Tutorial 11 - Colab and Video summarisation

Gidon Rosalki

2026-01-07

This tutorial is very short, since it is mostly examples. I suspect that these notes will not include much.

1 Colab

Google Colab is a Jupyter style interpreter, so it runs in blocks, that need re running after change. Note that if it does not auto save, then check your internet connection. Google Colab offers using a GPU, which allows significantly faster model training and execution. This is because the GPU is designed to do thousands, if not more, of simple independent operations simultaneously. Unfortunately, Google only gives each student one hour per day. If you need more, then you can pay, but you can plausibly do it all within this limit, so start early to give you as many hours as you can. Some people did jump between accounts to get more hours, but this sounds like a pain.
To run on the GPU you go to runtime, change runtime type, GPU, save. Files will be stored in Drive, so to mount your drive to access the files you perform `from google.colab import drive` and then `drive.mount("/content/gdrive")`.

2 Motion Segmentation

There are many goals for motion detection:

- Identify moving objects
- Detection of unusual activity patterns
- Computing trajectories of moving objects

Applications include:

- Indoor/outdoor security
- Real time crime detection
- Traffic monitoring
- Many intelligent video analysis systems are based on motion detection

Background subtraction uses a reference background image for comparison purposes, and the current image (containing target object) is compared to reference image pixel by pixel. Places where there are differences are detected, and classified as moving objects. A simple algorithm for this is as follows:

1. Construct a background image B as a median of a few images
2. For each actual frame I, classify individual pixels as foreground if $|B - I| > T$ (T is some threshold)
3. Clean the noisy pixels

Figure 1: Background subtraction with noise removal example

In order to compute the background, instead of simply taking the median, we may use the following algorithm:

$$\begin{cases} B_{t+1}(x,y) = (1-\alpha)\, B_t(x,y) + \alpha I_t(x,y), & \text{if } (x,y) \text{ is } \textbf{background} \\ B_{t+1}(x,y) = (1-\beta)\, B_t(x,y) + \beta I_t(x,y), & \text{if } (x,y) \text{ is } \textbf{foreground} \end{cases} \tag{1}$$

$$\beta \ll \alpha \tag{2}$$

This way, if there are brightness changes, then our background changes with the brightness changes, and we do not detect the *entire* image as moving when the brightness changes (think sun/cloud movement). As we see from the formula, this is applied to each pixel of the image individually, so the background is formed as an average of many different frames.

In summary, to detect and track an object in a static scene, we model the background, and subtract to obtain an object mask. We filter to remove noise, and group adjacent pixels to obtain objects. We then track these objects between frames to develop trajectories.

3 Video summarisation

The digital world contain huge amounts of recorded video. Video browsing is time consuming, which results in most of the captured video never being watched or examined. Video summarisation helps summarise the content of a long video into a compact representation, which enables us to quickly review a long video.

We will discuss two methods, summarising a video into a static mosaic image, and summarising a video into a video synopsis. In the second option, we can have many different times represented in the same frame, and note upon them at what time they were there.

We can see a giant difference when there is and is not video summarisation. Finding the 7/7 bombers in 2005, and the Dubai assassins in 2010 took weeks, but the Boston bomber (2014), and Sinwar (2024) merely took a couple of days.

3.1 Video indexing based on mosaic representations

We may represent it based on a scene: Videos contain many frames of the same scene over time This has high temporal redundancy. The idea is to transform a video from a sequential frame-based representation to a scene-based representation.
The scene-based representation is based on the three fundamental information components of video:

- Extended spatial information – appearance of the scene: Due to camera movement, the static appearance of the scene can be distributed among many frames – no way to see entire scene at once. Instead it is represented as a panoramic mosaic image, capturing an extended spatial view of the entire scene
- Extended temporal information: motion of moving objects in the scene: The evolution of events over time is distributed over frames. This is represented using the time trajectory of each moving object's centre of mass
- Geometric information: 3D scene structure, geometric transformation caused by camera motion. Here we relate frames to the mosaic coordinate system

We create a static background mosaic as follows:

- Align all frames in the scene to a single coordinate system
- Integrate the frames to form a single static mosaic image
- Remove the moving objects (e.g. temporal average or median)
- Save the geometric transformation relating the different video frames to mosaic coordinates

Moving objects are not represented in the static mosaic. To provide a summary of the events, a trajectory of each moving object is added on top of the static mosaic. This creates a visual summary of the entire dynamic foreground event that occurred in the video.

Figure 2: Synopsis mosaic

3.1.1 Nonchronological video synopsis and indexing

If we do not care that everything happens in chronological order within the video, we can have everything happen at once, so we can see a very compressed summary (Shmuel developed this).
Most previous work suggested a frame based summary, where we detect key frames, show short video segments, and adaptively fast forward. Video synopsis proposed an *object* based summary, where we simultaneously show multiple activities that may have originally occurred at different times. This does not include fast forwarding, instead it preserves the dynamics.

The steps are as follows, following a two phase process:

1. Online phase: Detect and track objects, and store them in a database
2. Query phase: According to a given user query ("30 second synopsis of the last day") select the relevant objects, and stitch them into the synopsis

We can call these objects *tubes*, since their movement across time forms a tube of the object shape. We can then include objects in the result by the times of the starts and ends of these tubes.

Video synopsis has the following properties. Let N frames of an input video be represented in a 3D space time volume $I(x, y, t)$, where (x, y) are the spatial coordinates of the pixel, and $1 \leq t \leq N$ is the frame number. The synopsis video $s(x, y, t)$ should have the following properties:

- s should be substantially shorter than the original video I
- The maximum "activity" from the original video should appear in the synopsis
- Dynamics should be preserved (objects should not be fast forwarding through the video)
- Visible seams and fragmented objects should be avoided

We need to identify the interesting objects. We may say that interesting $\approx$ moving objects. Some motions are not interesting, like clouds and leaves, but some interesting objects, like people, can be static. Object recognition can be incorporated to support these issues. We start with background construction $B(x, y, t)$, including time to avoid the appearance change between day and night (for example), generally using a temporal median over a few minutes before and after frame t. In practice, we use a temporal histogram per pixel.

To construct an activity tube, for each frame I_t, we create a labelling function f_t such that

$$\forall r \in I_t,\ f_t(r) = \begin{cases} 1, & \text{if foreground} \\ 0, & \text{if background} \end{cases}$$

The labelling function f_t creates blobs. Overlapping between consecutive blobs is used to create 3D $x \times y \times t$ connected components called **activity tubes**. Each tube b is defined over a finite time segment in the original video: $t_b[t_b^s, t_b^e]$. What we do is find M that enables us to create the optimal time shift for each tube. This has the shortest video, with minimum collisions between objects.

The length of the synopsis video is lower bounded by the length of the longest tube. This is a problem when very long tubes exist, so to solve this, we cut the long tube into shorter sub tubes, and display them simultaneously. This can result with a stereoscopic effect, where the same object appears in multiple locations.

Some objects can have different lighting then the background, for example, an object caught at night, on the daytime background. A seamless stitching method must be added like Poisson blending. Pyramid blending works as well.