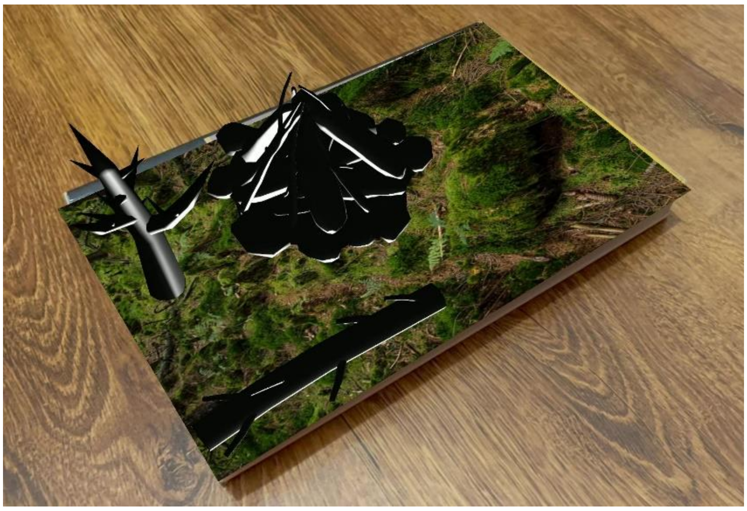
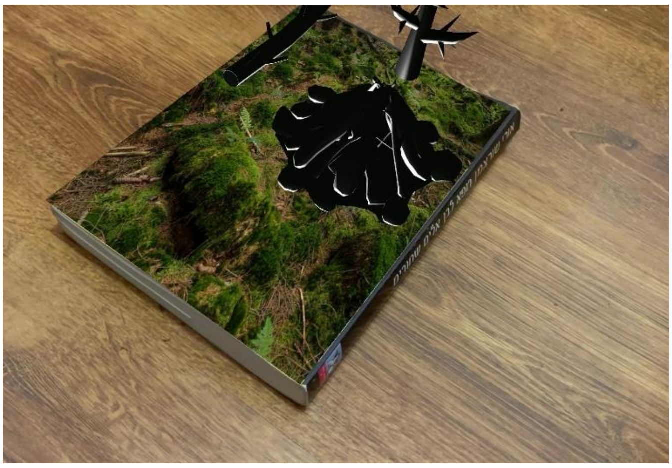
Augmented reality

OR LEVY 206263352 OFER FRITZ 313191983

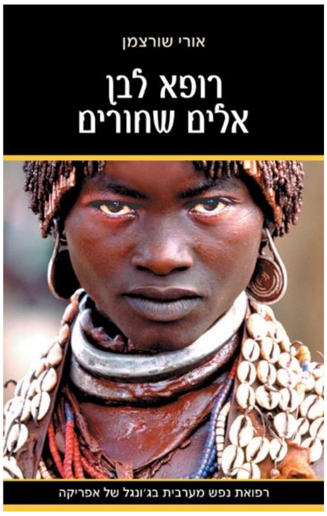
In this project we will produce a 3D scene on a given video while using:

- feature detection – recognizing the plane for our scene.
- camera calibration – reduce distortion from lance, orientation in space.
- linear transformations in 3d space.

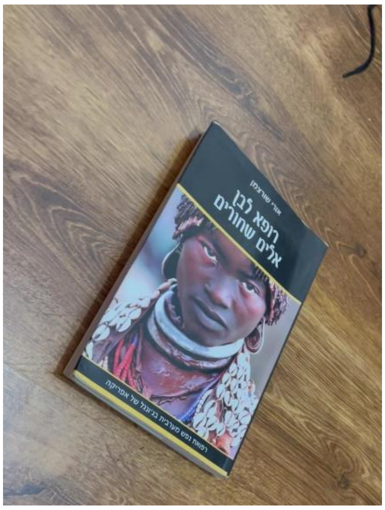




Part 1 – perspective wrapping



target



frame

the 3D plane – "forest"

Inputs: target image, wrapping image and a video with the target image in it.

1. Make SIFT feature detection on the target image (gray scale to reduce computing)
2. Make SIFT feature detection on one frame at a time(gray scale to reduce computing)
3. Make a list of the best corresponding features between target and frame:
    a. Match features using cv2.BFMatch – returns a list of match key points.
    b. Select only the best matches – ratio test grater than 0.5 between kp distances.
    c.
4. Find homography – finding the matrix witch will be the transformation ratio for the target shape and angle in the original image to the target figure detected in the frame. Steps:
    a. Make a list of target key points from 3.b
    b. Make a list of frame key points from 3.b

    Now the key points correspond (the only kp in the lists are related to the target pic).

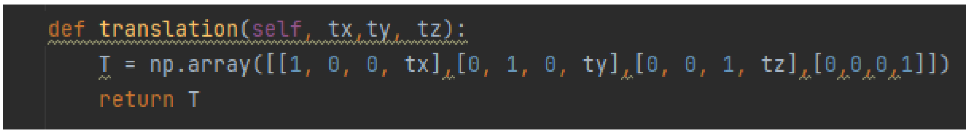
    c. Find the matrix and a mask using cv2.findHomography:
        - H_matrix – the matrix specified above.
        - Mask – a bitmask labeling the detected target in the frame.
5. Wrap the "forest" 3d plane on the detected area cv2.wrapPerpective:
    a. Make a new mask shaped like the target image, transformed according to H_matrix
    b. Transform the forest image the same way
    c. Lay over the 5.b image pixels that match to the 5.a mask on top the original frame.
6. Save the new frame, show percentage of progress.
7. Repeat 1 – 6 till the end of the video.

Part 2 – calibration and AR meshing

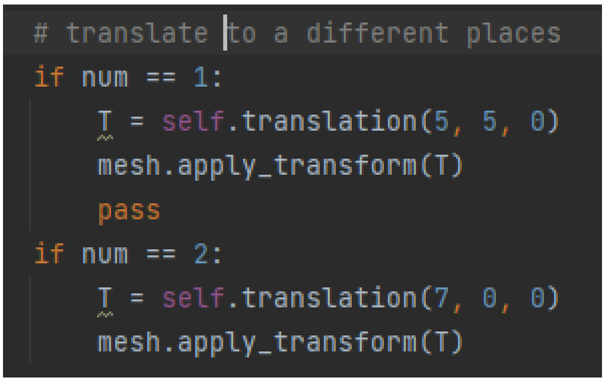
In this part we will find our specific camera calibration data to cancel camera distortion so we can plot and render our scene with better accuracy.

1. Camera calibration – run calib.py on my calib_chess video (removed from directory due to large size). This program saves the calibration data into a file so we can use it on the main program.
2. Find camera relative position:
    a. Reshape the key points (from 4.a) values according to the real size of the target image in the frame. And reshape to a 3d array.
    b. Find r, t vectors – responsible for the 3d object transformation relative to plane. We will do it using cv2.solvePnP.
3. Plot and render the 3d object:
    a. we made some changes in mesh_renderer.py so we can plot 3 different objects on the plane (adding translation function to put any object in a different place).
    b. Made 3 mesh_renderer object (onr for each)
    c. Draw the objects one by one, each one on top of the previous image given.
4. Save the frame
5. Repeat 1-4 for all frames in video.

- Mesh_renderer changes:
    - The object have one more variable – "num" – the num id for the object



```
def translation(self, tx,ty, tz):
    T = np.array([[1, 0, 0, tx],[0, 1, 0, ty],[0, 0, 1, tz],[0,0,0,1]])
    return T
```



```
# translate to a different places
if num == 1:
    T = self.translation(5, 5, 0)
    mesh.apply_transform(T)
    pass
if num == 2:
    T = self.translation(7, 0, 0)
    mesh.apply_transform(T)
```