

**Project Report**

Project Spot Check:

Accuracy Enhancement of Software to Aid in Recognition of Snow Leopards

and Containerizing the Code

Final Report to Panthera

**1 June 2020**

**ECE 20.4**

**Seattle University’s Project Center**

**College of Science and Engineering**

**Seattle University**

1 June 2020

Rana Bayrakçismith

Panthera

8 West 40th Street,

18th Floor,

New York, NY 10018

**Subject:** Accuracy Enhancement of Current Working Software to Aid in Recognition of Snow Leopards and Containerizing the Code.

**Document:** Letter of Transmittal

Rana Bayrakçismith,

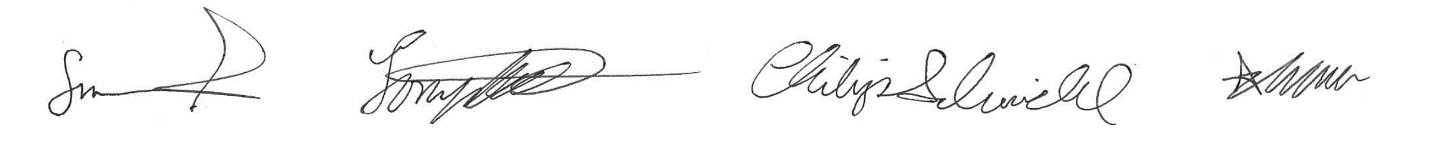
Attached to this letter is a final report to Panthera on the year’s work of modifying and containerizing the snow leopard identification program Recognition. Team ECE 20.4 has added functionality to improve the accuracy of the Python code provided in October 2019. The program now uses image editing techniques to create sharper images and a more accurate template generation method to automatically identify and mask snow leopards. A new user guide outlining these features is attached to the report.

The team has prepared Recognition for integration in the cloud with Databricks, to be used across different platforms. An individual will be able to run the program through a cloud service provider on a Linux, Mac, or Microsoft operating system. This can be done in Microsoft Azure by using steps in a separately attached user guide.

Thank you for this opportunity to help professionals at Panthera continue the ongoing work of conservation.

**Seattle University Design Team ECE 20.4:**

Sultan Alneif Anthony Caballero Philip Schmeichel Amudhan Sekar



CC: Dr. Agnieszka Miguel, Department Chair, Seattle University ECE Department

**Project Spot Check: Accuracy Enhancement of Software to Aid in**

**Recognition of Snow Leopards and Containerizing the Code**

Project Sponsor:

Panthera Corporation

Project Liaisons:

Rana Bayrakçismith, MES

Agnieszka Miguel, Ph.D.

Engineering Design Team ECE 20.4

Seattle University Project Engineers:

Sultan Alneif

Anthony Caballero

Philip Schmeichel

Amudhan Sekar

Faculty Project Advisor:

Paul Kostek

Electrical and Computer Engineering Senior Design Coordinators:

Shiny Abraham, Ph.D.

Henry Louie, Ph.D.

# Executive Summary

Panthera is focused on the research and conservation of wild cat species across the globe. Specifically, in this project, they are focused on snow leopards. Snow leopards currently span across 12 different countries in central to southern Asia. Panthera is tasked with determining the population of wild snow leopards. Panthera uses camera traps to capture a series of images of snow leopards passing by. To gain a more precise measure for an area’s population, Panthera biologists have to sort thousands of images by hand to match snow leopards.

ECE 20.4 furthered the development of a Python software program called Recognition.py, which, using image recognition technology, matches snow leopards in a given image set. The goal of Recognition.py is to automate this laborious manual matching process. ECE 20.4 used state of the art methods in image processing, machine learning, and cloud software to improve this software. The following improvements were made:

* Each image was filtered and labeled under the corresponding categories: taken at night, blurry, or clear. If the images were taken at night or were blurry, then certain image editing techniques were performed.
* Using the deep learning technique called Mask R-CNN, more accurate templates are generated at run time to be used by the software. This also allows for the identification of multiple snow leopards in a single image.
* The clustering method k-Means clustering was tested as a secondary identification of the snow leopards. k-Means showed promise in its results when clustering both the values of keypoint matches as well as the descriptors generated by the matching algorithm. The descriptors generated significantly more data, therefore the runtime greatly increased.
* The second clustering method tested was Markov clustering, as suggested by ECE 19.7. This method also showed promise in both the number of keypoint matches as well as descriptors that the matching algorithm generated. The results could be slightly difficult to read; however, the runtime was not affected.
* Using Azure Databricks, a Spark notebook is created to bring the program from a local machine to running in the cloud. Having the program running in the cloud allows for any number of virtual machines to work on the project at one time, matching image sets that a local machine wouldn’t be able to process. This also bypasses the need for operating system-dependent software packages.

Full documentation of both the local program as well as the version that is in the cloud is provided for future teams. The versions do not match perfectly, so the differences are noted. The documentation also includes the tests that were run and the results that were given.

ECE 20.4 recommends further testing of the clustering algorithms and a deeper look into what the matching algorithm outputs. If there is a way to extract only the most pertinent information to cluster, then this would be the best. ECE 20.4 also recommends moving the entire program to the cloud and not having a local machine version. This will allow for expansive testing on image sets of up to 3,000 images. The final recommendation is to develop a more precise value to include in the score matrix, as the current value isn’t as informative as it can be.

# Abstract

The Panthera organization is dedicated to the research and conservation of all wild cat species. To estimate a population of snow leopards in a study area, Panthera asked team ECE 20.4 to continue the development of their open-source Python program, Recognition. The program will help their biologists to sort and identify individual snow leopards in a given image set. Through the computer vision program OpenCV, Recognition uses the SIFT algorithm to identify key spots on the leopard’s body, giving them a series of numerical values. The team has used image processing and machine learning techniques to improve accuracy and optimize the Python program. Mask R-CNN is used to identify and create a template surrounding snow leopards in an image. Recognition uses two machine learning clustering algorithms, k-Means, and Markov clustering, to sort the data and determine the number of snow leopards in an image set. Microsoft Azure is used for remote execution of Recognition. For cloud computing, Azure Databricks is used to run a Spark notebook with image sets stored in an ADLS Gen 2 account. Recognition may use the Spark notebook which is not dependent on a specific operating system to run. The program is open-sourced on GitHub for further development for all operating systems.

# Acknowledgements

It is important to give credit to many peers, mentors, friends, and loved ones who have helped us during our work on this project for Panthera. ECE 20.4 would like to acknowledge Rana Bayrakçismith, Dr. Agnieszka Miguel, and Paul Kostek for their continuous support and guidance throughout the year. Giving us proper direction through their questions, concerns, and expertise helped us tremendously in achieving our goals. We would also like to acknowledge Dr. Louie and Dr. Abraham for their guidance. Whenever we had questions, they would give us a multitude of possible answers and allowed the team to pick the proper solution. ECE 19.7 gave us the foundation to push the project further. We would like to specifically acknowledge Devin DeWitt for his continuous presence in our project. He has left a big footprint on the project as a whole and always has had the technical solutions for our problems.

The team would also like to thank:

* Faculty in the ECE, CS, and Math Departments at Seattle University who have been open and eager to answer our questions about various research or technical areas of the project
* Rachael Brown and Jorge Vargas from the CSE’s Project Center for helping us organize our budget and presentation of materials for Projects Day
* Seattle University and public health officials for leadership in the face of the COVID-19 pandemic during our last months of working on the project
* Our families and close friends who have supported us in the long run of this project and our educational careers

**Table OF Contents**

Executive Summary ii

Abstract iii

Acknowledgements iv

List of Figures vi

List of Tables vi

I. Introduction 1

A. Background 1

B. Statement of the Problem 2

C. Purpose of the Project and Overview of the Project Report 4

II. Methods and Design Approach 5

A. Image Enhancement 5

B. Mask R-CNN Templating 7

C. Clustering Tests 8

D. Cloud Testing 10

III. Results 11

A. Image Enhancement 11

B. Mask R-CNN Templating 12

C. Clustering Tests 13

D. Cloud Testing in Databricks 16

IV. Conclusions 19

V. Recommendations 20

VI. References and Bibliography 21

Appendix A – Original Statement of Work 22

Appendix B – Student Resumes 25

# List of Figures

[Figure 1. Sample of Camera Trap Image of a Snow Leopard with the Lure Circled in Red 2](#_Toc41845610)

[Figure 2. Comparison Between the Regions of Interest in Two Images in a Dataset 3](#_Toc41845611)

[Figure 3. Example Score Matrix 3](#_Toc41845612)

[Figure 4. YUV Histogram Output on a Grayscale Night Image 6](#_Toc41845613)

[Figure 5. YUV Histogram Output on a Daytime Image 6](#_Toc41845614)

[Figure 6. Night Image with Blur Score 7](#_Toc41845615)

[Figure 7. Daytime, Not-Blurry Image with Blur Score 7](#_Toc41845616)

[Figure 8. Mask R-CNN Images with Masks of Two Snow Leopards 8](#_Toc41845617)

[Figure 9. A Sample of SIFT Keypoints Generated on an Image 9](#_Toc41845618)

[Figure 10. Equalized Grayscale Image after Editing 11](#_Toc41845619)

[Figure 11. Image with Blur Score Before Editing vs. Image After Editing 12](#_Toc41845620)

[Figure 12. Mask R-CNN Template and Keypoints 12](#_Toc41845621)

[Figure 13. Previous MATLAB Template and Keypoints 12](#_Toc41845622)

[Figure 14. A Sample of Clustering of Score Matrix Values (28 images, k=4) 14](#_Toc41845623)

[Figure 15. A Sample of Clustering of SIFT Descriptors (28 images, k=3) 14](#_Toc41845624)

[Figure 16. Markov Clustering Results with Score Matrix Values (10 images, 4 clusters) 15](#_Toc41845625)

[Figure 17. Clustering Results with SIFT Descriptors, on Image Set Tested in Figure 16 15](#_Toc41845626)

[Figure 18. Incorrect Clustering Results on Image Set with 3 Unique Snow Leopards 15](#_Toc41845627)

[Figure 19. Score Matrix Output Displayed in Databricks for 3-Image Set 17](#_Toc41845628)

[Figure 20. Score Matrix Accessed in Storage Container through the Azure Portal 18](#_Toc41845629)

# List of Tables

[Table 1. Summary of Clustering Results 16](bookmark://_Toc41845693#_Toc41845693)

# I. Introduction

Panthera sought a solution that improves the accuracy of their software program Recognition, which identifies snow leopards from images. Panthera wanted Recognition to be operational on multiple operating systems through the command line so that it could be used by their broader platform, PantheraIDS. Panthera requested detailed testing of Recognition, with larger image sets and on multiple operating systems. The team underwent a design process to identify the aspects of Recognition it would develop and how it would use these changes to advise further development and testing of the program.

## Background

Panthera, founded in 2006, is a non-profit organization devoted to the conservation of all 40 wild cat species and their ecosystems. True to its mission, Panthera is consolidating its efforts with local and international non-governmental organizations as well as scientific institutions to develop and implement conservation strategies around the globe. The range of Panthera’s efforts spans continents for all wild cat species. The design team ECE 20.4 was focused on the identification of individual snow leopards, using the Python program Recognition.

Snow leopards are native to Central and South Asia, with a habitat spreading across two million square kilometers. They are listed as “Vulnerable” on the International Union for Conservation of Nature Red List and are threatened by poaching and habitat destruction. Scientists studying the species estimate its population to be between 4,500 and 10,000 individuals. Snow leopards are naturally elusive, rarely seen by locals, and challenging to survey. A more accurate estimate would play a role in further studies and research to help in conservation endeavors. Panthera sought a solution by implementing software that identifies individual snow leopards from images in order to better estimate the population.

Panthera’s scientists use camera traps to capture images of the snow leopards in the surveyed area. Those camera traps are equipped with motion sensors and are supplemented by lures, as shown in Figure 1. When the camera’s motion sensor is triggered, the camera captures a series of images. Each image in the series becomes a point in the dataset of snow leopard images. After collecting the data, scientists have thousands of images that need to be classified. Biologists can manually classify the images by going through them as data points and comparing them to identify snow leopards by their unique spot pattern. This is inefficient and time-consuming. Recognition provides an alternative to that labor-intensive method of categorization.



Figure 1. Sample of Camera Trap Image of a Snow Leopard with the Lure Circled in Red

## Statement of the Problem

Recognition was inherited by last year’s Seattle University design team with Panthera, team ECE 19.7. As given, Recognition took an input dataset of snow leopard images taken by Panthera’s camera traps, as shown above in Figure 1. The program had two options for templating to define shapes surrounding snow leopards (called regions of interest) and their spot patterns, or rosettes. One option was to utilize the templates in the datasets generated by a MATLAB script from ECE 19.7 to define each image’s region of interest. The other option was to let the user manually draw a box around the region of interest in a given dataset. Upon specifying the method of highlighting regions of interest, Recognition compared these regions among the input images by matching keypoints via the scale-invariant feature transform (SIFT) algorithm.

Figure 2 shows an example of Recognition’s keypoint-matching with manually-drawn templates: the blue circles represent the keypoints within the user-defined region of interest, and the green lines are the connections between the matches in the two images. The keypoint matches were then written in a score matrix comma-separated-values (CSV) file showing the matching percentage of the keypoints between any two given images in the dataset, as demonstrated in Figure 3. The score matrix values were normalized or made into values between 0 and 1, so it is important to observe the comparison between images 1 and 2 (or 1 to B, in the figure). The image-pair with the highest number of keypoint matches in the score matrix was set to the maximum value of 1 or 100%.

A picture containing tree, grass, skating, photo

Description automatically generated

Figure 2. Comparison Between the Regions of Interest in Two Images in a Dataset



Figure 3. Example Score Matrix

Team ECE 20.4 was tasked with improving and testing the accuracy of Recognition with multiple image sets, primarily using the results of clustering algorithms on the score matrix. As given, there was also a problem with the robustness of Recognition. Recognition worked well with curated datasets; however, upon facing un-curated datasets, its identification accuracy significantly dropped. Recognition could not guess the correct number of individual snow leopards in an image set as much as 60% of the time, upon being given datasets of images that were not curated. The team considered how to treat images that were unlabeled, blurry, taken at night, contained multiple snow leopards, or contained only part of a snow leopard.

Team ECE 20.4 was tasked with solving this accuracy and image dataset robustness issue as well as making Recognition accessible to the biologists who will use it for verification. Recognition must be compatible with whatever operating system they use once they have access to it. The program was to be integrated into the software package PantheraIDS, for use with Amazon Web Services (AWS) by Panthera once completed by the team. Recognition’s availability to professionals required a method of containerizing the code.

## Purpose of the Project and Overview of the Project Report

This report details the work done by Seattle University’s engineering design team ECE 20.4 to improve the accuracy of Panthera’s software, Recognition, in identifying snow leopards. Beyond learning the function of Recognition from ECE 19.7’s guidance, the team focused on many areas for making Recognition more robust:

* Using image enhancement to make the SIFT algorithm produce a more even distribution of keypoints per image, yielding a more accurate score matrix
* Implementing automatic templating with a Masked Regional-Convolutional Neural Network (Mask R-CNN), to be used by a graduate student working on the project, and to be trained further
* Using statistical metrics to test the accuracy of using Recognition’s score matrix to guess the number of snow leopards in an image set
* Showing and comparing the results of tests with 2 clustering algorithms: k-Means and Markov
* Showing a proof-of-concept for running Recognition in the cloud, with a small image set on Azure Databricks’ Spark notebooks

The team gained and used knowledge of computer vision, keypoint generation with SIFT, machine learning with Mask R-CNN and clustering, and cloud-based platforms with Microsoft Azure. These areas of understanding aided in modifications to Recognition with the goal of testing the program’s accuracy in identifying individual snow leopards in an image set. A transition from Microsoft Azure to AWS should be simple. This allows for shorter runtimes on considerably larger data sets.

The team aims to use this report to explain the technical aspects understood to work with Recognition and the steps taken to add and test the functionality of Recognition during the project. Methods and Design Approach explains the technical areas of focus chosen by the team, and Results shows the outcomes and progress ECE 20.4 achieved. A list of recommendations to developers for continuing the improvement of Recognition is included. Further documentation of Recognition, both for the local machine and for the cloud, is attached to this report.

# II. Methods and Design Approach

The team was given a version of Recognition and areas for improvement of the program and implementation with Panthera IDS. The team chose to focus primarily on the modules that used keypoint generation with SIFT and improvement of “accuracy” in the program. Accuracy of Recognition is defined as the correct-guess rate of number of snow leopards in an image set by the clustering algorithm. Image enhancement improves the density of SIFT keypoints that are generated within the template of the snow leopard. Adding template generation with the Mask R-CNN gave a more focused region (containing the snow leopard) for SIFT keypoints that were compared between images. Different input methods of SIFT descriptors and Recognition’s score matrix were used for clustering. Finally, the goal of implementing Databricks for faster testing time of larger datasets was the last module ECE 20.4 worked to apply for the continuation of the project.

## Image Enhancement

The images that are in the datasets generated by Panthera aren’t always of the highest quality. There are times when the snow leopard was moving, or the image was taken at night. These two instances created an issue for the algorithm that generates keypoints to match. It was necessary to create a method to rank how blurry an image is as well as detect if the image is taken at night. If the image fell into either of these categories, it was subject to image enhancement. To detect if an image was taken at night, team ECE 20.4 used a luminance-bandwidth-chrominance (YUV) analysis. This analysis broke down the images into three different components, red, blue, and grayscale. If an image was taken at night, then the image would be in a black-and-white format and have no components of red and blue. Scanning each image using this YUV analysis would output an array of pixel values. If there were components of red and blue, then the range of these were seen in the output. However, if there was no color to the image, the image was taken at night; there would be no variance in these two factors. Filtering the array values with this analysis was a quick way to determine if an image was taken at night. An example of both an image that was taken along with its YUV analysis, as well as an image that was taken in the daytime, and its corresponding YUV analysis can be seen below in Figures 4 and 5.

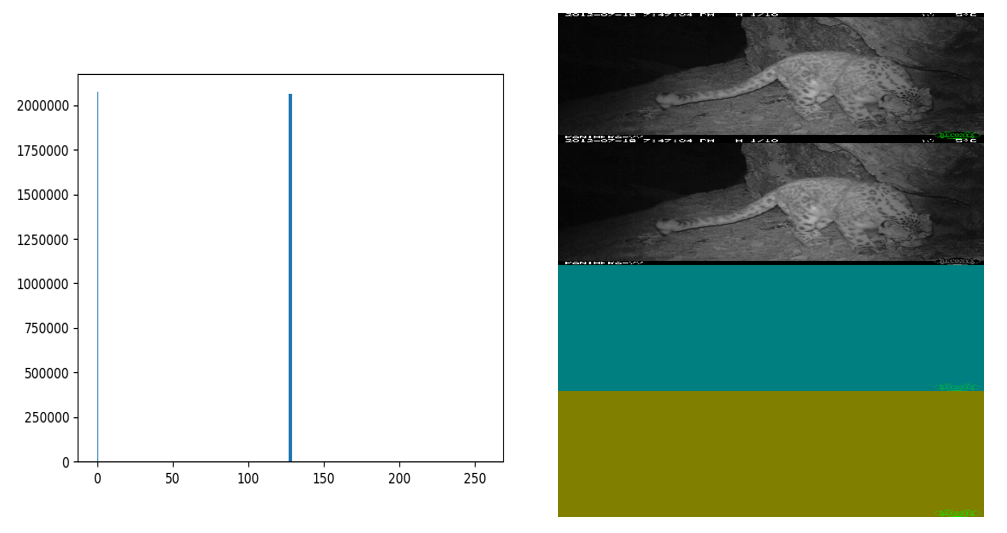


Figure 4. YUV Histogram Output on a Grayscale Night Image

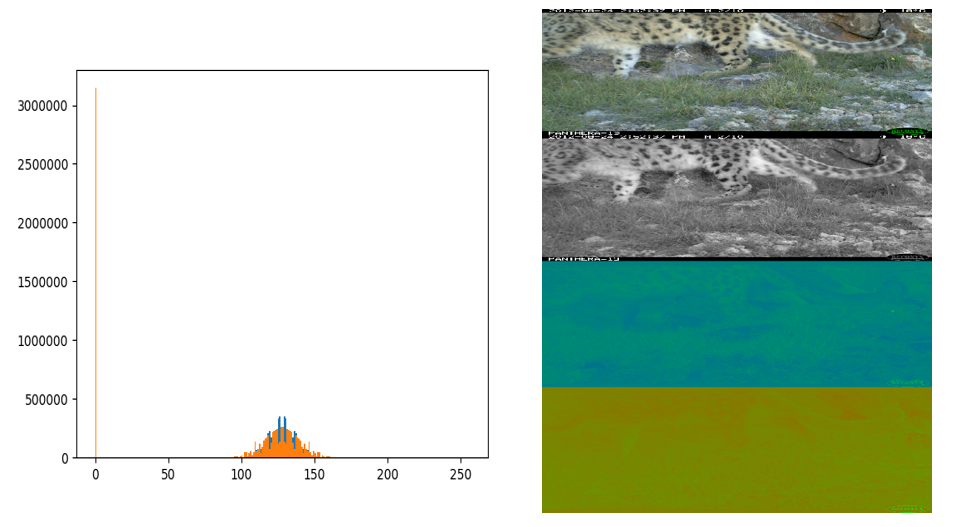


Figure 5. YUV Histogram Output on a Daytime Image

The next method that needed to be added was to determine if an image was blurry. This was accomplished by creating Laplace transformation. The image would go through this conversion and be compared against the original image. The variance between the two images was calculated. This calculated variance that was labeled the “blur score.” If the blur score passed a specific threshold, then the image was determined to be blurry and further editing would have to take place as well. Figures 6 and 7 illustrate when an image was determined to be blurry and when one wasn’t. The correct label can be seen in the top left corner of each image.



Figure 6. Night Image with Blur Score



Figure 7. Daytime, Not-Blurry Image with Blur Score

## Mask R-CNN Templating

Mask R-CNN is a deep learning technique, applied as a neural net for defining shapes from image or video. The shape can be labeled according to a “class” that is trained into the neural network. Team ECE 20.4 developed a new snow leopard class to automatically generate well-defined masks that outlined the correct shape of the snow leopard. The class is written in Python, containing the functions that correspond to the weight masks with the proper term, in this case, “Snow Leopard.” The class also outputs an accuracy metric to the right of the name that gives the confidence level of how confident it believes that this is a snow leopard. Figure 4 shows the Mask R-CNN successfully identifying two snow leopards in one image, drawing a mask around their bodies.

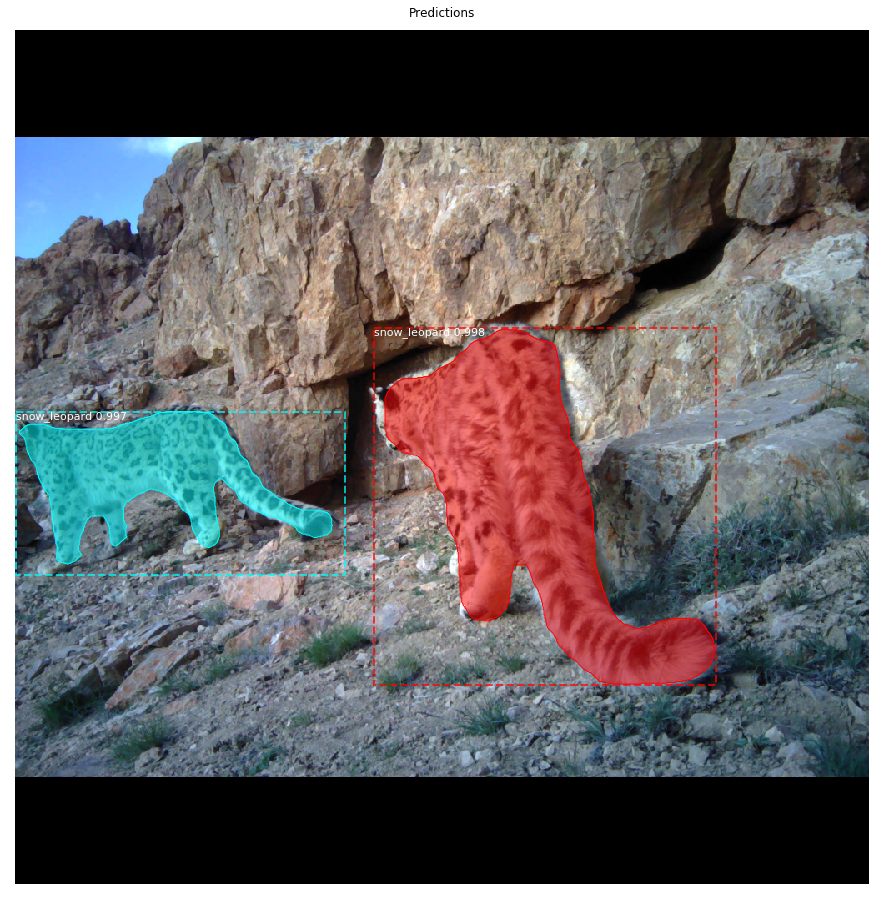


Figure 8. Mask R-CNN Images with Masks of Two Snow Leopards

The class had to be trained with 100 pictures of the snow leopards. Outputs from the cameras were outlined manually and then fed to new weights. The weights were saved and can be trained to make the snow leopard class better. The COCO (Common Objects in Context) dataset contains thousands of images of various objects and animals. Once the network was trained on the COCO dataset, it had an easier time recognizing other objects. This procedure of using the pre-trained neural network and adding our own network of weights is called Transfer Learning. When the Mask R-CNN is called to draw masks over new input images, these weights are loaded and used as the “learning” for drawing masks during runtime.

## Clustering Tests

Per team ECE 19.7’s recommendation, team ECE 20.4 explored and tested two clustering methods: k-Means and Markov. Team ECE 19.7 experimented with k-Means and Hierarchical clustering using a separate MATLAB program that ran these clustering algorithms on the score matrix values outputted by Recognition. Team ECE 20.4 experimented with k-Means and Markov in Recognition as functions that can be called from within the program once the matching process is finished. The team also experimented with additional clustering inputs, namely the clustering of SIFT descriptors, to compare the results of clustering tests and determine the most accurate results. A SIFT descriptor is a 128-dimensional vector describing the keypoint it represents, which is generated by the SIFT algorithm on each image in the dataset, as shown in Figure 9.

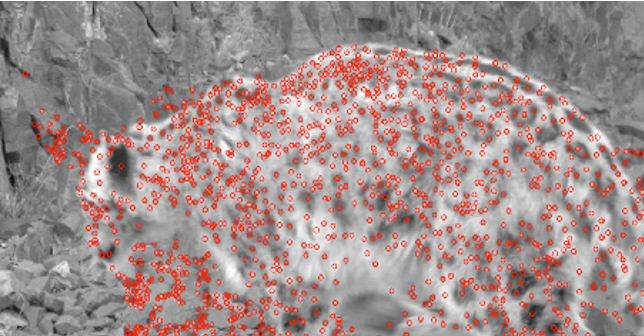


Figure 9. A Sample of SIFT Keypoints Generated on an Image

Team ECE 20.4 used the *sklearn* libraryto perform k-Means clustering and determine the unique number of clusters in a dataset. This uses the Elbow Method, which runs k-Means clustering on a given dataset using various numbers of k. It determines the optimal k by analyzing the change in distortion.

The second clustering algorithm that team ECE 20.4 tested was Markov clustering. This algorithm was used in Panthera’s previous snow leopard identification program called HotSpotter. It was team ECE 19.7’s suggestion to also use this algorithm as there was previous success with it in HotSpotter. This method works well with a large unknown set of data which is given to the program via the SIFT algorithm. The descriptors could be extremely large in number and difficult to handle, so Markov was deemed a good choice for this input.

Markov clustering uses a random walks theory. This theory explains that if one node takes a walk to surrounding nodes, it is more probable to stay within the cluster than it is to find a node in a new cluster. The probability is generated for each node, and Markov chains, or connections are created. The probabilities go through a series of smoothing, which is done by multiplication of each value with an inflation point. The purpose of this is to make larger probabilities larger and small probabilities smaller. Once the smoothing is finished, then a final matrix is complete.

The output of the Markov clustering algorithm team ECE 20.4 received was in a matrix form, and the number of clusters had to be counted manually. An example of the finished product of a dataset of ten images is in Section IIIC of this report. There is a simple Markov cluster function that is integrated into Recognition.py. This function is called twice, the first being the SIFT descriptor output, which is called after the matching has occurred, and the second being after the score matrix is generated. This function will output the matrix showing the clusters that are generated, as well as the shape of the matrix, where the number of clusters will be displayed.

## Cloud Testing

Team ECE 20.4 was recommended to improve the portability and containerization of Recognition for testing on larger image sets. This was needed because of the large runtime for a local machine when the set had more than 25 images. To accomplish a faster runtime, team ECE 20.4 sought an application that could use cloud computing rather than local machines.

The main approach was to use Microsoft Azure, the cloud computing service available to Seattle University students, to use virtual machines. The service Databricks was chosen to port Recognition into a Spark notebook. The Spark notebook in Databricks could then be used in AWS as well by Panthera’s team. The given online containerization of the Spark notebook also would make Recognition independent of operating system when run in the cloud.

ECE 20.4’s choice of Azure Databricks enabled multiple “worker” machines to be called to perform computations in the form of a Databricks “cluster.” Multiple workers would be used for matching keypoints and generating the score matrix. The score matrix would then be output to a cloud storage container within Azure, where it could then be downloaded as a CSV file. Attaching this storage container and running the Spark notebook for usage of Recognition in Databricks required these steps in Azure:

* Creation of a Recognition working directory in an Azure storage container,
* Creation of an enterprise application for accessing the storage container through Databricks, and
* Creation and usage of a Azure Databricks Service for running the notebook.

These steps are outlined in the user guide to the Azure Databricks version of Recognition, attached to this report.

# III. Results

Team ECE 20.4 documented the behaviors of functions added to Recognition after being tested by team members in May 2020. These results were documented for testing by future development teams of Recognition. Functionality of image enhancement and Mask R-CNN templating is included in this section. Accuracy of the newly-implemented clustering functions and timing results of the Azure Databricks notebook version of Recognition are also listed in these results.

## Image Enhancement

After Recognition completes the YUV analysis and blur score, the histogram equalization and edge sharpening occurs. The resulting image allowed the SIFT algorithm to be able to generate more keypoints for matching. Testing was done to compare the quantity of keypoints generated, by SIFT, and after an image was enhanced, it was able to generate 3%-6% more keypoints. This led to a greater number of keypoint matches between images than on the previous version of Recognition. Figure 10 shows the equalization that occurred to an image after it was edited. After the template was placed, the lighter area of the snow leopard aided the SIFT algorithm’s performance.



Figure 10. Equalized Grayscale Image after Editing

There was a more equal distribution of light and dark in the image because of the equalization that occurred. The image was enhanced and sent to replace the image that was first chosen and analyzed in Recognition.py. The original image was saved in a folder called “edited\_photos” that is loaded by Recognition through the helper script, easy\_run.py. This occurred so the original image was available and not lost. The original image would replace the edited version after the completion of the programming, which ensured that one image doesn’t continue to become over processed. An example of the editing that occurs on the image that was labeled blurry can be seen below in Figure 11.



Figure 11. Image with Blur Score Before Editing vs. Image After Editing

## Mask R-CNN Templating

The team confirmed that the Mask R-CNN-generated templates awere properly implemented in Recognition by comparing versions of the keypoint match image output from the *write\_matches* function. In Recognition, the masks are also saved as BMPs in their own folder of the working directory, as the Manual-ROI masks were previously. An example comparing the keypoint images drawn by *write\_matches* is shown below.



Figure 12. Mask R-CNN Template and Keypoints



Figure 13. Previous MATLAB Template and Keypoints

The Snow Leopard Class was implemented within the *mrcnn\_templates* function of Recognition. This would be an alternative option to the old *add\_templates*, which was used to input pre-generated or manually-drawn masks. The output of the Mask R-CNN was an array or arrays of the masks, in an OpenCV object of 1s and 0s. This array was then updated into the list of Recognition objects, a global array, which was then used as an input in *write\_matches*. Recognition must also deal with multiple cats in the image because Mask R-CNN was used to solve this problem. However, Recognition only makes one mask per image and therefore does not know what to do with the other masks. ECE 20.4 has started on this piece by creating an empty mask any time Mask R-CNN generates more than one mask. In theory there should be multiple instances of fully generated masks in the folder, but, ECE 20.4 did not complete this step.

Some masks include the bait which could increase error. Since the model was only made with 100-line drawn templates, the process of adding snow leopard pictures to Recognition can be used to better the snow leopard model. This would help our Mask R-CNN implementation get a more precise idea of a snow leopard and would yield higher accuracy. This can be done through an automated system in which as soon as the picture is take the template is drawn using an online annotator. This annotator should then upload the annotated pictures to the model and then be retrained. It is good that now multiple cats can be recognized; however, Recognition only can take an input of one mask. Therefore, either a separate algorithm, that deals with comparing cats in the same image, needs to be made, or, Recognition could ask the biologist for how they would want the algorithm to be used.

## Clustering Tests

Team ECE 20.4 tested k-Means clustering of normalized score matrix values with eight datasets. The clustering method determined the correct number of clusters in five datasets out of eight tested. Team ECE 20.4 also tested k-Means clustering of SIFT descriptors with eight datasets and the correct number of clusters was correctly predicted in five datasets out of eight. Samples of the results of clustering of score matrix values and SIFT descriptors are represented below using the elbow plot in Figures 14 and 15, respectively.

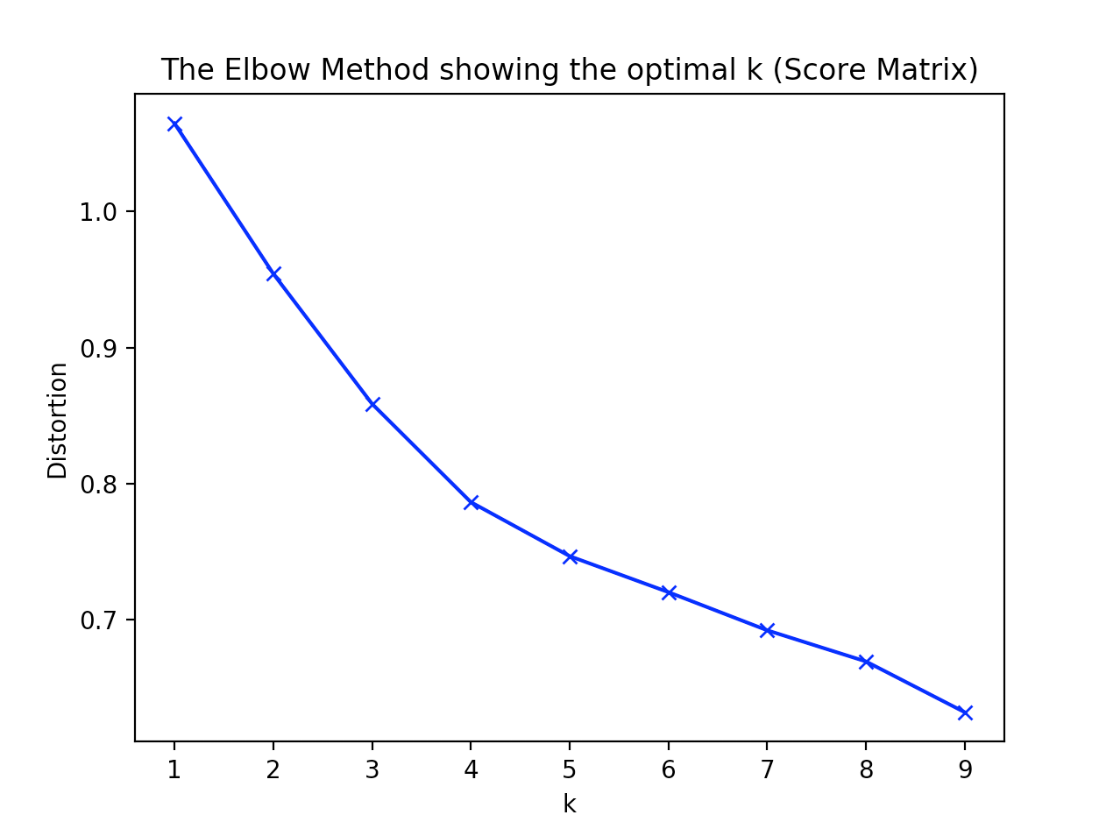


Figure 14. A Sample of Clustering of Score Matrix Values (28 images, k=4)

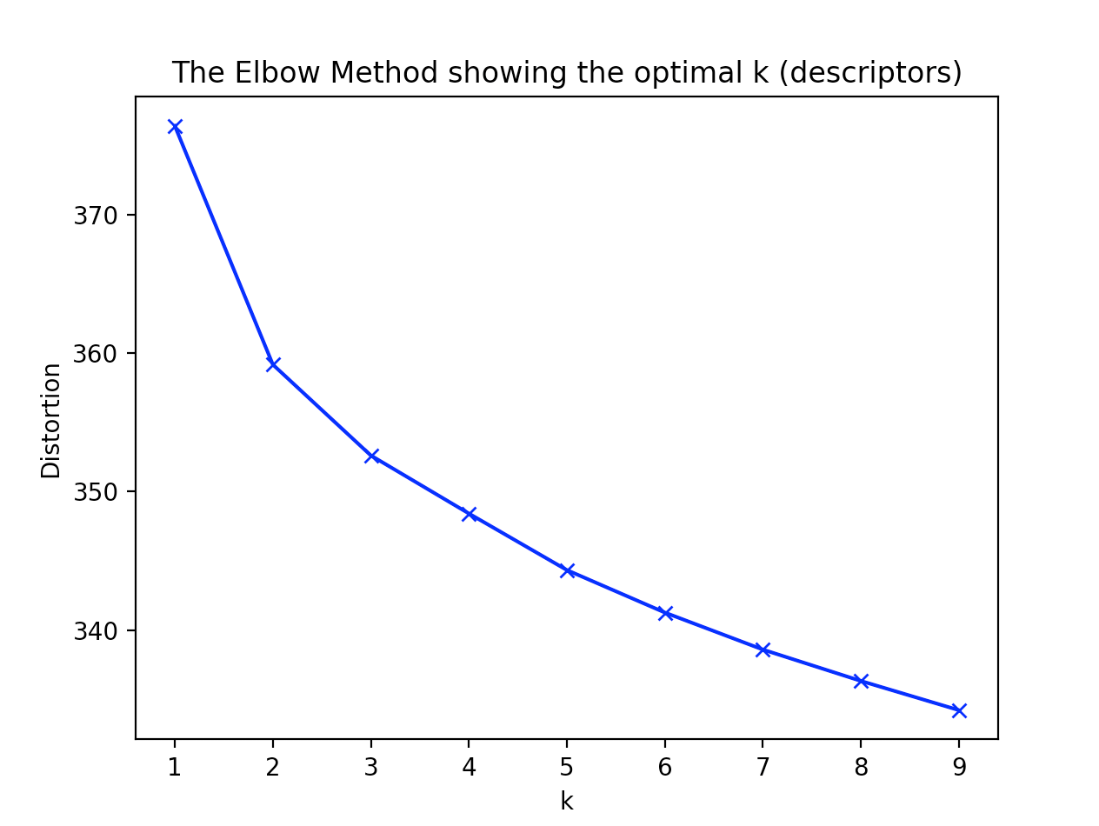


Figure 15. A Sample of Clustering of SIFT Descriptors (28 images, k=3)

Team ECE 20.4 also implemented the Markov clustering algorithm on different image sets. Below in Figure 16 is the result of a 10-image set that clustered the score matrix values.

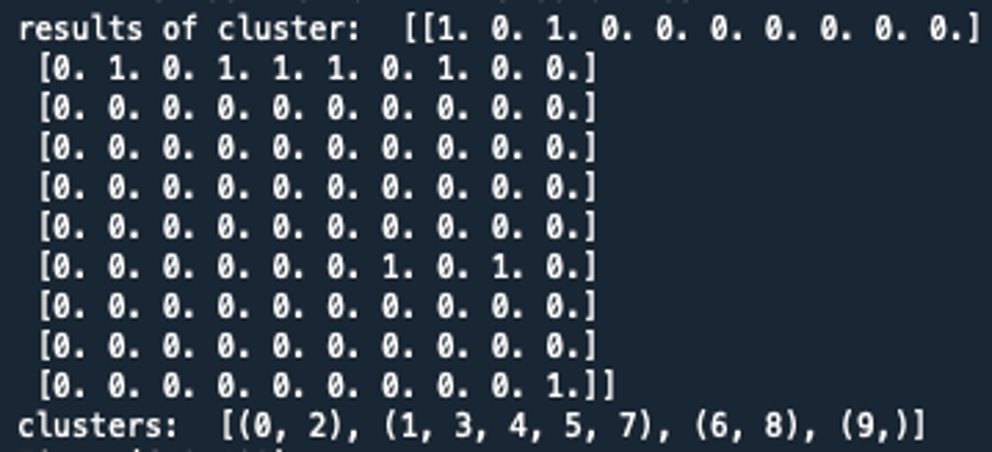


Figure 16. Markov Clustering Results with Score Matrix Values (10 images, 4 clusters)

The last line of the output labelled the different clusters. There were 4 different sets of clusters, which related to 4 different snow leopards that were identified in the image set. The matrix in Figure 16 is the transition matrix after all of the smoothing was completed.

Markov was able to correctly identify four different snow leopards with the given set of data. The clustering on the score matrix values was consistent with the results. Of the 5 tests that were run, it was able to identify correctly the correct number of cats each time. The clustering values on the SIFT descriptors was much less consistent. Only 4 tests were run on the SIFT descriptors, and it was only able to correctly identify the number of snow leopards in one of the image sets. Below in Figure 17 is the output of the clustering results on the same image set that was tested in Figure 16.



Figure 17. Clustering Results with SIFT Descriptors, on Image Set Tested in Figure 16

The green highlighted section is where the number of clusters is displayed. Here it is shown that there are 4 different clusters that were found in the 10-image data set. Figure 18 shows the wrong result of a different test that was run.



Figure 18. Incorrect Clustering Results on Image Set with 3 Unique Snow Leopards

Here, the algorithm determined that there were 8 different clusters to be formed, when in fact there were only 3 different snow leopards in the image set. Markov clustering is still inconsistent with the results.

A way to make the clustering of SIFT descriptors more accurate would be to use feature extraction of the SIFT descriptors using a “bag of words” technique. This technique would involve a user creating a matrix of commonly-used SIFT descriptors that are the most important values from a set of descriptors. If a new descriptor comes in, the values within the descriptor are compared to the bag of words that was generated. If the value within the descriptor is found in the bag of words, it is kept. However, if it is not found in the bag of words, then it is not saved to be clustered. This would leave only the descriptor values that hold the most weight to enter the clustering algorithm.

Table 1. Summary of Clustering Results

|  |  |  |
| --- | --- | --- |
| Clustering Method | Clustering Input | Correct Prediction of Clusters |
| k-Means | Normalized Score Matrix | 5 out of 8 datasets |
| Markov | Score Matrix | 5 out of 5 datasets |
| k-Means | SIFT Descriptors | 5 out of 8 datasets |
| Markov | SIFT Descriptors | 1 out of 4 datasets |

## Cloud Testing in Databricks

Team ECE 20.4 created a Spark notebook version of Recognition that was based on the Recognition version from team ECE 19.7’s GitHub. This notebook version used a multithreading library and assumed the working directory (created in a cloud storage container) contained an image folder, a template folder, and a *config.json* file. A user could start a Databricks cluster and copy the text of the Spark notebook into commands, adjust the mount information, and get results in their cloud storage container of choice. The Spark notebook is included in team ECE 20.4’s GitHub attached to this final report.

Mounting was accomplished by using an enterprise application in Microsoft Azure to get permission to the Azure Data Lake Storage (ADLS) Gen 2 storage accounts. This allowed access to a 3-image set (“quick\_set”) and a 10-image set (“ten\_images”) to prove that the Spark notebook version could generate a score matrix output. For final tests, the Databricks cluster used this configuration in Figure 15. There were 4 worker machines of the relatively cheap option of Standard\_DS3\_v2 machines.

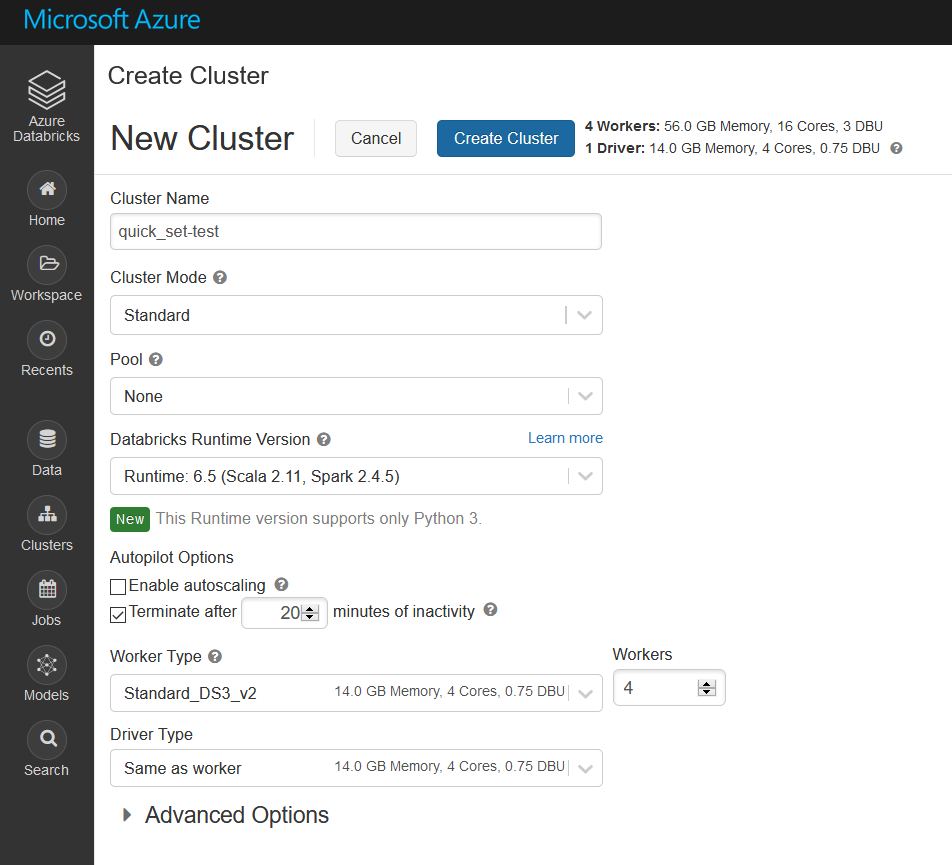


Figure 18. Databricks Cluster Configuration, with Spark 2.4.5 and Databricks Runtime 6.5

It took the Databricks notebook as long or longer than a local run of Recognition to generate a score matrix for both the image datasets “quick\_set” (2-3 minutes) and “ten\_images” (40-50 minutes). Due to the very long runtime of these sets, team ECE 20.4 believed that not all the worker machines are being used by the cluster to do the SIFT keypoint detection. This would be because of the multithreading version of Recognition used in this notebook.

Results of the keypoint matching between images were shown in the score matrix, outputted in the Spark notebook. A CSV file was then written in a chosen folder in the mounted cloud storage container, which could be downloaded and used with Recognition’s clustering functions. Figures 19 and 20 show the score matrix output in the notebook in Databricks and in the Azure cloud storage container.

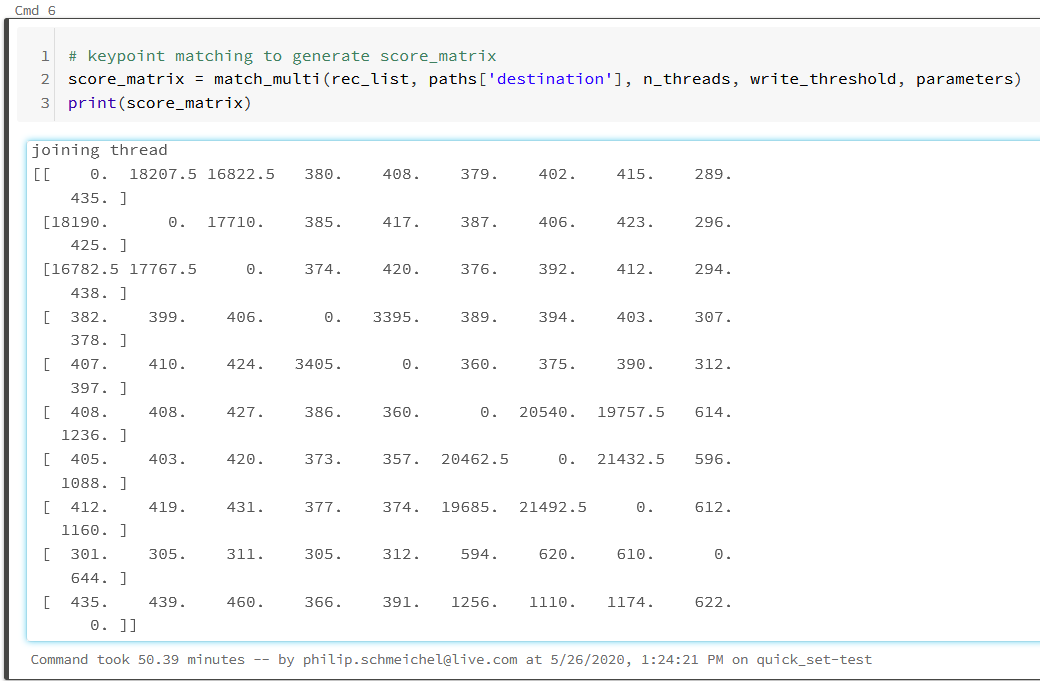


Figure 19. Score Matrix Output Displayed in Databricks for 3-Image Set



Figure 20. Score Matrix Accessed in Storage Container through the Azure Portal

Team ECE 20.4 concluded that multiprocessing may be implemented to utilize the Databricks cluster workers more efficiently. Ross Pitman gave team ECE 19.7 a version of Recognition in early 2019 that used multiprocessing. Team ECE 20.4 believes adapting this version to a Spark notebook should be the first step taken by a future development teams to more quickly get results from Recognition in the cloud. After improving this runtime, the image enhancement and Mask R-CNN templating must be added to fully test Recognition’s performance with large datasets.

# IV. Conclusions

The team met the great challenge of becoming familiar with Recognition and the interfaces of Python, OpenCV, and Azure Databricks. The main focuses of the project were to contribute to explaining accuracy, improving accuracy, and finding methods to let future teams get results in less turnover time. This resulted in the improvements in keypoint selection through the image enhancement and Mask R-CNN templates. Clustering and cloud testing were more difficult than anticipated to learn about and subsequently explain.

Team ECE 20.4’s finished version of Recognition is compiled and containerized and shows all the features that were developed during the course of the project. As part of the deliverable, a description of Recognition’s current components and features is attached to this report. The features added include:

* an automatic image enhancer for generating keypoints
* a template generator Mask R-CNN module for making masks that prioritize the snow leopard over the background
* functions for guessing the number of cats with Markov clustering using score matrix values
* a containerized Spark notebook version of Recognition for testing with larger datasets using Databricks

The next steps for Databricks include implementing the new versions of Recognition including the image enhancements and Mask R-CNN. The team has compiled a version of Recognition that uses pre-generated templates on a chosen image set in cloud storage. This is given as a Databricks Spark notebook as part of the deliverable for a containerized version of Recognition.

When handing this project off to later development teams, team ECE 20.4 aimed to outline the current function of Recognition, what progress has been made in these areas, and what methods the team recommends for further improvements.

# V. Recommendations

When considering the further development of Recognition and the project goal of improving testing of snow leopard image sets, team ECE 20.4 compiled these recommendations for future teams:

* Providing options to override Mask R-CNN’s decisions if it does not detect snow leopards in an image
* Improving Mask R-CNN functionality by automatically adding new images to training
* Implementing the Mask R-CNN version of Recognition into Databricks, as well as the image enhancement and clustering functions, allowing for testing on image sets of up to 3,000 images
* Implementing a “pipeline” mechanism for running tests with Recognition on the cloud automatically through Databricks, either through Azure or AWS
* Further testing with Markov and k-Means clustering algorithms to compare Recognition with HotSpotter
* Further investigating clustering of SIFT descriptors to extract only the most pertinent information to cluster, possibly through “bag of words” (bag of features)
* Developing a more precise value to include in the score matrix
* Considering options for creating a full neural network design to solve the problem of matching individual cats

# VI. References and Bibliography

Abdulla, Waleed. “Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow.” Github, 2017, github.com/matterport/Mask\_RCNN.

Almutawakel, Ali. “OpenCV: Filters & Arithmetic Operations.” Medium, Medium, 10 Apr. 2017, medium.com/@almutawakel.ali/opencv-filters-arithmetic-operations-2f4ff236d6aa.

“Azure Data Lake Storage Gen2.” Microsoft Docs, Azure Databricks Documentation, 28 Feb. 2020, docs.databricks.com/data/data-sources/azure/azure-datalake-gen2.html#mount-azure-data-lake-storage-gen2-filesystem.

Dan Mašek. “Rgb to Yuv Conversion and Accessing Y, U and V Channels.” Stack Overflow, 1 Mar. 1967, stackoverflow.com/questions/43983265/rgb-to-yuv-conversion-and-accessing-y-u-and-v-channels.

ECE 19.7 (DeWitt, Devin, Mai Dang, Jack Gularte, and Long Pham). “Project Spot Check.” Project Spot Check, n.d.

“Histogram Equalization.” OpenCV, Doxygen, docs.opencv.org/3.4/d4/d1b/tutorial\_histogram\_equalization.html.

Macropol, Kathy. Clustering on Graphs: The Markov Clustering Algorithm. University of California Santa Barbara- Computer Science Department, sites.cs.ucsb.edu/~xyan/classes/CS595D-2009winter/MCL\_Presentation2.pdf.

“The Mcl Manual.” Micans, 8 Mar. 2012, micans.org/mcl/man/mcl.html.

“MCL- a Cluster Algorithm for Graphs.” Micans, micans.org/mcl/.

Mishra, Anurag Kumar. “Demystifying Markov Clustering.” Medium, Analytics Vidhya, 8 Sept. 2019, medium.com/analytics-vidhya/demystifying-markov-clustering-aeb6cdabbfc7.

Sangam, Al. “Mask-RCNN-bottle-training.” GitHub, 4 Feb. 2019, github.com/AISangam/Mask-RCNN-bottle-training.

“What is Azure Databricks?” Microsoft Docs, Azure Databricks Documentation, 8 May 2019, docs.microsoft.com/en-us/azure/azure-databricks/what-is-azure-databricks.

Weber, Ben. “3 Methods for Parallelization in Spark.” Towards Data Science, Medium, 21 Jan 2019, towardsdatascience.com/3-methods-for-parallelization-in-spark-6a1a4333b473

# Appendix A – Original Statement of Work

ECE 20.4 Project Scope

Sponsor: Panthera

Title: Project Spot Check

Improve open source software to assist in individual recognition of spotted cats from camera trap photographs. Our objective is to produce software that will greatly speed up the process of identifying individual spotted cats in camera trap photo data sets and increase accuracy. The software, while open source, will be a collaborative effort between the Parties and will be credited as open source software created by Panthera and Seattle University.

Students will be modifying existing open source code software “Recognition” which was developed to recognize individuals in sets of images of spotted cats. Recognition, in turn, was developed in late 2018 by Panthera’s Ross Pitman by dramatically simplifying the software “HotSpotter” (<http://www.cs.rpi.edu/hotspotter/>). Recognition is a component of an integrated data management and analysis software system also created by Ross Pitman, “PantheraIDS”. Cat species included in this project will include both snow leopards and African leopards; previous Senior Design teams only worked with snow leopard photos.

In 2016-17, ECE 17.7 modified HotSpotter for snow leopards by adding automatic definition of regions of interest, querying and recording the results, clustering the measures of similarity between images to identify same cats in different pictures, and upgrading the GUI.

In 2017-18, ECE 18.7 modified HotSpotter by increasing cat identification accuracy via clustering refinements and adding user input regarding cat orientation and number (from 61% to 70%; 79% with cat orientation strategy which is not yet implemented), speeding up runtime by implementing multithreading, adding sorting of photos identified in clusters into individual folders, and additional GUI upgrades.

In 2018-19, ECE 19.7 switched from HotSpotter to working with Recognition. ECE 19.7 created Recognition class, a simple interface where all data for an image can be stored in a single object. They added a manual templating feature to allow the computer vision and image comparison process to utilize the entirety of the snow leopard’s body (versus limited regions of interest as in HotSpotter; templates are currently created independently in MATLAB). ECE 19.7 modified Recognition to permit multithreading, allowing the efficient creation of a score matrix for use with clustering algorithms, and implemented k-means clustering. They also created detailed developer and user documentation to allow future design teams to quickly familiarize themselves with Recognition and begin further development.

The goals of this project are:

1. Develop an understanding of the purpose, structure, and portability of the above source code using spotted cat photographs from camera traps.

2. Solve existing limitations of the software via improvements listed in objectives below.

3. Work with Panthera to containerize and integrate Recognition into existing PantheraIDS software platform.

4. Develop a usable system by June 2020 that is operational in Linux (within a containerized environment).

5. Develop detailed documentation on how to both use and modify the software, including User’s Guide, Developer’s Guide, and future development recommendations.

Project objectives (in order of importance):

1. Perform detailed testing of the complete software package using a wide range of different sets of spotted cat pictures and large data sets. Current test results are dependent on carefully curated, high-quality photos of the same orientation.

2. Containerize the source code so that the software can be easily deployed. Containerization should be via Docker (www.docker.com) and implemented within a Linux environment. A fully reproducible Dockerfile should be created and provided as part of the Developer’s Guide.

3. Ensure all core tasks and features are implementable using Python and executed from the command line (a Graphic User Interface (GUI) is less important for integration with PantheraIDS).

4. Implement automated template generation at runtime using Python, perhaps by adapting a pre-trained Mask R-CNN.

5. Improve cat identification accuracy to 95% (currently averages 40-65%) via some or all of the following:

1. Implement edge sharpening and contrast enhancement, such as adaptive histogram equalization, to improve quality of lowlight and blurry images which are too difficult for SIFT to effectively detect keypoints.
2. Improve clustering methodology: integrate Markhov clustering algorithm into Recognition.py as this has proven to be the most effective clustering methodology so far, and conduct further testing of k-means and ward linkages – both of which were found to be promising but with many limitations in 2018-2019. Continue to investigate alternative clustering methods as a means of improving accuracy.
3. Explore statistical methods of analysis, such as standard deviation, that can be used to improve the normalization process of the score matrix.
4. Investigate other metrics to create the score matrix. Currently each cell in the score matrix holds the number of keypoint matches between two images.
5. Implement Mask R-CNN to determine orientation of cat(s) and the number of cats in each photograph.
   1. Cat orientation: Separation of raw photo data by cat orientation increases accuracy and speed (left flanks will only be compared to left flanks, etc.).
   2. Multiple cats: Currently, multiple cats show up as Cat 1, Image 1 and Cat 2, Image 1. Then the user can sort by name to see if multiple cats were found in an image.
6. Need to test many different types of images to see what needs improvement – so many more cases need to be tested. Current testing has been done only with carefully curated data sets and the program must be able to work on a raw data set.
7. Implementation of a flexible spatial verification transform, or possibly a piecewise rigid transform. This allows the cat to transform more from one image to the next and still be recognized as the same cat. Since cats are flexible, fluffy creatures, this may be a more effective way of verifying the cat from one picture to the next. This is a lower priority task.

6. Test Recognition with datasets of 2,000–3,000 images. To date, it has only been tested with up to 100 images.

7. Implement the option to pause processes via the GUI in case user need to turn off/unplug computer but has already started the long process on a big data set.

# Appendix B – Student Resumes

**Sultan Alneif**

**Phone Number:** (206) 408-0055 **Email:** [alneifs@seattleu.edu](mailto:alneifs@seattleu.edu)

**Overview**

*A graduating Electrical Engineering with a Computer Engineering Specialization student with experience with multiple software and hardware projects and a passion for new learning opportunities*

**Education**

**Seattle University, College of Science and Engineering** Seattle, WA

*B.S in Electrical Engineering with a Computer Engineering Specialization;* January 2018 –June 2020

*Computer Science Minor*

*Engineering coursework: Electrical Circuits, Computer Tools, Semiconductor Devices, Digital Operations, Programmable Devices, Microprocessor Design, Signals and Systems, Junior Year labs, Data Structures, Foundations of Computer Science, Computing Systems, Embedded Systems, Robotic Manipulators, Machine Learning I, Object-Oriented Development, and Fundamentals of Databases*

**University of Washington** Seattle, WA

*Pre-Engineering* September 2015 –December 2017

**Projects**

**Panthera Spot Check Project,** Capstone Engineering Design Project, Sep 2019-June 2020

*Worked within a team to improve the accuracy and functionality of a python software, Recognition, that identifies and estimates the number of snow leopards in image datasets using Computer Vision, Microsoft Azure Databricks, and machine-learning techniques, including clustering and Mask R-CNN, for a non-profit devoted to the conservation of wild cats*

**Senior Year Projects,** Sep 2019-June 2020

* Fundamentals of Databases: *designed, implemented, and tested a business relational database using MySQL*
* Computing Systems: *implemented a simple client-server Network File System over a simulated disk in C++*
* Robotic Manipulators: *developed a python program for an AL5A robotic arm to sort objects by color using an external camera and OpenCV*

**Junior Year Projects,** Sep 2018-June 2019

* Microprocessor Design: *implemented a 32-bit single-cycle MIPS processor in VHDL*
* Electronics & Signals and Systems labs:
  + Electrocardiograph: *assembled a protype of an Electrocardiograph on a PCB connected to a Raspberry Pi and developed a simple Android app to show live ECG/EKG signals using Android Studio*
  + Robot design: *designed and built a chassis robot model with a Raspberry Pi Interface to drive the robot and equipped it with an IR sensor for obstacle detection*

**Skills**

* Computing Languages: *C++, C, C#, Python, MATLAB, and MySQL*
* Computer Software: *Multisim, AutoCad, ModelSim, Logisim, Visual Studio, and Android Studio*
* Hardware Languages and Software: *MIPS Assembly Language, VHDL, Questa Sim and Quartus*
* Equipment: AC/*DC Power Supplies, Signal Generators, Multimeter, and Oscilloscopes*
* Bilingual; fluent in English and Arabic
* Teamwork and project management
* Resourcefulness and eagerness for new learning opportunities

Anthony Caballero

206.458.4476 | [Tony.Caballero2@gmail.com](mailto:Tony.Caballero2@gmail.com)

**Objective:** Upon completion of a Bachelor of Science in Electrical Engineering degree at Seattle University, my goal is to pursue a career in industries focused on user interface optimization and performance. I have a strong interest in bridging the business and technology world where I can map the steady advancement of enabling technologies to ever changing consumer needs.

**Education:**

2016 – Current **Seattle University**

*Bachelor in Electrical Engineering with a Computer Engineering focus and minor in Innovation and Entrepreneurship* | *President’s List* | *Expected Graduation June 2020*

2013 – 2016 **Santa Clara University**

*Pursued Bachelor in Computer Engineering* | *Sigma Pi Fraternity* | *Men’s Crew*

**Projects:**

**Panthera Organization Project Spot Check:** *September 2019 – Current*

· Improved a Python program for Snow Leopard identification and matching

· Uses Microsoft Azure Databricks, OpenCV, and machine learning techniques: clustering and Mask R-CNN

**Data Acquisition with Enviro pHat Sensor:** *January 2020 – March 2020*

· Use Python, Raspberry Pi 3, and Enviro pHat Sensor to upload data on various IoT services

· Platforms used include AWS and Splunk; created AR dashboard for real time readouts

**Portable EKG with Android Studio Development:** *September 2018 – June 2019*

· Created and designed an embedded system PCB including various filters, defibrillator protection, Semiconductors and Signals and Systems methods

· Raspberry pi and MCP3208 ADC for real time reading and communication

**Experience:**

**Ruth’s Chris Steak House** *Server and Key Manager 2015-Current*

· New employee trainer | Team management experience | Required multitasking and rapid problem solving | Excellent communication skills and ability to operate well within in a team

**Buca Di Beppo***Server 2012-217*

· Adapted to new environment | Cooking to meet corporate standards | Exceeded customer expectation

**Technical Skills:**

· **Software:** Visual Studio, Spark, Solidworks, AutoCAD, Microsoft Excel certified, Microsoft Visio

· **Programming Languages:** C, C++, C#, Python, CSS, JavaScript, PHP, HTML, VHDL, ARM, RISC-V, R, and MatLab

· **Other:** Soldering, circuit design and test with Oscilloscope and Digital Multimeter, Altera DE1-SOC Cyclone V

**Personal Profile:**

· Self-motivated and disciplined with strong leadership characteristics

· Enjoys athletics, cooking, traveling, hiking, and camping

Philip Schmeichel

philip.schmeichel@live.com   | 253-797-8885

OBJECTIVE:

To work and participate in data communications and information technology for promoting more just and transparent access to education. To actively learn and improve skills in engineering design and technical communications while searching for employment. To apply my learning and talents creatively.

EDUCATION:

**Seattle University | Seattle, Washington | Sep. 2018 – June 2020**

*3.929 GPA through 92 credits, B.S. Electrical Engineering with Computer Engineering specialization*

* Computer hardware, machine learning, signals & systems, and data communications
* Coursework and projects with Python, C++, C, and VHDL
* Broadening skills with MATLAB, Simulink, Spice, AutoCAD, and lab projects

**South Dakota School of Mines & Technology  |   Rapid City, South Dakota  | Aug. 2016 - May 2018**

*4.000 GPA through 72 credits, progress towards B.S. Electrical Engineering*

* Solid foundation of circuits, math, and technical communications
* Collaborated with Mechanical and Industrial Engineering departments in design lab courses

PROJECTS:

**Panthera Project Spot Check | Senior Design | Sep. 2019 – present**

* Tasked to improve accuracy of a Python program that matches snow leopard images
* Worked with Microsoft Azure, Databricks for testing the program in the cloud
* Used OpenCV and machine learning libraries for a Mask R-CNN template generator
* Made technical documentation and weekly updates; boosted communication and management skills

**Android EKG Heart Monitor | Junior Lab | Sep. 2018 – June 2019**

* Applied course material from Circuits II, Semiconductors, and Signals & Systems classes
* Developed an Android phone app to show real-time heart monitor from probes on team-designed and self-assembled hardware board

**Alfred the Arm | Mechatronics Lab | Jan. 2018 – May 2018**

* Teamed with two Mechanical Engineering classmates to create a mechatronic system
* Created a mock Etch-a-Sketch using an Arduino Uno, potentiometers, and servo motors

EMPLOYMENT:

**Human Resources Clerk | The Northwest School (Seattle, WA) | Oct. 2019 – May 2020**

* State work-study helping a non-profit school with audit projects
* Managed email inbox for job inquiries and websites for job postings

**Summer Undergraduate Research Assistant | Seattle University | June 2019 – Sep. 2019**

* Designed challenge for upcoming junior class lab project as part of Keck Foundation grant
* Became familiar with setting up Raspberry Pi environment and testing with Python

**Cash & Carry Sales Associate | Smart Foodservice Warehouse Store | May 2018 – Sep. 2018**

* Improved work ethic, customer service, and communication to expand on retail experience

**Sports Information Worker  |   SDSM&T Hardrocker Athletics  |   Sep. 2016 – May 2018**

* Recording and relaying statistics for the football, volleyball, soccer, and basketball programs
* Handling various displays and promotions before/between games

INTERESTS:

* Audio design/engineering and video editing
* Speedrunning and fighting games
* Running recreationally and playing in team sports

**Amudhan Sekar**

|  |  |
| --- | --- |
|  |  |
| **Contact:**  Phone: 650-485-9136 Email: [amudhan1998@gmail.com](mailto:amudhan1998@gmail.com) | |
|  | |

**EDUCATION**

Bachelor of Engineering, **Seattle University,** Senior, Graduation Year: 2020

Major: Computer Engineering, Minor: Computer Science

**EXPERIENCES**

* **California Public Utilities Commission Software Engineering Internship**

Developed both enterprise and public web-based applications. Completed projects with the use of Oracle SQL, PLSQL, Oracle APEX, JavaScript, jQuery, HTML5, and CSS. Modified existing databases and database management systems. Worked as part of a project team to coordinate database development and determine project scope and limitations.

* **TipNow Software Engineering Internship**

Developed modules for TipNow Application Software using Python. Worked on supervised clustering methods from machine learning to filter and detect safe and unsafe keywords. Used open source package to create license plate recognition in C++ and implemented the software used in the TipNow app.

* **Amma Therapeutics Pharmaceutical IT Internship**

AMMA Therapeutics Pharmaceutical Inc. Hayward, CA- Involved in design, development implementation for systems tracking internal chemical data. Used AWS API S3 to generate automated uploads of data. Helped build LAN networks and setup electrical equipment.

* **1st Place Seattle University Inaugural Impactathon**

Given a task to help a nonprofit and create a new idea to help them. Worked in a small team to draft an idea, build a prototype and present the protype to judges.

* **Fry’s Electronics Computer Sales Associate**

Helped with customer computer problems. Salesmanship. Gained knowledge about Bluetooth, small electrical components, processors, laptops, desktops, monitors. Learned about the demand of these electronics and the process of in between the developer and the consumer.

* **Eagle Scout**

Logged over 200 hours of community service and held high leadership positions.

**SKILLS**

Computer Engineering

Risk-V, VHDL, Raspberry Pi 3, Questa Sim and Quartus on Cyclone V, Pipelining, Multithreading, Traps and Exceptions

Computer Science

C++, C, Java, Python, JavaScript, Cascading Style Sheets (CSS), PLSQL, HTML, ORACLE APEX

Electrical Engineering

AC and DC Power Analysis, BIBO stability, Frequency Response, Laplace Transforms

<https://github.com/amudhansekar/mygitrepos/>