# EXERCISE 3: GLOBAL OPTIMIZATION

GCT722 MATHEMATICAL METHODS FOR VISUAL COMPUTING  
20183151 Chaelin Kim

## PART 1: BRANCH AND BOUND FOR MAXIMUM COVERAGE

### **Description of implementation**

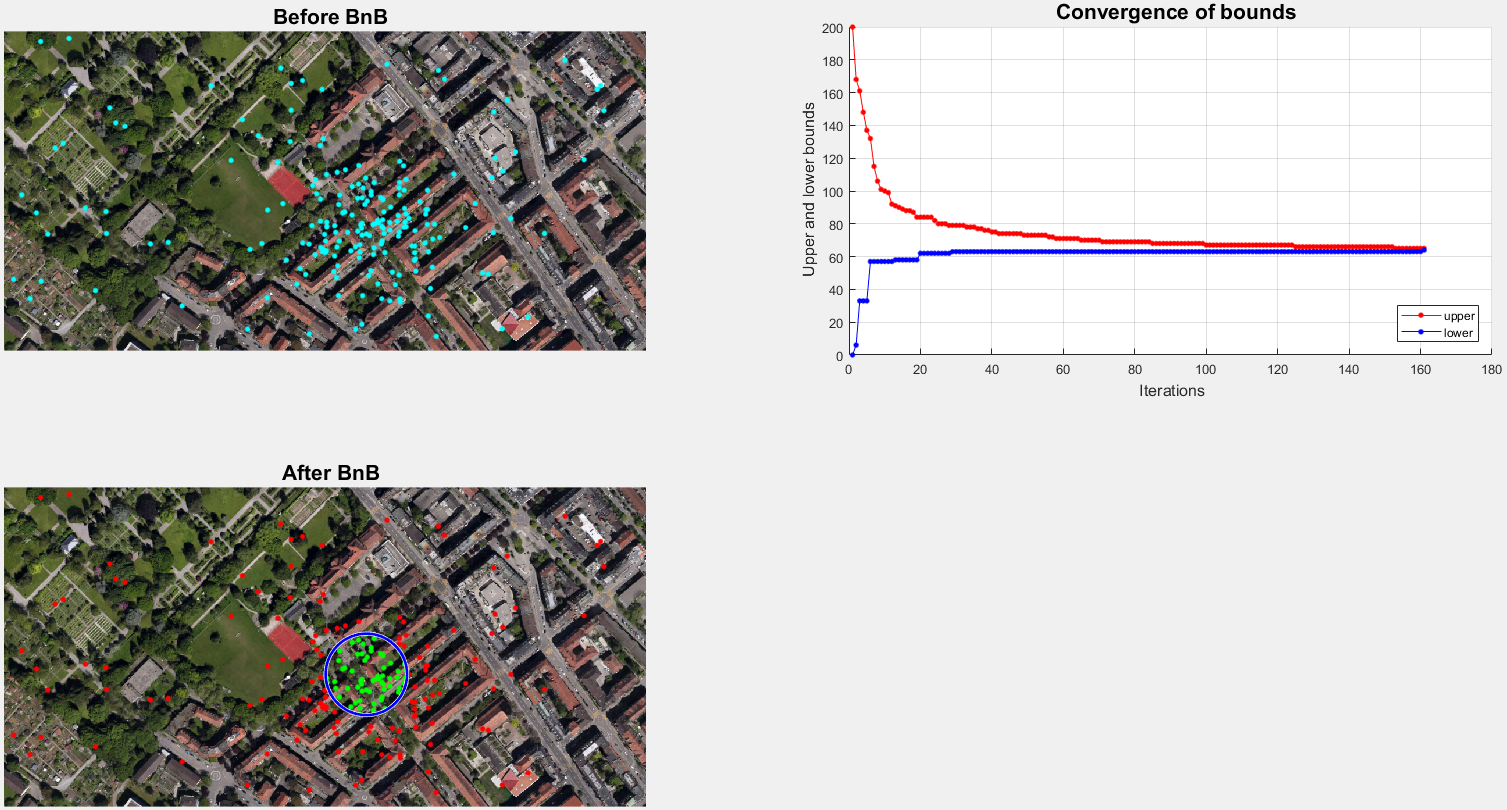
There are 1 script file & 2 function files for the exercise branch and bound.

* **Script file**
* **main.m**  
  : This is the main script that execute branch and bound, find inlier and outlier points, and draw images and convergence plot.  
   Branch and bound is done in while loop. At each iteration, take the best candidate in the list, split the space into two children, put them into the list and remove the parent space from the list. Then find the highest lower bound and the lowest upper bound in the list, and update the best bound values with them. After that, remove all the elements in the list that upper bound is lower than current best lower bound. The iterations stop when the current lower bound and upper bound are nearer than 1.  
   If the iteration is over, compute inlier and outlier points in the result antenna location and draw the plot on the image.
* **Function files**
* **makeChilds.m**  
  : This function is used to split the current space into two children along the longest dimension. It returns first and second child node that lower and upper bound is calculated in space.
* **calBounds.m**  
  : This function is used to calculate weighted centroids of source and target control points.

### **Instructions for running**

1. Open the file “main.m” in Matlab.
2. Execute that file.
3. The window that shows the result images and graph is opened.
   1. The left part shows result images of applying BnB before and after.
   2. The right part shows the result plot of convergence of bounds

### **Screenshots**



### **Discuss the results**

I have implemented image deformation with the affine, similarity and rigid transformations according to the reference paper “Image Deformation Using Moving Least Squares [Schaefer et al. 2006]” with Matlab.

Affine transformations contain shear and non-uniform scaling. Because it is simple than other transformations, the points except the control points are also very affected by direction of transformation for control points. So the result image appears to be larger and stretched globally. Similarity transformations contain translation, rotation and uniform scaling, so it preserves angles in the original image better than affine. The result is more natural than Affine, but the result is stretched a little over the original because of scaling. Rigid transformations, which removes the scaling from the similarity transformation, show the most realistic result image among them. The parts except control points are almost not deformed.

Through the above transformations, I have implemented forward warping and backward warping. Forward warping is a method of calculating the coordinates of the target with respect to the source coordinates to fill the color. Because the pixels in the image are integer, they need to be rounded. Then some pixels in the result image cannot be computed and holes (unknown values) are shown in the image. In order to correct, backward warping can be used. Backward warping is the reverse of forward warping. Since this can be seen as a transformation from target points to source points, I implemented it by reversing the source and target in the forward warping. The results are a little different with those of forward warping, but each result are performed well in each transformation. In addition, the bilinear interpolation is needed for antialiasing.