Video Processing Project

Team Members names & IDs:  
Yoav Dychtwald - 209518299  
Roy Sarafov - 209619477

Table of Contents

[1. Project Overview 2](#_Toc202992315)

[2. Code Structure 2](#_Toc202992316)

[3. Algorithmic Explanations 3](#_Toc202992317)

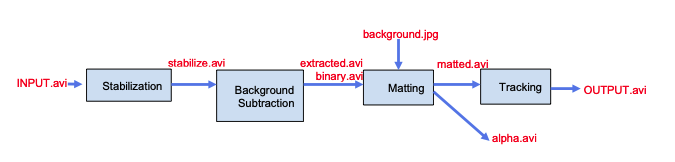
[Video Stabilization 3](#_Toc202992318)

[Background Subtraction 4](#_Toc202992319)

[Matting 4](#_Toc202992320)

[Object Tracking 4](#_Toc202992321)

# 1. Project Overview

In this project, we implemented a full video processing pipeline for analyzing motion in video and changing a background of a given video input with a given background image input. The pipeline includes steps for video stabilization, background subtraction, matting, and object tracking. Each module handles a specific stage of processing, and together they produce stabilized and segmented output for further analysis. The pipeline design can be seen in the following diagram:  


# 2. Code Structure

The project code is written in Python and organized into modules:

**main.py**

* **Purpose**: Controls the execution of all pipeline stages based on user input.
* **Main Tool**: argparse for command-line argument parsing.

**video\_stabilization.py**

* **Purpose**: Stabilizes video by compensating for camera motion.
* **Main Tool**: cv2.findHomography, cv2.goodFeaturesToTrack, cv2. calcOpticalFlowPyrLK , cv2.warpPerspective for frame alignment.

**background\_subtraction.py**

* **Purpose**: Detects moving foreground objects in video frames.
* **Main Tool**: cv2.createBackgroundSubtractorKNN for background modeling.

**matting.py**

* **Purpose**: Refines binary masks for cleaner segmentation.
* **Main Tool**: cv2.GaussianBlur for mask smoothing.

**tracking.py**

* **Purpose**: Assigns consistent IDs to objects across frames.
* **Main Tool**: Bounding box detection with cv2.findContours.

**utils.py**

* **Purpose**: Provides utility functions for filtering, video I/O, and kernels.
* **Main Tool**: Custom helper functions (disk\_kernel, load\_entire\_video, etc.)

**paths.py**

* **Purpose**: Manages file paths for all project inputs/outputs.
* **Main Tool**: Python class with predefined path attributes.

**logger.py**

* **Purpose**: Sets up logging for debug and status messages.
* **Main Tool**: Python logging module.

# 3. Algorithmic Explanations

## Video Stabilization

The stabilization algorithm uses feature-based motion estimation. Key steps include:  
1. Converting each frame to grayscale.  
2. Detecting good features to track using `cv2.goodFeaturesToTrack`.  
3. Estimating optical flow using `cv2.calcOpticalFlowPyrLK` to track movement across frames.  
4. Calculating an affine transform between matched features using `cv2.estimateAffine2D`.  
5. Accumulating and smoothing the transformations to reduce abrupt changes.  
6. Applying the smoothed transformations to each frame using `cv2.warpAffine`.  
  
This results in a more stable video with reduced jitter.

## Background Subtraction

The background subtraction is implemented using a K-Nearest Neighbors (KNN) model:  
1. A `cv2.createBackgroundSubtractorKNN()` instance is used to model the background.  
2. Each frame is passed to the model to identify dynamic foreground regions.  
3. Post-processing (morphological operations and median filtering) refines the foreground mask.  
  
This enables isolation of moving objects from static background scenes.

## Matting

Matting refines the binary masks from background subtraction. The steps include:  
1. Applying morphological closing to fill small holes.  
2. Applying median blur for smoother mask edges.  
3. Optionally using edge-aware matting (not detailed in current implementation) to preserve object contours.  
  
This improves segmentation quality, especially around object boundaries.

## Object Tracking

Tracking follows detected objects across frames using bounding box matching:  
1. Contours are detected from the matting mask.  
2. Bounding boxes are created around large enough contours.  
3. Each box is compared with boxes from the previous frame using intersection-over-union (IoU).  
4. Labels are assigned and updated based on best matches.  
  
This enables consistent labeling of moving objects.