**Technical Report**

Dima Vlad,

Computational Optimization Master, Year 1

**Content**

1. **Problem Presentation**
2. **State-of-the-art**
3. **Proposed System**
4. **Results**
5. **Comparison with other methods**
6. **Future work**
7. **Conclusions**
8. **Bibliography**
9. **Problem Presentation**

**The problem**: Given a query image, retrieve similar images from a large database of images.

**Requirements**: Implement a search system that has the following mandatory characteristics:

1. scalable
2. robust
3. heterogeneous
4. correctness and efficiency
5. **State-of-the-art**

In article [1] they present and implement an exploration system for large image databases that allows users to find, in an efficient way, similar images from a query image. The similarity level of images is given by the color information. The system is built in three stages:

1. Feature extraction stage in which images are represented in a way that allows efficient storage and fast results retrieval.
2. Clustering the images using k-means clustering in which the clustroid would allow quick human comprehension of the type of images within the corresponding cluster
3. Display the results to the user

The system performance was evaluated based on the retrieval accuracy, on the perceptual similarity order among retrieved images and time complexity. For that, a database of 2100 images was used. Time complexity for retrieving results can be reduced by using a hierarchy of clusters.

Technologies available:

1. **OpenCV**: OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library written in C and CUDA that allows the use of GPU.
2. **JavaEE**: it is a platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications. In our project we use it to develop the web application and to communicate between modules.
3. **Oracle Database**: database to store and retrieve related information such as image link, path to local stored feature vector for a specific image, account information of users.
4. **AspectJ**: offers support for implementation based on AOP.
5. **Proposed System**

We propose a distributed system that is divided in 3 main modules: Web Application, File Master and File Slave. We chose a distributed system because it allows us to save our database in a distributed manner and execute our search queries in a parallel way. Each module will be deployed on a web server and for communication between them, we used HTTP protocol and a partial defined structure of JSON message (as it can be seen in Figure 1).

Figure . JSON message Example



**Web Application** - The application users use it via a browser to access our searching system

**File Master** – Main purpose of this module is to manage the system, from registering or unregistering File Slaves to synchronization problems. Also provides an API for the Searching System that will be used by the Web Application or other clients.

**File Slave** – provides an API for File Master to upload images to its database by extracting feature vectors from them available at a given URI, to execute queries on its database of images.

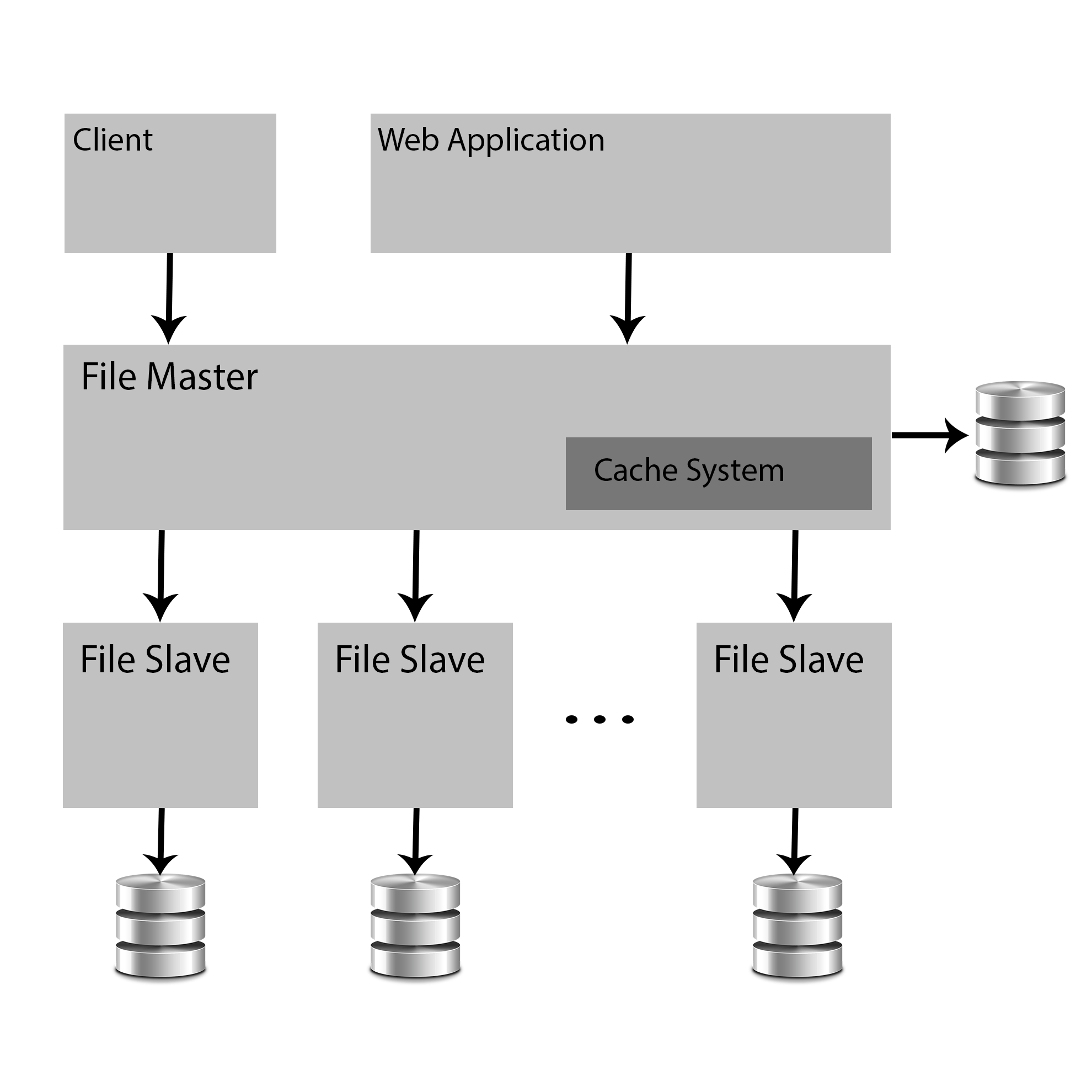


Figure 2. System Arhitecture

1. **Results**

The results are from a database of 8500images (feature vectors) and 40 clusters. For the Search Time, only the time for getting the results sets from DB was taken in consideration, other values such as the time it takes to send the results from File Slave to File Master via HTTP was ignored.

|  |  |  |
| --- | --- | --- |
|  | 1-NN | k-means |
| Memory Usage | 8500 \* 384 \* 8 | 40 \* n \* 384 \* 8 |
| Search Space | 8500 | 40 + n |
| Search Time (seconds) | 1.152 | 0.024 |
| Training Time (seconds) | 0 | 19.387 |

Table 1. Bad results from input image on the right

The value n represents the number of instances that belong to a specific cluster. The number 384 represents the size of feature vectors extracted by dividing the images in 8x8 blocks and for each block we calculate the mean and variance of red, green and blue channel.

In terms of results accuracy, we have some cases in which the results are less accurate in terms of colors but the structural similarity remains. This is due to the fact that the database doesn`t contain a large variety of images and because we used the RGB color space.

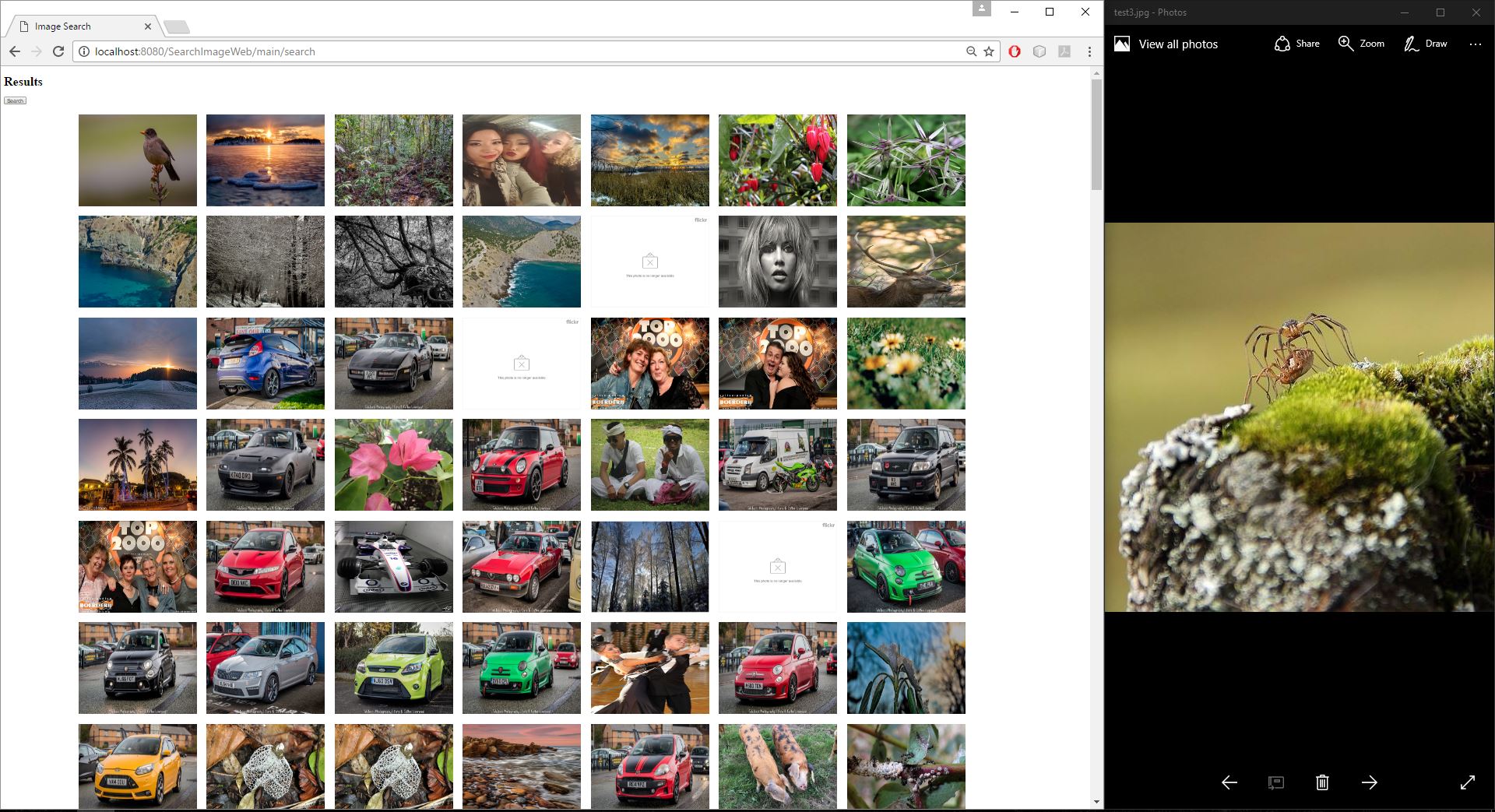


Figure 3. Bad results from input image on the right

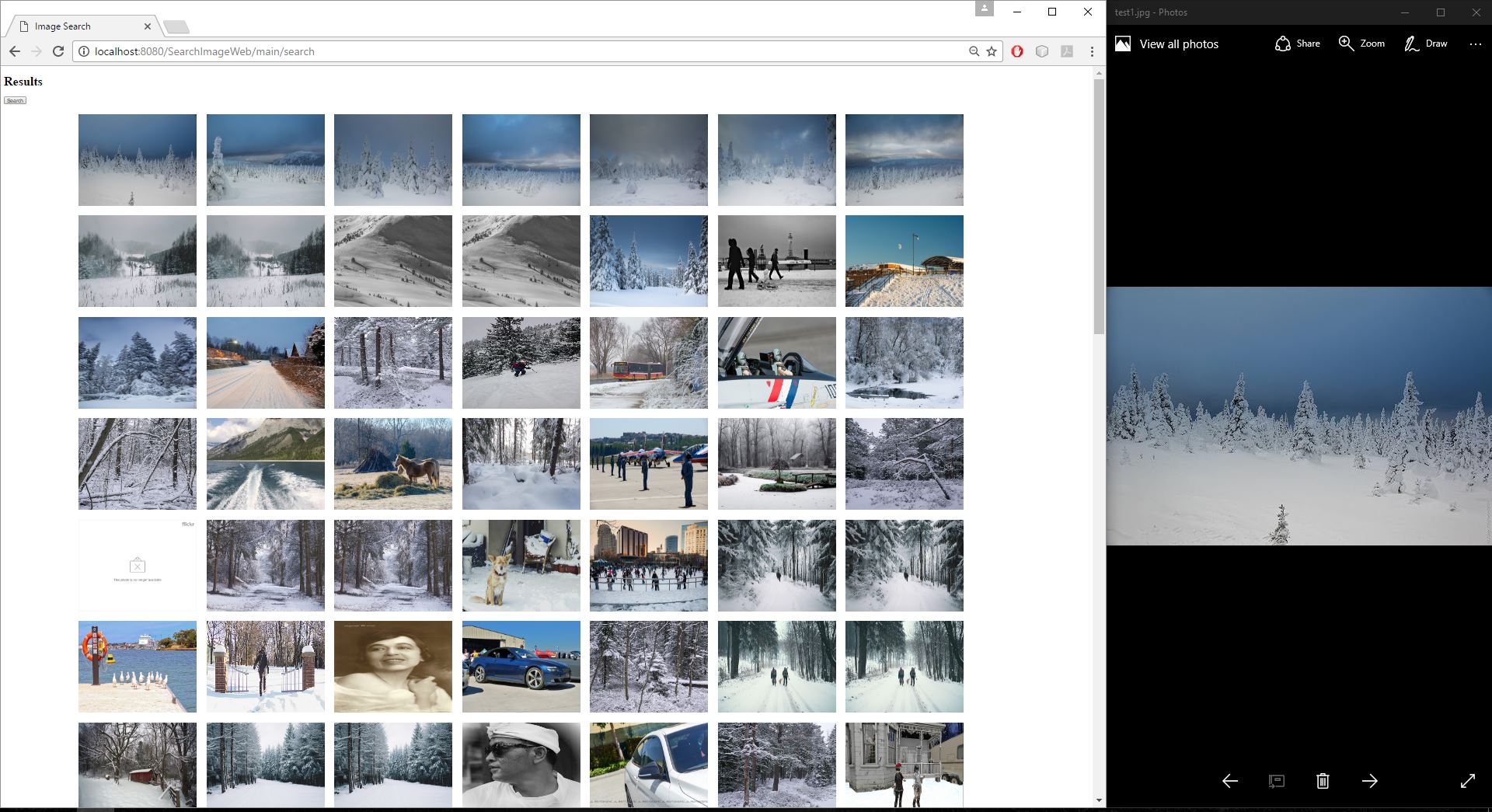


Figure 4. Good results from input image on the right

1. **Comparison with other methods**

In comparison with the system presented in article [1], in which they extracted feature vectors from images that were converted from RGB color space to HSV, I chose to extract them from images that were in RGB. As presented in the previous chapter, this decision affected the results in negative way. Also in article [1] they used a more complex similarity function that takes into consideration more or less each color channel, in contrast with the Euclidean distance I used.

1. **Future work**

In the future, I would like to test the system based on feature vectors extracted from images that are in HSV color space. In this case, I will use a different distance measure for each color channel, based on the values of that channel (for example, for Hue I would use a distance that work better on degree values, instead of 0 – 100 values range of Saturation and Value channel). To reduce the search space even more, I would like to use a hierarchical clustering algorithm.

Also, I would like to implement a caching system in File Master module that will save a number of query image (feature vector extracted from them) and their corresponding results and based on a threshold for the similarity function, return the results without sending the query to File Slaves.

1. **Conclusions**

In conclusion, I implemented a scalable system (easily extendable by adding more File Slaves to the system), fairly robust (“Single point of failure” occurs only if File Masters goes down) and heterogeneous (each module can be implemented in any language as long as it provides the necessary API for communication).

1. **Bibliography**

[1] “Searching Large Image Databases using Color Information”, Ioan Raicu, Department of Computer Science, University of Chicago. CMSC350: Artificial Intelligence – Winter Quarter 2004.

[2] “An image retrieval system based on the color complexity if images”, Yung-Kuan Chan, Department of Management Information Systems, Yi-Tung Liu, Rung-Ching Chen, Department of Information Management, Chaoyang University of Technology, Computing and Informatics, Vol. 24, 2005, 495–511.

[3] “State-of-the-Art in Content-Based Image and Video Retrieval”, [Remco C. Veltkamp](http://link.springer.com/search?facet-creator=%22Remco+C.+Veltkamp%22),  [Hans Burkhardt](http://link.springer.com/search?facet-creator=%22Hans+Burkhardt%22), [Hans-Peter Kriegel](http://link.springer.com/search?facet-creator=%22Hans-Peter+Kriegel%22). [Computational Imaging and Vision](http://link.springer.com/bookseries/5754) Volume 22 2001.