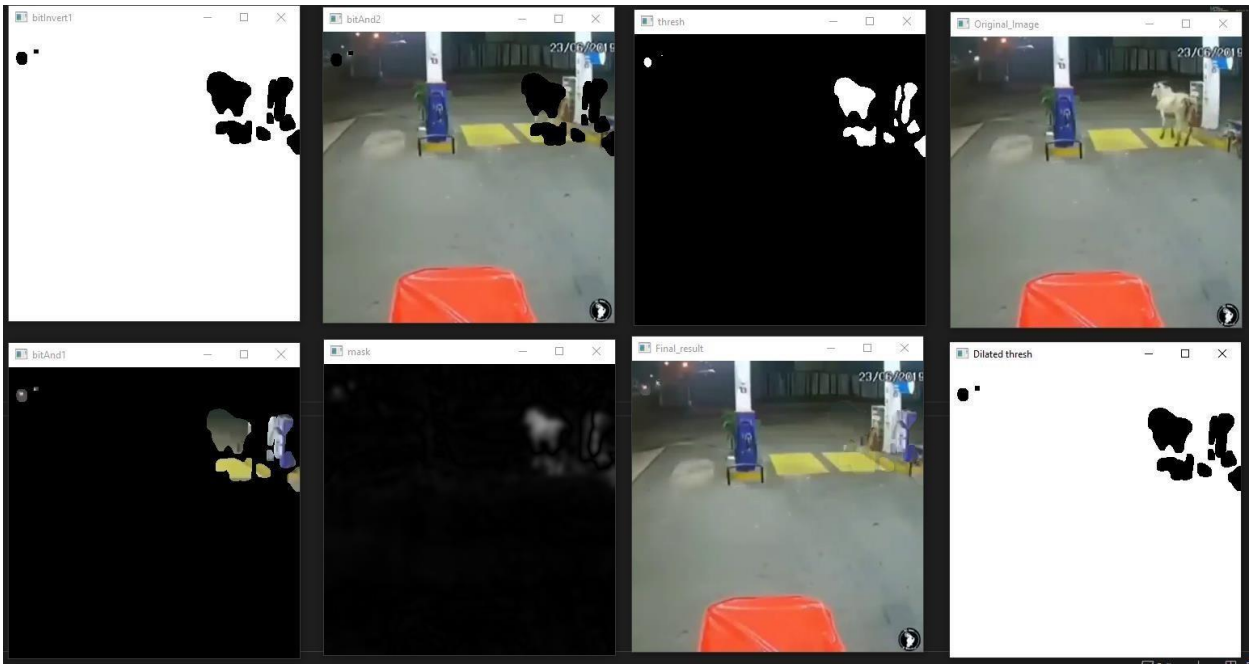
Moving object remover

Αρχικά, δηλώνουμε το video και ρυθμίζουμε την ανάλυση που θα χρησιμοποιήσουμε.

```python
import cv2
import numpy as np
from imutils.video import VideoStream
import argparse
import datetime
import imutils
import time
import cv2

#set the path of the video we are going to use
vs = cv2.VideoCapture('C://Users//spirt//Desktop//source2021//thema2//video.mp4')

#set video resolution
vs.set(cv2.CAP_PROP_FRAME_WIDTH, 600)
vs.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

#set first frame to none
Fframe = None


#get width and height of video
frame_width = int(vs.get(3))
frame_height = int(vs.get(4))
size= (frame_width, frame_height)




# set a writer so we can save our final video
writer = cv2.VideoWriter('C://Users//spirt//Desktop//source2021//thema2//finalResult.avi', cv2.VideoWriter_fourcc(*'XVID'),10.0, size)
```

Μετέπειτα, χρησιμοποιώντας μια λούπα ξεκινώντας από το πρώτο frame μέχρι και το τελευταίο του βίντεο αλλάζουμε το μέγεθος του frame, το χρώμα και το θολώνουμε. Επιπλέον αποθηκέυουμε το πρώτο frame με το αλλάγμενο χρώμα και το κανονικό για συγκρίσεις.

Στην συνέχεια, για κάθε frame του video εντοπίζουμε τα αντικείμενα που βρίσκονται σε κίνηση και δημιουργούμε την μάσκα. Ύστερα, χρησιμοποιούμε λογική πρόσθεση (cv2.bitwise_and ()) για τα pixels των bitAnd1 και bitAnd2 (το οποίο περιέχει την αντεστραμένη μάσκα) . Τέλος προσθέτουμε και παίρνουμε το τελικό αποτέλεσμα με το κινούμενο αντικείμενο να μην υπάρχει.




```python
while(1):
    ret, frame = vs.read()  #Read image frame


    if not ret:             #if we dont have more frames, then break while
        break



    frame = imutils.resize(frame, frame_width) # resize the frame
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)# convert to gray
    gray = cv2.GaussianBlur(gray, (21, 21), 0)# putting blur


    if Fframe is None:

        Fframe = frame #colored video

        Fframe2 =gray # black an white video

        continue


    # absolute difference between the current frame and first frame
    mask = cv2.absdiff(Fframe2, gray)
    thresh = cv2.threshold(mask, 25, 255, cv2.THRESH_BINARY)[1]

    #filing gaps of tresh by dilating the  thresholded image
    Dthresh = cv2.dilate(thresh, None, iterations=2)



    bitAnd1 = cv2.bitwise_and(Fframe, Fframe, mask=Dthresh)

    bitInvert1 = cv2.bitwise_not(Dthresh, Dthresh, mask=None)  ##invert the mask

    bitAnd2 = cv2.bitwise_and(frame, frame, mask = bitInvert1)
```