**Augmented Reality for books on a shelf**

**Implementation Plan**

**1. Detection**

* SIFT keypoint matching can be used to find the marker in the video stream *(Similar to Exercise 6)*

1. Pre-compute SIFT keypoints and descriptors for the clean marker image using OpenCV
2. Compute SIFT keypoints and descriptors for the current video frame.
3. Use OpenCV’s built-in keypoint matching functionality (FLANN matcher) to find matches between the sets of keypoints

Possibly:

1. Find a homography between the two sets of keypoints (RANSAC). If the number of inliers (consistent with the homography) is above a threshold, we consider it a match [2, 3].

**2. Tracking**

* Less expensive computational techniques for tracking the marker in upcoming frames.
* Using KLT optical flow *(Tutorial by Brian)*
* Given two frames from a video and a set of keypoints in the first frame, find the locations of those same keypoints in the second frame.

1. The keypoints and correspondences determined in the detection step are passed on to the tracker
2. Use KLT optical flow to track keypoints
3. **Find position and orientation of the camera**

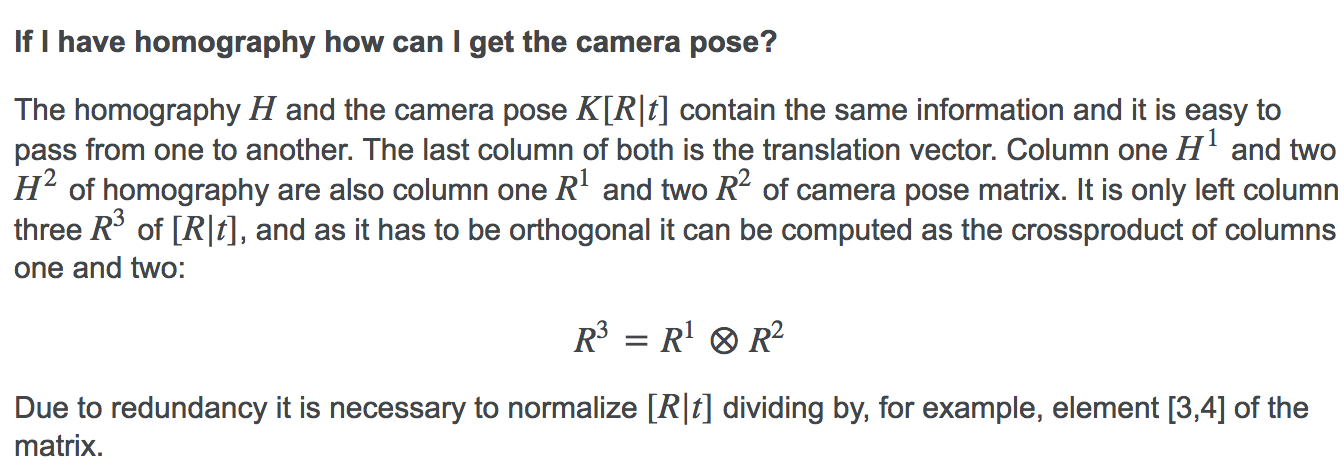
**-Pnp for markers ?**

- Detect the homography transformation between the marker detected in the scene, and a 2D image of the marker we have processed offline.

- **RAN**dom **SA**mple **C**onsensus (RANSAC) is an iterative algorithm used for model fitting in the presence of a large number of outliers.

1. Find [Homography](http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#findhomography) matrix from those matches. RANSAC can be used before to find inliers/outliers in the set of matches [3]
2. Extract **Camera Pose** from homography. [4,5]

[12]:



**4. Use camera pose to augment the book spine with augmented information**

**5. Implement gesture recognition for interacting with the AR menus**

* Fingertip tracking
* Optical flow

**References**

***General***

[1]<https://stackoverflow.com/questions/12283675/augmented-reality-sdk-with-opencv>

***Detection and tracking***

[2] Oliver Toole and Dave Dolben, “Marker Detection and Tracking for Augmented Reality Applications”

***Homography / camera pose estimation***

[3]<https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html>

[4]<https://stackoverflow.com/questions/8927771/computing-camera-pose-with-homography-matrix-based-on-4-coplanar-points/10781165#10781165>

[5]<https://stackoverflow.com/questions/7836134/get-3d-coordinates-from-2d-image-pixel-if-extrinsic-and-intrinsic-parameters-are/10750648#10750648>

***ArUco***

[6]<http://www.uco.es/investiga/grupos/ava/node/26>

***Example of AR Project in OpenCV***

[7]<https://bitesofcode.wordpress.com/2017/09/12/augmented-reality-with-python-and-opencv-part-1/>

***Pose Estimation***

[8] https://docs.opencv.org/3.1.0/dc/d2c/tutorial\_real\_time\_pose.html

***Basic Concepts of Homography Explained With Code***

[9] https://docs.opencv.org/3.4.0/d9/dab/tutorial\_homography.html

[10] https://docs.opencv.org/ref/master/da/d6e/tutorial\_py\_geometric\_transformations.html

[11] <https://rdmilligan.wordpress.com/2015/07/31/3d-augmented-reality-using-opencv-and-python/>

***Step by Step Camera Pose Estimation for Visual Tracking and Planar Markers***

[12]https://dsp.stackexchange.com/questions/2736/step-by-step-camera-pose-estimation-for-visual-tracking-and-planar-markers

***Perspective-n-Point Problem. Example: pose estimation, solvePnP***

[13] <https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/>

***Using Rodrigues to transform the rotation vector into a rotation matrix and then concatenate it with translation vector to get extrinsic matrix. Then multiply extrinsic matrix with the camera matrix to get the projection matrix.***

[14] <http://answers.opencv.org/question/150451/how-to-tranform-2d-image-coordinates-to-3d-world-coordinated-with-z-0/>