# Report 2: DCCP Applied in Obstacle Avoidence

Wu Lan

2019/7/25

## 1 Introduction

In this report I implement the disciplined convex- concave programming (DCCP), which combines the ideas of disciplined convex programming (DCP) with convex-concave programming (CCP). The Convex-concave programming is an organized heuristic for solving non-convex problems that involve objective and constraint functions that are a sum of a convex and a concave term (Gu & Boyd, 2016). Generally speaking, previously we can only solve problems with convex object function and convex domain. Thus when applying it to real applications like route-planning by the cvxpy package in python, it largely restricts the range of limiting conditions. Since obstacles are ubiquitous in real cases, the route-planning system should not depend on an algorithm which could not solve problems with restrictions. Fortunately, based on the Hartman theorem, the Convex Optimization Problem is proved to have more extensive implications and it is possible to use similar method solving DCCP problem. With the DCCP package in python, we are able to use several simplified functions calculating an optimal route object to verified limiting conditions.

## 2 Method

### 2.1 Turn Obstacles into Circles

**Algorithm 1** Obstacle Analyzing

```
1. **Take** a picture of the road condition including car and obstacles
2. **Load** the color image and access the pixel value by its row and column coordinates
3. **Distinguish** obstacles and find the contour of it
4. **Find** the smallest circle to include each obstacle
5. **Output** the origin and radius of each circle
```

- Code:

```
import cv2
import numpy as np
from picamera import PiCamera
from picamera.array import PiRGBArray
rawCapture = PiRGBArray(camera)
camera.capture(rawCapture,'bgr', use_video_port = True)
image = rawCapture.array
x_size = image.shape[1]
x_pic_length = rangefinder/range_pic_ratio
output_gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(output_gray,threshold,255,cv2.THRESH_BINARY
                                     )
useless, contours, hierarchy = cv2.findContours(thresh,cv2.
                                     RETR_EXTERNAL,cv2.
                                     CHAIN_APPROX_SIMPLE)
(x1,y1),radius1 = cv2.minEnclosingCircle(cnt1)
```

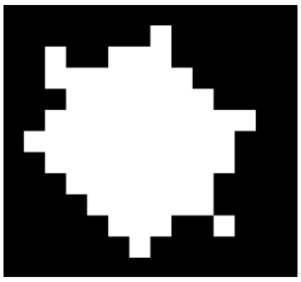
Figure 1: Obstacle Example

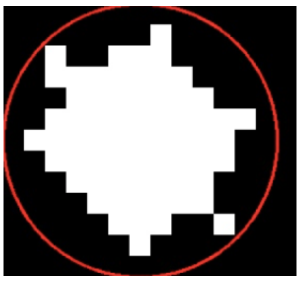
Figure 2: Finding the Smallest Circle

### 2.2 Solve the Shortest Path

- Problem Description
  The goal is to find the shortest path connecting points a and b in $R^d$ that avoids m circles, centered at $p_j$ with radius $r^j$, j = 1,...,m. After discretizing the arc length parametrized path into points $x_0$ , . . . , $x_n$ , the problem is posed as

$$\begin{array}{ll} \text{minimize} & L \\ \text{subject to} & x_0 = a, \quad x_n = b \\ & \|x_i - x_{i-1}\|_2 \leq L/n, \quad i = 1, \dots, n \\ & \|x_i - p_j\|_2 \geq r_j, \quad i = 1, \dots, n, \quad j = 1, \dots, m, \end{array}$$

  where L and $x_i$ are variables, and a, b, $p_j$ , and $r^j$ are given.

- Code:

```
import cvxpy as cvx
import numpy as np
import dccp
x = cvx.Variable(d,n+1)
L = cvx.Variable()
constr = [x[:,0] == a, x[:,n] == b]
for i in range(1,n+1):
  constr += [norm(x[:,i] - x[:,i-1])<= L/n]
  for j in range(m):
    constr += [norm(x[:,i] - center[:,j]) >= r[j]]
prob = Problem(Minimize(L), constr)
prob.solve(method = ' dccp ' )
```

- Sample Outcome (Cited)

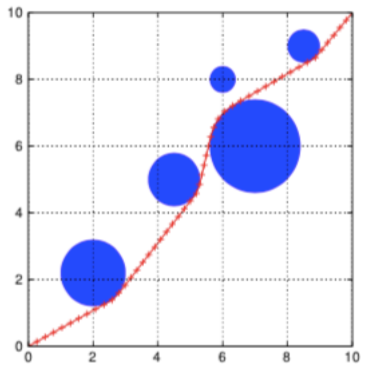

Figure 3: A Cited Sample with d=2, n=50

## 2.3 Simulation Setting

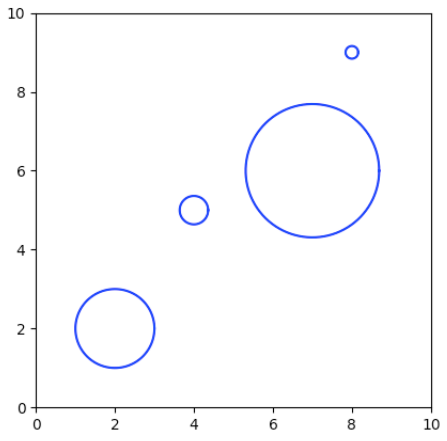

Figure 4: Simulation Setting with Four Obstacles

where $p_1 = (2, 2)$, $p_2 = (4, 5)$, $p_3 = (7, 6)$, $p_4 = (8, 9)$, $r_1 = 1$, $r_2 = 0.6$,$r_3 = 1.3$, $r_4 = 0.4$, n = 50, d = 2, $x_0 = (0, 0)$, and $x_n = (10, 10)$.

## 3 Outcome