Effects of Image Filters on Video Similarity Search

Ruochen Liu
Department of Computer Science
Columbia University
rl3160@columbia.edu

Directed Research Report - Fall 2020
Advisor: Prof. John Kender

January 3, 2021

Abstract

Videos with approximate topics can contain similar frames at different timestamps. To ease frame alignment, one idea is to improve frame quality using image enhancement techniques such as denoising and sharpening. Through experiments, Gaussian blur shows improvements in the similarity search.

1 Introduction

Video similarity search is a computer vision task that aligns similar frames among different videos. This task can help compare video preferences of differing affinity groups over the identical topics. To improve the search task, this paper studies the effects of three image enhancement techniques on different videos: Gaussian blur, unsharp masking, and grayscale. Intuitively, Gaussian blur removes noises or "incorrect information" while unsharp masking emphasizes edges or "correct information". Grayscale removes colors as "noise". The goal is to see whether or not these filters make similar frames more similar and distinct frames more distinct.

This paper will discuss the experiment procedures and results. The source code is available here.

2 Methods

2.1 Data Selection

`YouTube` is the primary video source and the video type is limited to news. Eight videos are selected and categorized into four groups:

- Noisy: on-scene footage with high level of noise
- Still: interviews with still backgrounds
- Dynamic: on-scene footage involving several movements
- Dark: dark backgrounds, high foreground-background contrast

Videos are then separated into frames using `ffmpeg` with one frame per second:

```
ffmpeg -i <input_video> -r 1/1 <output_directory>/%03d.png
```

Some frames are arbitrarily removed due to lack of information.

2.2 Noise Model

Gaussian noise model is an accurate approximation of noise in real world scenarios [1].

$$P(g) = \sqrt{\frac{1}{2\pi\sigma^2}} e^{-\frac{(g-\mu)^2}{2\sigma^2}}$$

Where $g \in [0, 255]$ is the gray value, μ is the mean, and σ is the standard deviation. A Gaussian noise model with $\mu = 0$ and $\sigma = 0.1$ is assumed across all videos.

2.3 Image Filters

Three image filters are applied to each frame to study their effects on similarity search: Gaussian blur, unsharp masking, and grayscale.

2.3.1 Gaussian Blur

2D Gaussian blur is an image-blurring filter applying the product of two Gaussian functions [8].

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Where x and y are respectively the horizontal and vertical distances from the central pixel/origin of the filter, and σ is the standard deviation. The Python Imaging Library (PIL) does not implement the Gaussian blur using the formula above. It instead implements an optimized version using extended box filter [3].

```
PIL.ImageFilter.GaussianBlur(radius=2)
```

Where radius is the size of the filter in one direction. The relationship between the radius and σ is [4, 5]:

$$\sigma^2 = \frac{r^2}{3}$$

The choice of radius/σ is arbitrary. For simplicity, the radius is set to 2 (default value).

2.3.2 Unsharp Masking

Unsharp masking is an image sharpening technique to emphasize edges. For each pixel:

$$\text{sharpened} = \begin{cases} \text{original} + (\text{original} - \text{blurred}) \times \text{percent} & \text{if } |\text{original} - \text{blurred}| > \text{threshold} \\ \text{original} & \text{otherwise} \end{cases}$$

PIL uses Gaussian blur in its Unsharp masking implementation [7].

```
PIL.ImageFilter.UnsharpMask(radius=2, percent=150, threshold=3)
```

The radius parameter controls the filter size of Gaussian blur. The percent determines the strength of the sharpening effect. The threshold specifies the minimal intensity difference ($[0, 255]$) required to perform sharpening. This third parameter is necessary to avoid sharpening noise [2]. For simplicity, the radius, percent, and threshold are set to 2, 150, and 3 respectively (default values).

2.3.3 Grayscale

Grayscale removes chromatic information from original images. Each resulting pixel contains the intensity information of the original one, ranging from black to white. PIL applies the `ITU-R 601-2 luma transform` to convert an image from RGB to grayscale [6]:

```
PIL.Image.convert(mode='L', matrix=None, dither=None, palette=0, colors=256)
```

$$\mathrm{L} = 0.299\ \mathrm{R} + 0.587\ \mathrm{G} + 0.114\ \mathrm{B}$$

2.4 Frame Comparison

2.4.1 Compression

A naïve approach to compare an image pair to pixel-to-pixel comparison. This method does not consider image features and may be computational expensive.

The Oxford Visual Geometry Group (VGG) models are very deep convolutional neural network classification models trained on the `ImageNet` database with 1000 categories. Pre-trained VGG models are handy tools to extract features and compress each image into a vector with 1000 values. In this paper, each frame is first pre-processed and then fed into a pre-trained VGG-19 model provided by the `PyTorch` library. The pre-processing steps are [9]:

1. Resizing each frame into a shape of $256 \times 256 \times 3$
2. Crop each frame at center into a shape of $224 \times 224 \times 3$
3. Convert the value of each pixel from $[0, 255]$ to $[0, 1]$ and reshape each frame into a shape of $3 \times 224 \times 224$
4. Normalize the RGB channels with the means $0.485, 0.456, 0.406$ and the standard deviations $0.229, 0.224, 0, 225$ respectively

Note that an image in grayscale should be converted into the RGB mode to have the correct input shape:

```
PIL.Image.convert(mode='RGB', matrix=None, dither=None, palette=0, colors=256)
```

$$\mathrm{R} = \mathrm{G} = \mathrm{B} = \mathrm{L}$$

2.4.2 Distance Function

Cosine distance measures the angular distance between two non-zero vectors ignoring magnitude.

$$\cos(\theta) = \frac{x \cdot y}{\|x\|\|y\|}$$

Where x and y are frame vectors and θ is the angle between them. $\cos(0) = 1$ indicates identical frames, while $\cos(\frac{\pi}{2}) = 0$ indicates distinct ones.

Cosine distance is considered as a better metric than the euclidean distance for frame comparison as the latter may suffer from large magnitude difference. Similar frames may vary a lot in magnitudes, but they should point to close directions.

3 Experiments

The experiments randomly pick 400 frames to form 200 pairs. Each frame has a 55% chance to pair with its neighbor on the right (if exists) and a 45% chance to pair with a random image. The former scenario gets a slightly higher percent since a neighbor may be distinct or absent.

For each image pair, apply the three image filters in the following manner:

- Original
- Gaussian blur
- Unsharp masking
- Grayscale
- Gaussian blur on Grayscale
- Unsharp masking on Grayscale

Compute the cosine similarity for each group mentioned above. The image pairs are then manually labelled as similar (1) or distinct (0).

There are 90 similar pairs and 110 distinct ones in the results. Below are some statistics, where deltas are relevant to the original pairs:

Similar Pairs	**Mean**	**Std**	**ΔMean**	**ΔStd**
Original	0.93847415	0.09640524	0%	0%
Gaussian Blurred	0.93981758	0.08502947	+0.14315%	-11.799947%
Unsharp Masked	0.93510603	0.10447835	-0.358894%	+8.374141%
Grayscale	0.93949947	0.09918524	+0.109254%	+2.883668%
Grayscale & Gaussian Blurred	0.93795136	0.089655	-0.055706%	-7.001945%
Grayscale & Unsharp Masked	0.93684464	0.10553472	-0.173634%	+9.469907%

Distinct Pairs	**Mean**	**Std**	**ΔMean**	**ΔStd**
Original	0.38322544	0.18785263	0%	0%
Gaussian Blurred	0.37103477	0.19482157	-3.181072%	+3.709796%
Unsharp Masked	0.40005726	0.18413773	+4.392144%	-1.977556%
Grayscale	0.45742863	0.19858088	+19.362803%	+5.710993%
Grayscale & Gaussian Blurred	0.46720121	0.19359079	+21.91289%	+3.054611%
Grayscale & Unsharp Masked	0.47475326	0.20114874	+23.883545%	+7.077949%

Observing the statistics above, Gaussian blur on average decreases the angular distances ($\cos(\theta) \uparrow$, $\theta \downarrow$) between similar pairs and increases those between distinct ones ($\cos(\theta) \downarrow$, $\theta \uparrow$). Unsharp masking offers the opposite effect. Grayscale in general decreases the angular distances, particularly when the frames are distinct.

4 Conclusion

The goal of the experiments is to evaluate effects of different image filters on the frame similarity search task. An ideal image filter should make similar frames more identical, and distinct frames more different. As the experiment results show, Gaussian blur achieves these requirements. In comparison, unsharp masking and grayscale may not be helpful to similarity search task. Feature works will focus on more rigorous experiments on Gaussian blur, possibly with more video categories and different parameter combinations.

In addition to Gaussian filter, object detection may also be a viable enhancement to explore. This technique can help avoid distractions such as news logo and concentrate on the main objects on the scene.

References

[1] Ajay Kumar Boyat and Brijendra Kumar Joshi. "A Review Paper: Noise Models in Digital Image Processing". In: *Signal and Image Processing : An International Journal (SIPIJ)* 6.2 (2015), p. 64. DOI: https://arxiv.org/ftp/arxiv/papers/1505/1505.03489.pdf.

[2] Cambridge in Colour. *GUIDE TO IMAGE SHARPENING*. URL: https://www.cambridgeincolour.com/tutorials/image-sharpening.htm.

[3] Andrés Bruhn Pascal Gwosdek Sven Grewenig and Joachim Weickert. "Theoretical Foundations of Gaussian Convolution by Extended Box Filtering". In: *Mathematical Image Analysis Group* (2011), p. 8. DOI: https://www.mia.uni-saarland.de/Publications/gwosdek-ssvm11.pdf.

[4] Pillow. *_imaging.c*. URL: https://github.com/python-pillow/Pillow/blob/master/src/%5C_imaging.c.

[5] Pillow. *BoxBlur.c*. URL: https://github.com/python-pillow/Pillow/blob/master/src/libImaging/BoxBlur.c.

[6] Pillow. *Image.convert*. URL: https://pillow.readthedocs.io/en/stable/reference/Image.html.

[7] Pillow. *UnsharpMask.c*. URL: https://github.com/python-pillow/Pillow/blob/master/src/libImaging/UnsharpMask.c.

[8] Linda Shapiro and George Stockman. *Computer Vision*. Prentice Hall, 2000, p. 168. DOI: http://nana.lecturer.pens.ac.id/index_files/referensi/computer_vision/Computer\%20Vision.pdf.

[9] PyTorch Team. *VGG-NETS*. URL: https://pytorch.org/hub/pytorch_vision_vgg/.