Team:

Cole Kingery | ckinger@purdue.edu

Will Stonebridge | jwstoneb@purdue.edu

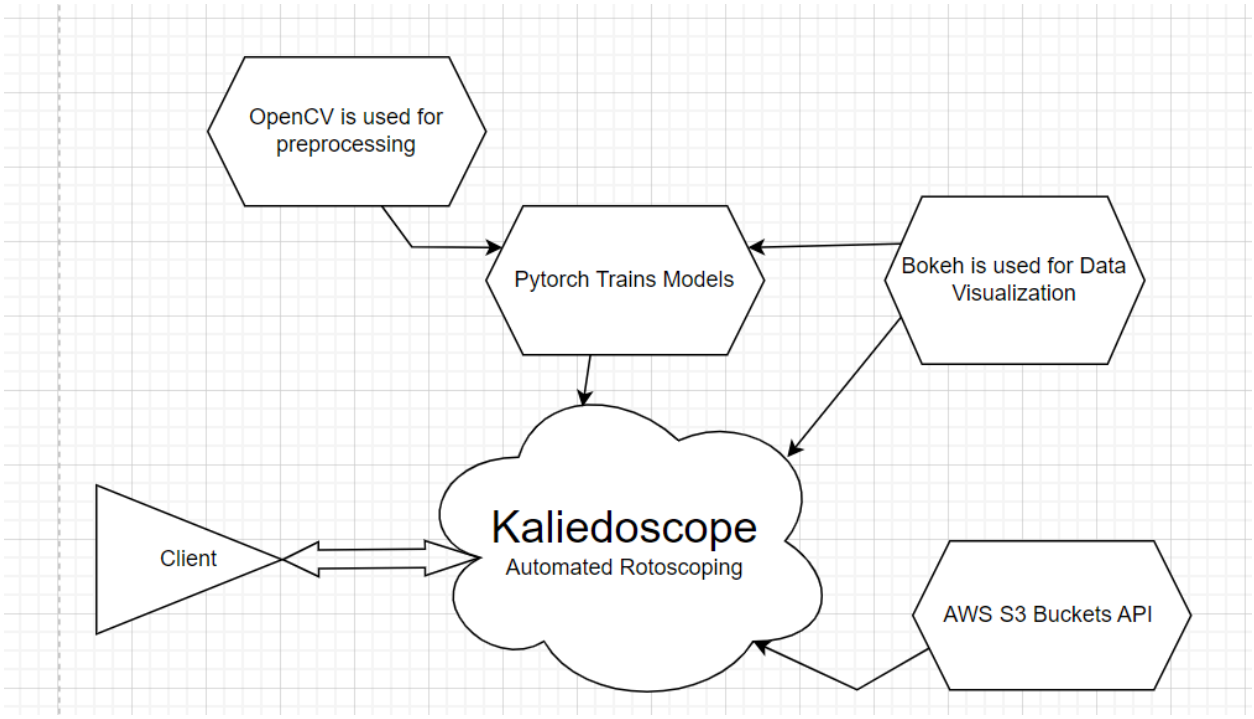
Connor Hise | chise@purdue.edu

Repository Information:

Project Name: Kaleidoscope

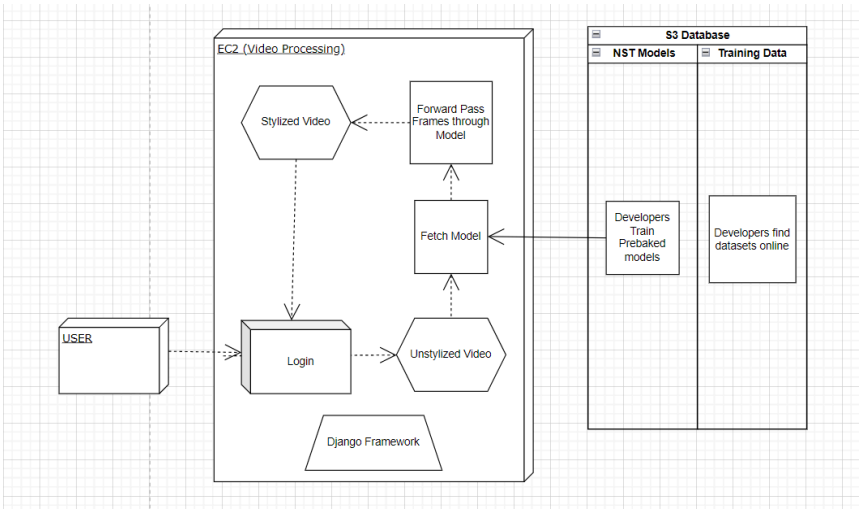
GitHub Repository URL: https://github.com/ColeMK/kaleidoscope GitHub will be used for team collaboration and project planning. We will use branches for component creation to minimize merge conflicts. GitHub projects will be used for task management and deadlines.

Context Diagram

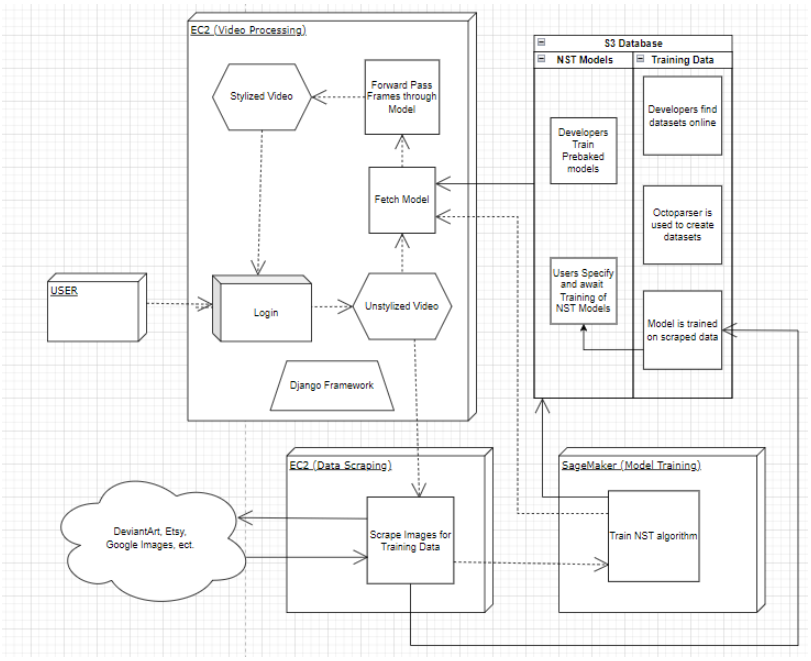


Component Diagram:

Minimum Viable Product:



Extended Requirements:



Requirements:

Minimum Viable Requirements

- Host Website AWS and setup CI/CD.
- Implement a database design. Must be capable of storing and accessing Models efficiently.
- Alter CycleGAN design to perform style transfer on video (Make recurrent possibly).
- Train two CycleGAN models capable of adding style to Video.
- Determine Website Design and implement UI.
- Implement Upload and Download Endpoints.
- Implement Video Downloading

Extended Requirements

- Implement a style scraping functionality.
    - Determine a method to identify images of a given style
    - Alter the website to have a customized style UI
- Use Sagemaker to train models on the fly.
- Adjust database to create, delete, and manage style training sets
- Attempt different style transfer architectures.
- Implement Video Streaming.

Success Criteria S.M.A.R.T

- Total Response Time: The application shall deliver the modified stylized video back to after the user input has been processed and show its progression through a progress bar visible to the user.
- Video Quality: The application shall provide a standard definition (SD) video at 24 fps. 24 fps is the norm for “cinematic” video. Standard definition was chosen as it is the “standard” for video resolution.

- Continuous Deployment: Pushes to the main branch of our Repo must be automatically reflected on the server/website. This shall be achieved within one minute of a commit.
- Responsiveness: The website entry page shall load within 5 seconds. According to Zona Research, 38% of users will leave the site after this timeframe.
- Video input mechanism: Users should be able to input videos through a video uploader on the website. Its associated data should be sent to the application for processing.

This represents what's going on behind the scenes of the minimum viable product scope. We see that the stylized video is sent to the EC2 instance for processing. This is the same website that the website is hosted on so it makes it simple to process the video after the user submits it. A pretrained model is then fetched depending on what style the user selected from the list of available models. The model is stored on the AWS S3 database that the website's EC2 instance would connect and pull from. The video is processed by splitting each second into 24 frames as that is the norm for cinematic video so it is what we base our processing off of. Each frame is then sent through the model and stylized after which the individual frames are joined back together into a fully stylized video. This video is sent by the website back out to the user for them to download and enjoy.

Data is an important aspect of this product as well which is why our pretrained models' training data will also be held in S3.

Extra Ideal Requirements

This diagram represents all of the features we would like to implement at the project's completion; however, we recognize it is also more ambitious. The primary difference between

this extended model and the minimum viable product is the ability for the user to specify the style. As you can see this will require another EC2 instance and a SageMaker Instance. By separating Data Scraping and Model training from the primary instance, we prevent our website from locking up whenever the user specifies a style. In order to scrape data we could use the google or deviant art APIs. However, it is also an option for us to use a service such as octoparse to accumulate this data. Finally, all training will occur on the cloud through Sagemaker, which provides GPU access.

Scope:

Minimum Viable Product: We aim to provide a service in which users can change the styles of their videos into a variety of preselected styles. The user would log into our website. Then they would be prompted to select a style they would want their video transferred to, these would be in a drop down menu given by the developer team. After a style transfer is selected the user would upload a short (i.e. 3 second) video. The website would show the progress in stylizing their video which would then be output back out to the user for them to download and do as they please.

Ideal Product: We would aim to provide a service to which users can change the style of their video into whatever style they would want. The user would log into our website. Then in a text box they would ask what style they would like. They would then upload the video they wanted stylized. The text and video would then be processed and a progress bar would appear indicating model training progress and time till their stylized video is complete. Once the video is complete is it output to the user to watch in the browser or download it for their own personal use.

List of Libraries and Services:

Website: To integrate our AI model to a client-available interface, we plan on using Python with Django as it is an industry standard for what we want to accomplish. Python especially allows for easy integration with the AI. Website and AI metrics will be displayed using the open source bokeh library, as it is an interactive data visualization library that will be helpful for our development. An AWS EC2 instance will be used for hosting our application as a website and running machine learning models for our transformations.

Machine Learning: PyTorch will be used for machine learning model training and testing. PyTorch will be used on the website's EC2 instance for loading the model and then generating the stylized video. The original CycleGan code library will be used for training our GAN and providing data for initial model training.

Extended Capabilities

Machine Learning: We believe that AWS Sage maker will provide us with the best platform to train on for the price. We understand our budget is limited and especially given the costs of training models and data scraping. Hence we will probably have this feature protected by a login page. PyTorch will also be used on the SageMaker instance for the model training and testing.

Data Scraping: Octoparse is a data scraping service that could be used if we want to export the data collection task.

Open Source Components:

Web Services: We will develop our web services using Django, which is an open source Python framework for developing websites. We will do our data visualization with bokeh which is an open source visualization library similar to matplotlib with more interactivity.

Machine Learning: PyTorch is an open source machine learning framework that we will use for our model training and development. The original CycleGan code library is open source as well and we will be using that as our initial model in our minimum viable product.

Extended Capabilities

Data Scraping: Octoparse is an open source data scraping library we will use to collect the data needed to train our AI.

Competitive Analysis:

There are a few similar products to what we are accomplishing, one example would be Vidio.ai (https://www.vidio.ai/tools/online-rotoscoping), which allows a user to select parts of a foreground and apply a masking rotoscoping to the selected object. Another example is Synthetik (https://synthetik.com/aiovg_videos/auto-rotoscoping-examples/), which takes a video and applies a rotoscoping effect to it based on a few preset options. The preset options are built off of an internal gallery that Synthetik provides. Our project differs from these in a few substantial ways, primarily being our data scraping. Our ability to find and train our model from the internet, free of cost for the user, is what separates our project from the current market.