# 3D to 2D: Using Image Segmentation to Automate Rotoscoping

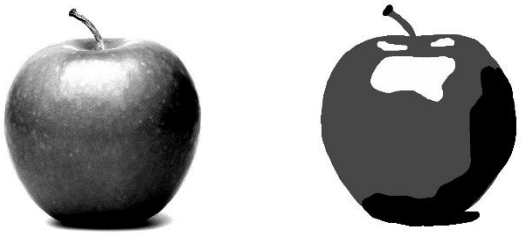
Group 10 - Logan White, Sari Pagurek van Mossel, Zeph Van Iterson

## Introduction

The research problem this project aims to address is using image segmentation techniques to simplify a given 3-dimensional image, for example, a photo of something 3D, to a flattened 2D image. The goal of this process is to extract the average tones from the given image and return simply the overall shape with distinct highlight and shadow sections. The ability to complete this successfully would be a step towards automating the animation process of rotoscoping, which currently requires significant manual effort from the animator. Rotoscoping in animation is the process of tracing frame by frame over live action footage to create an animated sequence. As the resulting animated sequence is typically a greatly simplified replication of the original footage, a tool which would extract the subjects using three distinct shades and separate the background would greatly speed up this manual process of rotoscoping and provide a base from which the artist could continue to animate.

## Research Problem and Objectives

The objective of this research project is to use neural network techniques, specifically U-Net models to create an image segmentation tool to extract shape, background, and 3 distinct shade values from an inputted image of a 3-dimensional scene. This project works towards the problem of automating rotoscoping within the field of animation, by transforming a 3D visual into a 2D representation. See the example below for the intended input and output of the model we wish to create.



## Literature Review

There are papers that use machine learning for rotoscoping such as “Rotoscope Automation with Deep Learning” [1], which uses a deep-learning based algorithm to automatically rotoscope people in any scene without any user input required. The method this paper uses to train a model for rotoscoping should be very relevant for our image segmentation, but since our model will need to separate more than just a person from the background, we will need to expand on their method or use something more advanced. An existing model that will be extremely helpful for this project is

the U-Net, a convolutional neural network developed for image segmentation. Since it was designed specifically for image segmentation, using it as our base and training it on our dataset should be much faster and yield more accurate results than a more general model. There are many papers that use U-Net networks to segment images for medical diagnosis to find lesions and tumours [2] [3]. The methods that these papers use are very relevant to our project since they are using the same model (U-Net) for a very similar use (separating image segments), even if the medical aspect isn't relevant to us. These papers will be read in depth to help develop our method and train our model. Additionally, there is literature that shows the U-Net architecture being used for animation purposes, specifically frame interpolation for 2D animation, and compares U-Net training to GAN training [4]. This illustrates not only that U-Net architectures are already being used to create tools for animators, but it also has measurable training data demonstrating that a U-Net is more stable than a GAN.

## Methodology

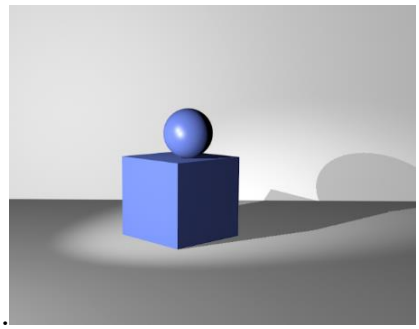
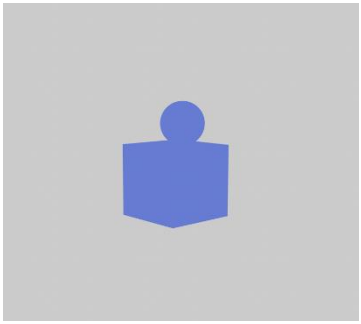
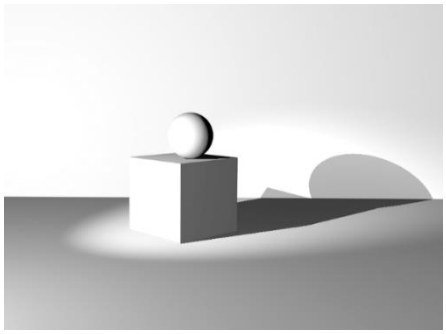
The goal of this project this to produce an image-to-image model capable of segmenting different regions of an image and producing a "simplified" image as output. While outputting region boundaries in a vector format is another possible solution for this problem, producing an image as output was preferred in order to simplify the network's output layer. Additionally, to avoid complexity related to the size of the network, the model will only handle greyscale images.

The specific type of architecture that will be implemented is known as a U-Net. The U-Net architecture is a type of fully convolutional neural network designed to optimize the process of image segmentation. As a U-Net is a type of CNN, the model will utilize supervised learning to optimize its weights. To reduce the required amount of data needed for training, the team will begin with weights from a pretrained U-Net model before training on the custom data.

## Data Collection and Preprocessing

As this project is implementing a novel solution, there is currently no existing dataset that meets our needs. As a result, we will be making our own dataset using the free and open-source 3D modeling software, Blender, and its Python API. Using the API, we will be able to procedurally generate several random scenes featuring spheres, cubes, and other simple 3D objects with varying lighting and camera angles. Renders of these scenes will be used as the input to our model. To acquire the expected output for each scene, we will once again use the Blender API to extract specific composites of the scene from the Blender renderer (see examples below). A simple thresholding system will then be applied to these layers to determine which category each area of the image belongs to and to create an "expected output" image.

In addition to using the Blender API to generate many different scenes, common dataset enhancing techniques such as cropping, horizontal flipping, and adding Gaussian noise will be used to increase the size of our custom dataset

## Experimental Design

Any tests using new data will be very difficult to measure objectively with metrics and would only be able to be judged subjectively. Therefore, we will set aside some of the dataset to use as a test set to test the trained model. The output image from the model can then be compared to the output image from the dataset. This comparison could be done manually, but in order to objectively measure the success of the model, we can compare what percentage of the output images are classified into each section and get what percent of the image is incorrectly classified. This would allow us to compare percentages of different iterations of our model and to any other methods to see if our model is better than other methods, and ensure it is better than random.

## Expected Results and Analysis

We expect our model to interpret the key subjects from a given 3D scene and reformat them as a flattened 2D image using only 3 distinct shades (highlight, mid-tone, shadow) to suggest the 3D form and remove it from the background. If done effectively, the outline of the subject should be accurate enough to the given image that the shape appears the same in the output. Additionally, the deconstruction into a maximum of 3 discrete sections should be accurate enough to indicate any potential light source, shadow, and distinction from other 3D elements in the scene. While comparing the output 2D image to the original given 3D scene is a good way to tell if it is an accurate representation, to effectively evaluate the performance of our model within the larger context of animation, and to test its usability as a rotoscoping tool, we will manually create a short and simple video and use our model to recreate it as an animated sequence. To do this we will use a short video of a ball bouncing, created and exported in Blender, and use a Python script to feed each individual frame through our model and then stitch it back together into video format. This will allow us to conclude if the performance of the model is accurate enough to provide a smooth frame by frame transition.

## Timeline

| Goal | Expected Deadline | Time Needed |
|---|---|---|
| Collect / create labelled dataset for model | October 18, 2024 | 1 month |
| Create initial model | November 1, 2024 | 2 weeks |
| Refine and iterate model to get final model | November 15, 2024 | 2 weeks |
| Analyze results | November 19, 2024 | 4 days |
| Write Final Report | December 3, 2024 | 2 weeks |

## Potential Contributions

Rotoscoping is an extremely common technique used within the field of animation, in which a reference video is traced over frame-by-frame to be reproduced as an animated sequence. On a basic level, rotoscoping can be done in software like Adobe Animate to just trace over each frame and immediately output an animation with a life-like style of movement, however rotoscoping is also used in professional workflows to trace over and add detail to 3D character rigs. A frequent first step in this rotoscoping process, as well as in many artistic practices, is to break down the 3D figure into its key shapes before then moving on to further detail. One can see how in a larger animation project that is minutes or even hours long at the standard of 30 frames per second, this frame-by-frame drawing process can become exceptionally tedious and time consuming. This project has the potential to ease this process by providing that initial step of tracing key shapes and colour blocks, and significantly cutting down on time for the animator. As a tool within the animation industry, this rotoscoping project would be used to create a preliminary base from a 3D reference which would then allow the artist to build from and add further detail in their own style.

## References


[1] O. E. Torrejon, N. Peretti and R. Figueroa, "Rotoscope Automation With Deep Learning," *SMPTE Motion Imaging Journal,* 2020.

[2] X. Li, W. Qian, D. Xu and C. Liu, "Image Segmentation Based on Improved Unet," *Journal of Physics: Conference Series,* vol. 1815, 2020.

[3] M. Krithika and K. Suganthi, "Review of Semantic Segmentation of Medical Images Using Modified Architectures of UNET," *Diagonstics,* vol. 12, 2022.

[4] F. A. Pazos, "In-between frame generation for 2D using generative adversarial networks," *Open Access Theses & Dissertations,* 2024.