Background Generation Algorithms:
Comparing Teknomo-Fernandez, Monte-Carlo, and Median Filter

Alejandro Castaneda
Renee Wu

Portland State University
Computer Vision 410
Feng Liu

Abstract

Background Image Generation is an advancing topic in Computer Vision. It allows for background subtraction to be used, thus isolating important objects such as pedestrians, vehicles, text, or other objects commonly in the foreground. Pre-existing algorithms are often either too computationally expensive or unable to maintain a desired background. The Teknomo-Fernandez is a simple, yet very powerful, algorithm which utilizes three images and performs boolean operations with layering to generate a 'background image.' We additionally compared the performance between TF algorithm and its Monte-Carlo inspired variance, and the simple Median filter.

Introduction

Background Subtraction is a technique used in detection and tracking of moving objects in a video sequence. In some lucky scenarios, we are provided a background image, known as the ‘ground truth’. In most realistic scenarios, however, there is the need to identify the background ourselves. This is a non-trivial challenge, simply due to the complexity of environments. We find that occlusions, shadows, moving cameras and objects are just so highly unpredictable, and it becomes difficult to find what the actual background may be when there are so many moving elements.

There are many algorithms out there that have a high precision with detecting backgrounds. However, a majority of them will use some form of object recognition, gradient filters, or filters which may all lead to a higher computational complexity. When some performance is needed in real time, that computational complexity may become a bottleneck to our functional use, and it simply may not be realistic in all scenarios.

Now, our goal was to implement the Teknomo Fernandez algorithm, which is a very efficient and inexpensive algorithm to generate the background. It runs on the premise of this one main idea: the background occurs in the majority of the shots. As we finished the first algorithm, we decided we wanted to try some more variance to it. What would happen when we had more than just the three images in our sample pool, and followed a result that showed the ‘Modal bit’, or the bit that occurs most in each pixel from each frame? After comparison, we found out that TF algorithm is an extremely efficient method to subtract background with limited time.

Methods

Teknomo-Fernandez is a uniquely powerful algorithm, yet it is very picky in where it can work. The idea is that it uses at least three frames, and using boolean operations, it will generate a background image with what occurs 'most'. With a simply effective design, it comes at the cost of many assumptions. It assumes that the camera will be stationary, the background is shown in the majority of the video, and there may not be any lighting changes to the image throughout the sequence. With these assumptions, it narrows the scope of what can be generated, but in that narrow scope this algorithm excels.

Having three images: x_1, x_2, x_3, we are able to formulate the background image using the following equation.

$$B = x_3(x_1 \oplus x_2) + x_1 x_2$$

When comparing this with the Monte Carlo algorithm, we increase the sampling size of the images, and compare the 'modal bit'. The idea behind this is that instead of three images, we increase the pool of images that vote on which bit should be turned on. By using this and layering, we're able to then have a 'tournament' style approach in which the bits that are favored by the images will stay on.

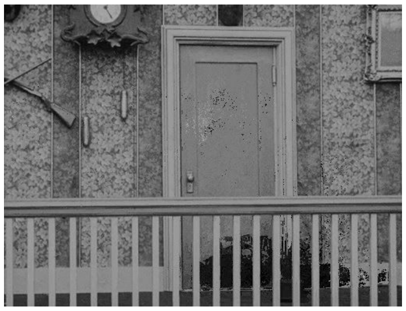
Our third algorithm to compare these to is the median filter. The idea behind the median filter is hidden in the name: take all pixel values of the uniquely, randomly selected frames from the data set, and find the median of those frames.

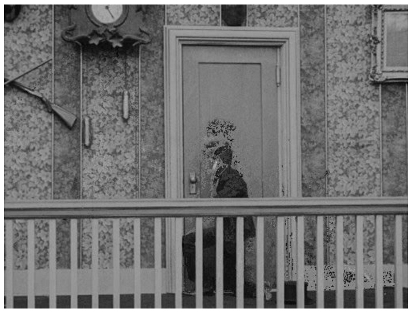
The results are commonly compared to a 'ground truth', but due to the selected datasets, there is no ground truth to compare to. Rather, we compare the results of each image to the others that were produced. Specifically, we're looking for noise, distortion, or inaccuracies in the algorithms. Additionally, we care about the speed a little, considering these algorithms may be needed in real-time in some cases. We measure them but we state this with the disclaimer that the algorithms were not created to be fully optimized, nor are we performance experts. Simply, we took whatever trivial approaches we could to make the algorithms run as quickly as possible.

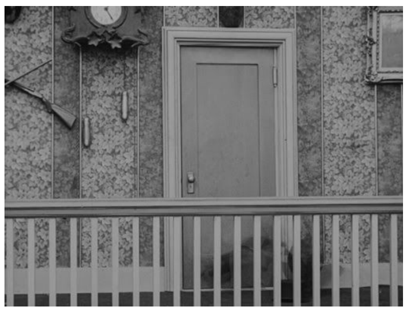
Data

The Charlie Data Set.

The Charlie data set is an animated gif from a Charlie Chaplin movie. This data set features 1 .gif file dissected into 27 image frames. In the original video, Charlie simply is knocked up by a 'grandfather' clock, and falls to the floor. This data set works well to estimate whether the door behind Charlie is shown, and is able to expose the weakness of the algorithm as when he falls, his lower body tends to hover around the same area that he was standing on.


Teknomo-Fernandez


Monte-Carlo

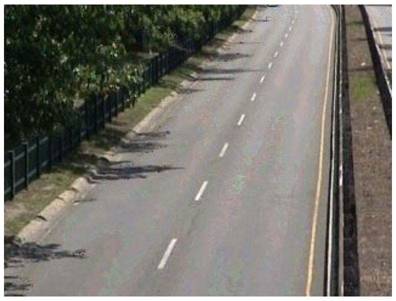

Median

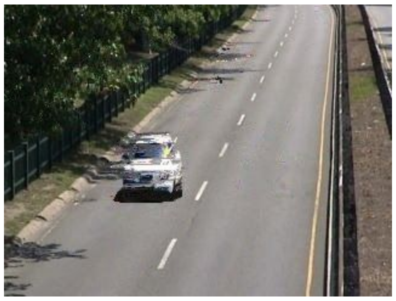
Time for each algorithm running the Charlie Data Set

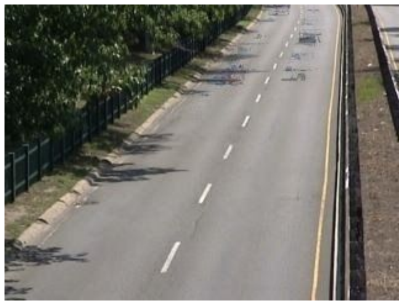
0.2906 seconds	315.6747 seconds	0.0607 seconds

The Highway Data Set.

The highway data set features a traffic camera in broad daylight with a series of cars passing through. This data set is composed of about 1700 images. There are cars in each frame, and they pass through from the top of the frame to the bottom of the frame in a sequence.


Teknomo-Fernandez


Monte-Carlo


Median

Time for each algorithm running the Highway Data Set

0.0871 seconds	134.8046 seconds	0.0298 seconds

The Park Data Set.

The park data set features a stationary camera in broad daylight set up in a park. This data set is composed of about 1100 images. Each frame features pedestrians walking through in any direction. However, this park features a 'scarce' amount of objects, perfect for testing the case in which the background is the majority of the image.

Time for each algorithm running the Park Data Set

0.07360 seconds	135.7947 seconds	0.0300 seconds

The Tunnel Data Set.

The tunnel data set features a stationary camera in a darker environment. The camera focuses on a highway with a frequent amount of cars passing through a tunnel. What is unique about this data set is that the camera has some sort of noise in the camera, causing for what looks like an 'oily' effect on the images. This data set features about 4000+ images.

Time for each algorithm running the Tunnel Data Set

0.6227 seconds	519.1620 seconds	0.1299 seconds

Discussion

In the Charlie data set, Median and Teknomo-Fernandez are very close to providing an amazing result while Monte-Carlo provides a very noisy, inaccurate result. The truth is that this data set is complicated due to the fact that the background is not the majority in one specific region of the image. With that assumption being denied, we already know that the Teknomo-Fernandez algorithm would struggle providing a perfect result. However, the median filter provides an almost 'wispy' after image to the result, still being able to see portions of the background.

In the Highway dataset, Teknomo-Fernandez (TF) seems to be the clear winner as it provides a more accurate result. TF produced a result with little to no discrepancies, producing an image so clean it could be considered the ground truth. In comparison, the Median filter produced an artifact of a vehicle, providing the assumption that if there is a lot of traffic in one specific area of the image with varying vehicle colors, we may find some 'noise' from the algorithm.

The park data set produced three accurate results straight in a row. However, the Median filter produced significantly better than the other algorithms, providing a result in half the time it would take the TF algorithm and almost 4500x faster than it would've taken the Monte-Carlo algorithm. The results of them all look the same, and thanks to the scarce amount of objects in the data set, we can assume that this was the data set which each algorithm would perform best in.

The best algorithm for the Tunnel data set was the Median filter, as it nearly beat the Teknomo-Fernandez algorithm and performed significantly better than the Monte-Carlo algorithm. The data set provided lots of noise through the recording which in turn makes the road look almost oily in the results. However, in the Median data set, the result looks like a real road: some wear-and-tear on the centers, but no purple or green hues to the side. This is additionally reflected in the earlier highway example, where the roads look much more natural in the Median result. This could be due to the fact that the noise occurs often, causing for the TF algorithm and the Monte-Carlo algorithm to confuse it for the background, when there is no reduction for that noise. Had this data set gone through some de-noising, or any sort of pre-processing, there may be a different result in the end.

In conclusion, there were two leading approaches to what generated the background accurately and at quick speeds. Teknomo-Fernandez and Median filter both worked exceptionally well, with Monte-Carlo being the clear "do-not-use" algorithm in these cases. We found that the Monte-Carlo was unreliable, added unnecessary computational time, and was a very inferior algorithm compared to the others provided in terms of accuracy and speed.