Студкемп по робототехнике и ИИ

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Реализация алгоритмов планирования пути

Пелевин Владимир

к.пед.н., доцент, УрФУ

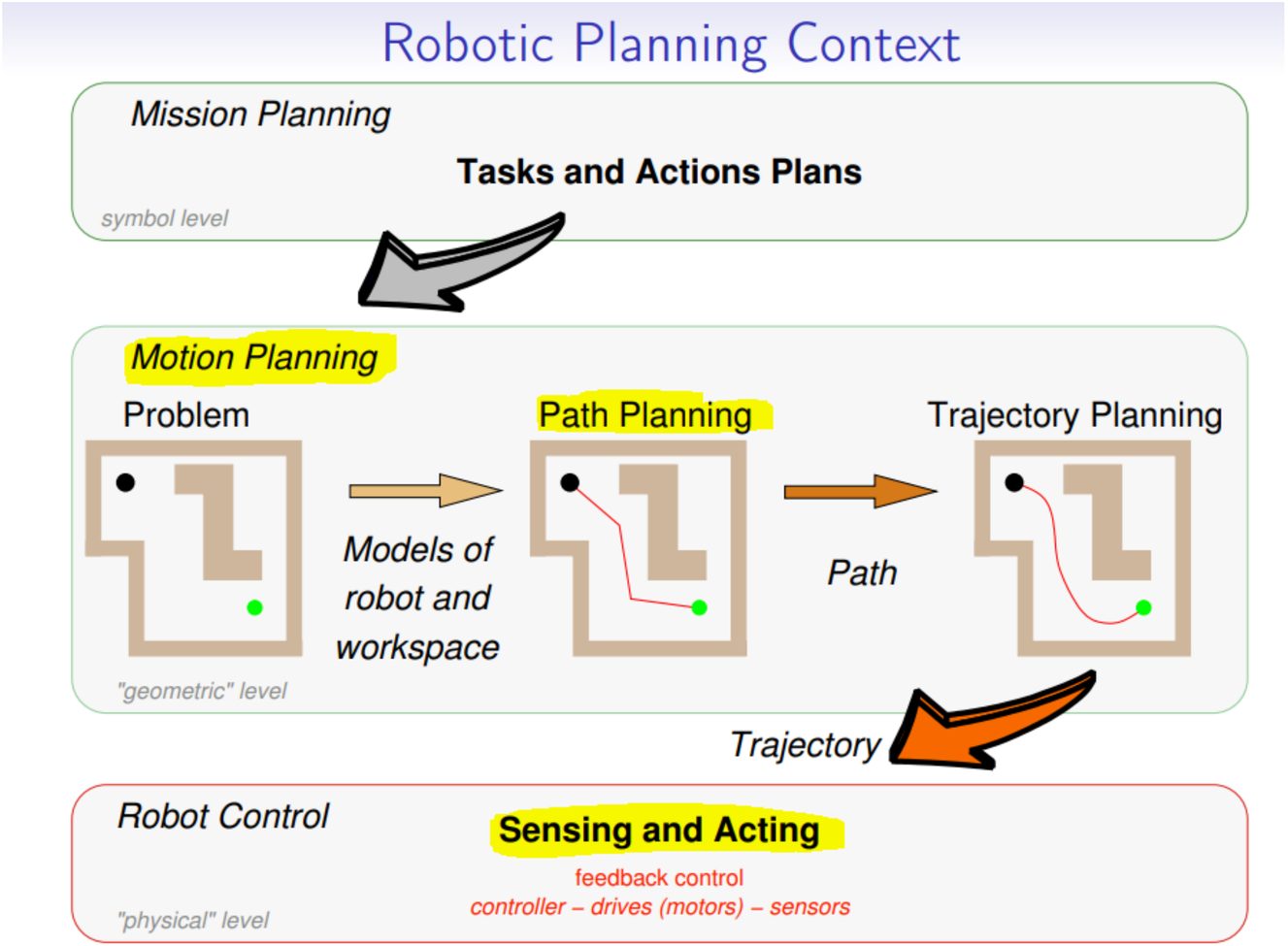
инженер, ООО Микроэлектроника и Робототехника

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Планирование пути и движения

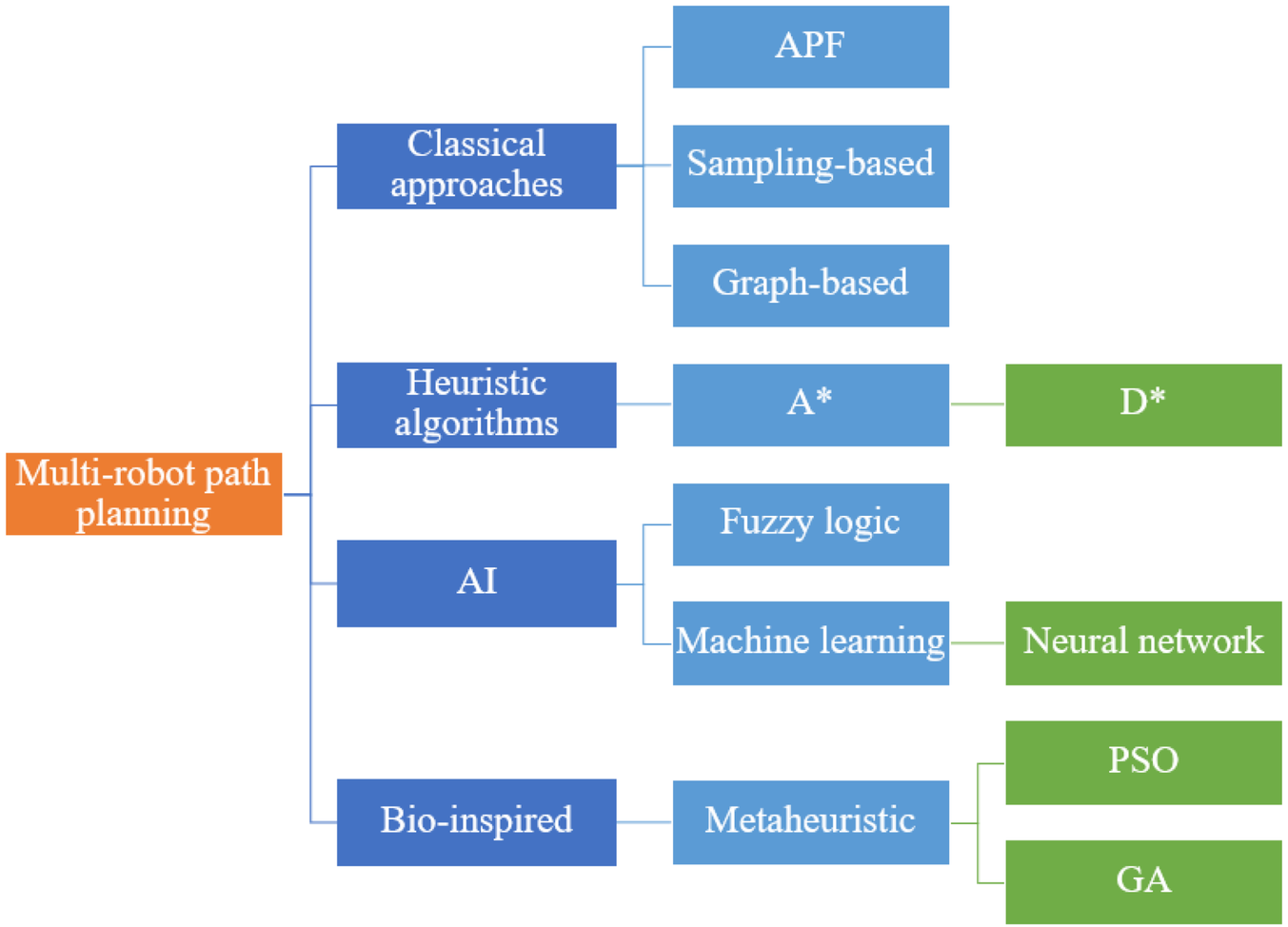
Планирование пути и движения



Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

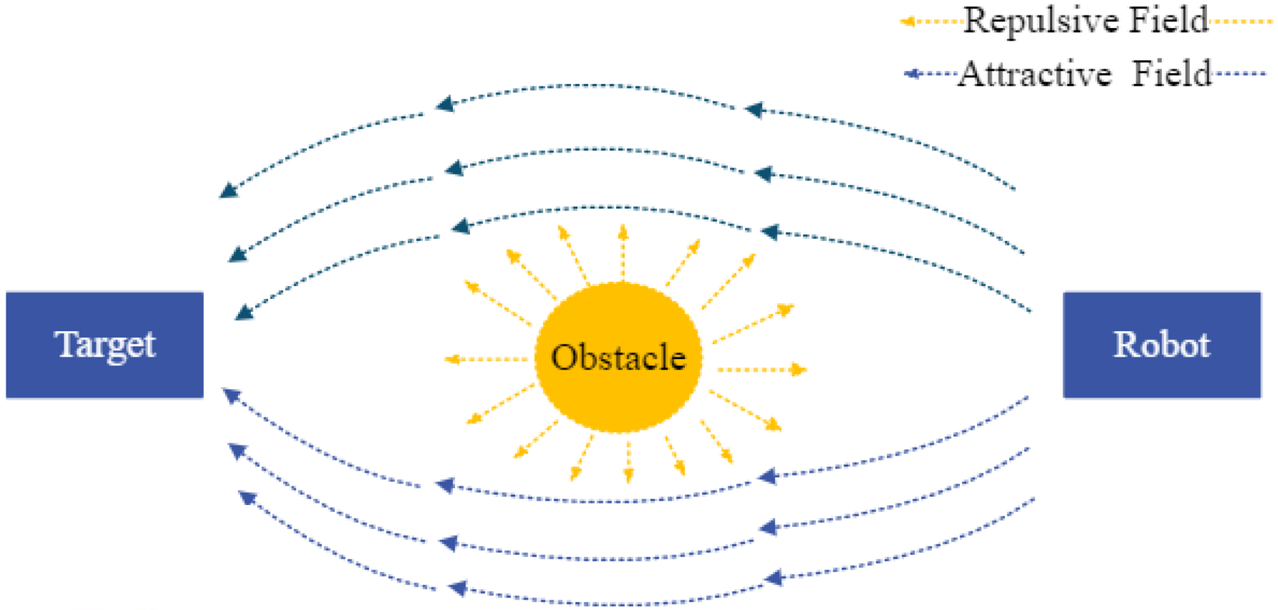
Виды алгоритмов построения пути

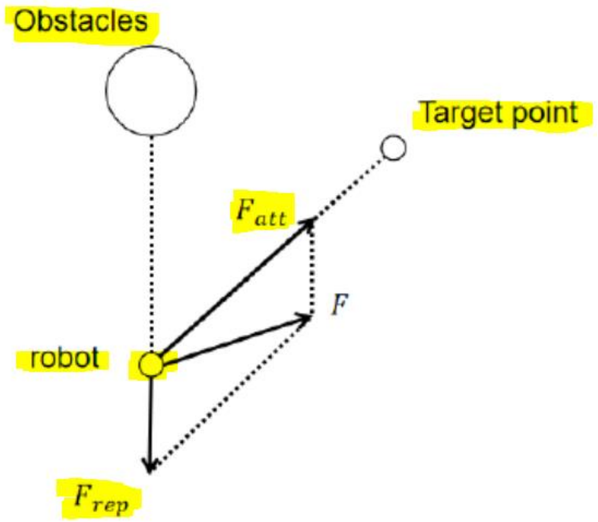


Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

APF





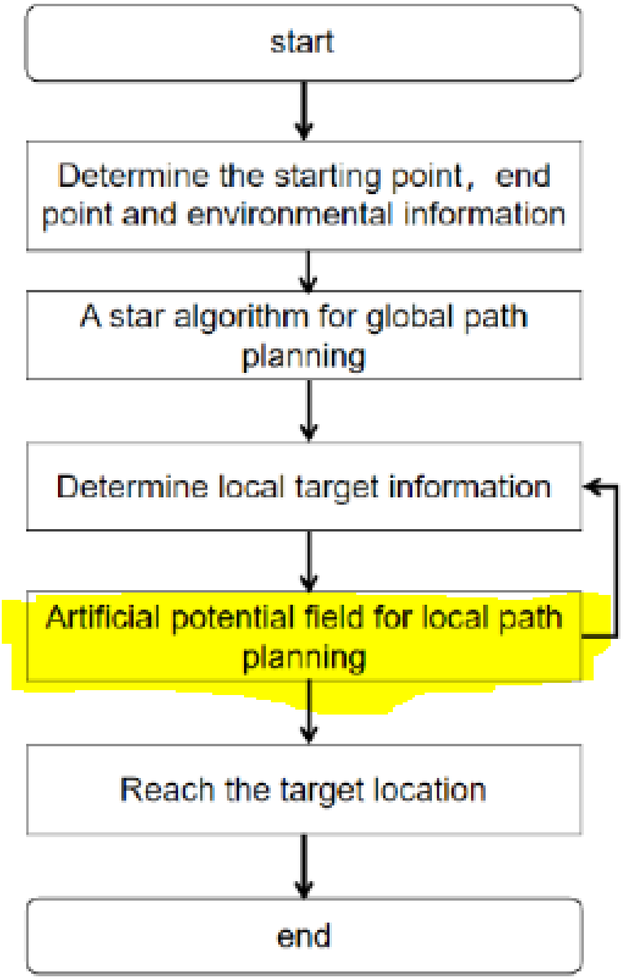
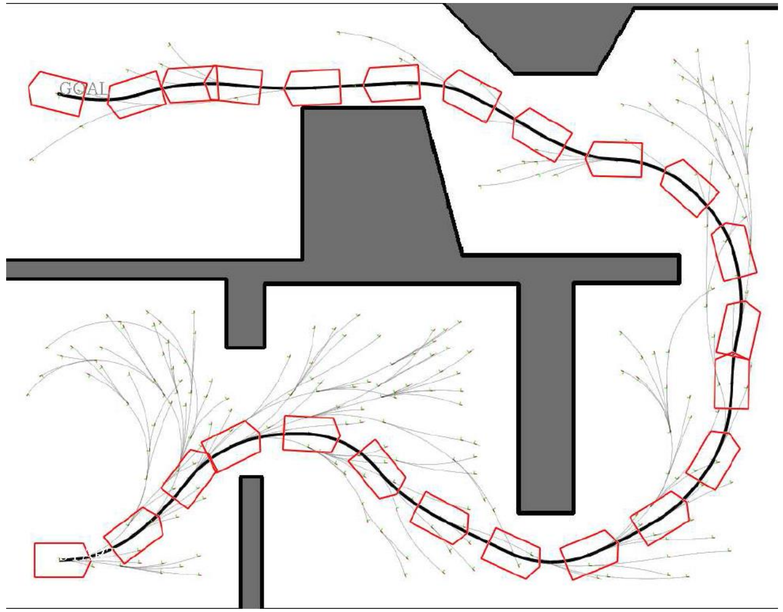


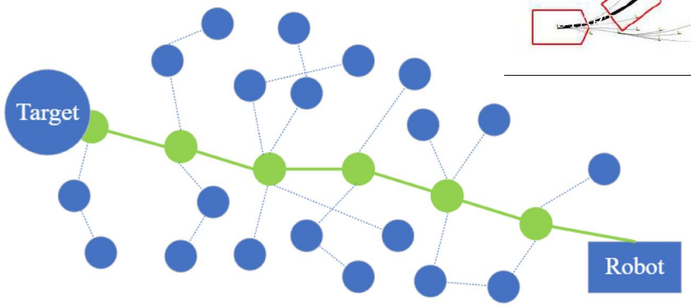
Figure 2. Mixed path planning flow chart.

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Sampling-based rapidly exploring random tree (RRT)

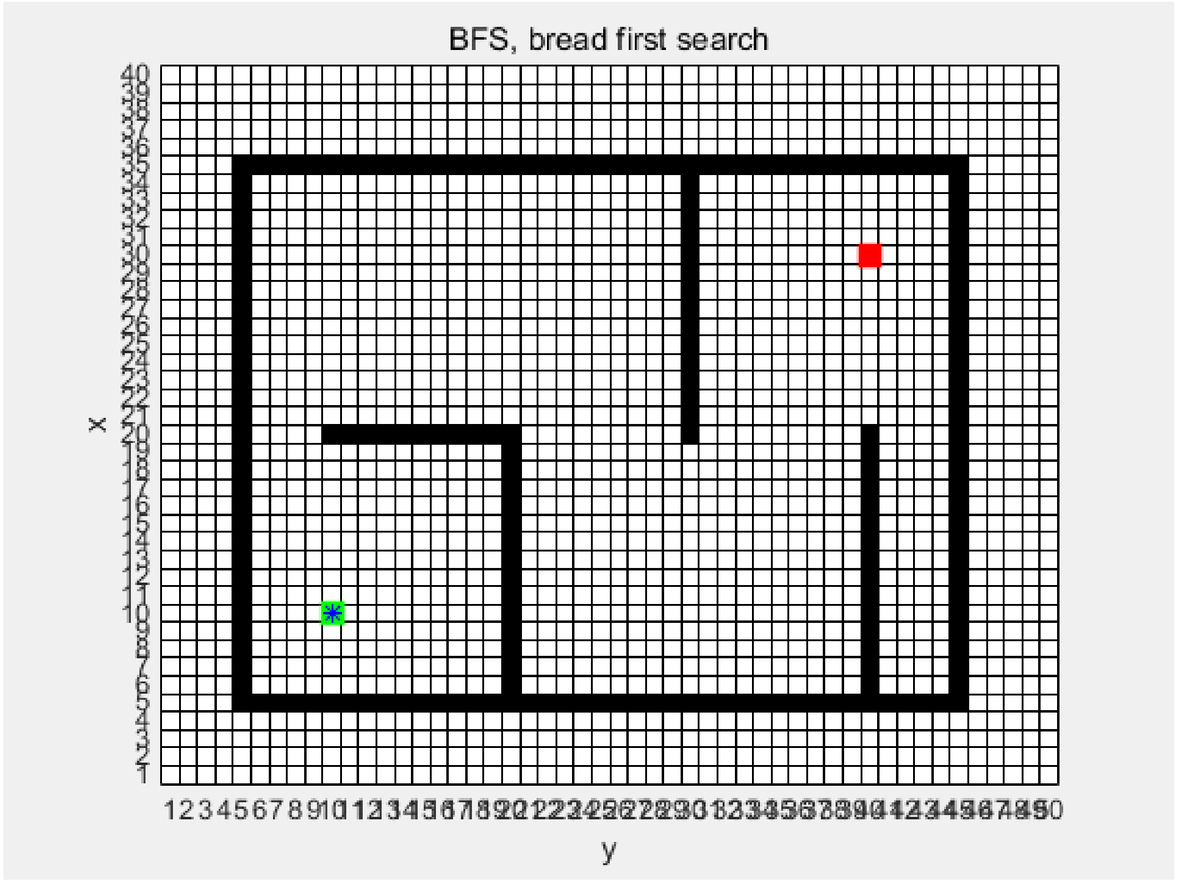


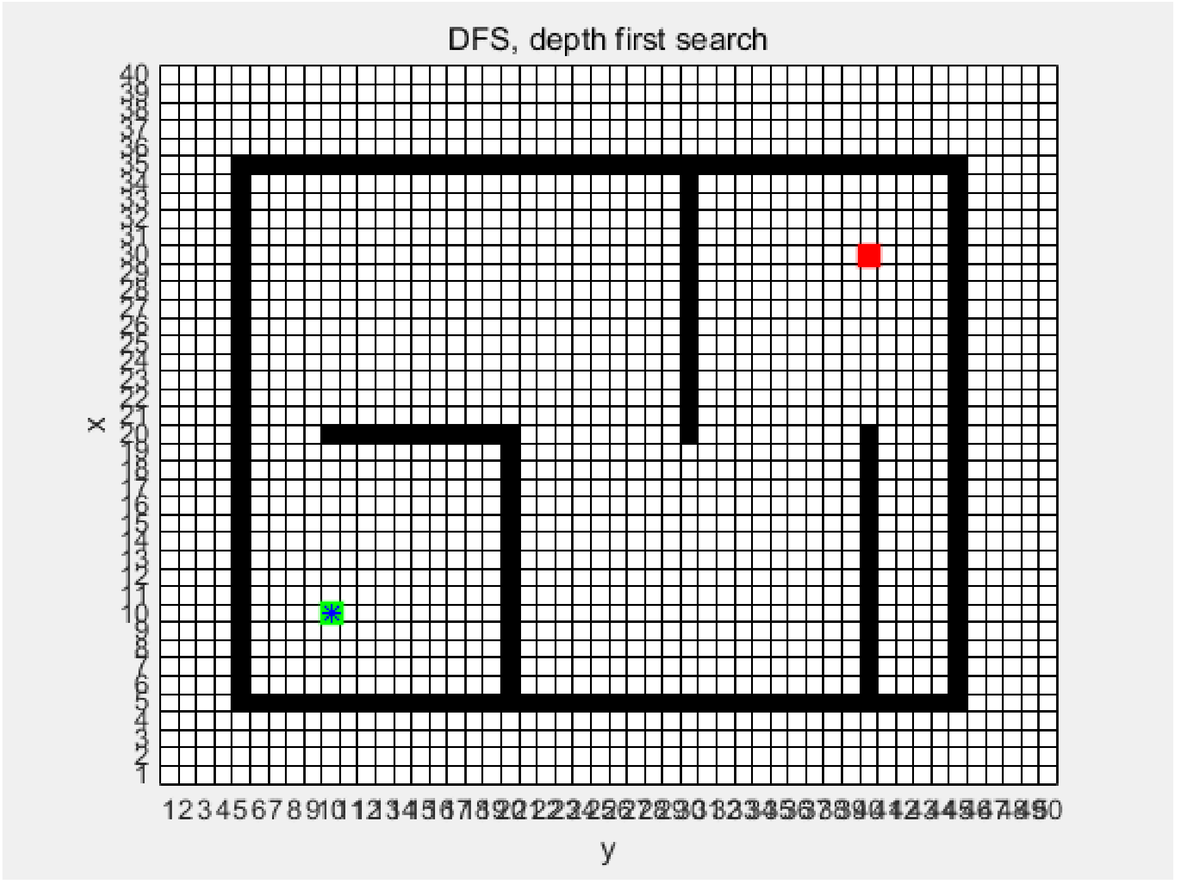


Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Search-based





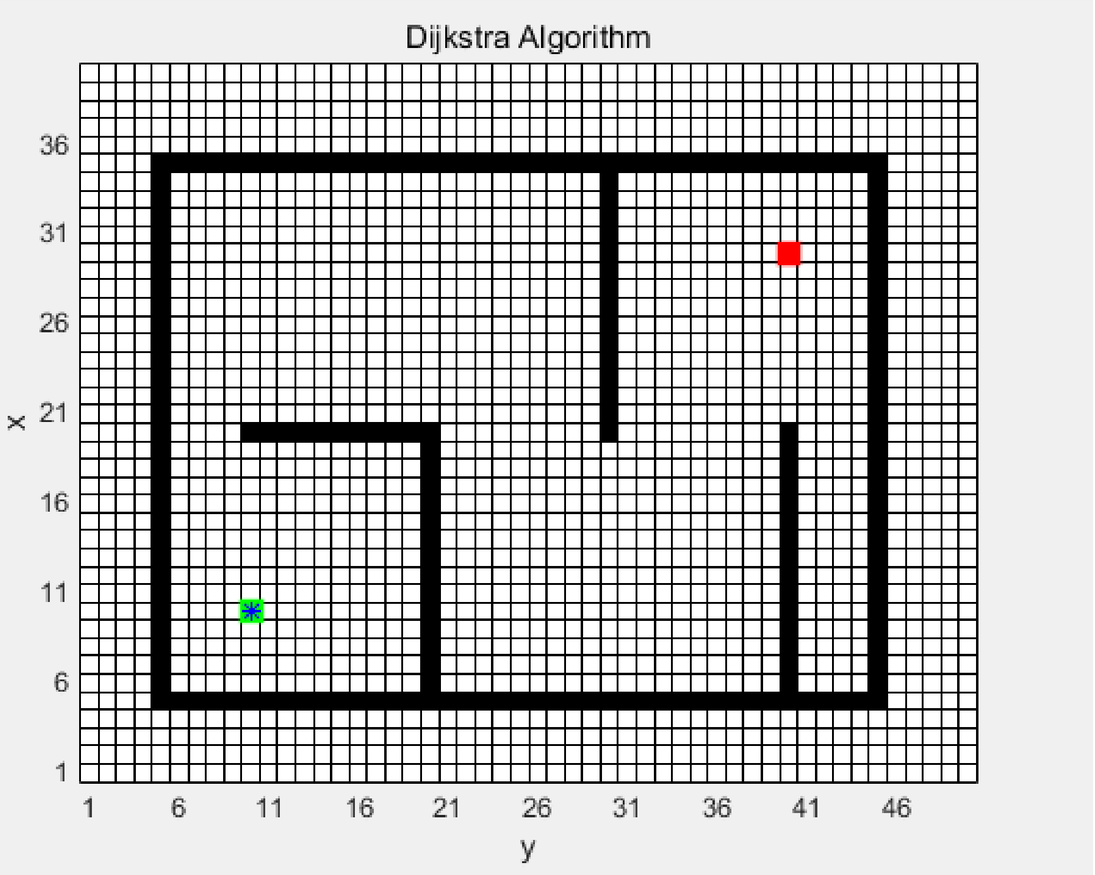
Обход графа в ширину (BFS)

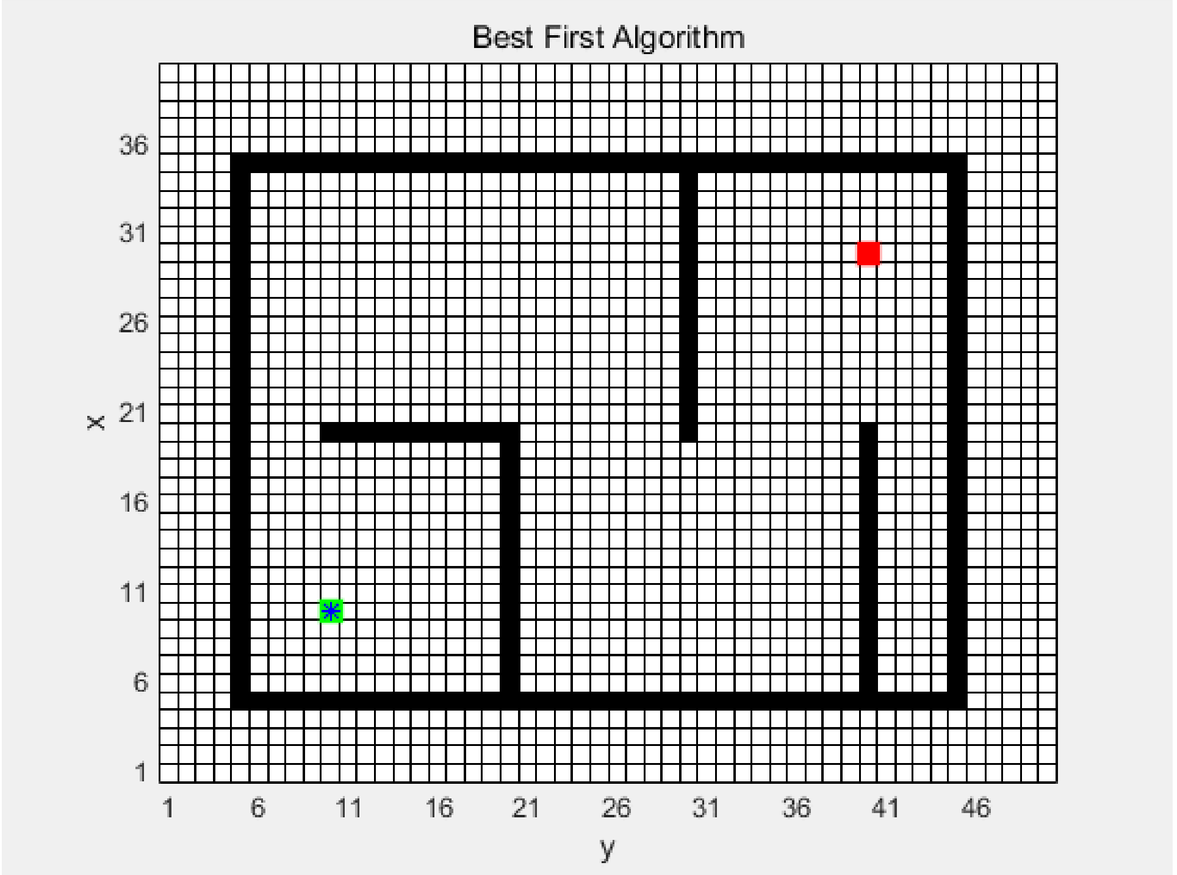
и глубину (DFS)

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Search-based



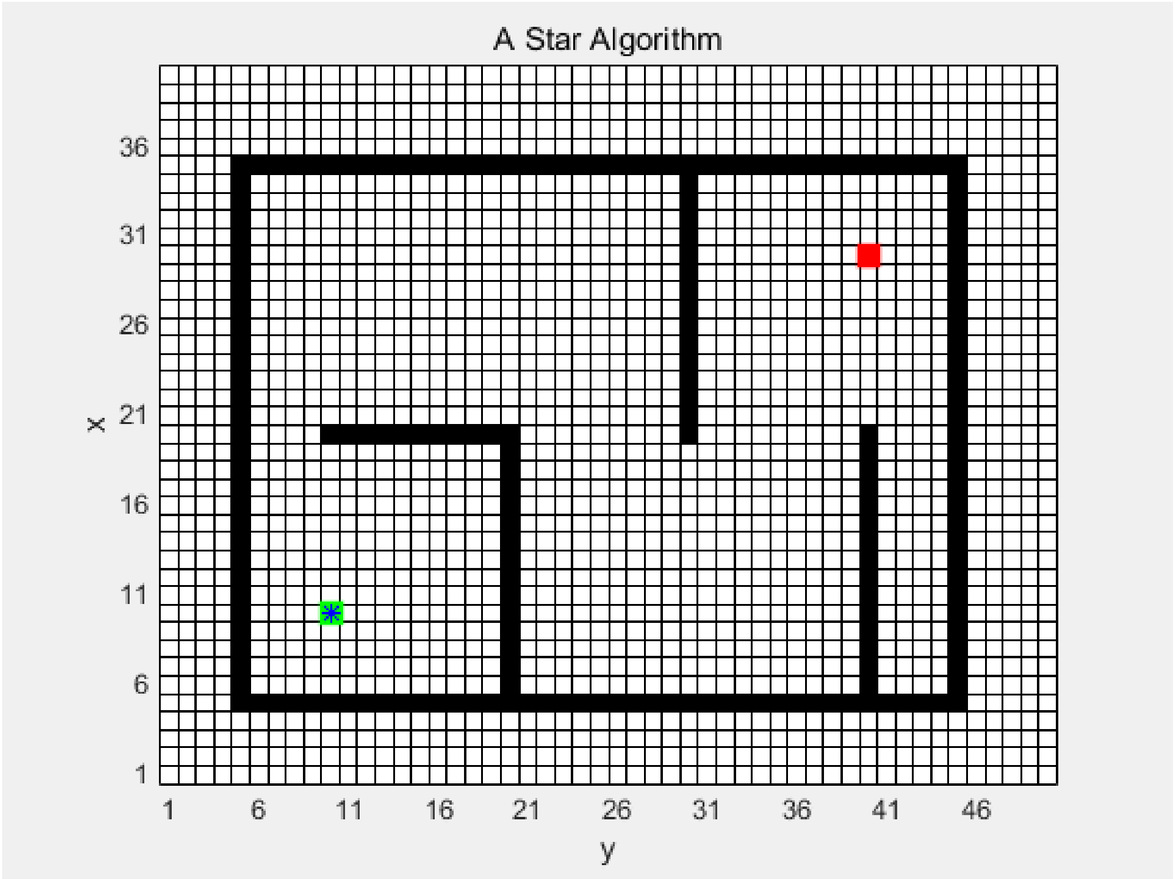


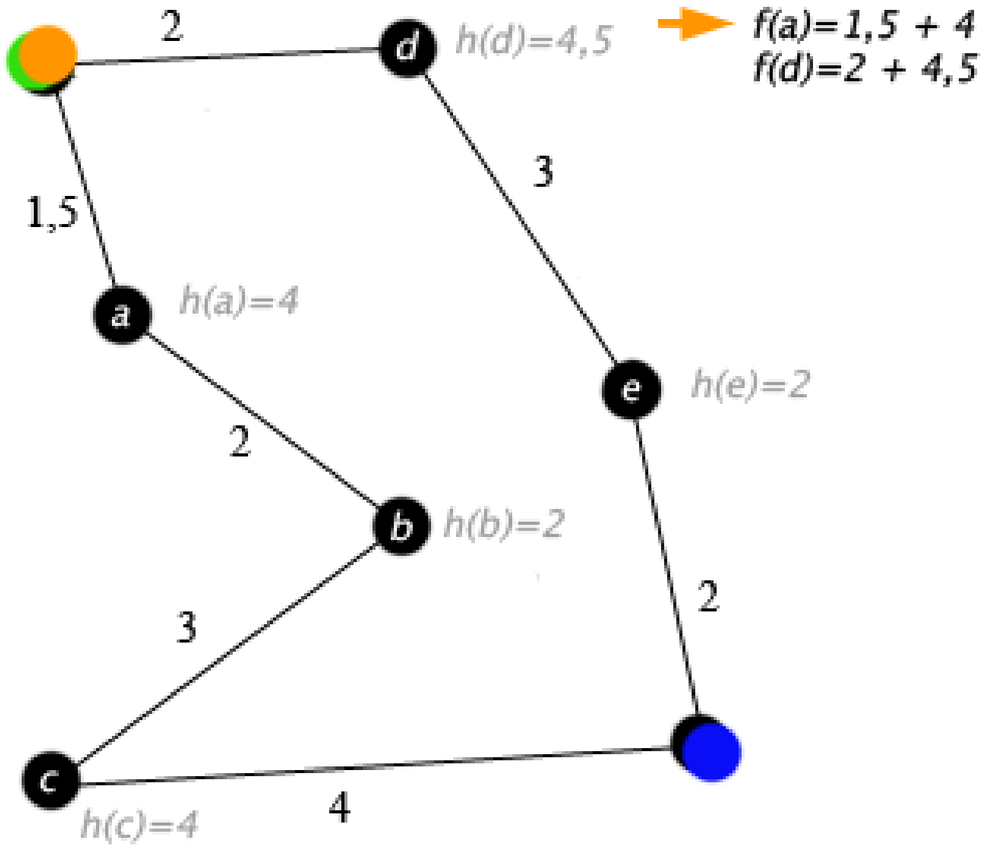
Алгоритм Дейкстры

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

Search-based



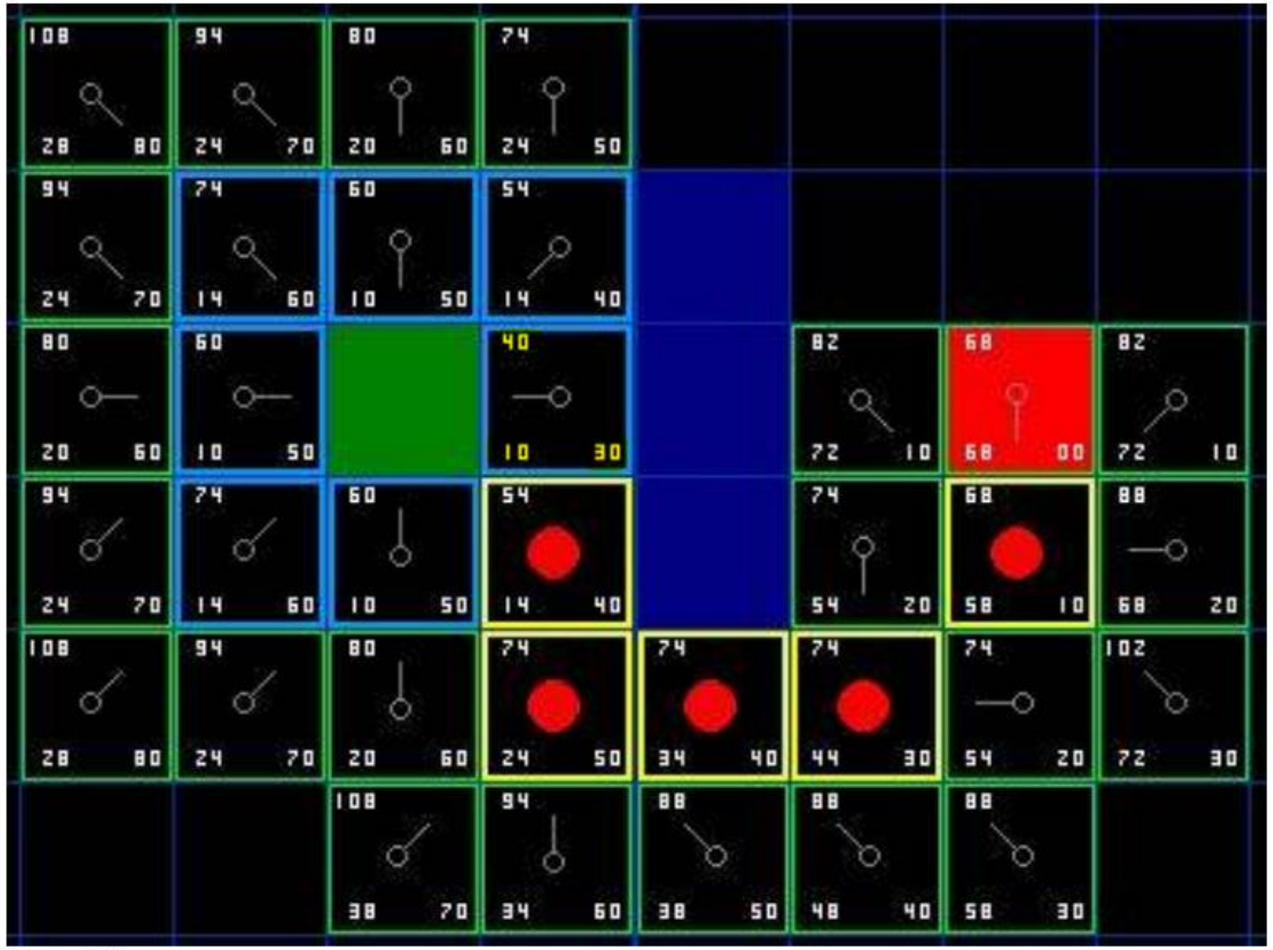


Алгоритм A*

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

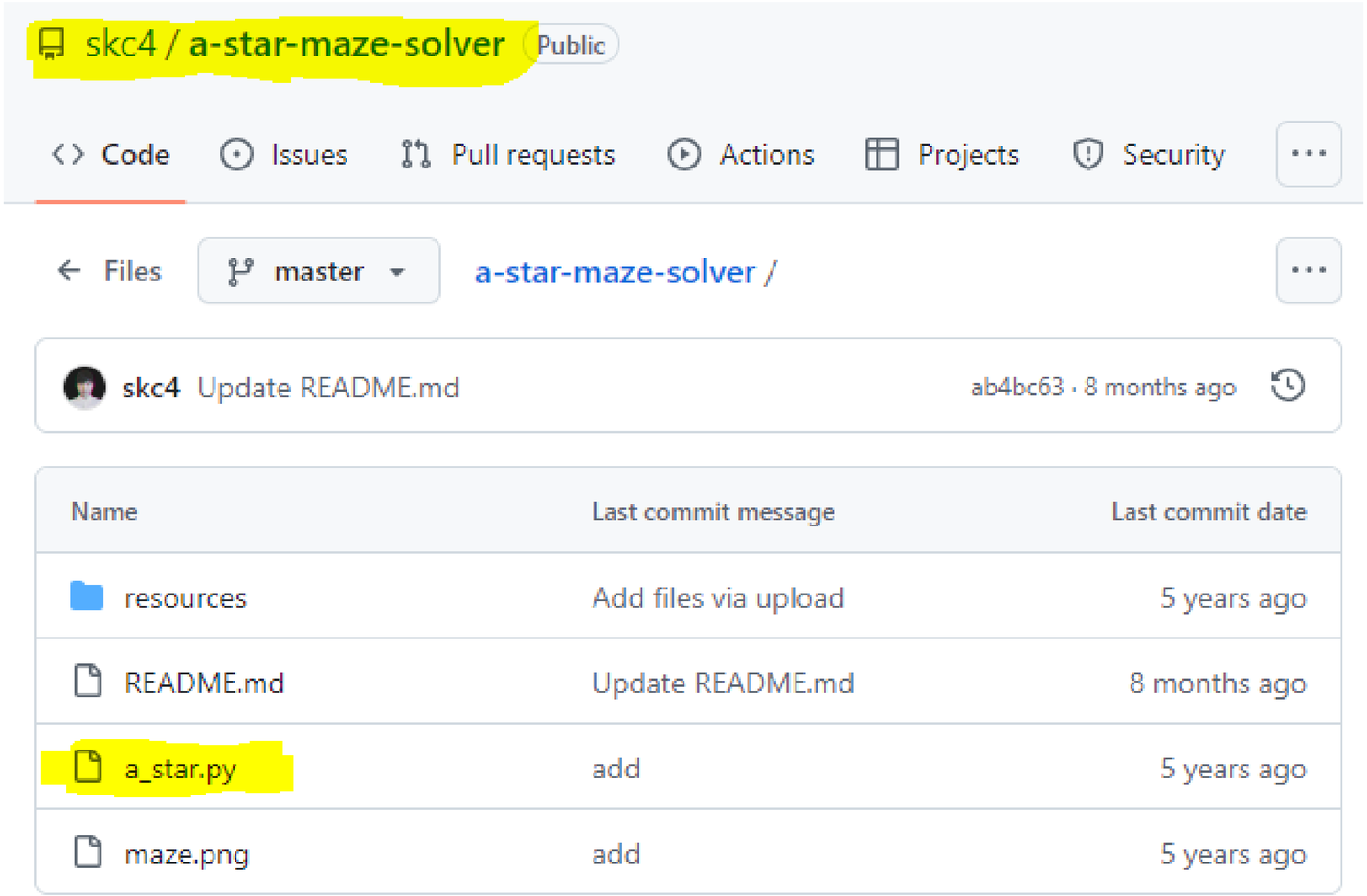
Пример реализации A*



Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

Пример реализации A*



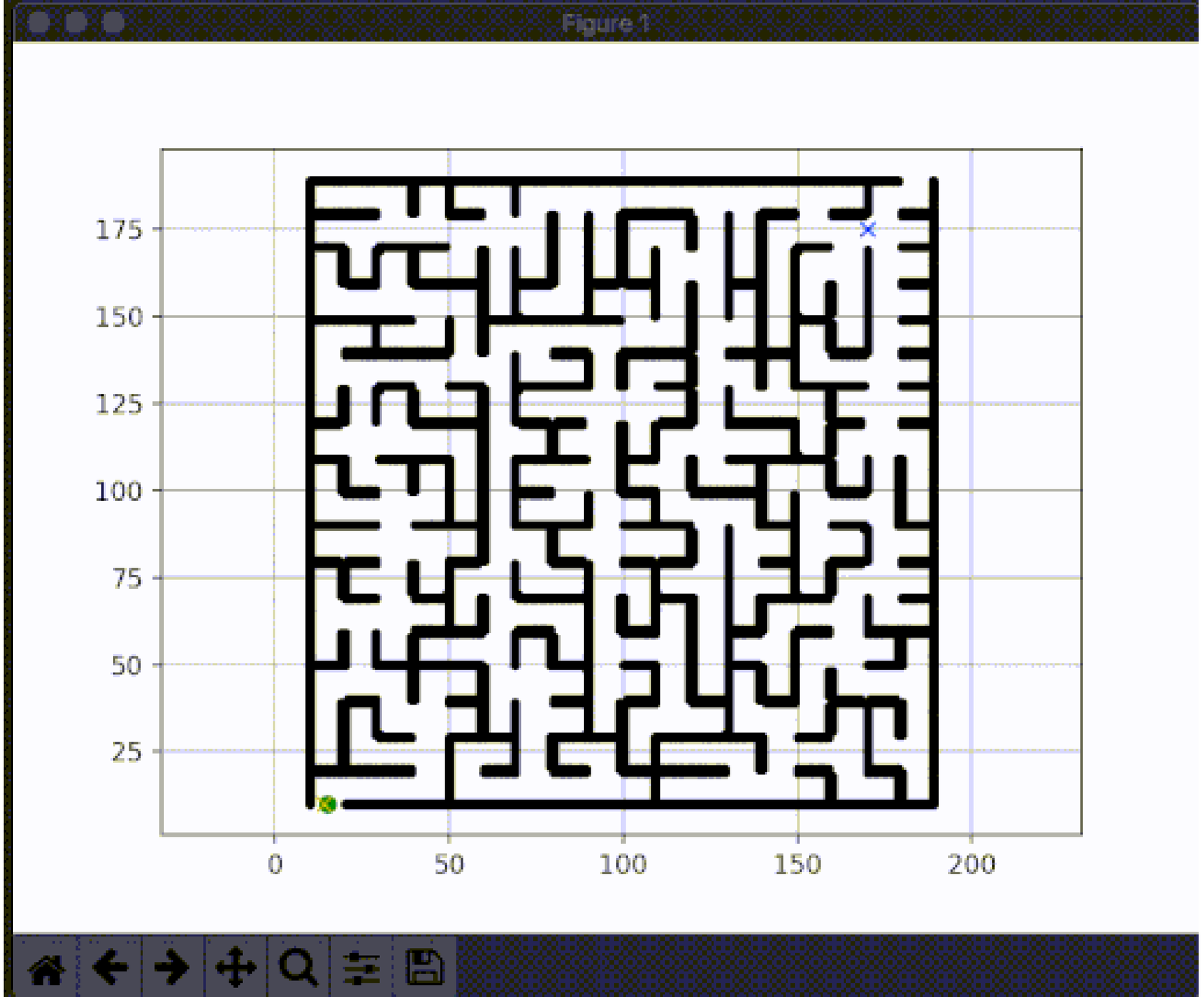


Figure 1
175
150
125
100
75
50
25
0
50
100
150
200

Студкемп по робототехнике и ИИ

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

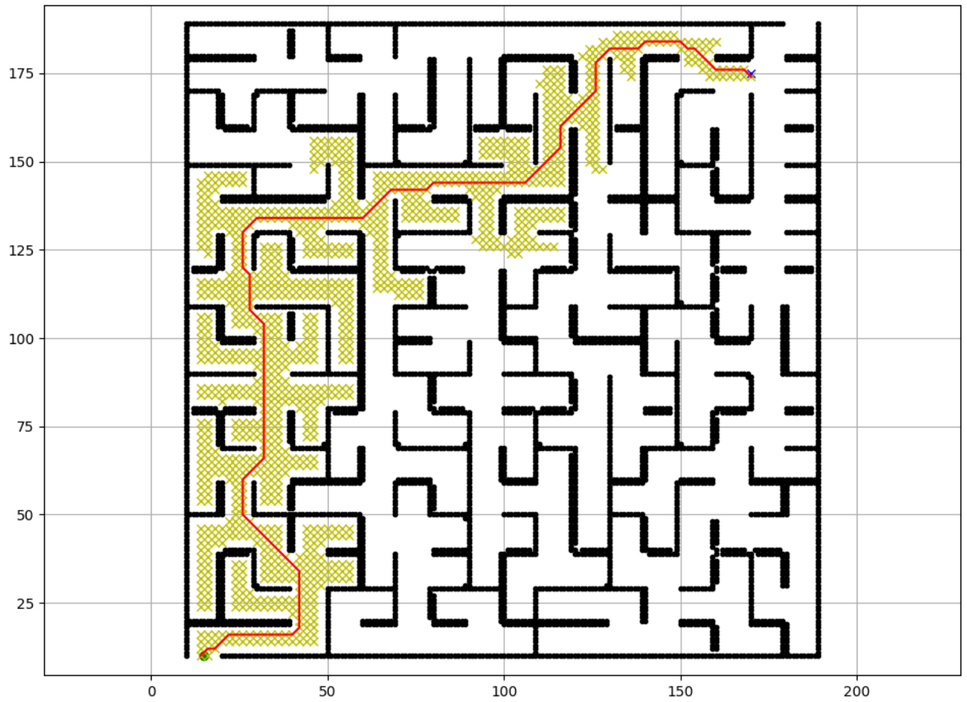
Astar.py

```
7   class AStarPath: ...
143         return x_out_path, y_out_path
144
145 def main():
146     start_x = 15
147     start_y = 10
148     end_x = 170
149     end_y = 175
150     grid_size = 2.0
151     robot_radius = 2.0
152
153     image = cv2.imread('maze.png')
154     image = cv2.resize(image, (200,200))
155     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
156     (width, length) = gray.shape
157     x_obstacle, y_obstacle = [], []
158     for i in range(width):
159         for j in range(length):
160             if gray[i][j] <= 150:
161                 y_obstacle.append(i)
162                 x_obstacle.append(j)
163
164     if show_animation:
165             plt.plot(x_obstacle, y_obstacle, ".k")
166             plt.plot(start_x, start_y, "og")
167             plt.plot(end_x, end_y, "xb")
168             plt.grid(True)
169             plt.axis("equal")
170
171     a_star = AStarPath(robot_radius, grid_size, x_obstacle, y_obstacle)
172     x_out_path, y_out_path = a_star.a_star_search(start_x, start_y, end_x, end_y)
```

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Координаты пути




```
Finished!
X-координаты:  [170.0, 168.0, 166.0, 164.0, 162.0, 160.0, 158.0, 156.0, 154.0, 152.0, 150.0, 148.0, 146.0, 144.0,
Y-координаты:  [174.0, 176.0, 176.0, 176.0, 176.0, 176.0, 178.0, 180.0, 182.0, 182.0, 184.0, 184.0, 184.0, 184.0,
```

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Подключаем свое изображение

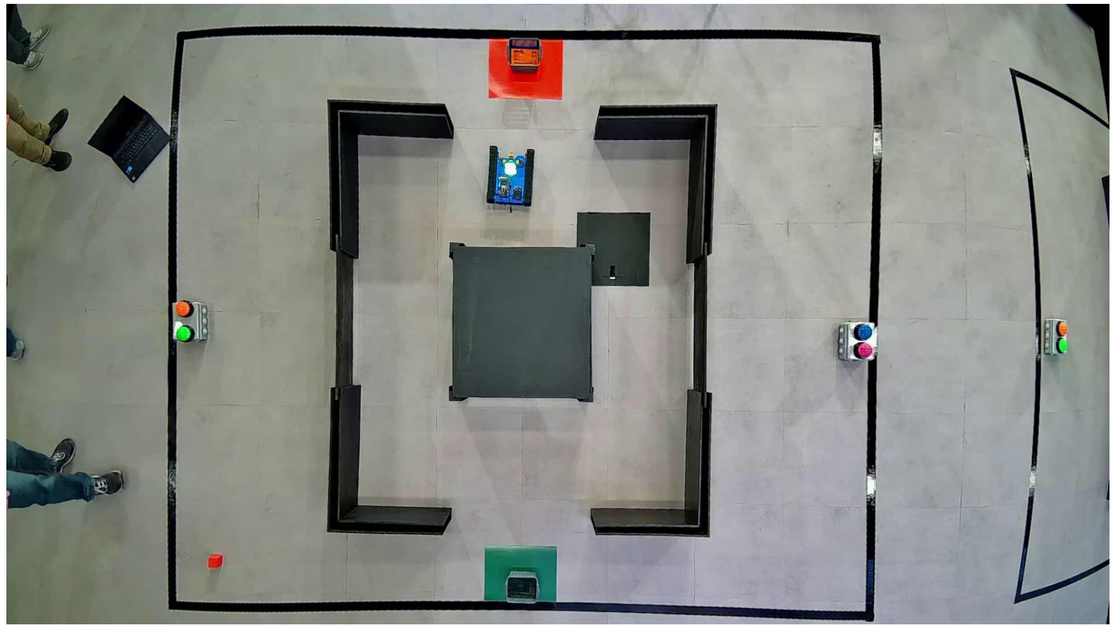
```python
start_x = 177
start_y = 218
end_x = 177
end_y = 25
grid_size = 4.0
robot_radius = 10.0

image = cv2.imread('forMap_1.png')
image = cv2.resize(image, dsize: (320, 240))
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
(width, length) = gray.shape
x_obstacle, y_obstacle = [], []
for i in range(width):
    for j in range(length):
        if gray[i][j] <= 90:
            y_obstacle.append(i)
            x_obstacle.append(j)
```


```
Finished!
X-координаты:  [176.0, 180.0, 184.0, 188.0, 192.0, 196.0, 200.0, 204.0, 208.0, 212.0, 216.0,
Y-координаты:  [24.0, 24.0, 24.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 32.0, 36.0, 40.0,
```

Яндекс
УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

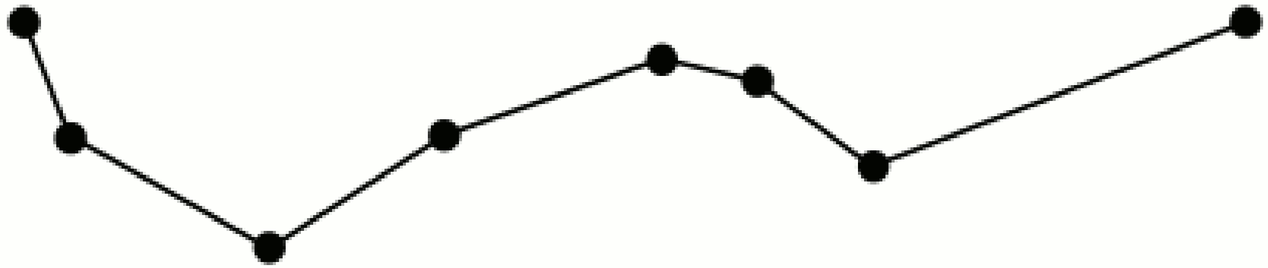
Подключаем свое изображение

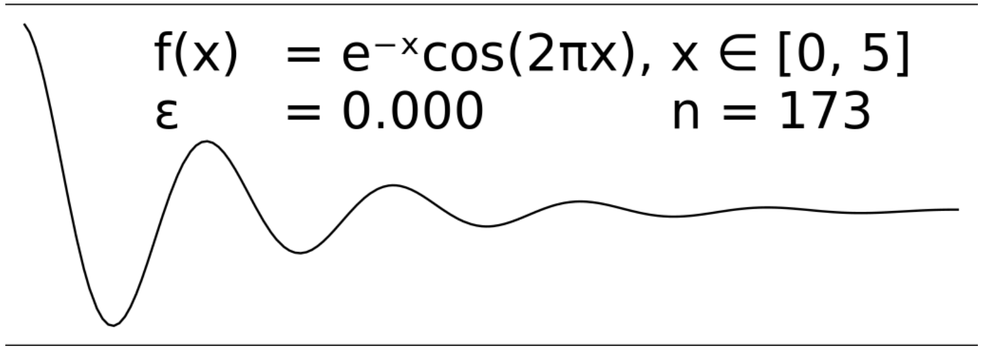


Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

Функция approxPolyDP





Douglas-Peucker algorithm

Python:

```
cv.approxPolyDP( curve, epsilon, closed[, approxCurve] ) -> approxCurve
```

curve Input vector of a 2D point stored in std::vector or **Mat**

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Преобразуем данные

```
########## создаем массив 2D координат точек пути #######
new_path = []
for i in range(len(x_out_path)):
    new_path.append((x_out_path[i], y_out_path[i]))
new_path_array = np.array(new_path).astype(np.float32)
print(new_path)
```

```
Finished!
X-координаты:  [176.0, 180.0, 184.0, 188.0, 192.0, 196.0, 200.0, 204.0, 208.0, 212.0, 216
Y-координаты:  [24.0, 24.0, 24.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 32.0, 36.0, 4
[(176.0, 24.0), (180.0, 24.0), (184.0, 24.0), (188.0, 28.0), (192.0, 28.0), (196.0, 28.0)
```

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

Упрощаем траекторию

```
########## аппроксимируем траекторию, уменьшаем количество  точек
path_len = cv2.arcLength(new_path_array,  closed: False)
epsilon = 0.05 * path_len
approx = cv2.approxPolyDP(new_path_array, epsilon,  closed: False)
print('длина пути: ',path_len)
print(approx)
```

```
длина пути:  262.6274166107178
[[[176.  24.]]

 [[216.  32.]]

 [[216. 208.]]

 [[176. 216.]]]
```

```
Finished!
X-координаты:  [176.0, 180.0, 184.0, 188.0, 192.0, 196.0, 200.0, 204.0, 208.0, 212.0, 216
Y-координаты:  [24.0, 24.0, 24.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 32.0, 36.0, 4
[(176.0, 24.0), (180.0, 24.0), (184.0, 24.0), (188.0, 28.0), (192.0, 28.0), (196.0, 28.0)
```

Рисуем новую траекторию

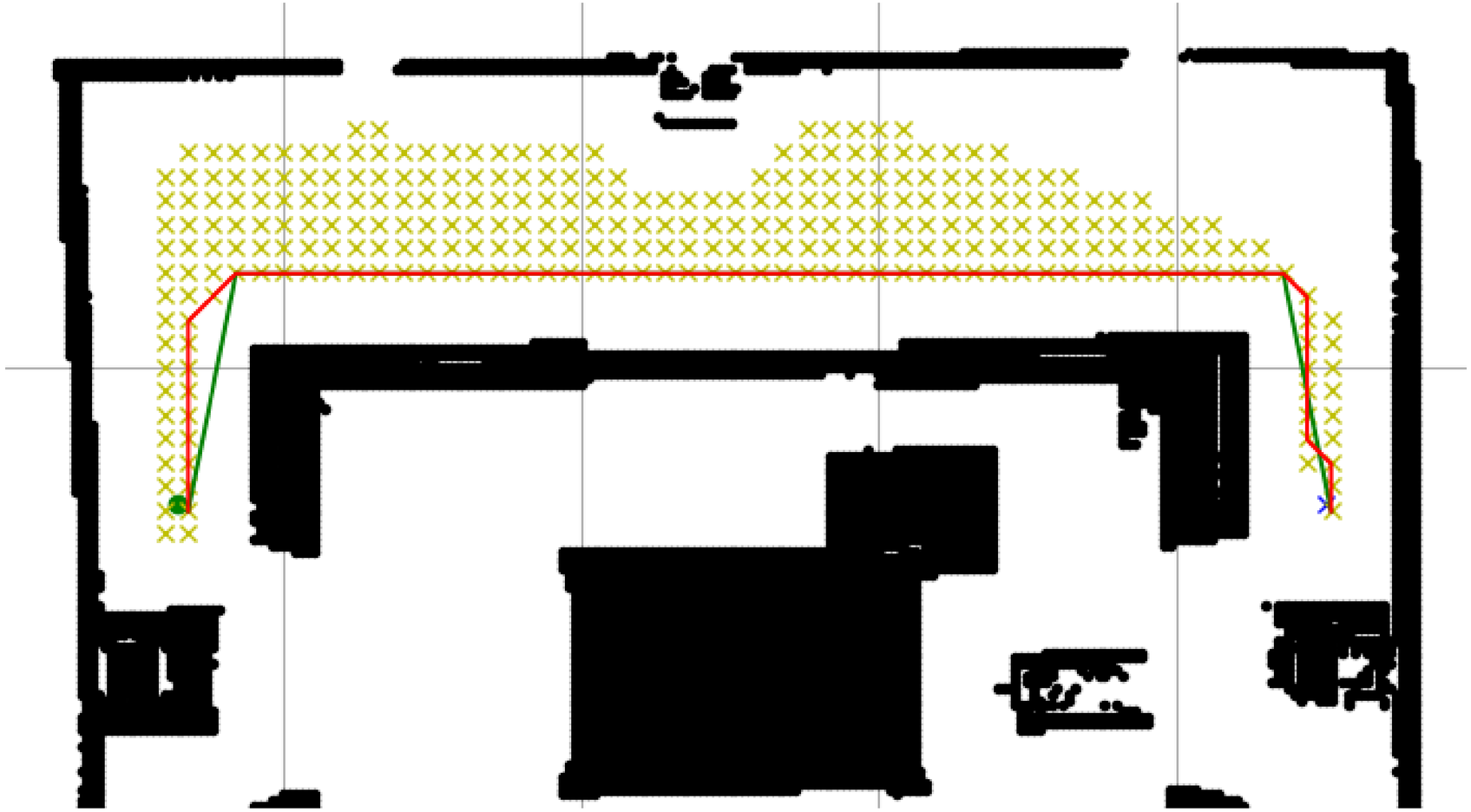
```
########## рисуем новую траекторию
x_approx, y_approx = [], []
for i in range(approx.shape[0]):
    for j in range(approx.shape[1]):
        x_approx.append(approx[i][j][0])
        y_approx.append(approx[i][j][1])
print(x_approx, y_approx)
plt.plot( *args: x_approx, y_approx, "g")
```

```
[176.0, 216.0, 216.0, 176.0] [24.0, 32.0, 208.0, 216.0]
```

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Рисуем новую траекторию



Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий – РТФ

Вычисляем длину отрезка

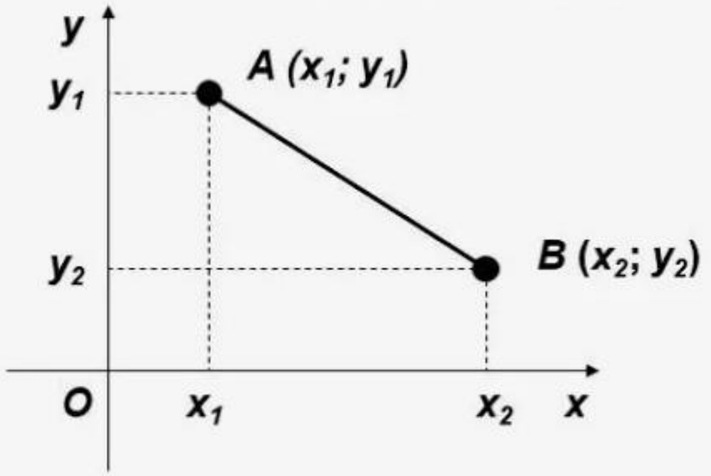
```
########## вычисляем расстояние между двумя точками
dist = hypot( *coordinates: x_approx[1] - x_approx[0], y_approx[1] - y_approx[0])
print('Расстояние = ',dist)
```

```
[176.0, 216.0, 216.0, 176.0] [24.0, 32.0, 208.0, 216.0]
Расстояние =  40.792156108742276
```

```
dist = sqrt( (x2 - x1)**2 + (y2 - y1)**2 )
```

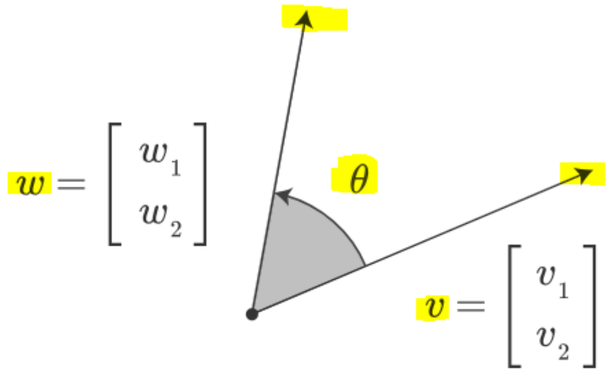
```
dist = math.hypot(x2 - x1, y2 - y1)
```

$$d(A,B)=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$



Вычисляем угол поворота

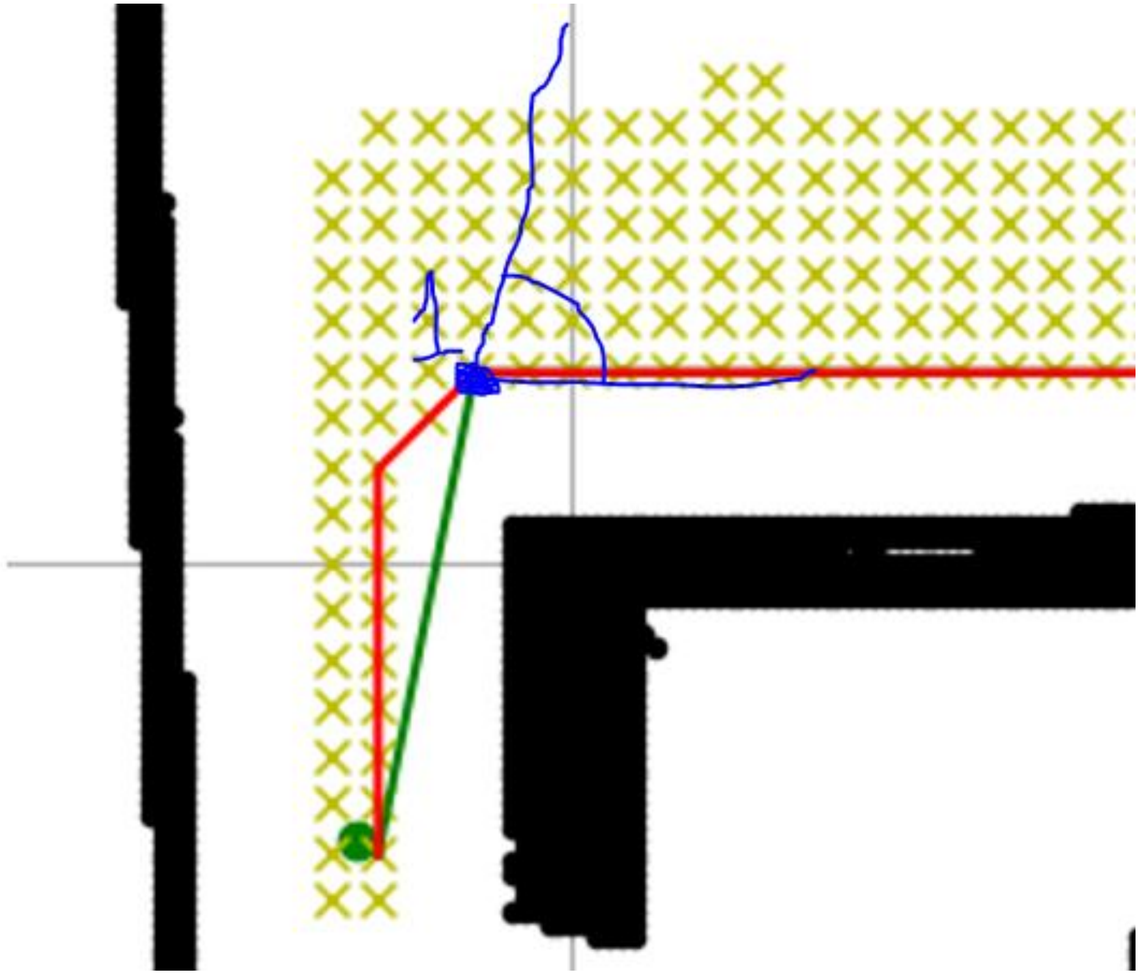
```python
######### вычисляем угол по трем точкам
def angle_between_lines(p1, p2, p3):
    v1 = (p2[0] - p1[0], p2[1] - p1[1])
    v2 = (p3[0] - p2[0], p3[1] - p2[1])
    angle = math.atan2(v2[1], v2[0]) - math.atan2(v1[1], v1[0])
    angle = math.degrees(angle) % 180
    # если нужен острый угол
    # return min(180 - angle, angle)
    return angle
angel= angle_between_lines( p1: (x_approx[0],y_approx[0]), p2: (x_approx[1],y_approx[1]), p3: (x_approx[2],y_approx[2]))
print('Угол поворота = ',angel)
```



$$\theta = \operatorname{atan2}\left(w_2 v_1 - w_1 v_2, w_1 v_1 + w_2 v_2\right)$$

Яндекс
УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Вычисляем угол поворота



```
Расстояние до первой точки =  40.792156108742276
Угол поворота =  78.69006752597979
```

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Спасибо за внимание!

Пелевин Владимир

к.пед.н., доцент, УрФУ

инженер, ООО Микроэлектроника и Робототехника

Студкемп по робототехнике и ИИ

Яндекс

УрФУ
Институт радиоэлектроники и информационных технологий — РТФ

Понравилось занятие?

Сканируй QR код и оставляй свой отзыв