An Implementation of the Stauffer-Grimson Background Estimation Method

Nikiforos Pittaras
Department of Informatics and Telecommunications
National and Kapodistrian University of Athens
Athens, Greece
Email: npittaras@di.uoa.gr

***Abstract*—In this report the Stauffer-Grimson method is presented, which is a process that implements background estimation on an input set of images. To begin, we present an overview over background estimation approaches and algorithms, followed by the detailed description of the method. We then outline the implementation structure, user interface, parameters and technical details, closing with a brief evaluation of the method's strengths and weaknesses.**

I. Introduction

Background estimation is the task where an image or set of images is segmented into a foreground and a background region. What typically follows is the subtraction of the background from the image, leaving the foreground, composed of objects of interest. This operation is important to a variety of fields, including medical imaging, event and object detection, motion capture, video processing and others. In non-trivial applications images can contain non constant lighting as well as various types and numbers of objects. Image elements can move, enter and leave the frustum at arbitrary speeds. A robust background subtraction method should be adaptive enough to handle such dynamic content scenarios.

II. Approaches to background estimation

There are many approaches to background estimation. In general, each method produces an estimate of the background image, and computes the difference of the background with each incoming frame. Thresholding the difference with a threshold T yields the segmentation result. Let $I(t)$ be the frame at time t, $BG(t)$ the estimation of the background, and $F(t)$ the binary image with the value of 1 at pixels belonging to the foreground. We then have:

$$F(t) = \left\{ \begin{array}{ll} 1, & |I(t) - BG(t)| \geq T \\ 0, & \text{otherwise} \end{array} \right\}$$

There are various methods of computing $BG(t)$, some of which are presented in the following paragraphs.

A. Naive approaches

If it is known that the first input image I_0 contains no foreground objects and it is static, the simplest approach is to consider that frame as the background. Subsequently, the foreground at time t is specified as:

$$BG(t) = I(0)$$

This method is unreliable in dynamic environments with changes in noise and lighting and demands some strong assumptions. In addition, it is not time invariant as it heavily depends on the content of initial frame. An improvement to the latter drawback is to consider the previous frame as the background and measure the difference with the next frame.

$$BG(t) = I(t-1)$$

This removes the static nature of the previous method, providing dynamic estimation of the background. It however retains the dependence on the frame content; outlier frames will provide poor background models. In addition, this system does not take advantage of past knowledge of the background.

B. Temporal aggregation

To disentangle ourselves from outlier frames, a common method is to perform an aggregation over a set of frames. This operation will diminish the effect of noise and high frequency values in the image. A set of consecutive K frames are pooled together to a single estimation. The pooling scheme can be the arithmetic mean of the frames:

$$BG(t) = \frac{1}{K} \sum_{m=1}^{K} I(t-m)$$

Averaging the previous images preserves the background pixels and smooths the intensity difference in the dynamic image elements. This however does not eliminate the latter, leaving traces of moving objects on the computed background. This distortion can be alleviated by choosing a large enough value for K. Alternatively, one can discard such artifacts by using median aggregation:

$$BG(t) = median(\{I(t-1), I(t-2), \dots I(t-K)\})$$

where the $median()$ operator works at the pixel level. The disadvantage of this method is that applying the median at each image pixel is computationally expensive. Moreover, both these methods require the storage of K frames in memory. A different approach is to take account of each incoming image to a running average:

$$BG(t) = \left\{ \begin{array}{ll} (1-lr)BG(t-1) + I(t-1)lr & t > 0 \\ 0 & \text{otherwise} \end{array} \right\}$$

This method requires the storage of a single image in memory, the estimated background. The parameter lr is the learning rate that controls the amount of contribution of new images in the running average. Note that setting lr to 0 and 1 the method degenerates to the first image and previous image methods respectively.

III. The Stauffer-Grimson method

A. Motivation and modeling

The Stauffer-Grimson method [1] attempts to construct a probabilistic model to represent the image background. In real world images, background pixel intensities have predictable values in the sense that they remain fairly constant. This normality can modeled with a Gaussian distribution. Image pixels that belong to a single object with constant lighting can be modeled using a single Gaussian distribution with constant parameters, $G_c(\cdot)$:

$$G_c(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where x, μ and σ represent the input intensity, the model's mean and standard deviation respectively. Image locations that correspond to single objects with dynamic lighting, can be represented with a single Gaussian distribution, with parameters that change as a function of time:

$$G_t(x_t, \mu_t, \sigma_t) = \frac{1}{\sqrt{2\pi\sigma_t}} e^{-\frac{(x-\mu_t)^2}{2\sigma_t^2}}$$

Intuitively, this corresponds to a shift of $G_c(x)$ along the intensity axis to a different μ value, namely μ_t, and a modification to its dispersion around that mean, σ_t. The next step of this thought process, is to generalize further and consider pixels that can belong to multiple objects, each of which can be illuminated in a dynamic lighting environment. This scenario can be modeled by a mixture of K adaptive Gaussians:

$$p(x) = \sum_{i=1}^{K} w_{it} G_{it}(x, \mu_t, \sigma_t)$$

where w_{it} represents the weight of the Gaussian $G_{it}(x, \mu_t, \sigma_t)$ in the combination.

B. Initialization and update

For a given image, a Gaussian distribution is initialized at each image pixel, with the pixel intensity as the starting μ value, a high standard deviation σ and a uniform weight factor $1/K_c$, where $K_c = 1$ the current value for K. At each subsequent frame F, the intensity values x of each pixel in F are matched to one of the K_c distributions, with the matching criterion specified as :

$$M_k(x) = \left\{ \begin{array}{ll} 1, & |x - \mu| \geq 2.5\sigma_k \\ 0, & \text{otherwise} \end{array} \right\}$$

where μ_k, σ_k are the mean and standard deviation for the k-th Gaussian respectively. The weights of the Gaussians are updated according to the following rule:

$$w_{k,t} = (1 - a) w_{k,t-1} + a w_{k,t} M_k(x)$$

where $w_{k,t}, w_{k,t-1}$ are the weights of the k-th Gaussian and a is the learning rate of the process. In addition, the parameters of the distribution G_m that matched the pixel value are updated as follows:

$$\begin{aligned} \mu_t &= (1 - \rho)\mu_{t-1} + x\rho \\ \sigma_t &= \sqrt{(1 - \rho)\sigma_{t-1}^2 + \rho(x - \mu_t)^2} \\ \rho &= a G_m(x, \mu_{t-1}, \sigma_{t-1}) \end{aligned}$$

This update brings the matched distribution closer to the recent pixel intensity, modifying the model to better fit the observed data.

C. Background estimation

For each pixel, the set of the K_c distributions are classified into background and foreground distributions. Image intensities of subsequent frames matched to a distribution that is assigned to the background are considered and marked as background points. The rest are classified as foreground points.

The Gaussians are sorted in descending order according to the quantity:

$$u_{kt} = \frac{w_{kt}}{\sigma_{kt}}$$

Higher u_{kt} values correspond to Gaussians with low variance and large weight, which means they are better candidates for representing a static background. A subset L is chosen for the background model:

$$L = \{l_1, l_2 \dots\}, l_i \in \{1 \dots K_c\}$$

where the number of background Gaussians G_{l_i} are determined as the first $N_L = |L|$ Gaussians, ordered by their u-values, such that the sum of their weights is greater than a specified threshold, T:

$$\begin{aligned} &\sum_{i=1}^{N_L} w_{li} \geq T \quad l_i \in 1 \dots K \\ &u_m \geq u_n, \qquad \text{for } n \geq m \end{aligned}$$

Pixels matched to the remaining $K_c - N_L$ Gaussians are classified as foreground.

With each incoming pixel intensity, if the intensity value does not match any existing distribution, a new one is constructed using the initialization scheme described in the previous section, using the input intensity as the mean and a large value for the variance. If the number of distributions K_c is less than the maximum number of Gaussians K, the new distribution is simply added to the mixture. If there is no space for a new distribution, the least probable distribution, which is the one with the smallest u_{kt} quantity, is replaced.

Stauffer-Grimson's estimation method is presented in algorithm 2. S_{init}, F_{init} are the high variance for initialized Gaussians and the initial frame respectively.

D. Post-processing

The response of the background & foreground detection can be littered with artifacts and outliers, due to the presence of lighting fluctuations, dynamic and shifting backgrounds or small moving objects of no interest. To alleviate these

responses, we can apply some morphological operators on the output image. Given a binary image I and a structuring element S, the operations of erosion, dilation, opening and closing are depicted as follows:

$$I_{er} = I \oplus S = \{z \in E | S_z \subseteq I\}, z \in \mathbb{Z}^2$$
$$I_{di} = I \ominus S = \{z \in E | (S^t)_z \subseteq I\}, z \in \mathbb{Z}^2$$
$$I_{op} = (I \ominus S) \oplus S$$
$$I_{cl} = (I \oplus S) \ominus S$$

whereS_z the translation of S by z and S^t is the symmetric of S, that is, $S^t = \{x \in E | -x \in S\}$. Intuitively, eroding a binary image diminishes each shape, dilation expands it, while opening (closing) removes shapes (fills holes) smaller than the structuring element, applying noise at the edge of each shape in the process. The operation of opening by reconstruction removes the aforementioned noise by repeatedly applying the dilation, followed by a conjunction with the original image. Similarly, closing by reconstruction fills response areas on objects. Conjunction guarantees no active pixels will remain that are not present in the original image. The method of opening by reconstruction is illustrated in algorithm 1 in pseudocode. Closing by reconstruction follows a similar procedure.

Algorithm 1 Opening by Reconstruction

```
I = I_init ⊖ S
C = []
while true do
    I = (I ⊕ S) ∩ I_init
    if I = C then
        break
    else
        C = I
    end if
end while
```

IV. Implementation

My implementation of the Stauffer-Grimson algorithm consists of the following components:

- The MAIN function and the OPTIONS class are responsible for displaying usage, receiving webcam and video input and displaying images, as well as receiving, storing and forwarding CLI parameters to classes and functions that need them.
- The PIXEL class runs the algorithm on each pixel of the image. It keeps and updates the Gaussian distributions, partitions and sorts background and foreground Gaussians. It is initialized with the first input frame captured from the webcam.
- The SGB class acts as a interface to the per-pixel implementation of the algorithm with the previous class. It receives the meta-parameters (learning rate, maximum number of Gaussians, the threshold, image dimensions, e.t.c.) that may be supplied from the user and forwards them to the pixel class, on every pixel of the input image. Additionally, it provides an interactive interface to the user so that she can control some of the above parameters in real time, as well as quit the application. The class collects the resonse of each pixel class and forms the output binary image, with white intensities representing foreground. Finally, it provides a description of the parameters the class expects.
- The POSTPROCESS, OPENREC and CLOSEREC functions apply opening and closing by reconstruction on the output binary image, with options supplied from the options class. It implements opening and closing by reconstruction. The size of the structuring element, as well as the initial number of erosion and dilation in each operation, are settable through CLI arguments.

Algorithm 2 Stauffer-Grimson

```
Input: K, T, a, S_init, F_init
procedure MAIN
    for each pixel x do
        INITGAUSSIAN(G_x)
    end for
    for each frame F do
        for each pixel x do
            [M, G_m]=Match(F[x], y)
            UPDATEWEIGHTS(G_m,M)
            SORTGAUSSIANS()
            if M = 1 then UPDATEPARAMETERS(G_m,a)
            else
                G_new =INITGAUSSIAN(F[x], S_init)
                if K_c = K then
                    G_last =FINDLEASTLIKELYGAUSSIAN()
                    REPLACE(G_new, G_last)
                else
                    K_c = K_c + 1
                    APPENDGAUSSIAN(G_new)
                end if
            end if
            BG = PARTITIONGAUSSIANS(T)
            if M AND m ∈ BG then output[x] = 0
            else output[x] = 255
            end if
        end for
    end for
end procedure
```

Each class is equipped with intuitive functions to carry out its purpose. Additional auxiliary functions can be found for debugging, visualization or design purposes.

A. User interface

The user can run the program using the following parameters:

- **main** parameters:
  - **method** *methodName*
    Where *methodName* is the name of the method of background detection to instantiate. Only the

Stauffer-Grimson method is implemented in this project, so *methodName* must be set to **SG**. The option was added for modularity in possible future expansions. This argument must always be provided.
  - **video** *videoPath* Run the method on a video file, located at path *videoPath*. Omitting this option will make the program get input from the default webcam, which is the default.
  - **resize** *cols rows*
    Where *cols, rows* $\in \mathbb{N}^+$ is the number of columns and rows to resize the images to. This is useful and often necessary as the SG method is computationally intensive. Omit this parameter to keep the image size provided by your webcam or video file, which is the default, but is not recommended.
  - **openclose** *strelOpen strelClose numOpen numClose*
    These parameters control the opening and closing by reconstruction. *strelOpen* and *strelClose* is the dimension of the square structuring element for the initial operations of erosion and dilation, in opening and closing respectively. The inverse process in each operation is done with the same structuring element. The *numOpen* and *numClose* specify how many times that original operation will be applied. In both cases, the inverse process is repeated until no change in the output image occurs. All four parameters should be members of $\mathbb{N}^+$. Omitting this argument skips opening and closing, which is the default.
- **Main** interface:
  - *a s, d f*: To decrease / increase the OPENREC / CLOSEREC structuring element size by 1.
  - *z x, c v*: To decrease / increase the number fo erosions / dilations of the above methods by 1.
  - *q ESC, p*: To quit and pause the program respectively.
- **SG** parameters:
  - *numGaussians learningRate threshold*
    Where *numGaussians*$\in \mathbb{N}^+$ is the number of Gaussian distributions per pixel, *learningRate* $\in [0, 1]$ is the learning rate a and *threshold*$\in [0, 1]$ the threshold T.
- **SG** interface:
  - *t r*: To decrease / increase the threshold value with an increment of 0.1.
  - *j k*: To decrease / increase the learning rate with an increment of 0.1.

Note that parsing the parameters of main is executed first. This however does not restrict the user in order she will provide command line parameters - any arguments that do not match the parameters of main, are stored and passed down to SG. Documentation for each class, function and method was generated by the comments and is available along with this report. A snapshot of the program running on a video input[1] is depicted in figure IV-A.

[1] https://www.youtube.com/watch?v=f66GcJGQ_OI

Fig. 1. SG background detection in action. In the left image is the raw video feed, while in the center and the right there is the segmentation result, before and after post-processing respectively.

B. Parameter selection

The algorithm parameters heavily influence the segmentation result. On low quality video feed, such as a laptop's webcam, one must set a threshold value $T \geq 0.6$, as lower values will let excessive noise corrupt the result. Post-processing is necessary on low to medium quality video to eliminate noise artifacts, and the number of Gaussians K and resize image dimensions depend on the hardware the program will be deployed on. Furthermore, the video content itself plays a role in the parameter selection; videos with dynamic, high frequency changes and small objects will require a larger value for K and a to achieve similar segmentation results to videos with a more static and slow changing content.

C. Technical Details

The algorithm was implemented in C++, using gcc 6.1.1. Opencv 3.1.0-3 was used to handle video and webcam input (videoio) and image output (highgui), morphological processing (imgproc), storage and basic operations (core). The code was written in qtcreator 4.0.2-1 and the project documentation was generated using doxygen 1.8.11-4. Compilation and execution was successfully tested in Kubuntu 15.04 and Arch Linux, on a Intel Core i5-3210M 8GB machine. The code is accompanied by a makefile as well as the pre-built binary it produces.

V. Conclusion

The Stauffer-Grimson background estimation method provides a robust method of modeling the image background. The per-pixel probabilistic model it constructs allows for an adaptive process that can deal with dynamic illumination and multiple objects. The adaptation speed and accuracy, as well as the response threshold are configurable parameters, providing flexibility that can be optimized with respect to the task at hand. However, the pixel-wise nature of the model renders it computationally intensive for large images and large numbers of Gaussian components. This can be remedied by exploiting the inherently distributed nature of the algorithm, and adopt a parallel processing scheme in multiple threads in the CPU or the GPU.

References

[1] C. Stauffer and W. E. L. Grimson, *Adaptive background mixture models for real-time tracking*, Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., Fort Collins, CO, 1999, pp. 252 Vol. 2.