# Assignment 3 - Edge Detection

**Peter Thramkrongart and Jakub Raszka**

## Github link

Link to the repository: <https://github.com/JakubR12/cds-visual.git>

Link to the asssignment folder: <https://github.com/JakubR12/cds-visual/tree/main/assignments/assignment-3>

## Contribution

Both Peter Thramkrongart and Jakub Raszka contributed equally to every stage of this project from initial conception and implementation, through the production of the final output and structuring of the repository. (50/50%)

## Description

The purpose of this assignment is to use computer vision to extract specific features from images. In particular, we’re going to see if we can find text. We are not interested in finding whole words right now; we’ll look at how to find whole words in a coming class. For now, we only want to find language-like objects, such as letters and punctuation.

The instructions are:

* Draw a green rectangular box to show a region of interest (ROI) around the main body of text in the middle of the image. Save this as **image\_with\_ROI.jpg**.
* Crop the original image to create a new image containing only the ROI in the rectangle. Save this as **image\_cropped.jpg**.
* Using this cropped image, use Canny edge detection to ‘find’ every letter in the image
* Draw a green contour around each letter in the cropped image. Save this as **image\_letters.jpg**

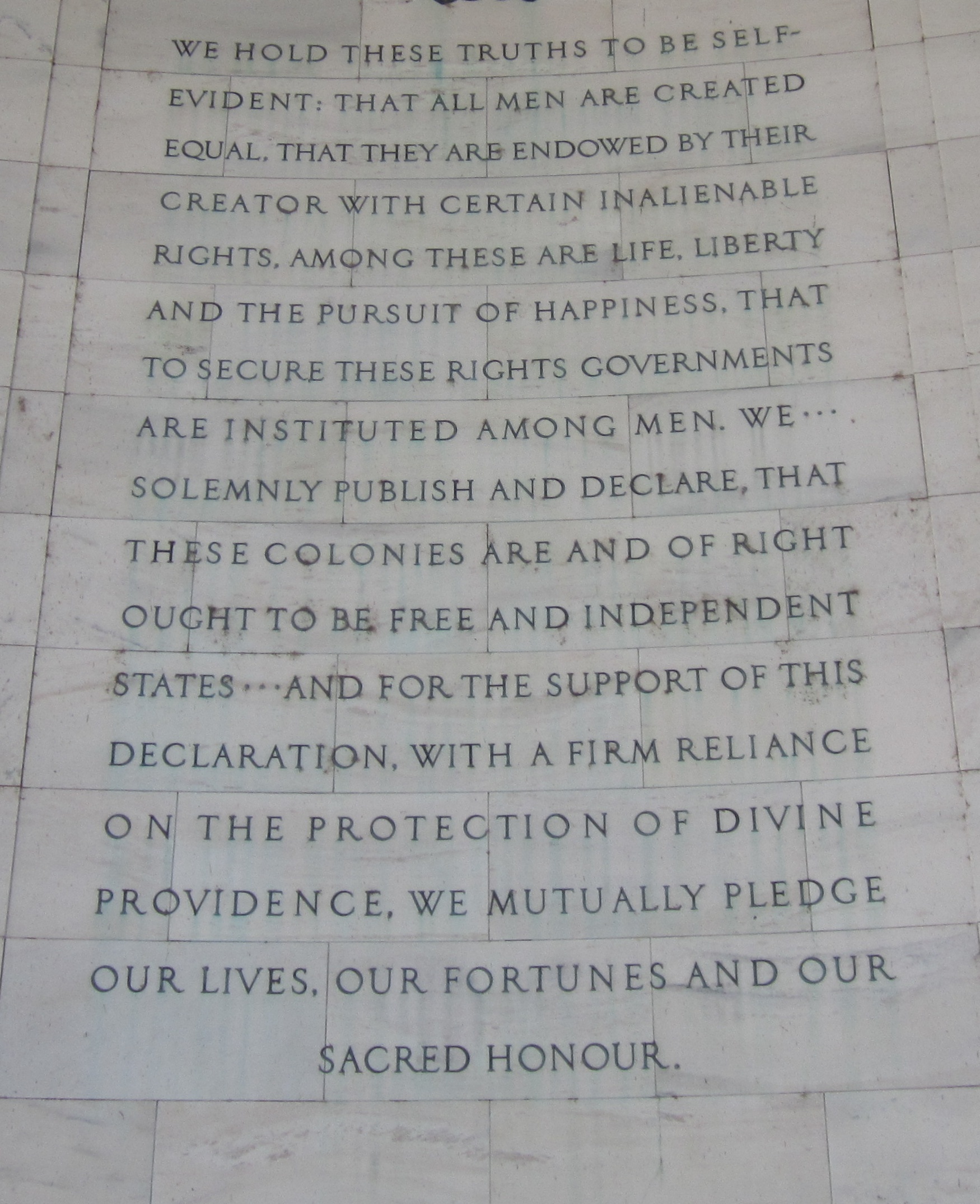
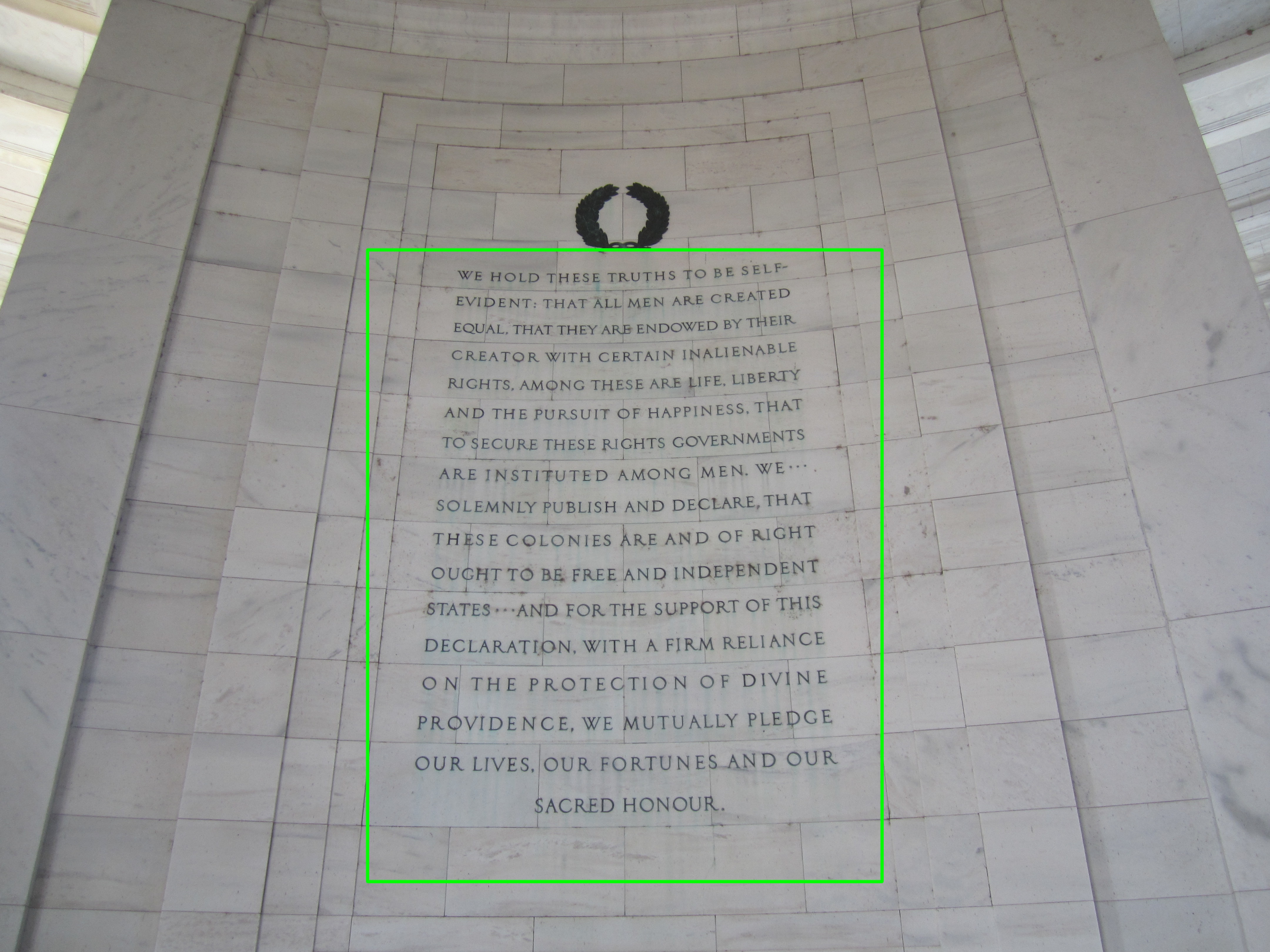
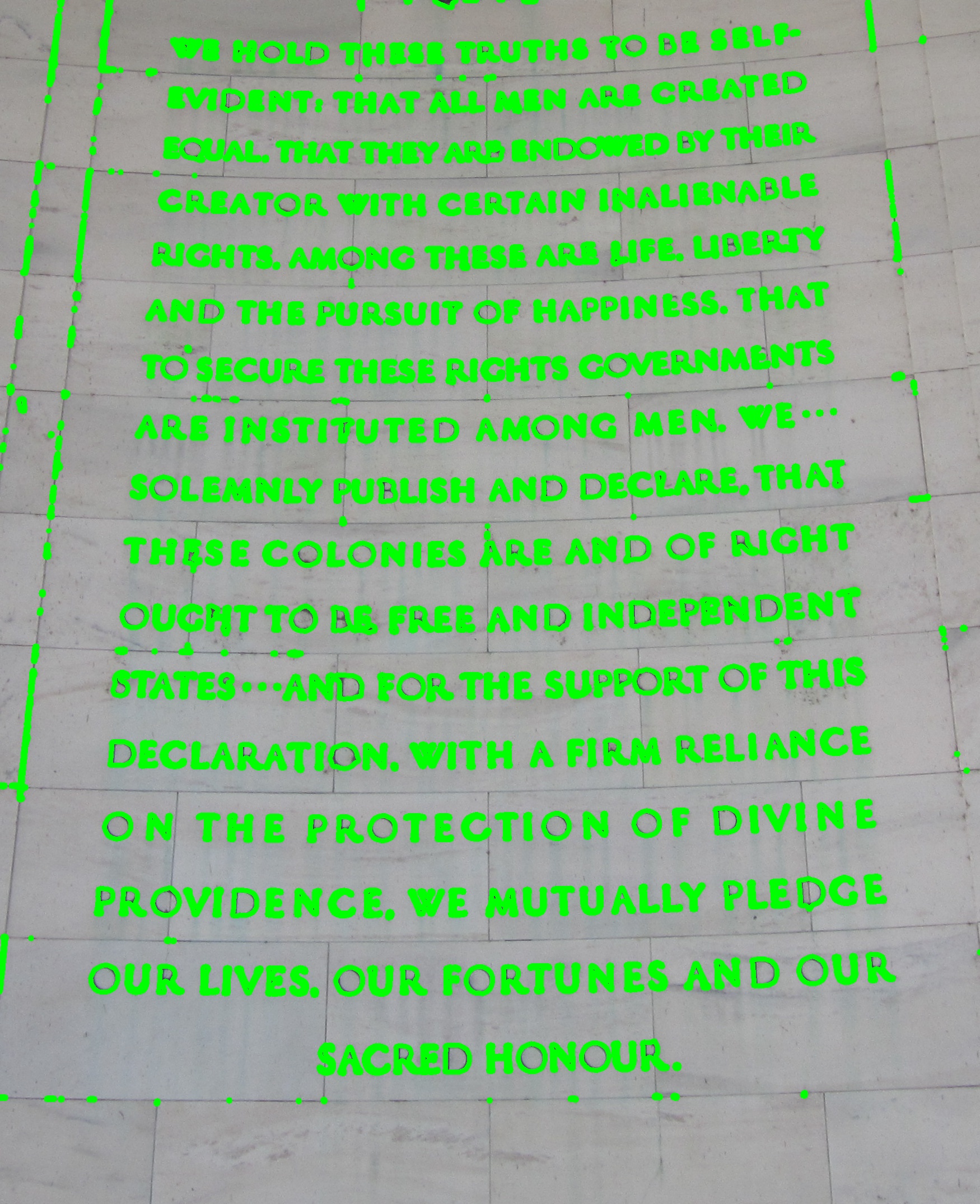
## Methods

The problem at hand concerns finding a text in an image. In order to do so, we first cropped the image to obtain only a text part of the image. Secondly, we transformed the image into a grayscale, used a median blur with a manually set threshold to achieve the best result. The contours of text were found by canny edge detection and colored green.

## Results

We managed to highlight text in a given picture using computer vision. However, as can be seen, the result is far from perfect. Some of the brick interstices were picked up along with the text. Also, the approach relies on manual cropping and the blur parameters are hardly generalizable outside this assignment. Using object-character recognition, such as Tesseract along with a smart pre-processing and natural language processing software to predict the most plausible outcome would yield much better accuracy.

Here are the images:

## Reproducibility

**Step 1: Clone repository**  
- open a linux terminal - Navigate the destination of the repository - run the following command

git clone https://github.com/JakubR12/cds-visual.git

**step 2: Run bash script:**  
- Navigate to the folder “assignment-3”.

cd assignments/assignment-3

* We have written ad *edge\_detection.sh* to set up a virtual environment, run the python script, save the images, and kill the environment afterwards:

bash edge\_detection.sh

## Running the project on something else than Linux

Our projects are mainly made for Linux/mac users. Our python scripts should run on any machine, though our bash scripts may not work. For this case, we recommend using the python distribution system from <https://www.anaconda.com/> to setup environments using our requirements.txt files.

## Project Organization

The folder structure of our projects are based on a simplified version of the cookiecutter datascience folder structure <https://drivendata.github.io/cookiecutter-data-science/>. For the sake of generalizability some folders will remain empty for some projects, but overall this will make folder navigation easier.

├── LICENSE  
├── README.md <- The top-level README for developers using this project.  
├── data  
│   ├── interim <- Intermediate data that has been transformed.  
│   ├── processed <- The final, canonical data sets for modeling.  
│   └── raw <- The original, immutable data dump.  
│  
├── models <- Trained and serialized models, model predictions, or model summaries  
│  
├── utils <- utility scripts with reusable functions and classes  
| └──\_\_init\_\_.py <- Makes utils a Python module  
|  
├── notebooks <- Jupyter notebooks. Naming convention is a number (for ordering),  
│ the creator's initials, and a short `-` delimited description, e.g.  
│ `1.0-jqp-initial-data-exploration`.  
│  
├── references <- Data dictionaries, manuals, and all other explanatory materials.  
│  
├── requirements.txt <- The requirements file for reproducing the analysis environment, e.g.  
│ generated with `pip freeze > requirements.txt`  
│  
└── src <- Source code for use in this project.  
 └── \_\_init\_\_.py <- Makes src a Python module