# FP.3 Associate Keypoint Correspondences with Bounding Boxes

Prepare the TTC computation based on camera measurements by associating keypoint correspondences to the bounding boxes which enclose them. All matches which satisfy this condition must be added to a vector in the respective bounding box.

It is implemented in the function `cluserKptMatchesWithROI`:

Iterates over all the matched keypoints of the current frame:

1. For a given matched-kpt, gets the corresponding points in both the previous-frame and current-frame.
2. Then. checks if both points belong to the given bounding-box.
3. If yes, calculates the euclidean distance between the two points and add it to the `distanceVec`
    a. The idea is that, if two points are matched then they should be in a similar position in the image; hence, the euclidean distance should be small and for all truly matched kpts should be of similar value.
    b. Thus, later we can find outlier kpts by comparing their euclidean distance (should be much larger than average).
4. Calculate `mean` and `stddev` from the `distanceVec`
5. For each pair of points, (from current frame and previous frame) that belong to the bounding-box; check if the difference of the distance from the mean is smaller than the `2*stddev`: `abs(distance - meanDistance) < 2.0*stddevDistance`.
    a. If yes, then it's a in-lier and hence we would add it to the final list.

```
// key matched points that are within 2*stddev distance from the mean
    for (const auto& match : matchesInROI)
    {
        cv::KeyPoint* prevPt = &kptsPrev[match.queryIdx];
        cv::KeyPoint* currPt = &kptsCurr[match.trainIdx];
        double distance = sqrt((prevPt->pt.x - currPt->pt.x)*(prevPt->pt.x - currPt->pt.x) + (prevPt->pt.y - currPt->pt.y)*(prevPt
        if (abs(distance - meanDistance) < 2.0*stddevDistance)
        {
            boundingBox.kptMatches.push_back(match);
        }
    }
```

6. print how many outliers found and removed from the list:

```
Removed 3 kpt outliers from bounding-box 1
```