1. 2 weeks ago, the obstacles in a map were built by circles, and they are terribly inefficient (Fig 1). Now I use polygons instead and it has much better efficiency towards obstacles avoidance. Fig 2 show a comparison of real micro channels and a data map. The new micro channels were made by our new acrylic material and double-side sticker.

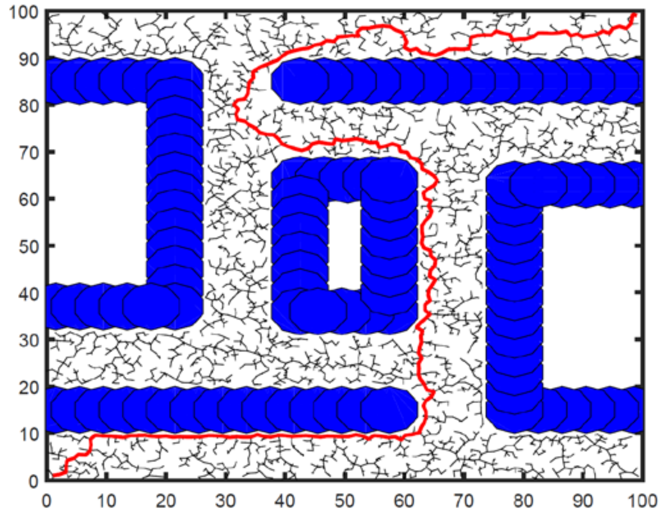

Fig. 1

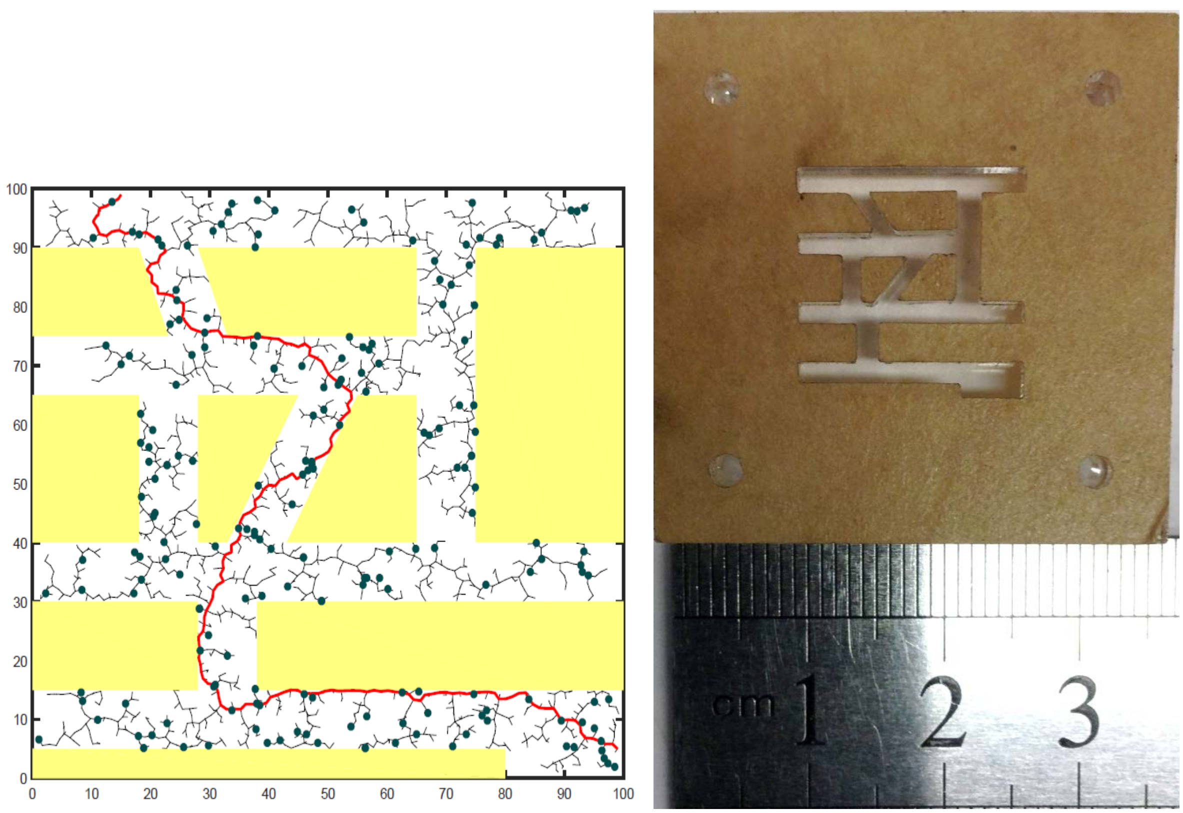

Fig. 2

2. In fig 2, dark blue points stand for random particle swarms. And they can move following inputs of 4 directions with aggregation next to obstacles. I tried random walk input and it has nice aggregation for the first few hundreds of steps. I'll plot the changes of total cost to a target and entropy of the swarms next week. These two variables play key roles in determining a control input.

3. Search neighbors by binning data vs brute-force
   Total random spatial data: 10^6
   Map size: 100*100
   Binning partition along each side: 10 bins
   Bin_length = 100/10=10
   Total bins: 100

   Given a random spatial data point, search its neighbor in the range of 5, namely 0.5*bin_length. In fig 3, we have almost the same output of neighbors, and search time by binning data is about **10%** of the brute-force search.

   In fact, if we shrink the search range to 0.2*bin_length, we can see in fig 4, search time by binning data is about **1%** of the brute-force. Note that fig 4 left does not have a complete circle range because it only searched 1 bin in this situation (range is small, and probably the point is near the boundary.). This can be solve by add some optimization conditions later.

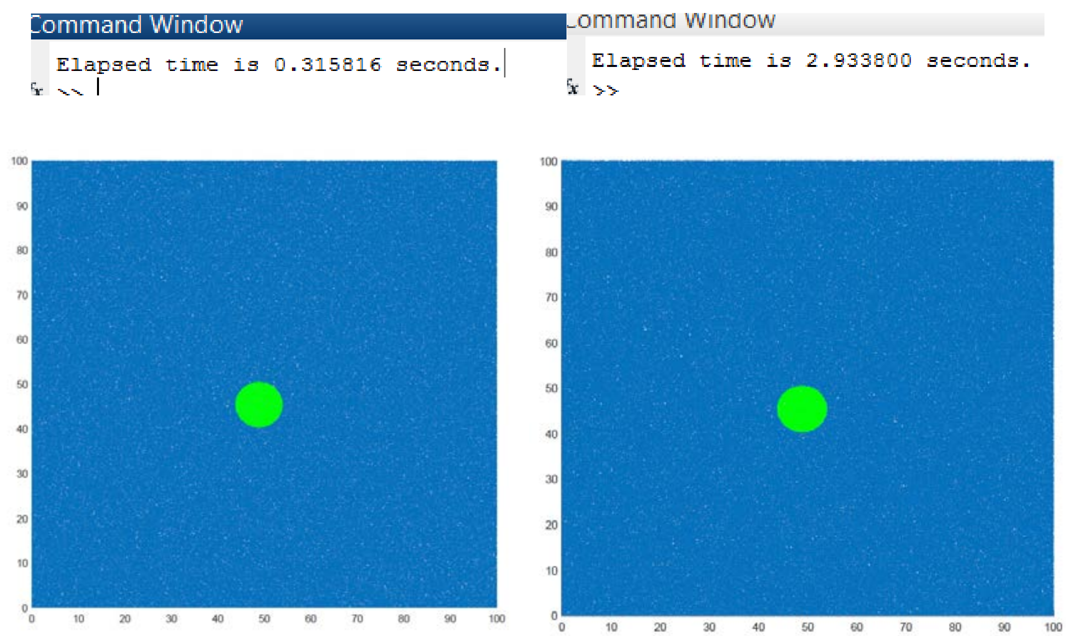

Fig 3: left: binning data; right: brute-force

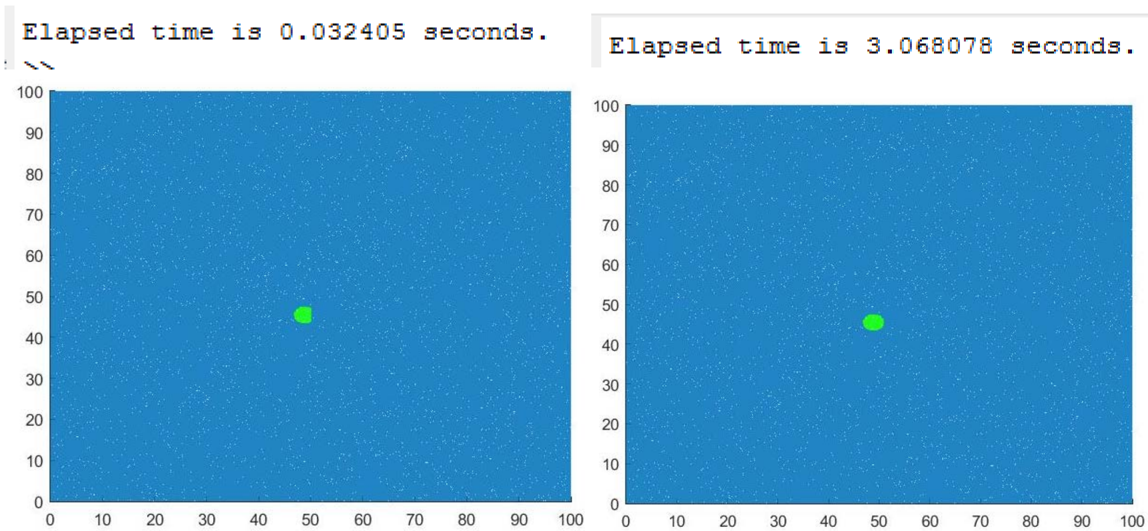

Fig 4: left: binning data; right: brute-force