Available: https://web.mit.edu/6.005/www/fa14/classes/17-concurrency/. [Accessed: 22- Nov- 2019].

[6]"Concurrency, Parallelism, Threads, Processes, Async and Sync — Related? 🤔", Medium, 2019. [Online]. Available: https://medium.com/swift-india/concurrency-parallelism-threads-processes-async-and-sync-related-39fd951bc61d. [Accessed: 22- Nov- 2019].

[7]"What are the common algorithms used in image processing? - Quora", Quora.com, 2019. [Online]. Available: https://www.quora.com/What-are-the-common-algorithms-used-in-image-processing. [Accessed: 22- Nov- 2019].

[8]"How face recognition is taking over airports", CNN Travel, 2019. [Online]. Available: https://edition.cnn.com/travel/article/airports-facial-recognition/index.html. [Accessed: 22- Nov- 2019].

[9]"Facial Recognition At U.S. Airports. Should You Be Concerned?", Forbes.com, 2019. [Online]. Available: https://www.forbes.com/sites/kateoflahertyuk/2019/03/11/facial-recognition-to-be-deployed-at-top-20-us-airports-should-you-be-concerned/#79c3278e7d48. [Accessed: 22- Nov- 2019].

[10]"What is the Working of Image Recognition and How it is Used?", Maruti Techlabs, 2019. [Online]. Available: https://marutitech.com/working-image-recognition/. [Accessed: 22- Nov- 2019].

[11]"Image Recognition", Mathworks.com, 2019. [Online]. Available: https://www.mathworks.com/discovery/image-recognition.html. [Accessed: 22- Nov- 2019].