Pixel Based Complex Background Subtraction for Video Compositing

Sunny He (slhe) and Yannis Karakozis (ick)

January 16, 2018

Project Codebase:
`https://github.com/I-C-Karakozis/Complex-Background-Subtraction-using-Gaussian-Models`

1 Introduction

The task of rotoscoping presents a major bottleneck in modern visual effects pipelines. The term rotoscoping generally refers to the process of cutting out a foreground subject from a segment of video footage in order to be combined with other footage. This is commonly used to create scenes featuring actors in dangerous or inaccessible locations or to integrate live-action shots with computer generated imagery.

Most professional rotoscoping today is still performed manually by skilled professionals. Artists carefully create paint masks at a pixel level or trace out regions of the frame with splines. The outcome of this rotoscoping is an alpha matte, a series of grayscale video frames indicating foreground and background pixels. This process is a expensive and time-consuming part of the visual effects pipeline.

The other commonly used method of performing background subtraction is chroma keying, also colloquially known as "green screening." Using this method, the foreground subject is filled in front of a uniformly colored fixed-color background. A single color model for the solid background color is used to differentiate foreground elements from background [1]. While this method is technically simple to implement, it requires significant setup and set preparation. For large scenes, it may be very difficult or impractical to set up a sufficiently sized screen to isolate the background. In addition, the screen must have a high degree of uniformity and not share any colors in common with the foreground, placing strict restrictions on lighting and filming options.

This project aims to perform automatic background subtraction with the same ease as chroma-keying methods without the need for extensive set preparation. More specifically, we aim to create a video processing pipeline that will create foreground mattes from video footage with a known, static background. We begin by examining a few existing algorithms, then discuss improvements to enhance their performance in real-world environments.

2 Related Work

General background subtraction tasks have been addressed before in a variety of contexts. For instance, variants of the background subtraction task have been the focus of many machine learning segmentation challenges, such as the Common Objects in Context (COCO) challenge. However, leading algorithms such as Mask R-CNN [2] developed for these contests specialize in the 80 or so predefined classes of objects used in the training data set, and so are not as generally applicable for segmenting the arbitrary objects a visual effects project might encounter.

Another class of algorithms attempts to automate segmentation by using "trimaps." Instead of requiring the artist to produce a pixel-perfect matte, the artist paints rough areas of known background and foreground. For instance, a Bayesian approach to trimap based segmentation is described in [3]. While these approaches produce excellent results, they still require an artist to produce an initial rough segmentation. Our goal is to automate the segmentation process entirely, and the need to generate trimaps would have complicated the segmentation and testing pipeline.

In the end, the models we used as the basis for our investigation centered on statistical methods for modeling pixel color distributions. In particular, we examined the pixel-based Mixtures of Gaussian Model initially introduced in [4] and then later refined in [5]. This model utilizes a mixture of Gaussian (MoG) model with M components, such that

$$\hat{p}(\vec{x}|X_T, BG + FG) = \sum_{m=1}^{M} (\hat{\pi}_m \cdot N(\vec{x}; \hat{\vec{\mu}}_m, \hat{\vec{\sigma}}_m^2 I)) \tag{1}$$

BG indicates the event that the pixel is background and FG indicates that the pixel is foreground. T is a reasonable time period expressed such that, at time t, the training dataset $X_T = (x^{(t)}, ..., x^{(t-T)})$. For each new sample, the algorithm updates the training dataset and re-estimates $\hat{p}(\vec{x}|X_T, BG)$.

$\{\hat{\vec{\mu}}_m\}_{i=1}^M$ are the estimates of the means and $\{\hat{\vec{\sigma}}_m^2\}_{i=1}^M$ are the estimates of the variances that describe each Gaussian component. This makes the important assumption that the covariance matrices are diagonal. Finally, the mixing weights $\{\hat{\pi}_m\}_{i=1}^M$ are non-negative and add up to one. $\{\hat{\vec{\mu}}_m\}_{i=1}^M$, $\{\hat{\vec{\sigma}}_m^2\}_{i=1}^M$ and $\{\hat{\pi}_m\}_{i=1}^M$ are updated using an exponential decaying envelope to limit the influence of old data. More details can be found in [4] and [5] for the specific implementation.

This proposed algorithm is an online clustering algorithm. The background pixels are modeled by the clusters with the B most dominant mixing weights $\hat{\pi}_m$, while the clusters with the additional least dominant weights model the foreground pixels. Therefore, the background mixture model is modeled by Equation 2

$$\hat{p}(\vec{x}|X_T, BG) \sim \sum_{m=1}^{B} (\hat{\pi}_m \cdot N(\vec{x}; \hat{\vec{\mu}}_m, \hat{\vec{\sigma}}_m^2 I)) \tag{2}$$

If the components were sorted in descending order with respect to their mixture weights and c_f is the maximum portion of the pixel that can belong to the foreground at any instant, then

$$B = \arg\min_b (\sum_{m=1}^{B} \hat{\pi}_m > (1 - c_f)) \tag{3}$$

3 Per-Pixel Multivariate Gaussian Model (MGM)

Given the scope of our project is limited to static backgrounds with very small variations over time (e.g. small lighting changes), we hypothesize that the mixture of Gaussians model described by [4] may introduce a lot of complexity to update the model over time that would not be as necessary in the context of a known static background.

Therefore, we also constructed an algorithm using a simpler per-pixel Gaussian model constructed from the known background footage. The proposed algorithm then performs a statistical test for every pixel of every frame to determine whether the pixels belong to the foreground or background.

For a video resolution of $L \times W$ pixels, each with k color channels, this corresponds to estimating $L \times W$ multivariate Gaussian distributions with dimensionality k. Since the process for each of the

$L \times W$ pixels is the same, we will only consider the process for a single pixel. Represent the intensity of this pixel with k color channels as the vector $\vec{x} \in R^k$. We assume the user has access to n frames of static background to train the model with, where the pixel takes on intensity values $x_1, \ldots, x_n$. Thus we can model the background color at the location of x with a Multivariate Gaussian Distribution $\vec{x}_{BG} \sim N(\hat{\vec{\mu}}, \hat{\vec{\sigma}}^2 I_k)$. To stay consistent with the MOG model, we assume the channels of each pixel are independent and, thus, all covariances are zero, rendering the covariance matrix diagonal.

The parameters of $\vec{x}_{BG}$ are estimated using the Maximum Likelihood Estimators of the Multivariate Gaussian Distribution.

$$\hat{\vec{\mu}} = \frac{\sum_{i=1}^{n} \vec{x}_i}{n} \tag{4}$$

To be consistent with the channel independence assumption used in both MOG and MGM, the variance of each channel need to be estimated independently by using the Maximum Likelihood Estimator of the simple Gaussian Distribution:

$$\forall j \in \{1 \ldots k\}, \hat{\vec{\sigma}}_j^2 = \frac{\sum_{i=1}^{n} (v_{i,j} - \hat{\mu}_j)^2}{n} \tag{5}$$

Where $\hat{\mu}_j$ and $\hat{\sigma}_j^{\,2}$ are the j-th color channel of $\hat{\vec{\mu}}$ and $\hat{\vec{\sigma}}$ respectively.

Since we make no assumption on the color or shape of the foreground object, one cannot model any prior distribution for the foreground channels of each pixel ahead of time. Therefore, generative binary classification is not directly possible.

By defining the foreground pixels as pixels that are not background, one can use hypothesis testing on the modeled population $\vec{x}_{BG}$. The null hypothesis states that an observed color x' is well-modeled by $\vec{x}_{BG}$ and, thus, belongs to the background, while the alternative hypothesis states that the pixel is a foreground pixel. For the pixel in question, given a new sample x' to be classified, the test-statistic T is given by:

$$T = (x' - \hat{\vec{\mu}})^T \cdot (\hat{\vec{\sigma}}^{\,2} I_k)^{-1} \cdot (x' - \hat{\vec{\mu}}) \tag{6}$$

The null hypothesis now states that T is distributed under Chi-Squared while the alternative hypothesis states that T is distributed under non-central Chi-Squared. By determining the degrees of freedom from the dimensionality k of the channels vector (DoF $= k$, as the channel components are assumed independent of one another), picking the right confidence level and then using the appropriate threshold values for the test-statistic [6], one can determine whether an observation belongs to the foreground or the background for a given pixel. This process is done in parallel for each pixel in the video frame.

4 Background Subtraction Pipeline

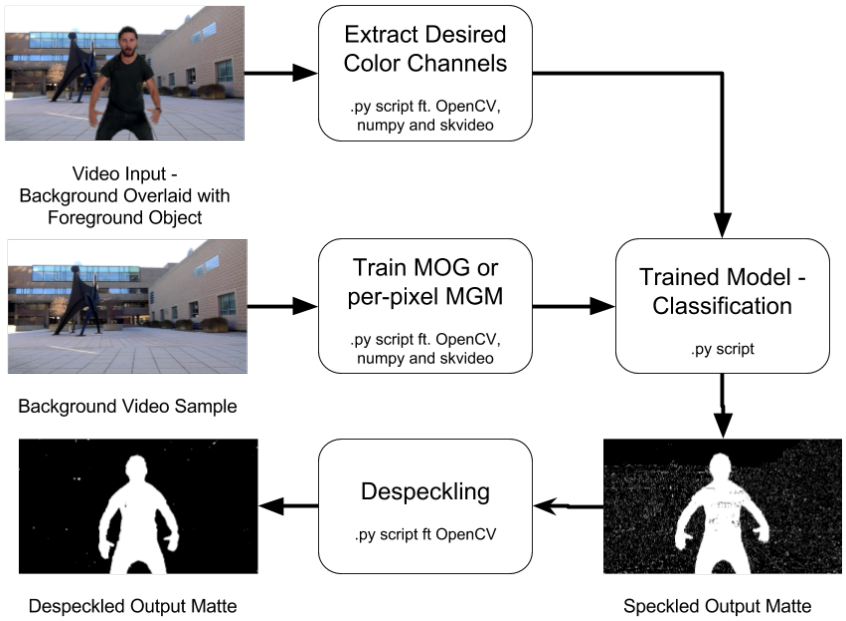

Figure 1: Background Subtraction Pipeline

This section describes the pipeline developed to perform background subtraction on a video sequence. The pipeline has been modularized so that it is compatible with both algorithms evaluated. Algorithms were implemented in Python, leveraging the scikit-video [7], NumPy [8], and OpenCV [9] libraries. An illustration of the pipeline is shown in Figure 1.

The pipeline is initially fed a video sequence of pure background. The pipeline then trains the selected algorithm (MOG or MGM) on the background video sequence. For MOG, this implies creating an OpenCV MOGSubtractor and running it through the video to populate its training dataset with purely background frames.

The pipeline is then given the video sequence of the background overlaid with the foreground objects. At each frame, the pipeline extracts the desired color channels to match the target color space. More details on the choice of color spaces and channels used can be found in section 5. The resulting frame is fed into the trained model, which classifies every single pixel as foreground or background and outputs a matte, typically with a good amount of speckles and small holes on the foreground object. The speckled matte is then despeckled by having a median filter of fixed size be applied multiple times on it. Thus, the final matte is produced.

The pipeline code for the MGM model is found in `gmm_mov.py` and for the MOG model is found in `mog_mov.py`. All code implementation was performed in Python. The MGM model was developed from scratch with the help of NumPy. For the MOG model, an OpenCV implementation was used. The Despeckling median filter implementation was also lent from OpenCV, while OpenCV assisted in the color channel extraction as well. The videos were processed with the help of scikit-video.

5 Testing Methodology

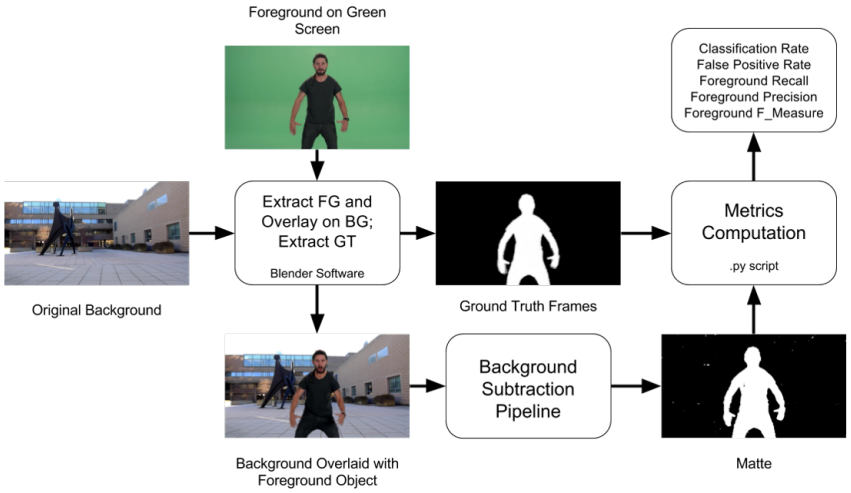

Figure 2: Testing Pipeline

To avoid having to manually label thousands of frames of test footage for evaluating our algorithms, we generated test videos using the Blender open source video effects software. Three foreground clips were used, one from the famous "Just Do It" video [10] to represent a human subject, and two recorded in a studio with a mix of slow moving and fast moving subjects. These foreground clips were segmented using standard chroma-key processes, then overlaid onto one of two backgrounds. The two backgrounds used were an indoor scene and an outdoor scene captured on the Princeton campus.

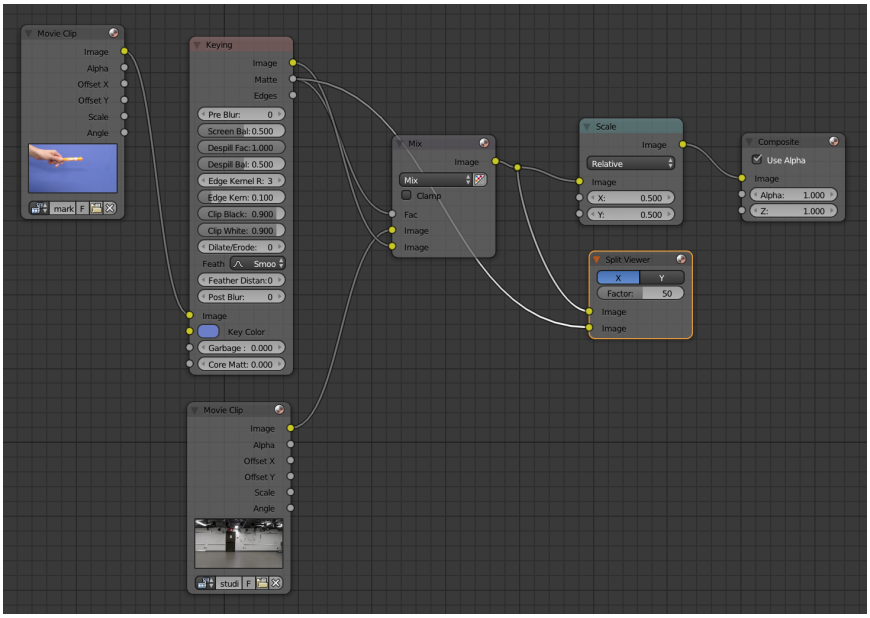

Figure 3: Blender Testing Footage Generation

The chroma-key process's output mattes were used as ground truth data for evaluating our segmentation pipeline. Custom Python scripts utilizing Scikit-video and NumPy compared the output segmentation produced by our algorithms with the ground truth mattes and calculated key performance statistics.

6 Algorithmic Optimization

The initial output produced by the algorithm was promising, but two notable issues remained. First, on some testing sequences there were frames where the camera autofocus caused the brightness of the entire frame to change. While this made a very small perceptual difference, it caused the segmentation algorithms to fail spectacularly.

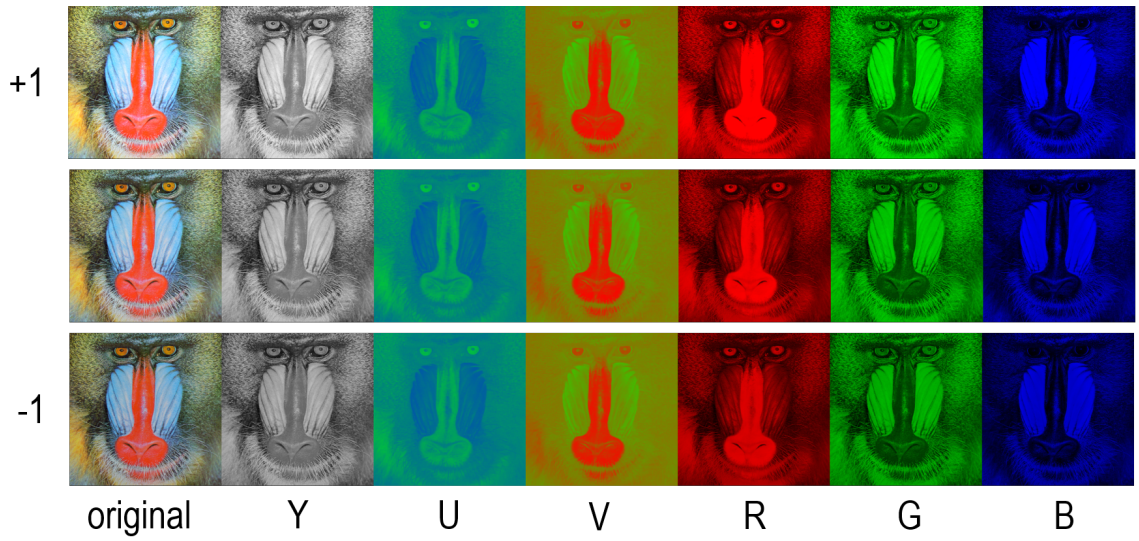

Figure 4: Color Space Brightness Sensitivity

To solve this sensitivity to brightness variations, we explored different color spaces instead of the default RGB. In particular we explored the YUV color space, which explicitly breaks out perceptual brightness into the Y or luma channel. Since the color content remained isolated in the two chroma channels (U and V), it was hypothesized that the simple color models being used would better be able to distinguish a change in brightness from a change in actual color content. In contrast, using RGB color space all three channels would vary in unpredictable ways under lighting changes.

The other major issue that appeared early on in testing was the existence of noise in the output mattes. Likely the result of compression artifacts or sensor noise, these manifested themselves as small specks a few pixels wide randomly strewn across the output matte. To combat this noise, a variety of despeckling approaches were considered including Gaussian blur and edge detection. Ultimately, a simple median filter appeared to produce the best results with minimal computational cost. The impact of various parameters including filter size and iterated application of the filter were examined.

7 Performance Evaluation

Performance Evaluation is comprised of two parts: Color Space and Median Filter Configuration. In the former, the metrics measured in each video are averaged to identify which color space and channel configuration have the best performance for each algorithm. The only exception is the Foreground (FG) F_Measure which is recomputed for each configuration from the average Recall and Precision across all videos. The F_Measure is the harmonic mean of the FG Precision and FG Recall and is a measure of the algorithm's accuracy in identifying specifically Foreground pixels successfully.

In both parts, the evaluation metrics used are the following:

1. Classification Accuracy to evaluate the classification methods used in overall
2. False Positive Rate, to identify how often the algorithms selected background elements for foreground, which quantifies the extent to which the Binary Masks produced are distorted by background noise and misclassified objects.
3. Foreground Recall and Precision, to be able to compute the Foreground F_Measure, a metric for meaningful comparison of the various algorithms and configurations in their ability to correctly identify the foreground pixels.

Therefore, these metrics adequately capture the two core goals of the task: A) Correctly identify foreground pixels and B) Minimize misclassificiation of background noise and objects as foreground.

Color Space oriented performance evaluation is undertaken to identify the best-performing color space and color channels for the background segmentation task. The color spaces considered are YUV, UV, RGB and RGB whose luminance has been set to a constant level for all pixels (RGB-No luma). In both methodologies, the YUV color space seems to outperform all other alternatives in the two key metrics: Classification Accuracy, by a small margin of 0.5%-1.0%, and FG F_Measure, by a wider margin of 2%-7%, as evident from the first table below. It is important to note that the lowest False Positive rate (False FG on the first table), measuring percentage misclassified of BG pixels as FG, is attained by the configuration that completely isolates out luminance by considering only the U and V channels in the YUV color space. Finally, the configuration fixing the luminance value for all pixels, seemed to be completely outperformed in all metrics.

All metrics are means over all frames of the video		Average Performance over All 6 Videos				
		Accuracy	False FG	FG Recall	FG Precision	FG F-Score
	Color Space and Channels Used					
MOG	RGB	97.46%	0.58%	85.15%	91.73%	87.38%
	RGB - No LUMA	95.90%	3.95%	80.66%	83.82%	81.55%
	YUV	97.86%	1.10%	89.05%	90.61%	89.16%
	UV	95.93%	0.41%	75.62%	92.05%	82.07%
Per Pixel MGM	RGB	96.18%	4.16%	98.04%	78.79%	86.23%
	RGB - No LUMA	96.83%	0.65%	83.90%	89.85%	86.56%
	YUV	97.51%	2.92%	97.61%	83.08%	89.24%
	UV	96.79%	0.41%	82.36%	92.36%	86.87%

Figure 5: Effect of Color Spaces on Background Subtraction Performance

Given despeckling is introduced into the pipeline in the form of median filtering, the median filter configuration is being investigated. The goal is to identify if despeckling improves performance

at all and, if it does, what the best configuration is. In both algorithms, introducing despeckling significantly improved results, as evidence from the very sizable improvement we see across all three metrics with the introduction of median filtering, as evident from Figures 6, 7, and 8. On the contrary, there are very minor improvements in performance as the size of the median filter increases, seeming to perform the best at size 5 by a very small of margin in all three metrics (see Figures 6, 7, and 8). At the same time, sizable improvements are identified across all metrics as the number of iteration one applies the median filter increases. However, increasing the number of iterations above the range 10-20 not only leads to gradually deteriorating performance, but also significantly increases the computational time needed to perform the matte extraction. This is because the magnitude of the operations increases proportionally to $O(N^2)$, where N is the number of pixels on each side of the frames.

	Median Filter Application Iterations - Accuracy					
Filter Size	Zivkovic MOG - YUV			Per Pixel MGM - YUV		
	1	10	100	1	10	100
0 (no filtering)	90.57%			97.46%		
3	95.78%	96.22%	96.23%	99.27%	99.53%	99.53%
5	96.19%	96.19%	96.04%	98.30%	99.61%	99.60%
7	96.19%	96.04%	95.62%	99.57%	99.57%	99.20%

Figure 6: Effect of Median Filter Parameters on Classification Accuracy

	Median Filter Application Iterations - FG F_Measure					
Filter Size	Zivkovic MOG - YUV			Per Pixel MGM - YUV		
	1	10	100	1	10	100
0 (no filtering)	66.13%			90.89%		
3	81.70%	83.42%	83.43%	97.23%	98.19%	98.21%
5	83.29%	83.27%	82.57%	92.29%	98.52%	98.51%
7	83.28%	82.60%	80.85%	98.36%	98.39%	97.07%

Figure 7: Effect of Median Filter Parameters on Foreground F_Measure

	Median Filter Application Iterations - False Positive Rate					
Filter Size	Zivkovic MOG - YUV			Per Pixel MGM - YUV		
	1	10	100	1	10	100
0 (no filtering)	6.81%			2.62%		
3	0.83%	0.32%	0.31%	0.63%	0.39%	0.38%
5	0.36%	0.32%	0.36%	0.35%	0.32%	0.34%
7	0.32%	0.37%	0.57%	0.34%	0.34%	0.66%

Figure 8: Effect of Median Filter Parameters on False Positive Rate

8 Results and Discussion

Based on the quantitative analysis performed, the YUV color space seems to be the best color space configuration to perform background segmentation with Gaussian-based models. It achieves 97.86% and 97.61% classification accuracy, and 89.16% and 89.24% FG F_Measure for MOG and MGM respectively, both pairs being the highest compared to the other color spaces used. Therefore, it seems that isolating out luminance, although significantly minimizing the false positive error rate, it does not give rise to the best performance overall. This is most probably because luminance also encodes useful information in discerning background and foreground pixels, which is consistent with the state of the art Mask R-CNN approach described in [2], which does not exclude any color channels.

The explanation of why the YUV space works the best compared to RGB alternatives stems from the RGB equations themselves:

$$R = Y + 1.140V$$
$$G = Y - 0.395U - 0.581V$$
$$B = Y + 2.032U$$

All three color channels depend on the luminance channel Y. Therefore, these three channels vary in the same direction as the image brightness varies, which implies a high covariance between the three color channels. This is also evident in Figure 4, which indicates how the RGB channels behave with lighting variations. On the contrary, the U and V channels do not vary as color brightness changes because they only encode chroma; all the brightness changes are captured by the Y channel. Therefore, there is less correlation due to lighting between the Y, U and V channels compared to the RGB color space. This is more consistent with our independence assumption that the covariance between any two pixel channels is zero, because of the assumed independence of the pixel channels. Since YUV better represents this assumption, it makes sense why YUV performs moderately better than the rest of the color space configurations.

Despeckling also improved performance substantially, as classification accuracy increased by 2%-5%, the false positive rate dropped by 2% - 6.5% and the FG F_Measure improved by 7%-15% with the introduction of a single iteration of a median filter. The median filter effectively removes isolated and tiny neighborhoods of speckles while filling up gaps in foreground objects. The choice of the filter does not matter as much, and 3x3, 5x5 and 7x7 median filters performed equivalently (see Figures 6, 7, and 8). The number of times that the filter was applied seemed to matter though, as applying it a single time did not deal with larger neighborhoods of speckles, while applying it too many times was eroding edges and corners, leading to the classification of a lot of background pixels close to the object as foreground. Therefore, applying the median filter approximately 10 times seemed to give out the best results both qualitatively and quantitatively.

According to Figure 6, MOG and MGM have comparable performance across all videos given, despite each performing better than the other on particular video sequences. This is because their average performance on each of the three metrics used was equivalent. Therefore, it seems that for complex backgrounds with multiple or single objects performing various types of motions in the foreground, a running Gaussian model performs equally well as a pre-trained Gaussian model.

9 Conclusion and Future Work

In this project we have successfully created a system for performing background subtraction in video footage with a known static background. Building off of existing global and per-pixel approaches to pixel color modeling, we examined enhancements in the form of color space conversion and median filtering to improve the robustness of these background subtraction models to lighting variation and

noise. Over a diverse dataset of composited foreground and backgrounds, we achieved average pixel classification accuracies of over 97% and F-scores of over 99% with near real-time performance.

However, there remains room for further exploration. The fundamental assumption of a static background is widely applicable for studio-shot footage, but it would be highly beneficial to account for small camera motion. Assuming the camera doesn't make sudden movements, it may be possible to use optical flow to track simple camera pans or deviations and update the per-pixel color models accordingly.

Furthermore, the exploration of different color spaces could be expanded to include other mappings such as the Lab color space. While the YUV color space improved performance for most subjects, we found that it would struggle with certain subjects with desaturated gray colors.

In the end, rotoscoping represents an important element of the visual effects pipeline, and improvements to segmentation algorithms stand to greatly improve the flexibility and accessibility of advanced compositing techniques.

References

[1] A. R. Smith and J. F. Blinn, "Blue screen matting," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96, New York, NY, USA: ACM, 1996, pp. 259–268, ISBN: 0-89791-746-4. DOI: 10.1145/237170.237263. [Online]. Available: http://doi.acm.org/10.1145/237170.237263.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *arXiv preprint arXiv:1703.06870*, 2017.

[3] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, "Video matting of complex scenes," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '02, San Antonio, Texas: ACM, 2002, pp. 243–248, ISBN: 1-58113-521-1. DOI: 10.1145/566570.566572. [Online]. Available: http://doi.acm.org/10.1145/566570.566572.

[4] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, Aug. 2004, 28–31 Vol.2. DOI: 10.1109/ICPR.2004.1333992.

[5] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.11.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865505003521.

[6] M. Software. (2018). Chi square table, [Online]. Available: https://www.medcalc.org/manual/chi-square-table.php (visited on 2018).

[7] scikit-video. (2018). Scikit-video, [Online]. Available: http://www.scikit-video.org/stable/.

[8] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011. DOI: 10.1109/MCSE.2011.37. eprint: http://aip.scitation.org/doi/pdf/10.1109/MCSE.2011.37. [Online]. Available: http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37.

[9] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[10] S. LaBeouf, *Introductions*, 2015.