Supplementary Materials: A Learning-Based Approach to Parametric Rotoscoping of Multi-Shape Systems

Intel Corporation

1. Training Data

Our data comes from a professionally rotoscoped stop-motion animated film. We use a limited set of rotoscoped data from the completed movie to train our supervised learning models. The rotoscoped data is primarily for facial features, such as eyes, eyebrows, sideburns, and more. As can be observed in Supplementary Table 1 not every shot is equally useful. We only used frames in our dataset that included all of the shapes with which we are concerned. Sometimes shapes are not available when a character exits a scene, turns around or is occluded by an object.

Shot	Labeled Frames	Total Frames	Usable Fraction
Shot A	125	314	0.398
Shot B	327	348	0.940
Shot C	267	289	0.924
Shot D	398	440	0.905
Shot E	321	331	0.970
Total	1438	1722	0.835

Table 1. Ground truth dataset by the numbers.

The ground truth dataset is composed of image data paired with rotoscoped Bézier shape data. From this shape data, we will eventually mine the landmark points, but first we automatically crop the images with a margin around the rotoscoped shapes. This makes training and processing significantly faster, while reducing the size of our database. We chose to retain images of size 400x400 pixels, as opposed to the raw 2348x1566 images, a reduction of just over 95% of the initial data. This reduced image data (alongside the transformed shape data) is used for training. In spite of the relatively low-resolution training data, our models appear to perform well even after the predictions are transformed back to the full resolution images.

1.1. Data Augmentation

Deep learning methods usually require huge amounts of training data to perform at their full potential, and often require expensive manual labeling. Using small amounts of data for training results in overfitting of the model to the inherently less variable dataset. To solve for this and ensure data efficiency and robustness, we perform a significant amount of data augmentation (Figure 1). Along with the usual augmentations like brightness, hue, blurring, scaling, warping and flipping, we introduce a background and occlusion augmentation that we describe here.

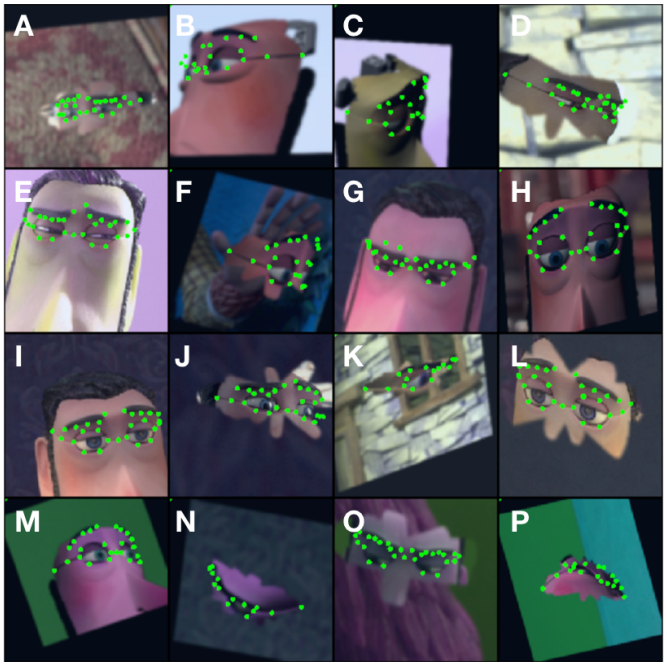

Figure 1. A-P are sample augmented images. Augmentations applied to the training data include: flipping, rotation, warping, random motion, hue, color, and background replacement.

We first compute the convex hull of all the landmark points and apply a small dilation to obtain a base matte. We then add a random cloud around the base matte to form the final matte (Figures 1A, D, J and L illustrate the random cloud matte). We form a database of likely background images including both plain (e.g., blue and green screens) and normal (e.g., objects, rigs, non-face features of other characters from the shots) and we select random swatches while augmenting. By randomizing the background, we force the attention of the model towards facial features only. We observed that models are better able to generalize on faces with small occlusions such as hats when they are trained

with background augmentations that use a random cloud to define the matte, as opposed to a standard geometric shape.

1.2. Synthetic Data Generation

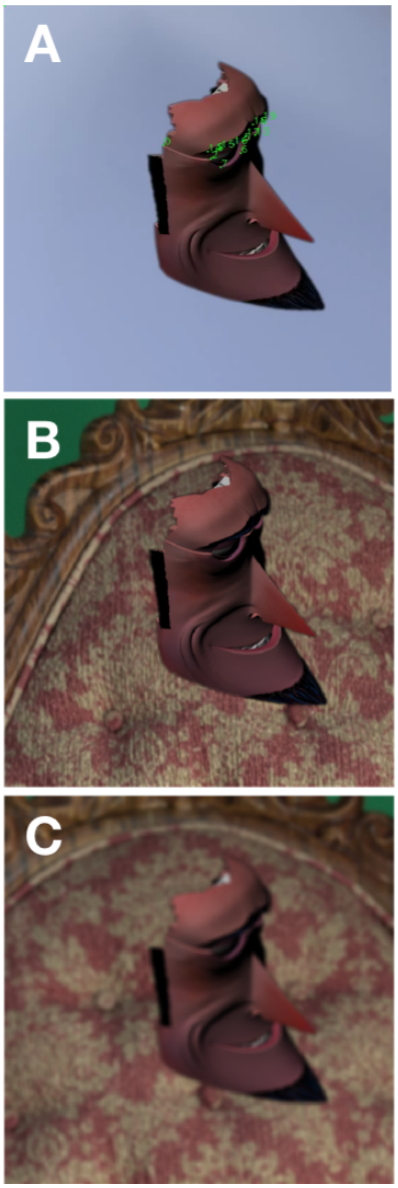

Figure 2. (A) an example rendered labeled data, (B) the same image placed on a random background, (C) and further augmented with blurring.

To address the data variability issue inherent to small training sets, several approaches have been proposed to combine synthetic and real images, but synthetic rendering pipelines are usually unable to reproduce the results of their real-world counterparts. This is often referred to as the domain gap between synthetic and real data and the transfer from one to another usually results in deteriorated performance, as observed in [5].

Several approaches have tried to overcome this domain gap. For instance, [1, 2, 4] use synthetic images in addition to real ones to boost performance. While these approaches result in good performance, there are other approaches that rely on synthetic data alone [3].

Our synthetic data generation pipeline currently leverages 8 facial expressions of our character. The studio uses 3D printing to create their stop-motion animation puppets. We use these pre-existing 3D object files that portray our character with different facial expressions to generate a large set of 3D transformations uniformly covering the pose space in which the character's face is visible (Figure 2A shows one example labeled pose). The character is rendered in a random pose on a randomly selected background image using a uniform distribution (Figure 2B). The selected background image comes from a set of real background images used in the film. To increase the variability of the background image set, we randomly flip and rotate the images. We also experimented with a set of backgrounds of uniform color, however we discovered that using real background images improved the model significantly (This result is discussed in the Experimental Results section). Additionally, some 3D points from the synthetic data can suffer from self occlusion, so we do not consider occluded points during training.

We used Maya for rendering the synthetic dataset with identical rendering parameters for each image. Since the features of the synthetic data are much sharper and lie within a different domain than the real images, we add random blurring during training so that the model does not overfit to the synthetic data's domain (Figure 2C).

We use standard data augmentation techniques as well as synthetically generated images[1, 2, 4] to boost performance during training (see Supplementary Materials Section 1). While we also tried freezing layers, similar to [3], we found that this did not result in a significant performance improvement after implementing data augmentation.

2. Training Details and Software Optimizations

Training was performed using Tensorflow 2.0 using the built-in Keras API. We use the Adam optimizer. The learning schedule is as follows: The base learning rate is set as $1e3$ for the first 10 epochs, and then follows a smooth curve of intermediate learning rates between $1e-3$ and the final learning rate $1e-6$ by linearly interpolating between the log10 values of the stepped learning rate with the following rate milestones–epoch 30: $1e-4$, epoch 50: $1e-5$, epoch 70: $1e-6$. The learning rate then remains set at $1e-6$ for the remainder of training. The training process is terminated within 140 epochs.

With a goal of enabling fast and accessible computation, we performed training on two hardware configura-

tions: The first utilized an Intel Xeon(R) CPU E5-2699A v4 @ 2.40GHz 44, as well as a TITAN RTX/PCIe/SSE2. The second configuration solely utilized an Intel Xeon CLX 8280. In order to optimize training on the Xeon CLX 8280, we tuned the OpenMP environment variables (OMP_NUM_THREADS=14, block_time=0), we used hyperthreading (_logical=1) and we tuned the TF runtime knobs (intra_op=28, inter_op=2). OpenMP related knobs were set via "export" in bash, and inter_op and intra_op knobs were set by modifying the training script. Additionally, we performed a Keras-related optimization. We set the Keras learning phase ("keras_learn_phase") to a constant value of 1 instead of, by default, leaving it as a runtime variable, which would cause dropout and batch normalization steps to go into a different routine in each phase. With these parameter adjustments, we achieve a 24% improvement in software efficiency on the Intel hardware. These optimizations bring training time down to 77 seconds per epoch on the Intel-only configuration as compared to 60 seconds per epoch on the Intel/NVidia configuration. In future work, we will enable cross-socket and cross-node distributed training on the Intel-only configuration, which will significantly boost Xeon-based training performance.

3. Additional Shape model results

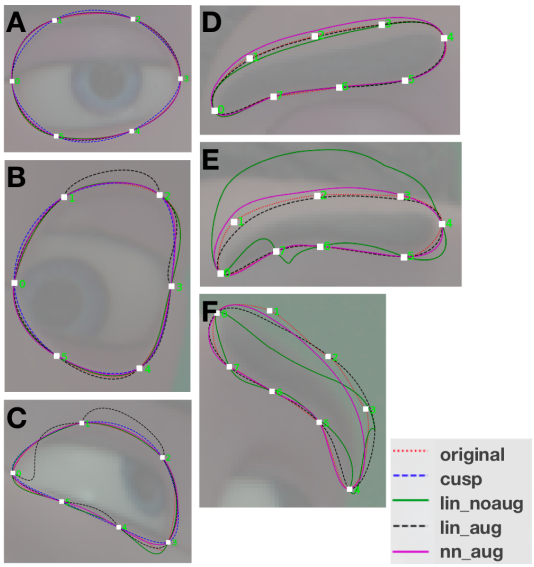

Figure 3. Examples of shape completion based on learned models. (A, B, C) are eyes and (D, E, F) are eyebrows.

4. Additional Landmark Model Results

Figure 4. Experiment 1: Results of the model trained with raw, unaugmented data, 1 shot of ground truth data with augmentations but no synthetic data, synthetic data with augmentations but no ground truth data, and 1 shot of ground truth data with augmentations and synthetic data.

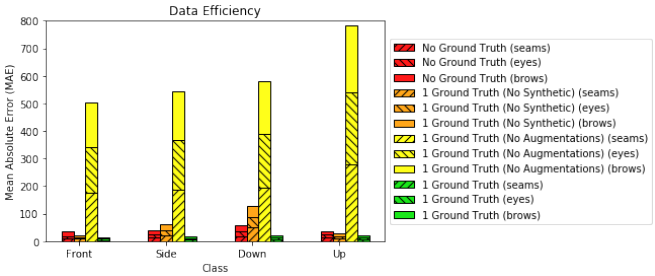

Figure 5. Experiment 1: Numerical analysis of models trained with raw, unaugmented data, 1 shot of ground truth data with augmentations but no synthetic data, synthetic data with augmentations but no ground truth data, and 1 shot of ground truth data with augmentations and synthetic data on four face poses: front, side, down and up. Contributions to the MAE are broken out by shape category: seams (crosshatched to the upper right), eyes (crosshatched to the lower right), and eyebrows (no pattern).

	Seams	Eyes	Brows	Total MAE
Raw 1GT, No Synth	209.75	200.10	192.39	602.22
No GT, Synth	14.17	12.12	16.34	42.64
Aug 1GT, No Synth	22.93	17.94	19.39	60.26
Aug 1GT, Synth	6.45	5.56	5.30	17.31

Table 2. Experiment 1: Landmark prediction performance on test dataset: the numbers in the table are MAE broken down by part resulting from models trained with one shot of raw ground truth data without synthetic data, no ground truth data and augmented synthetic data, one shot of augmented ground truth data not supplemented by synthetic data, and one shot of augmented ground truth data supplemented by synthetic data.

	Seams	Eyes	Brows	Total MAE
No GT	14.17	12.12	16.34	42.64
1 GT	6.45	5.56	5.30	17.31
2 GT	5.08	3.91	4.27	13.26
3 GT	4.73	3.33	3.36	11.42
4 GT	3.34	2.95	3.36	9.65
5 GT	2.45	2.55	2.17	7.17

Table 3. Experiment 2: Landmark prediction performance on test dataset: MAE broken down by part resulting from models trained with 0, 1, 2, 3, 4, 5 augmented ground truth shots and supplemented with synthetic data.

	Seams	Eyes	Brows	TotalMAE
No Synth	3.92	3.25	4.04	11.21
No Back	4.08	3.05	4.50	11.63
No GT, No Back	19.27	11.73	15.21	46.20
No Synth, No Back	4.96	3.57	5.92	14.45
No Scale	2.95	2.56	2.67	8.18

Table 4. Experiment 3: Landmark prediction performance on test dataset: MAE broken down by part resulting from models trained with all shots and augmentations, (1) but without synthetic data, (2) without background and occlusion augmentation, (3) without ground truth data and without background and occlusion augmentation, (4) without synthetic data and without background and occlusion augmentation, and (5) without scaling adjustments to the loss function.

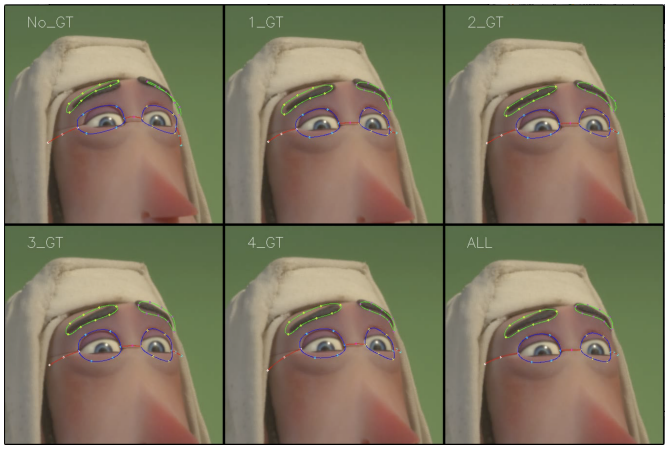

Figure 6. Experiment 2: Results of the model trained with synthetic data, background and occlusion augmentations and scaling, but (A) without ground truth data, (B) with 1 shot of ground truth data, (C) with 2 shots of ground truth data, (D) with 3 shots of ground truth data, (E) with 4 shots of ground truth data, and (F) with all 5 shots of ground truth data.

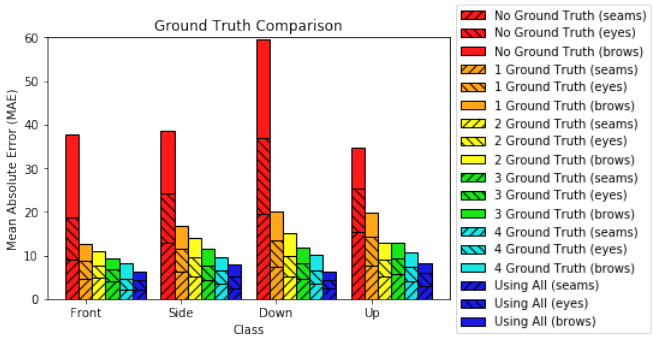

Figure 7. Experiment 2: Numerical analysis of models trained with synthetic data, background and occlusion augmentations and scaling, but (red) without ground truth data (orange) with 1 shot of ground truth data, (yellow) with 2 shots of ground truth data, (green) with 3 shots of ground truth data, (cyan) with 4 shots of ground truth data, and (dark blue) with all 5 shots of ground truth data on four face poses: Front, Side, Down and Up. Contributions to the MAE are broken out by shape category: seams (crosshatched to the upper right), eyes (crosshatched to the lower right), and eyebrows (no pattern).

Figure 8. Experiment 3: Results of the model trained on (A) ground truth and synthetic data with background augmentation, (B) synthetic data with background augmentation, (C) ground truth data with background augmentation, (D) ground truth and synthetic data with no background or occlusion augmentation, (E) synthetic data with no background or occlusion augmentation, F) ground truth data with no background or occlusion augmentation.

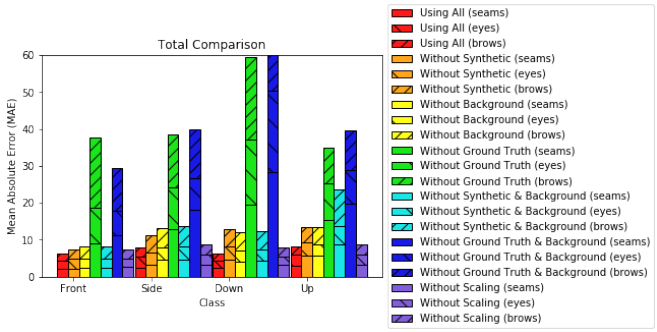

Figure 9. Experiment 3: Numerical analysis of models trained on ground truth and synthetic data with background augmentation (red), without synthetic data (orange), without background and occlusion augmentation (yellow), without ground truth data (green), without synthetic data and without background augmentation (cyan), without ground truth and without background augmentation (dark blue), and without scaling adjustments to the loss function (purple) on four face poses: Front, Side, Down and Up. Contributions to the MAE are broken out by shape category: seams (no pattern), eyes (crosshatched to the lower right), and eyebrows (crosshatched to the upper right).

Figure 10. Experiment 3: Results of the model trained (top) with scale-aware parameters in the loss function, and (bottom) without scale aware parameters in the loss function. Some improvement in the seams, but no significant difference.

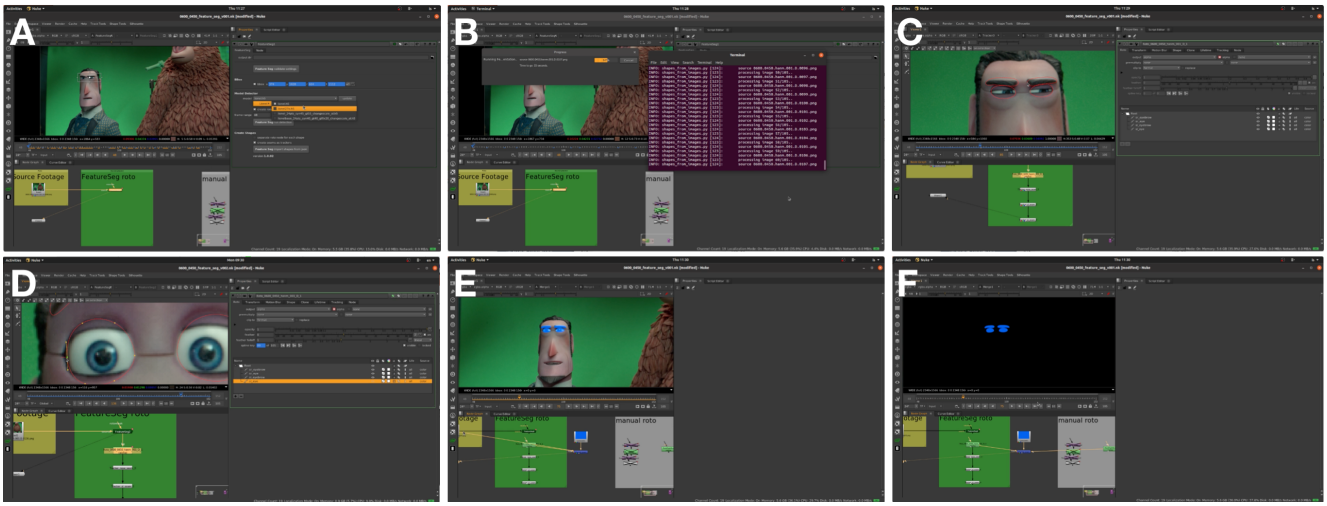

Figure 11. Tool Workflow: (A) A shot is imported into Nuke, a rough crop of the face is placed on the frames and a model is selected for landmark inference. (B) The plug-in calls an external subprocess that runs inferencing on the shot. (C) Landmark points and initial shapes are output across all frames. (D) The points in each shape can be displayed and tweaked, and a revised shape output. (E) The final splines are overlayed on the face. (F) We can also visualize the final roto matte. These finished shapes can be exported to a JSON file and used to further train the model.

5. Artist Workflow and Tool Interaction

Images of our Interactive Tool are reproduced here for convenience (Figure 11).

We chose to allow the artist to feed the input crop to the tool as that is an easy step to perform in Nuke. Rather than focus on automating this step, instead we chose to make the tool robust to rough crops as traditional keypoint models expect a tight crop of the face or object to be fed into the model. We designed this tool with the expectation that multiple artists will use it and we wanted to keep tool-specific training to a minimum. In addition, providing the artist with intermediate outputs that they can visualize and and modify also ensures that the input into the next stage (shape generation) is guaranteed to perform well. Shape predictions rely on good landmark predictions, so we expose this layer of keypoints as tracker nodes. Since rough localization of keypoints is quick to repair manually, we did not automate that step. We have received feedback from VFX post-processing artists that good tracker nodes, can also be utilized for other post-processing tasks in addition to the specific roto task that we set out to solve.

In order for the artists to be able to interact with the trained models, we needed to build an interface within a paradigm that they work with. Nuke is a special effects software platform developed by Foundry, which is used by many special effects artists.

We use a "Read" node to read in a set of images from the file system, these images usually pertain to continuous shots within a scene. Once the frames are in the Nuke buffer, the artist can use a traditional "Roto" node to put a bounding box around the facial features area by creating a rough shape (also known as a "Garbage Matte"). Next the artist will select our custom "FeatureDetect" node, connect it to the read node and then select the required parameters (Figure 11A).

The user can then select the model they want to apply from a drop-down menu and set the frame range for inferencing. The user can choose to create a roto node, export the node to a JSON file or whether to overwrite an existing node. The user may also select their desired output directory.

At this stage, the user may choose to run the model (Figure 11B). The tool reads in the rotoscoped shape coordinates, calculates a bounding box region that fits the input feature size and runs the inferencing portion of the tool. Once this subprocess has completed, a custom "Roto" node is created. The roto node consists of a set of Bézier curves that represent a series of shapes that the model was trained to recognize (Figure 11C). In our example, these shapes include eyes, eyebrows and seams across the nose bridge and from the corners of the eyes to the edge of the face.

The output Bézier curves behave as any other Nuke roto shape where each key and handle can be adjusted to fit the artistic needs of the scene (Figure 11). Figures 11E and F show the finished splines and matte. This set of shapes can also now be exported as a JSON file to be used for training future models.

Our supplemental video shows an artist creating eye and eyebrow holdout masks using our trained model and tool. The artist was able to complete this task for 105 sequential frames within 6.5 minutes. When the same artist creates a holdout mask just for the left eye of the same character

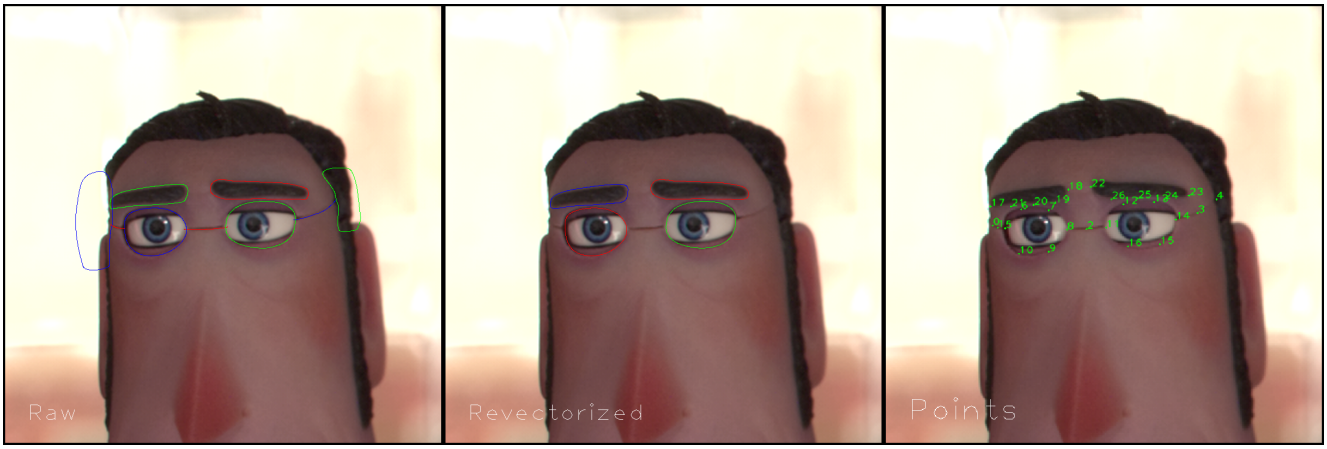

Figure 12. Sample ground truth data with ground truth Bèziers overlayed (”Raw”, left) , with revectorized Bèziers overlayed (”Revectorized”, middle), and with point labels overlayed (”Points”, right).

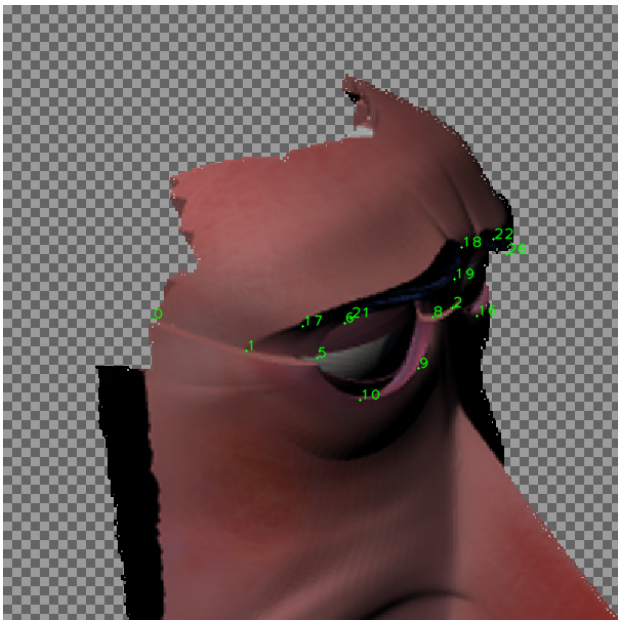

Figure 13. Sample computer generated data with point labels overlayed.

without our tool in Nuke, the task takes 31 minutes to create the left eye mask across the same 105 frames. In this example, the artist's use of our tool allowed her to complete the task roughly 5X faster than with Nuke alone. More detailed experiments studying artist interaction with our tool on multiple characters and across multiple shots will be needed in order to accurately compare our tool's performance and usability with other tools.

6. New Dataset: Professionally Rotoscoped Multi-Shape Fine Feature Systems

We are unable to attach the entire dataset in the supplementary materials for submission. Instead we include a small sample of the new dataset, which we plan to publish in conjunction with this paper.

Training Set: The training set includes 800x800 cropped images from 5 shots of an animated feature film, unaltered ground truth rotoscoping data (Bèzier information) for 9 shapes (left eye, right eye, left eyebrow, right eyebrow, left seam, right seam, nose seam, left sideburn, and right sideburn) in a JSON file, as well as our revectorized annotation data (Bèzier information) for 4 shapes (left eye, right eye, left eyebrow, right eyebrow) and 27 landmark points in an additional JSON file.

The dataset also includes 936 frames of computer Generated synthetic data across 8 facial expressions for the character, with an accompanying JSON file that includes all 27 landmarks points. In addition, there are visualization for each frame with the annotation overlayed on the image. We provide a sample of the film data, with both raw and revectorized annotations and visualization 12 and a sample of the computer generated data, along with its annotation and visualization here.

Test Set: The test set is comprised of 240 full-resolution images taken from an animated feature film, as well as a JSON file that includes the Bèzier information for 7 shapes (left eye, right eye, left eyebrow, right eyebrow, left seam, right seam, nose seam).

7. Acknowledgements

We would like to thank LAIKA Studios for their cooperation and support, specifically: Jeff Stringer, James Pina, Taku Wakisaka, Andrew Gardner, and Veronica Hernandez.

We would also like to thank Dipika Jain, Matthew Pinner, Shanmugam Thangavel and David Rosales.

References

[1] D. Dwibedi, I. Misra, and M. Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1310–1319, Oct 2017.

[2] Georgios Georgakis, Arsalan Mousavian, Alexander C. Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. 2017.

[3] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 682–697, Cham, 2019. Springer International Publishing.

[4] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3848–3856, Oct 2017.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, Sep. 2017.