

编程作业一实验报告

崔士强 PB22151743

2024 年 3 月 17 日

1 算法

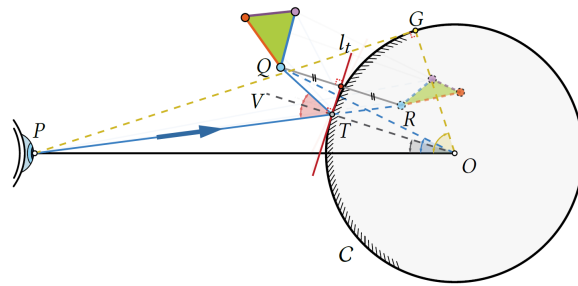


图 1: 示意图

在 Computational Mirror Cup and Saucer Art 中有以下结论:

当 $\angle POT$ 从 0 增加到 $\min \angle POQ, \angle POG$ 时, 其中 $G \in \mathbb{C}$ 且满足 $PG \perp OG$, $\angle PTV$ 严格增加而 $\angle QTV$ 严格减小, 并且 $\angle PTV - \angle QTV$ 从负值严格增加到正值.

因此, 我们可以采用二分法找到点 T , 使得 $\angle PTV - \angle QTV = 0$. 以 $\angle POQ$ 为上界, 0 为下界, 取角平分线, 若 $\angle PTV - \angle QTV > 0$, 则取左半边, 否则取右半边, 直到 $\angle PTV - \angle QTV$ 足够小.

Algorithm 1 Binary search for T

Input: x_P, x_Q, y_Q

Output: $T \in \mathbb{C}$, $\angle PTV - \angle QTV = 0$

1: $l \leftarrow 0$ 2: $r \leftarrow \angle POQ$ 3: **while** $\angle PTV - \angle QTV > \varepsilon$ **do**4: $m \leftarrow (l + r)/2$ 5: **if** $\angle PTV - \angle QTV > 0$ **then**6: $l \leftarrow m$

```

7:     else

```

8: $r \leftarrow m$ 9: **end if**10: **end while**11: **return** T

找到 T 后，通过延长 PT 可以找到 R ，延长的长度为 QT

2 实验结果

实验中对 8 个测试样例进行计算,通过`np.set_printoptions(precision=20, suppress=True)`设置输出精度以及格式. 上述算法中 $\epsilon = 10^{-8}$. 程序实现中使用递归实现二分法查找, 输出结果包括递归深度, T, R 的坐标以及每个样例的计算时间. 结果如下图所示:

```
def find_image():
    if delta < 0:
        return find_t(mid_angle, ceiling, error, depth+1)

    1个测试
    def find_image():
        d1 = np.linalg.norm(P-T)
        d2 = np.linalg.norm(Q-T)
        vector = (T-P)*d2/d1
        return vector + T

    # Read 8 testcases
    for i in range(0, 8):
        P_inputList = input().strip().split(' ')
        Q_inputList = input().strip().split(' ')

        start_time = time.time()

        P = np.array([float(P_inputList[0]), float(P_inputList[1])])
        Q = np.array([float(Q_inputList[0]), float(Q_inputList[1])])
        print('-----Testcase', i + 1, '-----')
        T = find_t(0, pointAngle(Q), error=0.0000001)
        print(T)
        print(find_image())

        end_time = time.time()

        print('Time:', end_time - start_time, 's')
        print()

    for i in range(0, 8)
```

```
-----Testcase 1 -----
Recursion depth: 42
[[-0.9999999999999999 0.00000199998196798966]
 [0.0000088127590278649 1.9999999999999999]]
Time: 0.000768999176025391 s

-----Testcase 2 -----
Recursion depth: 46
[[-0.9999999999999999 0.00000099999999999985]
 [-0.9999999999999999 0.00000099999999999999]]
Time: 0.0006690025329589844 s

-----Testcase 3 -----
Recursion depth: 23
[[-0.9892790406550636 0.1468375969420106]
 [1.18242397072476637 0.38258963826949956]]
Time: 0.00037789344787597656 s

-----Testcase 4 -----
Recursion depth: 26
[[-0.9226152822830498 0.3857214550639873]
 [-0.7869281378807857 0.41891685822882775]]
Time: 0.00040221214294433594 s

-----Testcase 5 -----
Recursion depth: 26
[[-0.0270283302809022 0.562160244870431]
 [8.380295758335354 2.944147841113247]]
Time: 0.0003960132598876953 s

-----Testcase 6 -----
Recursion depth: 23
[[-0.9874084505423346 0.15819150355687855]
 [1.187635841642585 0.3291341555532053]]
Time: 0.0003689657287597656 s

-----Testcase 7 -----
Recursion depth: 25
[[-0.9593115089613334 0.28234983402548076]
 [0.3042141580296597 0.3218110169561312]]
Time: 0.0005979537963867188 s

-----Testcase 8 -----
Recursion depth: 25
[[-0.970865736488883 0.2428424737262458]
 [7.00089435931097 0.2447345861687136]]
Time: 0.0004229545593261719 s

进程已结束, 退出代码为 0
```

图 2: 计算结果

3 实验分析

3.1 精确度

本算法的精度问题主要来自于二分法中设置的判定阈值 ϵ 以及 `int` 类型的精确度.

3.2 稳健性分析

1. 此程序并未对不合法输入进行处理, 默认输入的两个点符合规则
2. 对于边缘情况, 程序中通过提高输出精度的方法保证结果准确度. 可以看到对于边缘情况的处理需要更多层的递归.