Image Processing Optimization

Advanced Python Project

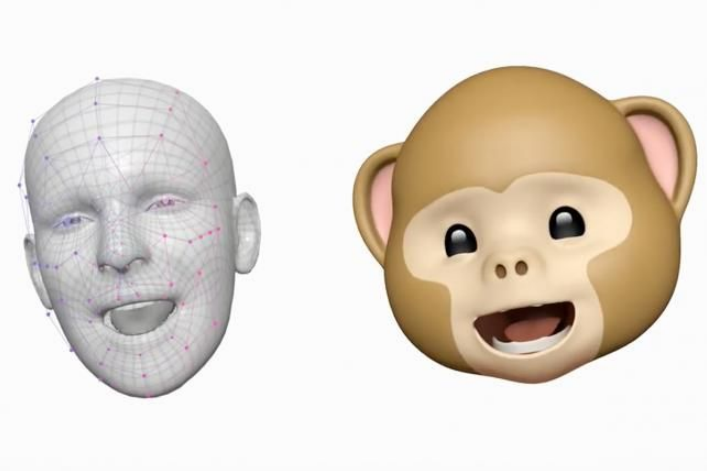
Our Team

- **Stella Sun** - ss8955 - MS Data Science ‘19
- **James Guo** - xg626 - MS Computer Science
- **Ksenia Saenko** - ks4841 - MS Data Science ‘19

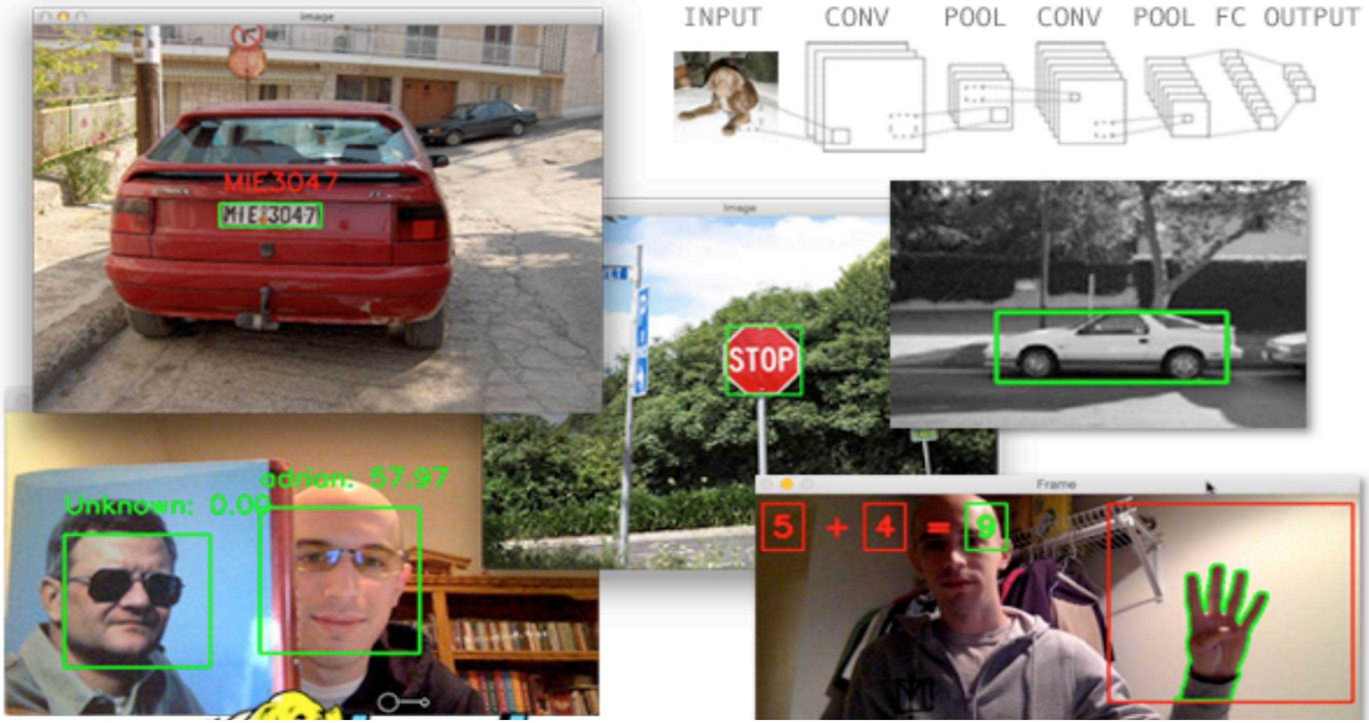
Problem Statement

Image processing:
Pixel accessing and calculating is slow due to high dimensionality, in addition, the usage of “for loops” and “if-else” statement can also slow down the coding process.

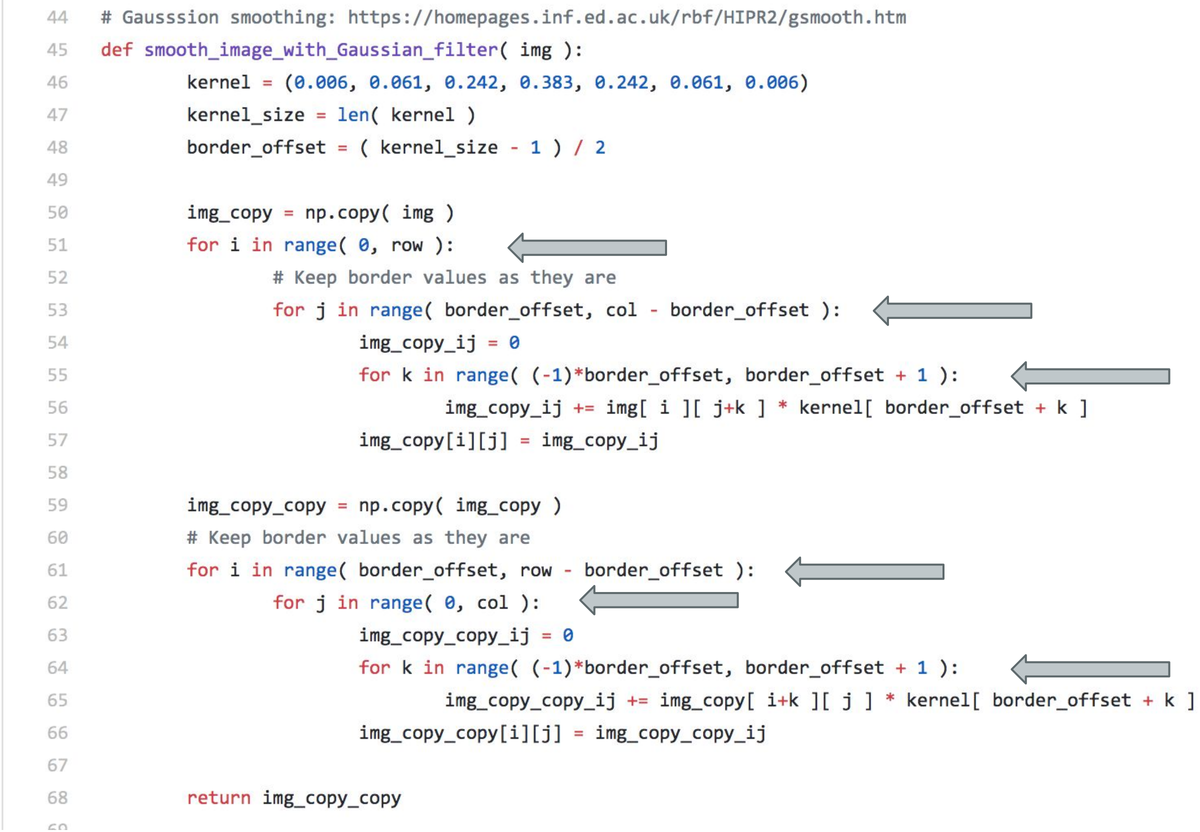
Application Fields of Image Processing:
eye-tracking, face recognition, etc.



Examples: Image Recognition



Code Example without optimization



```
# Gausssion smoothing: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
def smooth_image_with_Gaussian_filter( img ):
        kernel = (0.006, 0.061, 0.242, 0.383, 0.242, 0.061, 0.006)
        kernel_size = len( kernel )
        border_offset = ( kernel_size - 1 ) / 2

        img_copy = np.copy( img )
        for i in range( 0, row ):
                # Keep border values as they are
                for j in range( border_offset, col - border_offset ):
                        img_copy_ij = 0
                        for k in range( (-1)*border_offset, border_offset + 1 ):
                                img_copy_ij += img[ i ][ j+k ] * kernel[ border_offset + k ]
                        img_copy[i][j] = img_copy_ij

        img_copy_copy = np.copy( img_copy )
        # Keep border values as they are
        for i in range( border_offset, row - border_offset ):
                for j in range( 0, col ):
                        img_copy_copy_ij = 0
                        for k in range( (-1)*border_offset, border_offset + 1 ):
                                img_copy_copy_ij += img_copy[ i+k ][ j ] * kernel[ border_offset + k ]
                        img_copy_copy[i][j] = img_copy_copy_ij

        return img_copy_copy
```

It takes about
1.23 min to run!

Methodology

1. **Vector Operation/Matrix Multiplication**: recall the last homework in the heat problem
2. **Built-in libraries:**
   - **OpenCV:** built-in function like “ForEach(allows you to use all cores on your machine to access every pixel in the image)
   - **Scikit-Image**
3. **Cython**
4. **Python optimization tricks:** reduce function call overhead, etc.
5. **Optional:** OpenMP(open multi-processing), PySpark
6. **Version Control** (Github)

Expected Results

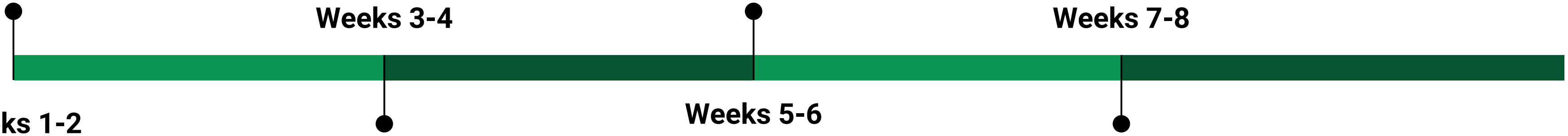
At least 20% faster for the core parts.

Focus on the speed instead of readability or anything else.

We'll directly apply the tools we learned in class

- Python Performance
  - Reducing function overhead
  - Efficient membership testing
- Itertools
- Numpy
- Cython
- Numba
- Timing and profiling tools
  - cProfile
  - Prun
- Julia?
- Spark?

Schedule



Weeks 1-2

Code review and analysis

- Time performance
- Determine areas for improvement and necessary tools

Weeks 3-4

First phase optimization

- Implement planned optimization
- Track performance, measure & document improvement

Weeks 5-6

Second phase optimization

- Improve upon first stage (apply more challenging techniques)
- Build additional functionality for the code

Weeks 7-8

Final phase of the project

- Finalize, test and measure overall optimization
- Document methods, challenges, and outcomes
- Prepare the final report and presentation

Questions?