Motivation

Low contrast and noise remains a barrier to visually pleasing videos in low light conditions. Recording appealing video at concerts, social gatherings, and in security monitoring situations is still an unsolved problem, with many groups searching for a solution. Our target application for this research is finding a better software solution for mobile low-light video, particularly in concert venues.

Although mobile image processing products such as Instagram create nice-looking photos and videos, particularly in concert venues.

Dealing with Low-Light

Low Light Video Processing on Mobile Devices

Patrick Martinchek¹, Nobie Redmon², Imran Thobani³

¹Graduate School of Business, Stanford University
²Department of Applied Physics, Stanford University
³Department of Electrical Engineering, Stanford University

GPU-Accelerated Computation

Because histogram equalization is a pixel-wise operation, we can accelerate these calculations using the onboard iPhone 5s GPU. About 75% of our mobile solution executes on the GPU, which is fast enough for our algorithms to operate at video frame rates with an iPhone 5s device.

To make our development faster, we used an iOS framework called GPUImage. The GPUImage library uses OpenGL ES 2.0 shaders to perform image and video processing significantly faster than CPU-based routines, and the framework provides a simple, lightweight Objective-C interface into the complex OpenGL ES API. With this interface, we are able to customize our own shader programs using the Graphic Library Shader Language (GLSL), which controls how each pixel value is determined, calculated, and displayed on the GPU.

Experimental Results

Conventional denoising algorithms temporally average individual pixels. However for videos with motion it is important to neglect pixels in motion to avoid motion artifacts.

A Hidden Markov Model: x = hidden variable, z = observed variable

For each pixel p at frame k, a 3-vector a is chosen to be a Bernoulli variable representing whether the current RGB values are caused by significant motion occurring at the pixel as opposed to random noise. After hand tuning the 2x2 transition matrix A and initial probability P(x=1) to reasonable values, we are able to efficiently perform HMM filtering on any video to compute m = P(x=1 | z, a) for each pixel and each frame k. m represents the model’s belief that motion is occurring at the pixel at time k. We then set the pixel’s RGB values at frame k, a 3-vector a, to be a weighted average of its current RGB values p and the running average m of its RGB values as follows:

\[ a_p = a_{p-1} + \frac{1 - \beta}{1 + \beta} \left( \frac{1}{\beta} \right) m_{(z, a)} \]

Incorporating m into the calculation of running average a effectively resets the running average to the current RGB values whenever the model believes that motion occurred at a pixel.

Future Work

Iphone Implementation

- Parallelize complete algorithm for use on GPU
- Implement Video Stabilization
- Probabilistic Denoising
- Train HMM Transition Matrix
- Implement Online Learning
- Model direction of motion
- Static Denoising

References


G. Toderici, J. Yagnik. Automatic, efficient, temporally-coherent video enhancement for large scale applications. ACM Multimedia 2009: 609-612