Fast start OVO

1. If you want use library visual odometry, you want install Python package OVO from github repository will use «pip».

2. Next step, create project in PyCharm, and create file \*.py.

3. Add next libs:

import cv2  
import time  
import math # For math.nan if needed  
from ovo\_types import CameraParams, Pos\_i2  
from video\_processor\_ovo import VideoProcessorOVO  
from ovo\_constants import OVO\_ANGLES\_FROM\_VIDEO # or OVO\_ANGLES\_FROM\_SOURCE

\*\*We use opencv in backend for work with video files or streams, feature matching mechanism ORB, and visualization video in examples

4. After, you should create code construction like this:

4.1. Pre start objects:

def main():  
 cam\_p = CameraParams()  
 cam\_p.fov = 86 # Assuming 86 was degrees, convert to radians if map\_scale expects it  
 # Check how map\_scale\_py uses fov. If it's directly in tan(fov), it should be radians.  
 # If fov in C++ Camera\_params was degrees, then the C++ mapScale would convert.  
 # Python's math.tan expects radians.  
  
 res = Pos\_i2(1920, 1280)  
 cam\_p.resolution = res  
 cam\_p.type = 0 # Example type  
  
 video\_file = "./examples/test.LRF" # Make sure this file exists

4.2. Init class for work with video VideoProcessOVO:

vpo = VideoProcessorOVO(  
 p\_params=cam\_p,  
 source\_info=video\_file,  
 api\_reference\_or\_source\_flag=OVO\_ANGLES\_FROM\_VIDEO, # SOURCE\_FLAG  
 custom\_shape\_or\_max\_points=100 # maxPoints

except IOError as e:  
 print(f"Error initializing VideoProcessor: {e}")  
 return

vpo.set\_custom\_shape(640, 640)  
h\_altitude = 300.0 # meters  
k = 0 # key code

4.3. Grab frame and send TMI information about altitude

while k != 27: # 27 is ASCII for ESC  
 start\_time = time.perf\_counter()  
  
 vpo.set\_data\_for\_one\_iteration(h\_altitude) # Only setting altitude  
  
 if vpo.grab\_frame\_and\_data():  
 end\_time = time.perf\_counter()  
 processing\_time\_ms = (end\_time - start\_time) \* 1000  
 print(f"Full tick: {processing\_time\_ms:.2f} ms")  
  
 pos = vpo.trajectory.get\_curr\_pos()  
 print(f"Position x,y,z: {pos.x:.2f}, {pos.y:.2f}, {pos.z:.2f}")  
  
 current\_display\_frame = vpo.get\_frame() # This is the processed gray frame  
 if current\_display\_frame is not None:  
 cv2.imshow("Capture OVO Python", vpo.frame)  
 else:  
 print("No frame to display from VPO.")  
 else:  
 # print("grab\_frame\_and\_data failed or no new data.")  
 # If video ends, cap.read() will return False, bgr\_frame will be None  
 # Check if cap is still good  
 if not vpo.cap.isOpened() or vpo.cap.read()[0] == False : # Heuristic check if stream ended  
 print("Video stream ended or cannot read frame.")  
 break  
  
  
 k = cv2.waitKey(1) & 0xFF # Wait for 1 ms

5. Destroy and Escape

cv2.destroyAllWindows()  
 if vpo.cap.isOpened():  
 vpo.cap.release()  
 print("Processing finished.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

6. Press Run (Shift + F10) if all requrements is OK

7. Console out want looks like this:

Position x,y,z: -277.98, 280.68, 300.00

Full tick: 12.98 ms

Position x,y,z: -282.94, 284.75, 300.00

Full tick: 12.48 ms

Position x,y,z: -287.45, 288.81, 300.00

Full tick: 12.37 ms

Position x,y,z: -291.97, 292.87, 300.00

Full tick: 13.50 ms

So, its simple example how use visual odometry algorithm. Information about position its «Position x,y,z: -291.97, 292.87, 300.00» for local coordinate system. Zero Point – its Point when start video stream\file. Data x,y,z – on meters. If you want, analyze rotate angle Oy you should check affine\_params.angle from VideoProcessorOVO (video\_processor\_ovo.py).