FP.1 Match 3D Objects

Matching Bounding-Boxes with the highest number of keypoint correspondences implemented in `matchBoundingBoxes` function and is solved in the following steps:

1. **Bounding-Box Overlap:** For each pair of Bounding-Boxes from current and previous frames, count the number of matched keypoints that belongs to both and save the results to `map<pair<int, int>, int> countBoxOverlap`.
    - For Robustness, we are considering a smaller-box (with `shrinkFactor = 0.1` similar to the section where we assign Lidar points to bounding-boxes) when counting the number of kpt-matches that belongs to the bounding-box. This avoids having too many outlier points around the edges of the bounding-box which can make matching bounding-boxes erroneous.
2. **Bounding-Box Matching:** Done using a helper function: `helperMatchCurrBox`.
For each bounding box in the current frame, it finds the best matching bounding box from the previous frame.
    a. For current bounding-box, we loop over previous bounding-boxes and keep track of the one that has the most overlap using `int` `maxOverlapMatches` and `pair<int, int>` `bestMatchPair` variables.
    b. If a previous bounding-box is already paired (tracked with `alreadyMatchedBB` variable), check the confidence ratios.
    c. The confidence ratio: `numOverlapKpts / totalKpts` (threshold value is set to `0.5`)
    d. If the current confidence ratio is larger than the one that is already paired we go further and check if it also has more overlap than the one we found until now. If yes, update the `bestMatchPair` and `maxOverlapMatches` variables.
        i. But, **don't update** the `alreadyMatchedBB` variable since we are still in for loop and best match is not finalized yet.
    e. After the loop over previous bounding-boxes finished, check whether confidence ratio for the best matched found is actually larger than the threshold.
    f. If it passes the threshold, we check whether the chosen previous bounding-box as best match is already paired with other bounding-box.
    g. If yes, it must be that the new confidence ratio is higher than the old one (as it was checked before).
        i. Therefore, we replace the old pair with the new pair.
        ii. Find a new match for the one that is being replaced, recursively using the helper function `helperMatchCurrBox`.
    h. If no, it's a new and unmatched pair, so added it to the `alreadyMatchedBB` variable.
    i. Final best bounding-box matches is stored in `bbBestMatches` variable.
3. **Visualization:** added extra input `bVis` and implemented visualization for bounding-box matches for debugging purpose. Here is the sample image (each bounding-box pair has same color):