


Object Glorification

By Shani Bidgary and Shani Bar-Gera

Supervisors: Boaz Sternfeld and Yaron Honen

Spring 2021

Table of Contents

Introduction……...……...……....3
Application Overview……...……...……..…....4
Technologies and Platforms……...……...7
Development Process……...……...…9
Stage 1 - Object Segmentation……...……..……..9
Mask-R-CNN……..……..…..…..…9
TensorFlow……..……..……...12
Grabcut……..……..…..14
Stage 2 - Connecting the Segmentation Script to an App……...…..…..17
Stage 3 - Creating the UI……...……...18
Future Work……...……..19
Conclusion……...……...20
References……...……...21

Introduction

The goal of the project was to create an application that glorifies a chosen object within a given image. After the glorification, the user would have some sort of ability to “play” with the object by performing various manipulations on the object.

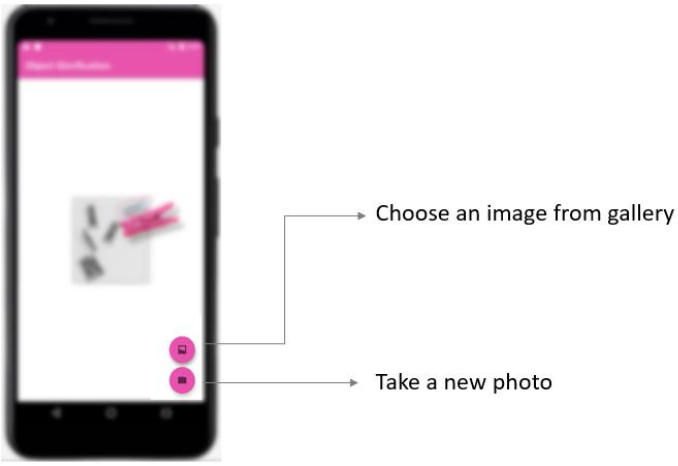
We decided to achieve this goal by creating an Android application that allows users to select an image, or to take one, mark an object in that given image and perform the glorification process. In this case, glorifying means blurring the background and magnifying the object. After the object is glorified the user can perform manipulations on the object such as scaling and rotating. We also decided to add a bonus option - allowing the user to select two objects at once.

To achieve our goal we need first to find a segmentation algorithm to segment the image and by that separate the desired object from the background. After that, we need to find a way to connect that segmentation algorithm to an Android application. And lastly, we need to create a simple UI for the Android application.

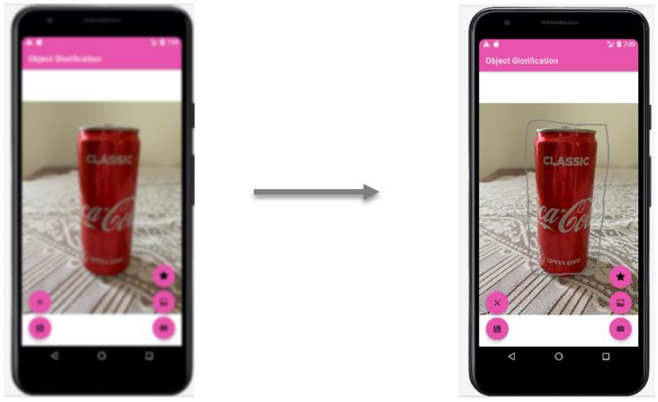
Application Overview

1. Opening Page

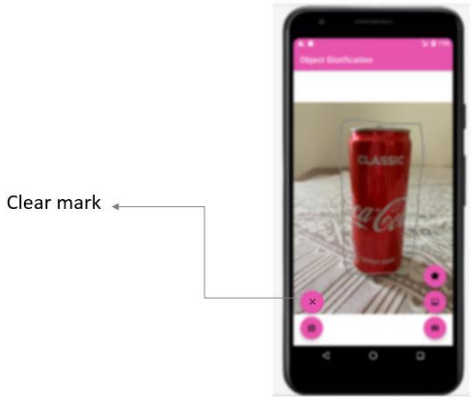
2. Select Image - the user can choose between uploading an image from the gallery or taking an image with the camera



3. Mark an Object - the user can mark with their fingers the general area the desired object is in the photo



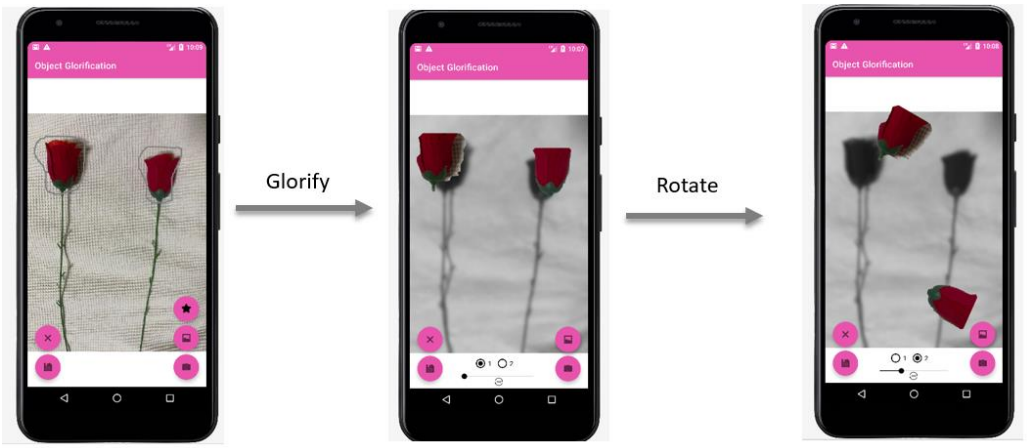
4. Clear Mark - if the user wishes they can clear the mark and try again



5. Glorify the Object - by pressing on the star button the Python script is called and the image processing is occurring. A new image will be displayed with the object magnified and a blurred black and white background. To undo the changes the user can use the ‘clear mark’ button (X button).

6. Manipulate Object - the user can magnify the selected object with their fingers, rotate the object with the bar, and save the entire image with the save button.

Bonus: Two objects - the user can perform the whole process also on two objects simultaneously. After the glorification, the user can choose which object to manipulate with radio buttons.



Technologies and Platforms

- **Python 3**
  The image processing for this application was written with Python. The Python script receives an image and a rectangle that the user marks and includes the desired object, and returns an object and a blurred black and white background.

- **OpenCV**
  The image processing includes segmenting the object with the GrabCut algorithm, blurring the background, changing it to black and white, and magnifying the segmented object. All these were done with the OpenCV package offered by Python.

- **Android Studio - native**
  Android Studio is an IDE for Google's Android operating system, designed specifically for Android development. We used the native **Gradle**-based version.

- **Gradle**
  Gradle is a build automation tool for multi-language software development. It supports different languages, we used **Java**. Gradle uses DAGs to determine the order in which tasks can be run, and provides dependency management. Gradle is distributed as an open-source software.

- **Chaquopy plugin**
  Chaquopy is a plugin for Android Studio's Gradle-based build system. It provides a simple API for the Android developer to include Python components in an Android app.

The Development Process

In order to create our app we separated our development process into three main parts:

1. Segmenting an image with Python
2. Connecting the segmentation to the Android app
3. Creating the app's UI

We will now talk about each stage, our thought process, challenges, and resolutions from each part.

Stage 1 - Object Segmentation

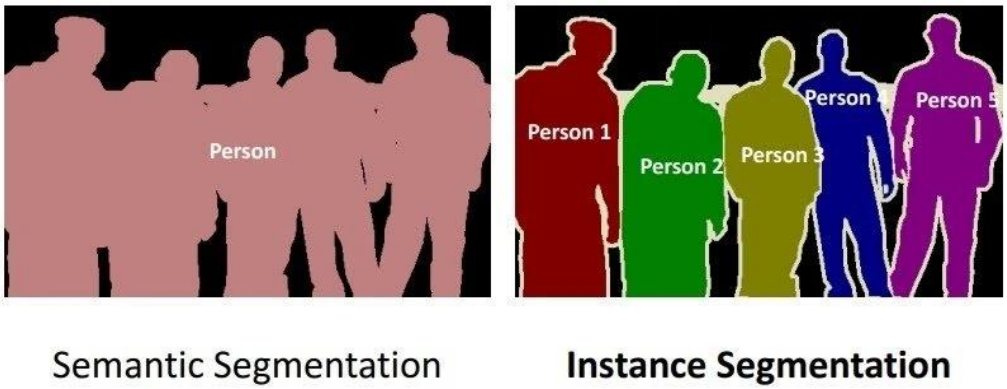
Object segmentation is the process of segmenting an image into portions of smaller size images. Meaning, the computer will be able to tell how many objects there are in the image and to tell for each object if it corresponds to the foreground or the background.

In order to glorify an object from an image, we wanted to find a good segmentation algorithm to catch that object and separate it from the background. We read about various segmentation algorithms and studied and tested three main ones: Mask-R-CNN, Tensorflow, and Grabcut.

<u>Mask-R-CNN (1st attempt)</u>
Mask R-CNN is a Region-Based Convolutional Neural Network (CNN). It detects objects in an image and generates a segmentation mask for each one of them.

There are two types of Mask R-CNN:

1. **Semantic Segmentation -** classifies each pixel into a fixed set of categories without differentiating object instances.
2. **Instance Segmentation -** deals with the correct detection of each object in an image while also precisely segmenting each one of the instances.

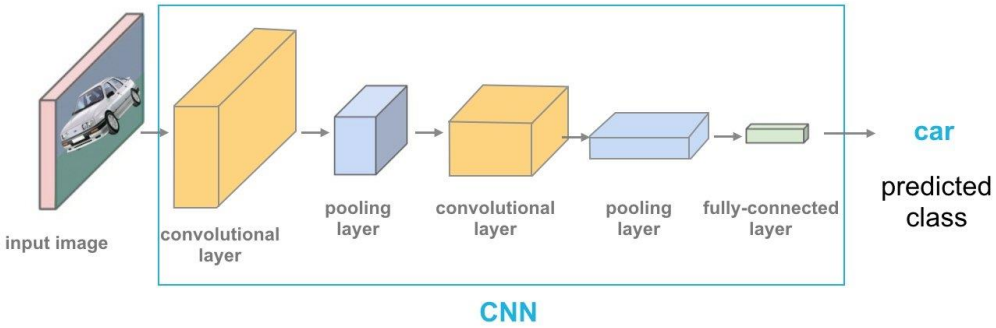


What is CNN?
Convolutional Neural Network (CNN). Artificial neural networks are used in image recognition and processing.

CNN Architecture consists of three main layers:

1. **Convolutional layer -** uses filters and kernels to create a feature map from the image.
2. **Pooling layer -** downsizes the feature map by summarizing the presence of features in patches.
3. **Fully connected layer -** connects every neuron in one layer to every neuron in another layer.

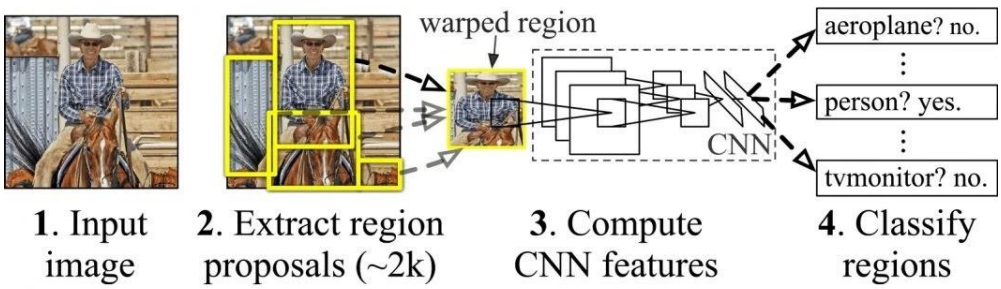
When combining these layers the neural network can learn how to recognize the desired object in the input image.



A simple CNN architecture works well for one object but isn't optimal when multiple objects are wanted. In this case, we will want a more complex algorithm like R-CNN.

What is R-CNN?

R-CNN utilizes rectangles across the object regions and then evaluates all of the regions of interest independently to classify them into the proposed class.

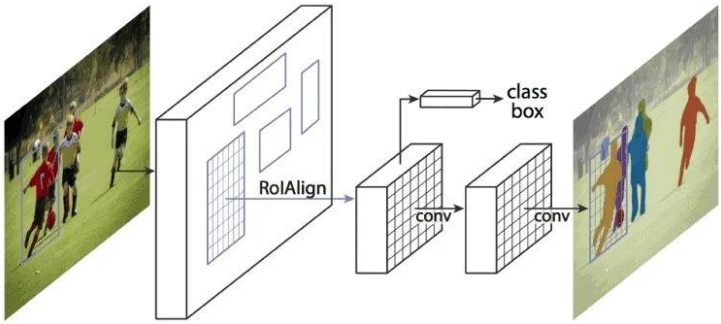


Fast R-CNN

Fast R-CNN is an improved version of R-CNN architectures that has two stages:

1. **Region Proposal Network**- a neural network that proposes multiple objects within a particular image
2. **Fast R-CNN** - extracts features using Regions of Interest from each rectangle and performs classification and regression

How Does Mask-R-CNN Work?
Mask R-CNN is an extension of Faster R-CNN. A new branch for predicting an object mask was added in parallel to the existing branch for bounding box recognition.



Should We Use It?

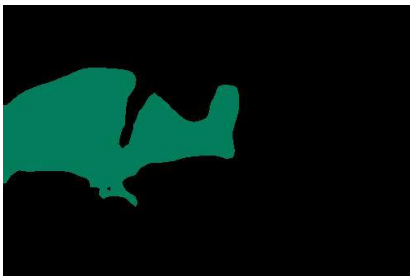
We tried to find implementations of Mask-R-CNN online but had a lot of problems due to version conflicts of packages of the various implementations. After many attempts to find a proper implementation, we decided to not use this method for our application.

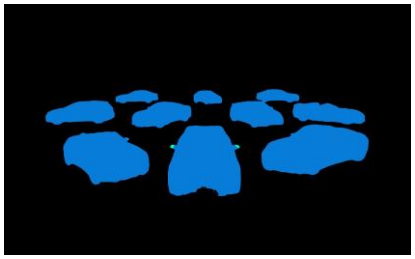
TensorFlow (2nd attempt)

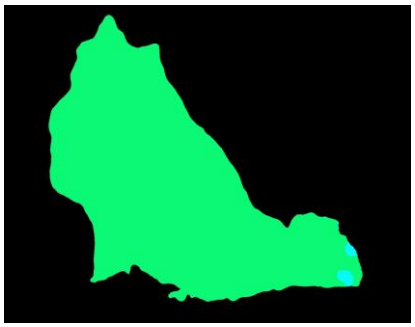
What is TensorFlow?
Tensorflow is an open-source library for numerical computation and machine learning. It involves Machine Learning and Deep Learning models and algorithms.
The computations of Tensorflow use tensors, which are vectors/matrices of n-dimensions that represent data. TensorFlow has many applications like voice recognition, series prediction, and the application that is relevant for our project - image recognition.

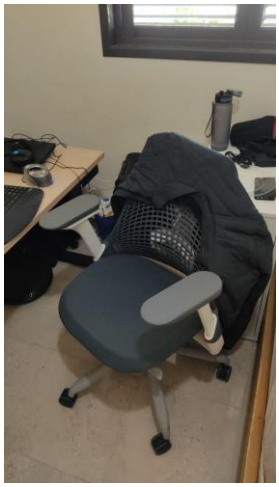
Image Recognition with TensorFlow
Deep Learning gets some images that are labeled manually and trains the system to identify them. After, the system will be able to identify other examples that are not previously shown. The advantage of TensorFlow is that it helps to identify and categorize different objects within one image.

Should We Use It?
We Tested TensorFlow of various images to see if it fits the needs of our application.
Here are some examples of our testing attempts:





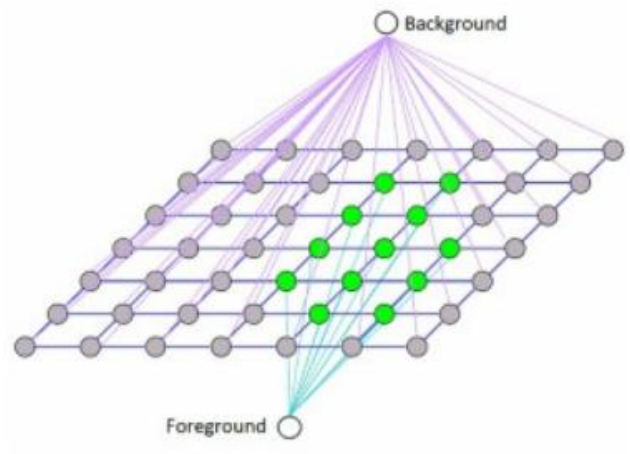


While it seemed to work for the more popular objects like dogs, cars, and birds it was not very well trained and other objects like the chair or the screws. We understood that in order to use this algorithm we need to have a large number of photos to train the program so a wider range of objects could be segmented. We preferred an app that works well straight away so we decided to not use TensorFlow for our segmentation algorithm.

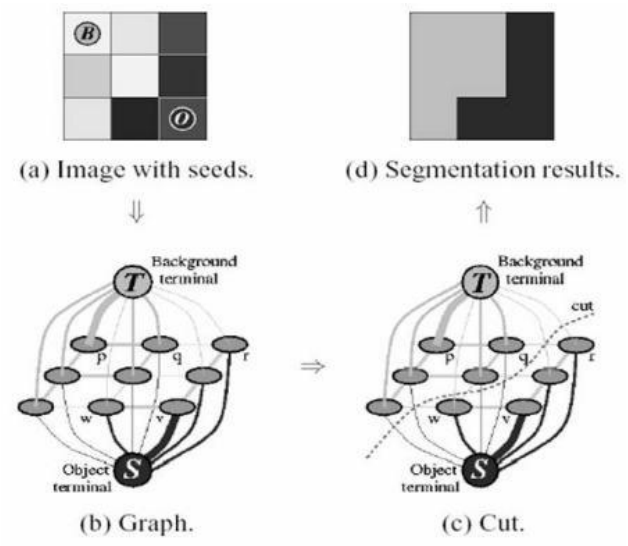
Grabcut (3rd attempt)

What is Grabcut?
Grabcut is an image segmentation method based on graph cuts that segments an image into the foreground and background. It estimates the color distribution of the image using GMM (Gaussian mixture model).

The algorithm receives a rectangle on an image that includes the selected objects. The area lying outside of the rectangle is defined as the background and the algorithm defines the area in the rectangle as a color distribution model using GMM. Every pixel is connected to one another by a gradient and each one is given one of three labels: foreground, background, or unknown. Neighboring pixels of similar color distribution are likely to have the same label.



Then, the algorithm creates a weighted graph which is solved by using the Min-Cut algorithm. The vertices are the pixels in the image, and neighboring vertices are linked with edges whose weights are defined by color similarity. Additional two nodes are added, Source node and Sink node. Every foreground pixel is connected to Source node and every background pixel is connected to Sink node. The weights of these edges are defined by the probability of a pixel being foreground or background. If there is a large difference in pixel color, the edge between them gets a lower weight. Finally, iterations of Min-Cut algorithm are used to define a label for each pixel. The pixels that are connected to the source node are labeled as foreground and those connected to the sink node are labeled as background.



Should We Use It?

We tested Grabcut on various images:

The Grabcut testing seemed to show much better results, as seen in the images above, than Tensorflow. It has many simple and easy Python implementations that we found online unlike Mask-R-CNN. It works for all pictures and does not require any training, has a decent runtime, and light in memory consumption. Therefore, we chose Grabcut as our segmentation algorithm.

Creating the Segmentation Script

After choosing the Grabcut algorithm we improved the Python script to fit our needs and by using the OpenCV package we blurred the background, turned it to be black and white, and enlarged the returned object. In doing this we faced the challenge of separating the object from the background that we eventually solved with the use of masks.

The improved script's results:

Stage 2 - Connecting the Segmentation Script To an App

After creating a segmentation script in Python we wanted to connect the script to an Android app. This proved to be a challenge since we didn't find an easy default option to run Python scripts in Android apps that are written in Java/Kotlin.

First Attempt - HTTP Server

In our first attempt, we wanted to set a Python server that will run in the background and communicate with the app using HTTP requests. It will receive an image from the app and a rectangle, and will return an image of the segmented and enlarged object and another image of the blurred black and white background.

We created a Flutter Android app written in Kotlin and a Python server. While the app and server seemed to work fine separately we had many issues with transferring an image between one another, creating a 2 sided communication stream, and transferring multiple objects. In addition, we were worried about how to make sure the server always runs in the background and preferred an app that can be standalone after being downloaded to the Android device. We also preferred an app that is not reliant on network availability and that could work also without reception or WiFi. Due to all those reasons, we decided to look for other options.

Second Attempt - Chaquopy Plugin

After deciding to look for other options to connect our Python script to an Android app we did some research online. We found the Chaquopy Plugin that allows creating Gradle-based Android apps objects from Python scripts that can accept input, send it to the python function, and return the output. With this plugin, we were able with a simple API to run python functions from an Android app, without the need to set up a separate server or for the app to rely on any network connection.

But by switching to Chaquopy we had to do some adjustments in our implementation.

Firstly, those Python objects required us to accept input in string format so we had to convert our input into string format before sending it and the output out of string format after receiving it. We also had to convert the input in the Python script from string to Image and after the Image processing back to a string.

Secondly, to use Chaquopy we needed to change the Flutter app to a Gradle based app and in addition we switched to writing in Java instead of Kotlin mainly because we found a very helpful tutorial that explained how to use Chaquopy that used Java.

Another issue we encountered was that our Grabcut implementation used the python-igraph package which is not supported by Chaquopy.

The Chaquopy development team opened an issue following our request to add igraph package support:

https://github.com/chaquo/chaquopy/issues/539

Since they mentioned the issue isn't of high priority for them we decided in the meanwhile to use the Grabcut implementation offered by Python OpenCV. This implementation was much simpler and required far less code but also took more time to run. To improve the script's performance we resized the uploaded image to a smaller size before sending it to the Python script.

Stage 3 - Creating the UI

In our last stage, we had to learn Java and some Android App development in order to create a comfortable and user-friendly interface.

We created buttons for the user to add an image from the gallery or from the camera.
We created a canvas layer for the user to easily draw on the selected image and mark the desired object. This required us to learn thoroughly how to use the Canvas and Paint libraries. We had to master the asynchronous functions in order to identify when the user finished drawing, allow the user to use the buttons in order to call the Python script and process the image and block the buttons from use while the processing occurs.
We added various features like rotating the object, magnifying it with the user's fingers and saving the new image created.
We improved the UX by enabling selection cancelation.
And lastly, we added a bonus option of selecting 2 objects simultaneously and enabling manipulation of each one separately.

Another challenge we faced was that the app didn't act the same on different devices, for example for one device the image uploaded from the camera was displayed horizontally while on a different device it was displayed vertically. To solve this we learned about ExifInterface. **Ex**changeable **I**mage **F**ile Format (Exif) is a standard that specifies detailed information about a photograph or other piece of media recorded by a camera. It is portable to different devices. For example we were able to access the image's rotation and set it to be standard no matter what is the default rotation for the specific device.

Future Work

Improving the Segmentation

The algorithm segments iteratively to get the best result. But in some cases, the segmentation won't be fine, like, it may have marked some foreground regions as background and vice versa. In that case the user needs to do fine touch-ups by giving some strokes on the images where some faulty results are there. The Grabcut function offered by OpenCV allows you to enter a rectangle to mark the object but also allows you to enter a mask with the touch-ups. In future work, the segmentation could be improved by allowing the user to do some touch-ups as well.

In addition, as mentioned before, we simplified the Grabcut algorithm by using the OpenCV implementation instead of the igraph based one since Chaquopy doesn’t support the igraph package. If in future Chaquopy versions igraph support will be available we could improve the Grabcut implementation by using igraph again and by that improve the image processing performance.

Improving the UI

Currently to segment the object the user needs to mark the object and press a button. In the future, another more comfortable options could be to simply tap the marked object. This will give the app a more simplistic UI with fewer buttons.

More possible improvements could be adding more manipulation options to the object like moving it around, coloring it, rotating it with fingers only, and selecting an infinite number of objects.

Publishing the App

Currently, the app is only available by direct download from our computer to a device. We wanted to upload it to the Google Store but this required a fee for both Google and for Chaquopy in order to get a distribution license. In the future, a small investment could allow the app to be published to the masses.

Conclusion

In conclusion, the project granted us an opportunity to learn many new and interesting things in the programming world and develop useful skills.

We got our first experience with Android development that included mastering the use of asynchronous functions, useful android libraries like Paint, Canvas, and Exif and learning Java and XML language. We also learned about flutter apps and the Gradle build-tool.

We learned about various image segmentation methods and improved our skills with the OpenCV library.

We also learned about HTTP servers, Chaquopy plugin, and the challenges of integrating pieces of code written in different languages.

We got first-hand experience with being a software architect and making technological decisions.

Overall this project gave us a wider base of knowledge in diverse fields and a useful experience that could serve us greatly in the future.

References

Grabcut Algorithm Theory

- https://medium.com/analytics-vidhya/computer-vision-understanding-grabcut-algorithm-without-the-maths-9a97ef4c5ba3
- https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html

About Chaquopy, OpenCV, Gradle

- https://chaquo.com/chaquopy/
- https://opencv.org/about/
- https://docs.gradle.org/current/userguide/what_is_gradle.html
- https://en.wikipedia.org/wiki/Gradle

TensorFlow, Mask-R-CNN Theory

- https://www.mygreatlearning.com/blog/what-is-tensorflow-machine-learning-library-explained/
- https://viso.ai/deep-learning/mask-r-cnn/

Chaquopy igraph Issue

- https://stackoverflow.com/questions/68977565/trying-to-install-python-igraph-on-chaquopy-android-studio
- https://github.com/chaquo/chaquopy/issues/539

Segmentation Algorithms

- https://datahacker.rs/top-10-github-papers-semantic-segmentation/

Exif Interface

- https://developer.android.com/jetpack/androidx/releases/exifinterface

Chaquopy Tutorial

- https://www.youtube.com/watch?v=dFtxLCSu3wQ&ab_channel=ProgrammingFever