Facultées des sciences appliquées

LIÈGE université
Sciences Appliquées

Computer Vision : Project - Background subtraction

Brandoit Julien - s200710

de Thibault Adrien - s202309

Gerard Manon - s201354

2023 - 2024

Algorithm

To perform background subtractions we have chosen to employ the ViBe algorithm. This choice is consistent with our commitment to understanding the components of state-of-the-art computer vision algorithms and implementing them from the ground up. Our implementation is inspired by the article [1], which gives a full description of the algorithm, as well as various ways of modifying and exploring its parameters. Although our algorithm is largely inspired by the one described in the article, it is not a carbon copy, and this report also aims to highlight any differences between our version and the one described in the article.

A two parts algorithm

A fundamental feature of our algorithm (and one that differs from the version described in the reference article) is that it is separated into two distinct parts : the construction of a background model and the creation of a mask video, segmenting the input video into a foreground and a background.

Building a background model includes the steps of initiating the model and updating it. Segmentation consists of just one main step, based on a background model. These steps are described in detail later in this report.

We have chosen to separate the algorithm into two distinct parts for two main reasons :

1. The video background is relatively 'simple', since it is relatively constant over time and almost uniform. It is therefore not necessary to update the background model constantly, as is done in the reference article, in order to obtain correct results. We develop this notion of 'correct results' in the last part of this report.
2. Separation allows us to play on the performance-accuracy trade-off by drastically reducing the algorithmic costs of segmentation. More details will be given in the section on this step.

The entire implementation of our version of the algorithm is carried out in `C`, to take advantage of the performance that can be achieved with this language, from a memory and computation time point of view. We've also taken advantage of the possibility of parallelizing tasks.

Construction of the background model

The aim of this part is to build a background model for the next segmentation stage. This construction can be carried out off-line, and the background model saved for future use. In practice, this part takes a series of images forming a video as input and gives a background model as output. This model is built in two stages. Our implementation builds a grayscale model based on a series of grayscale images. A model is defined by a set of $N_{samples}$ values for each pixel in the image. Determining these values is the fundamental goal of background construction.

Initialisation

The first step is to carry out an initialization of the background model : this model is used as a basis for building the final model. There are several methods for building this base, and we have chosen to follow almost faithfully the method used in the reference article.

In particular, a prior background model is built on the basis of a single frame. For each pixel, the model is initially filled with the values of neighboring pixels (the 8 nearest neighbors), using a uniform draw with replacement.

In the classic ViBe version, an inherent challenge is the presence of ghosts during the model initialization process. However, in our two-part code version, ghosting is not a concern because definitive segmentation is only performed after obtaining the final model, at which point ghosts have typically disappeared. It's important to note that while the initialization step is traditionally critical for achieving a rapid response, in our two-part algorithm, a fast response is not fundamentally necessary, as no "final" segmentation occurs before obtaining the ultimate background model. This observation plays a pivotal role in selecting algorithm parameters, prioritizing "good results" over "rapid response."

Update

An update step is then carried out on a certain number of frames in order to iteratively obtain the final version of the background model. This step is performed successively from one frame to another using the background model derived from frame $n-1$ to update the model at frame n. The prior of the model is used for the first frame.

More specifically, there are two mechanisms (identical to the reference article) that drive the model's evolution : spatial diffusion and temporal updating. These mechanisms come into play when a pixel is detected as belonging to the background (using the current background model). They do not occur systematically and, in practice, each of them has a separate probability of $1/\phi$ of occurring.

Pixel belonging to the background : A pixel is identified as part of the background when its value is sufficiently close (in absolute value, within a radius of R) to a minimum of V_{min} values associated with the background model of that pixel.

Temporal update : One of the values - randomly chosen - from the background model associated with this pixel is replaced by the current value of the pixel.

Spatial diffusion : One of the neighbors - randomly chosen - of the pixel has its background model updated by replacing - randomly - one of the values in its background model with the value of the pixel detected as belonging to the background.

Segmentation

The segmentation step is the simplest algorithmically, but it is also the most critical in terms of performance (computational time). It is designed to be carried out "online" and, therefore, must be fast. In practice, it takes a series of grayscale images as input, along with a background model, and outputs a series of images : masks that provide the estimated separation by our algorithm between the pixels belonging to the background and those belonging to the foreground (defined respectively by a value of 0 and 255 in the mask).

In practice, each pixel is tested in the same way as in the update step to be classified.

We aimed to make this step faster by leveraging task parallelization tools. The '`OpenMP`' library is used as follows : one thread is responsible for loading the images and the model, and, if necessary, for performing the model construction step. The other available threads are responsible for segmenting a certain number of different frames for each thread.

It's worth noting that this step would have been particularly well-suited for implementation on a GPU. We decided not to implement it on a GPU due to time constraints, lack of expertise, and hardware limitations, but it's definitely a performance enhancement avenue (with no loss of precision in the results) that would be very interesting to explore.

Evaluation

Different parameter values needed to be fixed. Here is the discussion of the chosen value.

We kept the same value as in the article for a few parameters. The first one is the radius R of the sphere used to compare a new pixel value to pixel samples. R equal to 20 was explained as the perceptible difference in luminance, so we kept it. After this, to classify a new pixel value as background, V_{min}, equal to 2, of close pixel samples were needed. With V_{min} equal to 1, there was a loss of precision, so we opted for 2 which was a compromise between speed and accuracy. Lastly, considering only the 8 nearest neighbors of each pixel, like in the article, gave us a good result. Therefore, we did not try to include further neighbors as the further the neighbor is the less correlation there would be, so the quality of the result could decrease.

The update factor ϕ is equal to 1 for the initialisation of the first 100 frames. As it represents the probability of updating, with a value of 1, it means that there is an update of every background pixel. This could remove quickly the ghosts from our initial background model. The update factor will become 16, like in the article, if the video is longer than 100 frames for the segmentation part. The ghosts would have already been taken away so there is no need to resample too frequently in that case.

As the background is uniform, there was no need to have too many samples stored in each pixel model. Therefore, we chose N, equal to 10, as it was sufficient and preserved the accuracy.

To have a quantitative evaluation, we would have needed to be able to tell whether the pixels are correctly classified or not. For this, we would have needed a reference of the real background for each image. Obtaining it manually is impossible, therefore we need another background subtraction algorithm. This background would then serve as our reference background in our metrics. The most used metric is the Percentage of Correct Classification. Of course, during this analysis we must not forget that we did not use the real background but a background decided by an algorithm. Hence, this background is not 100% perfect.

The masks that we obtain are non-filtered masks. Some small isolated dots are not considered as background in each image. These are possibly due to some noise, or the small cells floating around. To take out these cells and have a mask that contains only the regions of interest, we will need to apply filters to our masks. This additional step will be considered in the next task and not in this report.

Références

[1] Olivier Barnich et Marc Van Droogenbroeck. « ViBe : A universal background subtraction algorithm for video sequences ». In : *IEEE Transactions on Image Processing* 20.6 (juin 2011), p. 1709-1724. doi : `10.1109/TIP.2010.21016131`.