16K codec

BACKGROUND SEGMENTATION

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

[Type the author name]

[Pick the date]

16K codec

# Problem statement

Our main goal is to segment the background and the foreground from a High Resolution video. There are some constraints in the capture of the video. The camera has to be recording still to avoid background movement and objects in the scene can move freely. The objects’ movements must to be smooth because if not, flickering pixels will be detected as part of objects We want to make 2 streams: one for the background and another one for the moving objects. This way we can reduce the amount of data required to transmit the video and High Resolution.

A pixel is defined as a tuple , where and are the rows and columns respectively and is the time step (frame) of the video. Any pixel that has same color at is considered background → ∆color < λ, where λ is a real number threshold. We consider as the same color all the pixels that change less than λ distance using RGB values of the pixels as cartesian coordinates.

# First pass: preprocessing

In the first pass, we preprocessed the video to detect the local changing pixels in a pair of consequent frames. Those pixels determine the moving objects in the scene and thus they are part of the foreground at that timestep. The result of this first pass is an image in which each pixel is classified as *background* or *foreground*. The *foreground* pixels are discarded for next steps of the algorithm but the *background* pixels are kept as **possible** *background* candidates.

## Compute local frame difference

To detect the moving objects we computed the difference between the frame and . The difference tells us how much a pixel has changed its color, which leads to the detection of the movement if there is no drastic lighting change. This operation is done locally using two consequent frame and thus it is not applicable to the entire video to detect video movement nor to detect same objects in different frames.

## Classify each pixel as background or foreground

Since our goal is to detect the moving objects and generate a background image, we classified each pixel as part of the background or foreground which is created from the moving objects. Using the already computed frame differences of the video we threshold the those images using the following equation:

We considered as part of the background every pixel . Therefore, we generated a new image in which every pixel of is assigned the label *background* using the following equation and the remaining ones as foreground:

# Second pass: object identification

In our case we consider a pixel as part of an object at timestep if it has changed the color therefore that pixel has been classified as *foreground* in . In fact, every in the video is a binary image which segments the objects in the scene from the background. There are some problems with this approach: (1) Objects may be still and then start moving → in previous frames the object has been detected as background, (2) Objects may be moving and then stop → in next frames the object is going to be detected as background, (3) Objects with same color/texture are not going to be detected as moving objects because there is no enough color difference in their surface.

## Unique blob identification

In each frame we detected the *foreground* pixels and for each frame we segmented unique object blobs. We considered a unique object blob as all the pixels that are part of the *foreground* and that are connected using 8-neighbor connection. Using Connected Component Labeling (CCL) technique we can retrieve, for each frame, a unique id for each connected region leading to the segmentation of unique objects. This technique does not work well with overlapping objects because the two objects are recognized as a single blob.

## Mask generation

Once we segmented the unique blobs for each object, we generated an individual binary mask for each of those ones. This masks are kept for the next passes to detect when an object has traversed some specific pixels. For each timestep we stored a set of binary masks representing the unique objects in the frame. It is crucial to assign the same *id* for the blobs that are the same object in different frames. This way, we know when an object has occluded an specific pixel and when, and when the pixel is part of the background.

For a better understanding of the workflow of the object identification see the following flowchart:

# Third pass: pixel classification

In this last step, we know when and where an object has moved. With such information it should be possible to detect which objects occluded each pixel in the video. A problem arises when there is a still object and then moves or when an object is moving and stops. Because the frame differences are not going to provide temporal information for still objects we cannot determine if a pixel is part of the *background* or not.

Our premise is that if there is a pixel and there exists an object , where is the set of all detected objects at timestep , is considered part of the *foreground*. If that is not the case can be part of the *background* or part of an object. In this situation there are three cases: (1) a still object that starts moving, (2) a moving object that stops and (3) an object that is moving. This approach is based supposing the tracking has no errors (e.g. blinking).

**Case 1:** for some , overlaps with an object at which is the closest timestep where an object has overlapped the pixel. If the same pixel in the next timesteps until are overlapped by and there is not the same object present at time step we consider it as *background*. We executed this algorithm for every object in the video. If the pixels passes al the tests for all the objects in the video we assigned that pixel to the final result.

**Case 2:** this is the same case as **Case 1** but reading the video in reverse. We applied the same method as before but processing previous frames. In this case .

**Case 3:** in this last case, if the object is moving, its mask is not covering at timestep and thus the pixel is a candidate for being part of the *background*. Each object in frame which is not covering the pixel is excluded from next computations increasing the efficiency. Once is a candidate for background we have to compute cases **Case 1** and **Case 2** to determine if the pixel is part of a still object.

If after all the cases the pixel is not considered part of the *foreground* we assign its value to the final background image.