Contents

1 Telescope

- Connor O'Hara: Image Processing (cohara1@stevens.edu)
- Kevin Poli: Application/ Artist Tools Developer (kpoli@stevens.edu)
- Philip Vitale: Application & Systems Developer (pvitale@stevens.edu)
- Brendan von Hofe: Machine Learning (bvonhofe@stevens.edu)

1.1 Advisors:

Hong Man (hman@stevens.edu), Jeff Thompson (JThomps4@stevens.edu)

2 Intro

Telescope is a machine learning assisted toolkit for digital video compositors with applications in visual effects, matte painting and diverse use cases accross the video post production pipeline. Tools from existing compositing packages will interact with a novel ML core to assist or completely automate the rotoscoping process. Rotoscoping is the process of masking and segmenting poritons of an image accross multiple moving frames, any feature length movie will often consist of hundreds of rotoscoped shots with multiple tracked mattes per image, and this is the primary job of thousands of roto artists accross the world.

We hope to make this process, fast, intuitive, and accesible to alleviate the manual and time consuming process that makes up a huge chunk of the man hours required to produce even low budget features. We believe that machine learning is in the process of revolutionizing image processing, and that user driven toolkits rather than black box command line workflows will bring our intelligent core into the hands of the artists where they can thrive.

2.1 Demonstration

Rotoscoping is the process of frame by frame selecting and isolating a given feature (usually an object or person) in a video, such that you can produce a video clip of exclusively that selection on a transparent background

Lets walk through this step by step:

- First, our source image at frame 1, of Marceu the Mime

- Lets start by creating a selection just of the Mime's face and hand - these are the features that are actually being "rotoscoped" out

- This purple selection represents a 'mask' which are the points and curves that make up the boundary of what we are looking to isolate. Traditionally, artists will digitally paint this selection in a software of their choice, by hand.
- This selection or mask is different from a matte, which is another important piece of terminology. A matte is a single channel image; meaning rather than pixels having red,green,blue values, they only contain 1 value from 0-255 called 'alpha'. 'Alpha' will often be displayed in software as white. The Matte of this selection is an image where only the pixels corresponding to the selection are white, and all other pixels are black.

  - this is so that, under the hood, all we need to do is pixel-wise 'multiply' the source image to the matte,

meaning any pixels with a black 'zero value' in the matte will become transparent, and any pixels in the white '255 value' in the matte will remain.

- Here is the result of that multiply, an image containing only the pixels we selected before

2.1.1 Frame By Frame

Much of the challenge and tedium of rotoscoping comes from repeating the above process for every frame, traditionally, artists will go frame by frame through the video and manually adjust their selections to match the feature they are isolating, here is the next frame of that video, with an adjusted selection for clarity

to see how the selections should move as features in the video move, check out this gif that displays the matte on the left, with the source on the right, and has selection lines on both https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj4poGei_bdAhVvTt8KHYSXBs0QjRx6BAgBEAU&url=https%3A%2F%2Ftaukeke.com%2F2014%2F07%2Frotoscoping-in-nuke%2F&psig=AOvVaw0rzB0nhBNxm_0WD1VdybtL&ust=153906208645

2.1.2 Use Cases

With our selection isolated, we can start to play with the image accordingly

By layering the source footage and our rotoscoped hand and face, we can apply an effect, like the 'colorama' effect to only the pixels we roto'd previously

1. Compositing The most popular use case for rotoscoping is Compositing, which is the process of combining multiple images into one. Consider three layers to see how this is done.

   Say we want this red square video clip to appear 'behind' the Mime's face and hand (note what appears black is acutally transparent)

We can grab our source clip and place the square image on top

Then grab our rotoscoped face and hand and place that on top

and here is the desired effect

3 Technical Plan

3.1 Components

Telescope as a product will consist of two primary modules, the Telescope Core, which is a machine learning core assisted by traditional algorithmics that implements the novel functionality of Telescope, and an exchange plugin that allows existing professional compositing tools to interact with our proccesses. Telescope For Nuke is our chosen example exhange plugin, designed to demonstrate how the Telescope core can interact with existing artist workflows - but the separation of core and plugin is designed such that Telescope can be implemented into other software packages like Adobe After Effects or Blackmagic Design Fusion at a later date.

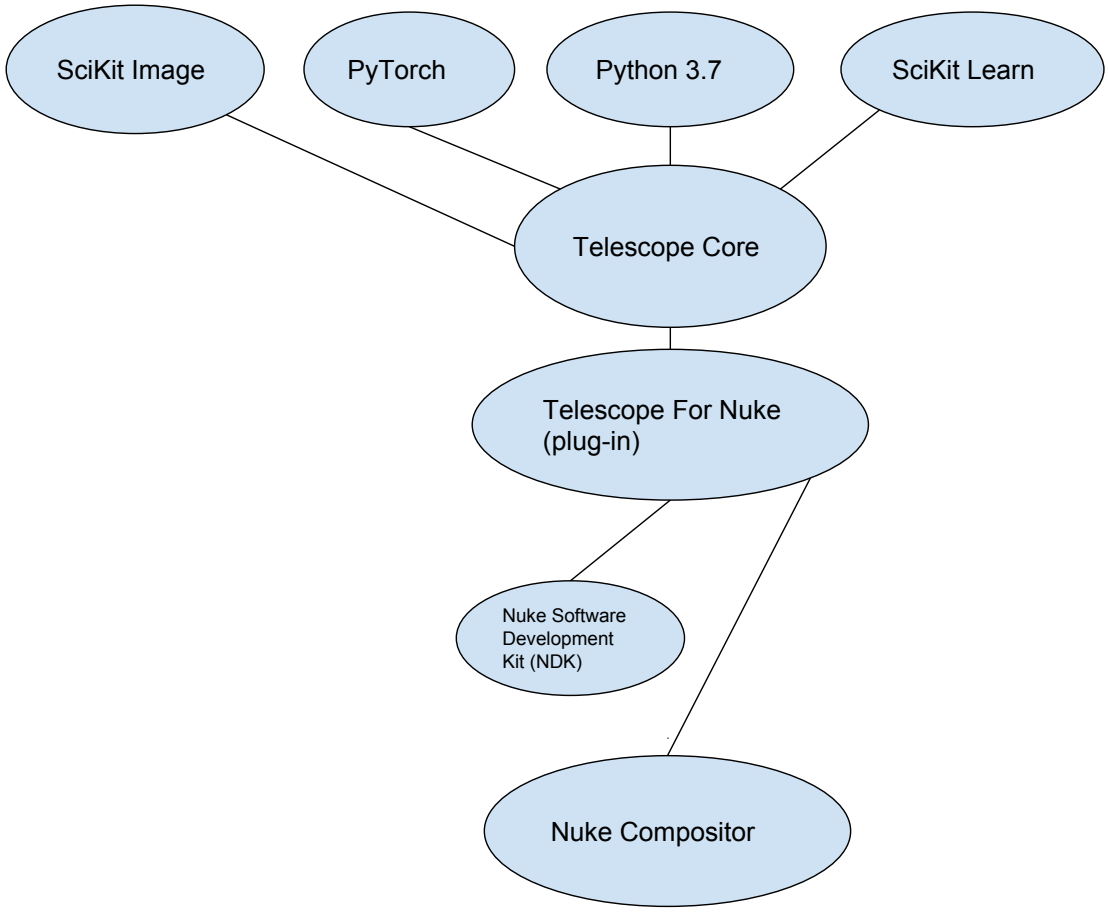
Category	What are we using?
Communication	
Email	Gmail
Web Conferencing	Facebook Video
Instant Messaging	GroupMe
Collaboration	
Document Collaboration	Google Drive
File Sharing/Data Tracking	GitHub
Plugin Development	
OS Supported	Windows, Mac OS, Linux
Host Application	Nuke
Development Language	C++
Machine Learning Development	
Development Language	Python
Packages	PyTorch

3.2 Algorithmics

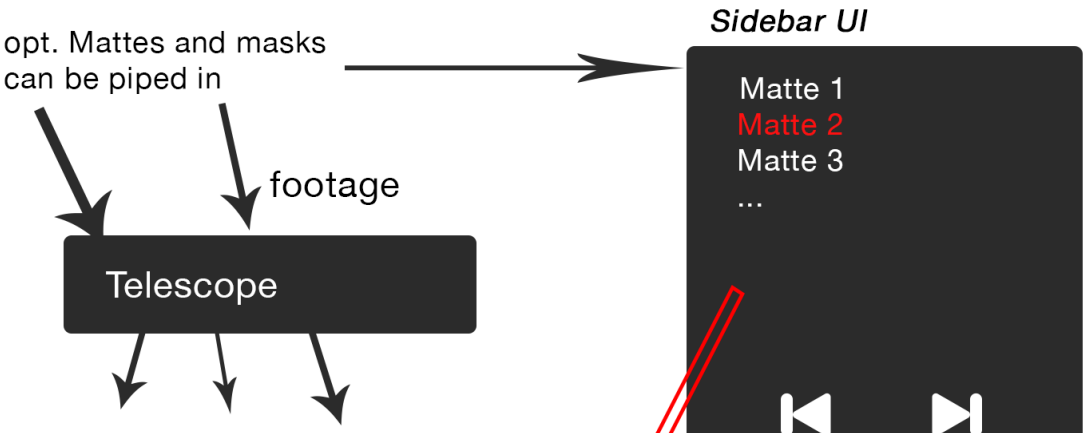
The algorithmic core of our plugin will take images (frames of videos) as input and output segmentation masks (mattes) as output. The goal of the masks is to identify all the discrete objects in the image. It is class-agnostic and therefore does not need to determine what the objects are (e.g. cat or dog) but rather the fact that they are discrete. Our criteria

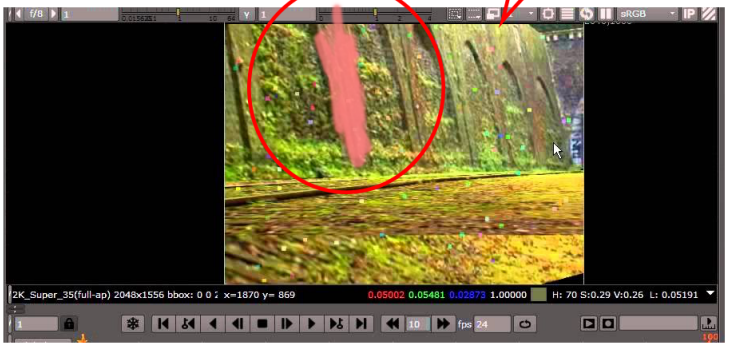
for determining how well our model is accomplishing the task is the Intersection-over-Union metric (IoU). We have yet to determine what an acceptable IoU score is for industry applications. The model will be a convolutional neural network. Specifically, we will begin with the UNet model (`https://arxiv.org/abs/1505.04597`). Initially, our primary dataset to train the model with will be the Panoptic Detection COCO dataset, modified for a class-agnostic task. Further iterations of the model will take advantage of the additional information in EXR images to refine object mattes and the DAVIS video object segmentation dataset.

3.3 Dependency Model



3.4 Plugin UI Mockups





Selecting a matte in the sidebar will expose it to the viewport for frame by frame editing

playhead controls:

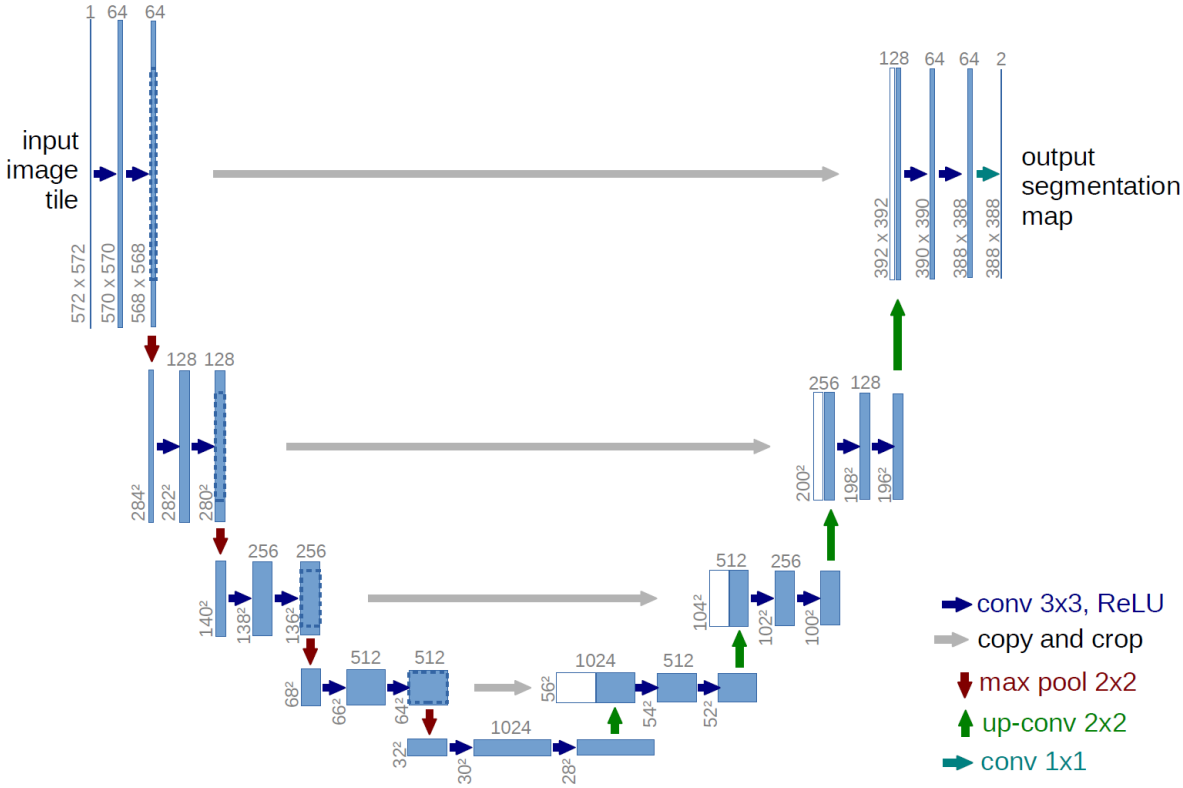
-pause (default)

-interpolate forward (auto-adjusts mask for next frame)

-play/stop(continuosly adjusts frame by frame unless user interrupts)

3.5 Deep Learning Core

The core of our rotoscoping program is the deep learning model that takes the image to be cropped and an associated trimap as input (the two leftmost frames respectively), and outputs the cropped portion (rightmost frame). The image to be cropped can be of anything the user wishes. The associated trimap is used to identify the subject (foreground) in the image that the users wishes to crop out. It is drawn with auxiliary tools of our software to identify the definite foreground in white, the unsure foreground (e.g. hair) in grey, and definite background in black. The output is an alpha matte that can be used to crop the subject out of the original image.



The deep learning core is defined by the architecture (type of neural network) and its training process. The architecture is composed of two convolutional neural networks. The input data first passes through an encoder-decoder style network, commonly used for segmentation tasks (in the image above, the encoder is the first half of the 'U' and the decoder is the second half). Through the successive steps of the network, the image is transformed into different representations called feature maps. The initial representation is the input image itself along with the trimap. As a multidimensional array it has a shape of [height, width, channels]. In this case, there are 4 channels. Three are the RGB channels of the input image, and the last is the associated trimap. As the image passes through the encoder, the feature map representing the image becomes shorter and thinner, but much deeper (e.g. shape of [7, 7, 2048]. The receptive field[1] of the convolutions grows, theoretically allowing it to make higher level abstractions about the subject matter of the image. The decoder then uses this information to eventually generate a single channel image with the same height and width of the

original as it uses transposed convolutions. The second convolutional neural network is much simpler, composed of only a couple convolutional layers similar to the first of the encoder's. The input to this network is the original image along with the alpha matte produced from the first network. It outputs a refined version of the alpha matte. The training process involves feeding the networks images that we also have ground truth alpha mattes for. After the network outputs a prediction for the alpha matte, we compare it with the ground truth alpha matte using a loss function. This loss function is a combination of two loss functions, the alpha prediction loss, and the compositional loss. The former simply calculates the pixel-wise squared error between mattes. The latter does the same calculation but with the original RGB image composited with the ground truth and predicted alpha mattes. Because of the differentiable nature of the loss functions and backpropagation[2], the weights of the neural network are able to be updated to perform the associated task of rotoscoping slightly better after training on each image. We repeatedly train on the entire dataset until improvements become negligible.

3.5.1 More Information

- `https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e`
- `https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611`

3.6 Completion Schedule

As the models have taken to exceptionally long training times, we have pivoted slightly to the goal of having single frame at a time calculated at every frame via the plugin utilizing the ML core. This is in contrast to our previous goal which would use motion within the image as part of the ML core. These features may still come to the plugin, but it is likely that they will arrive as traditionally algorithmic and not ML based features, as there are already algorithmic techniques for making mattes and trimaps more consistent across frames.

Applications	Machine Learning	Week
~~Integrate ML Module Loader into existing plugin Node~~	Explore new training schedules and perform hyper-parameter tuning	4
Test ML Module Loader on Various Models	Continued refinement on the single frame model	5
Research Image Interpolation	Research into multi-resolution support	6
Test interpolations	continued research into resolution independence	7
Begin integrating interpolation tools	Begin implementing resolution independence	8
Refine interpolation tools	Research ML based frame Interpolation	9
Research any additional artist tools required	Research additional ML artist tools	10
implement additional convenience tools if necessary	implement ML frame interpolation and tools if necessary	11
continued refinement	continued refinement	12

4 Team

4.1 Roles

- Connor O'Hara: Image Processing (cohara1@stevens.edu)
- Kevin Poli: Application/ Artist Tools Developer (kpoli@stevens.edu)
- Philip Vitale: Application & Systems Developer (pvitale@stevens.edu)
- Brendan von Hofe: Machine Learning (bvonhofe@stevens.edu)

4.2 Delegation of Tasks

4.2.1 Connor O'Hara

1. Last Week
    - research into adapting matte to curves
2. Update
    - still unsure about curve-ifying complex mattes with transparency
3. For Next week
    - Research potential alternatives for cleaning up an auto-roto'd shot

4.2.2 Kevin Poli

1. Last Week
    - Resuming work in Nuke, and exploring API for novel interactions
2. Update
    - Looking to complete TriMap drawing node with support for edge dialations, masks etc.
3. For Next Week
    - Functional 1-channel 3 color drawing node for TriMaps

4.2.3 Phil Vitale

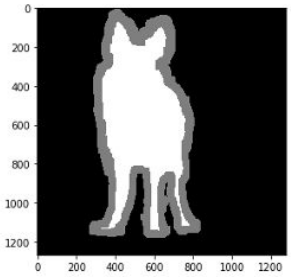
1. Last Week
    - work will be split into TriMap drawing and module loader nodes
2. Update
    - work has began on the ML module loader
3. For Next Week
    - rough draft of the module loader, may be as simple as shellscript/watch folders

4.2.4 Brendan Von Hofe

1. Last Week Utilizing dilation and erosion algorithms to automatically generate trimaps from image mattes. See attached image of dialated trimap

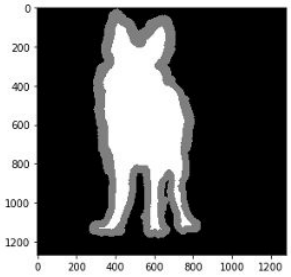
```
In [115]:
kernel = np.ones((26,26), np.uint8)
k2 = np.ones((37,37), np.uint8)
io.imshow(((0.5 * morphology.dilation(vmap(m), kernel))+(0.5 * morphology.erosion(vmap(m), k2))).astype(np.uint8)

Out[115]: <matplotlib.image.AxesImage at 0x7f02f9315b70>
```




```
In [10]:
io.imshow(tri)

Out[10]: <matplotlib.image.AxesImage at 0x7f02fcf8e9b0>
```



2. Update

    - We have begun training the model, initial results even at low training times are looking good. A high level explaination of our techniques has been included above.

3. For Next Week

    - continue iterating on the model, and explore options for further fine tuning