Literature Review 2

Computer Graphics

Gargi Gajjar (01745061)

Gargi_Gajjar@student.uml.edu

3/21/2018

Computer Graphics Literature Review 2

Gargi Gajjar (01745061) **Gargi_Gajjar@student.uml.edu**

Primary Paper: Fast Smoke Detection for Video Surveillance Using CUDA

Published in: IEEE Transactions on Industrial Informatics

Date of Publication: September 27, 2017

Authors: Alexander Filonenko (University of Ulsan, South Korea)
Danilo Cáceres Hernández (University of Ulsan, South Korea)
Kang-Hyun Jo (University of Ulsan, South Korea)

Secondary Paper: Smoke Detection for Static Cameras

Published in: Frontiers of Computer Vision (FCV), 2015 21st Korea-Japan Joint Workshop

Date of Conference: 28-30 January 2015

Authors: Alexander Filonenko (University of Ulsan, Republic of Korea)
Danilo Cáceres Hernández (University of Ulsan, Republic of Korea)
Kang-Hyun Jo (University of Ulsan, Republic of Korea)

Smoke Detection for Static Camera discusses the smoke detected during day time using static camera. Smoke can be an early precursor of fire that is why it should be detected as fast as possible. There are many kinds of sensors can be used to detect smoke regions including IR cameras, but using them leads to issues like high cost and complexity of maintenance. Cheap surveillance systems tend to use low resolution cameras. So, the algorithm should work in real time and be able to detect smoke in low resolution videos. The main difference is that camera is static and it gives an opportunity to detect smoke by using background subtraction technique and a single processor. But this allowed obtaining fast processing time. Smoke detection algorithm for static camera makes it possible to use background subtraction to distinguish moving objects and static ones. This approach exempts from the false positives given by sky regions and lane markings.

Smoke detection algorithm works as following steps in order: Background Subtraction, Morphology Transformations, Calculate Color Probability, Find Intersection of Probabilities, Perform Connected Components Labeling, Filter Components by Color Characteristic, Filter Components by Boundary Roughness, Filter Components by Edge Density and Compare Result to a Previous Frame. It uses Gaussian mixture model for background subtraction. In the result, the foreground image obtained had many pixels belonging to the same smoke cloud separated. To unite them into the same blob, two morphology operations were applied to the foreground mask: dilation and erosion. In order to apply next step of algorithm, it required to train the system. The smoke training data was gathered for different lightning conditions to depict smoke color characteristic. All images of database were united into a single one and for all the pixels probability density functions (PDF) of normal distribution were built for: Red, Green and Blue channels for RGB color space and saturation channel of the HSV color space.

Artificial objects like trucks may appear with same color characteristics with smoke. Usually their shape is similar to the convex hull while the smoke regions are random in their shape. But another filter called Boundary Roughness can distinguish real smoke regions from that object. Some moving objects with a random shape may have color close to one in training dataset. Smoke regions usually contain many edges. The edge density is another mean to distinguish smoke and another object. The last step of algorithm uses the advantage of using video sequence. Blobs which remained in the current frame are compared to their closest neighbors in previous frame. If their areas vary,

then the blob is decided to represent a real smoke region. Otherwise, it is another object which should not be considered anymore.

The most expensive part is the calculation of color probability. Processing time increased by 2 for 4 times larger image resolution. Performance must be bottlenecked by RAM clock because there are too many requests to the memory to process pixels of all 4 channels where each task does not require extraordinary processing power. There are possible ways of improving speed using separate CPU cores for each smoke blob and utilizing GPU to calculate color probability. The better the recording quality gives higher the detection rate. Further analysis can be done to neglect reflections on metal surfaces and water. This algorithm does not work with smoke detection at night. Some blobs extracted by the background subtraction, could not be recognized because the system was trained for smoke color characteristic at daytime. They proposed for future work some of steps will be reconsidered to be used in CUDA kernels. In this paper all the parameters were chosen manually, but for fully autonomous systems they should be adjusted according to lightning conditions.

Fast Smoke Detection for Video Surveillance Using CUDA extended the work of previous paper. They introduced some techniques to identify smoke and different objects. In primary paper, a smoke detection method for surveillance cameras is presented that relies on shape features of smoke regions as well as color information. They also use background subtraction method to detect changes in the scene. Their algorithm uses all the steps which were used in secondary paper. But they provide fast processing for both low resolution and high definition videos.

They used one popular way to improve processing speed by using General Purpose Graphic Processing Unit (GPGPU) cores. One type of GPGPU is Nvidia compute unified device architecture (CUDA) cores. These cores allow parallel processing of massive amounts of data. This discrete nature of the images makes it possible to process each pixel separately by means of variety of algorithms. This paper presents the hybrid approach of utilizing CPU and GPGPU in one algorithm where steps that benefit from parallelization are implemented using CUDA. In their comparison to the implementation of the proposed method on CPU, they only show that the hybrid approach applied to HD videos allows to keep processing time less than 200 ms, which is appropriate for surveillance systems while CPU only way provided less than 2 fps performance. A state-of-the-art method is re-implemented to provide a direct comparison using the same hardware and data.

This manuscript is focused on improving the performance of processing data from the camera. The method proposed in this work achieved data processing speed of up to 61 fps for 320 × 240 video sequences and more than 5 fps for HD video. The contributions of the present work are as follows: the proposed combination of CPU and GPGPU processing is presently the fastest computer-based implementation of smoke detection for surveillance cameras and the performance of the CPU+GPGPU implementation proposed herein drops slower than a CPU-only implementation when the frame size is increased. The main steps of Smoke Detection Algorithm are:
For GPU: Background Subtraction, Morphology Transformations, Calculate Color Probability, Find Intersection of Probabilities, Perform Connected Component Labeling, Apply Canny Edge Detector.
For CPU: Filter Components by Color Characteristic, Filter Components by Boundary Roughness, Filter Components by Edge Density and Compare Result to a previous Frame.

Computer Graphics Literature Review 2

Gargi Gajjar (01745061) **Gargi_Gajjar@student.uml.edu**

There are many possible ways to separate the foreground and the background. They used the modern approach named the self-balanced sensitivity segmenter at first. But then they used update strategy and neighbor-spread rules that are used to update the background model. This method is resistant to intermittent object motion and camera shaking. As the result of background subtraction, a foreground mask was obtained. Many parts belonging to the same smoke cloud were separated in mask. To unite them they used two morphology operations to the foreground mask: opening and closing. Smoke and nonsmoker regions can be distinguished by the color characteristics of moving objects. A smoke training dataset was created by using actual smoke regions to depict smoke color characteristics. For all the pixels in dataset, probability density functions (PDFs) of normal distribution were computed for Red, Green and Blue channels of RGB color space. It was noted that in most scenarios, the smoke is poorly saturated. The PDF for the saturation (S) channel of the Hue, Saturation and Value (HSV) color space was also calculated. Mean and standard deviation values were obtained for each PDF.

In the OpenCV 3.0 library, many algorithms are already implemented on GPUs. The background subtraction implementation and Canny edge detector used in the current work can be directly retrieved from the OpenCV. The same is true for morphology transformations. The computation of color probability should be implemented using CUDA C++ directly. According to the some data, in which only the CPU was utilized, the calculation of color probability was the most time consuming step. In preparing the color probability binary image, every pixel can be processed independently. Memory transitions are expensive. Therefore, all four color probability values for a pixel were calculated in a single thread. Furthermore, the intersection of probabilities of four “events” and binarization were also included in the thread. In this case, only five reading (four channels and training data) and one writing global memory operations were performed for each frame.

Summing up, this paper presents the smoke detection algorithm for video surveillance. A single CPU is not able to rapidly process HD video sequences. Processing of the most time consuming parts should be carried out by specialized devices like PPGA or GPGPU. This work was based upon the use of GPGPU. The implementation of the proposed method could achieve time performance for HD resolution video, which is appropriate for video surveillance task. The method is also sensitive to noise in the video. The best environmental condition for the proposed algorithm is indoors with artificial light and it outperforms the state-of-art work by processing frames four times faster when HD resolution is used. The algorithm is even more sensitive to low quality of video input.