

**Министерство науки и высшего образования**

**Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Институт цифровых интеллектуальных систем

Дисциплина: «Мобильная робототехника»

Лабораторная работа № 2

Алгоритм А\*

Вариант 1

Выполнил:

студент группы АДМ-21-05 \_\_\_\_\_\_\_\_\_\_ Абдулзагиров М.М.

(подпись) (ФИО)

Принял

преподаватель: \_\_\_\_\_\_\_\_\_\_ \_Собольников С.А\_

(подпись) (ФИО)

Дата:\_\_\_\_\_\_\_\_\_

Москва 2022

**Цели работы**

* Научиться реализовывать алгоритм А\* с помощью SCILAB;
* Исследовать преимущества и недостатки алгоритма А\* и его модификаций.

# Теория

Поиск A\*— алгоритм поиска по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной).

Данный вид поиска — это модификация алгоритма Дейкстры, оптимизированная для единственной конечной точки. Алгоритм Дейкстры может находить пути ко всем точкам, A\* находит путь к одной точке. Он отдаёт приоритет путям, которые ведут ближе к цели.

В начале работы просматриваются узлы, смежные с начальным; выбирается тот из них, который имеет минимальное значение f(x), после чего этот узел раскрывается. На каждом этапе алгоритм оперирует с множеством путей из начальной точки до всех ещё не раскрытых (листовых) вершин графа — множеством частных решений, — которое размещается в очереди с приоритетом. Приоритет пути определяется по значению f(x) = g(x) + h(x). Алгоритм продолжает свою работу до тех пор, пока значение f(x) целевой вершины не окажется меньшим, чем любое значение в очереди, либо пока всё дерево не будет просмотрено. Из множества решений выбирается решение с наименьшей стоимостью.

Чем меньше эвристика h(x), тем больше приоритет, поэтому для реализации очереди можно использовать сортирующие деревья. Множество просмотренных вершин хранится в closed, а требующие рассмотрения пути — в очереди с приоритетом open. Приоритет пути вычисляется с помощью функции f(x) внутри реализации очереди с приоритетом

# Ход работы.

Листинг 1.

*//Задание начальных условий*

clear; xdel(winsid());

*// возвращает индекс точки в списке*

*// 0 - точки нет в списке*

function **iPoint**=in\_list(**pointList**, **Point**)

**iPoint**=0;

for i=1:length(**pointList**)

if **pointList**(i)(1) == **Point**(1) & **pointList**(i)(2) == **Point**(2) then

**iPoint** = i; *// записываем индекс точки*

break

end

end

endfunction

*//выбор минимального значения из списка*

function **x**=search\_min(**pointList**)

y = zeros(length(**pointList**));

for i = 1:length(**pointList**)

y(i) = **pointList**(i)(3);

end

[z **x**] = min(y);

endfunction

function **f**=path\_cost(**g**, **h**)

*//Весовые коэффициенты*

kg=0.1; kh=1;

**f**=kg\***g**+kh\***h**; *// стоимость пути*

endfunction

*//Создание карты*

size\_ = 32;

data = read ('goo1.txt', size\_\*size\_, 1);

map = zeros(size\_,size\_);

i = 1;

j = 1;

k = 0;

for i =1:size\_

for j=1:size\_

map(i,j)=data(k\*size\_+j);

end

k=k+1;

end

*//Начальная и конечная точка*

s\_pos = [3 3];

e\_pos = [30 30];

map(s\_pos(1), s\_pos(2)) = -10;

map(e\_pos(1), e\_pos(2)) = -15;

map1 = map;

*//Определение открытого и закрытого списка*

*//[ 1,2- родительская точка; 3-стоимость пути ; 4 - расстояние от начала до текущей точки ;5,6- координаты точки ]*

open\_list = list([s\_pos(1) s\_pos(2) 0 0 0 0]);

close\_list = list();

*//Начало цикла проверки. Осматриваем каждую соседнюю клетки относительно текущей*

while in\_list(open\_list,[e\_pos(1) e\_pos(2)]) == 0

t = search\_min(open\_list) *// интекс минимальной (текущей) точки*

side\_p =[

open\_list(t)(1)+1 open\_list(t)(2);

open\_list(t)(1)-1 open\_list(t)(2);

open\_list(t)(1) open\_list(t)(2)+1;

open\_list(t)(1) open\_list(t)(2)-1;

];

for i = 1:4

*// проверка на препядствие и эта точка не в закрытом листе*

if map(side\_p(i,1), side\_p(i,2)) ~= 1 & in\_list(close\_list,side\_p(i,:)) == 0 then

g = open\_list(t)(4)+1;*//+map(side\_p(i,1), side\_p(i,2)); // путь от начала (Дейкстра)*

h=abs(e\_pos(1)-side\_p(i,1)) + abs(e\_pos(2)-side\_p(i,2)); *// расстояние до конца*

f=path\_cost(g,h); *// стоимость пути*

*// если точка не находится в открытом списке*

if in\_list(open\_list,side\_p(i,:))== 0 then

open\_list($+1)=[side\_p(i,1) side\_p(i,2) f g open\_list(t)(1) open\_list(t)(2)]

end

end

end

close\_list($+1)=open\_list(t);

map(open\_list(t)(1),open\_list(t)(2))=open\_list(t)(3);

open\_list(t)=null();

end

*//Построение пути от начальной до конечной точки*

path=list(close\_list($));

map1(path($)(1), path($)(2))=10;

while in\_list(path, [s\_pos(1),s\_pos(2)]) == 0

path($+1)=close\_list(in\_list(close\_list,[path($)(5) path($)(6)]));

map1(path($)(1), path($)(2))=10;

end

map = map/ (max(map)\*1);

figure(3);*//clf;*

subplot(1,2,1);

title('проверенные точки')

hist3d(map);

subplot(1,2,2);

title('итоговый маршрут')

hist3d(map1);

Изменяя параметры kg и kh, можно сделать алгоритм с большим перекосом к алгоритму Дейкстры.

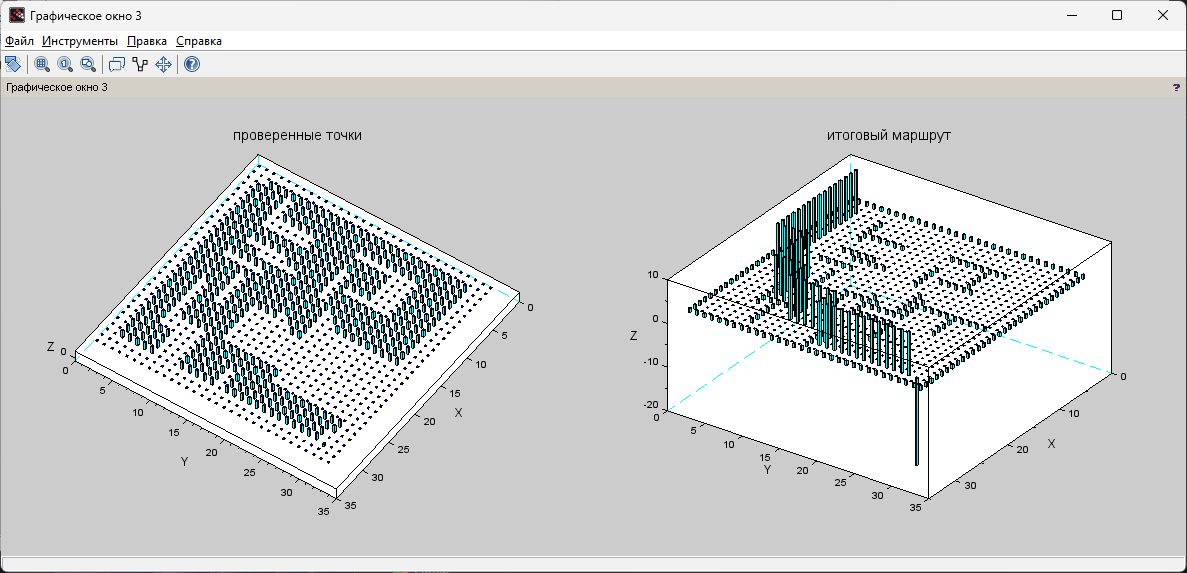


Рис.1. Найденный путь при kg=1 kh=1.

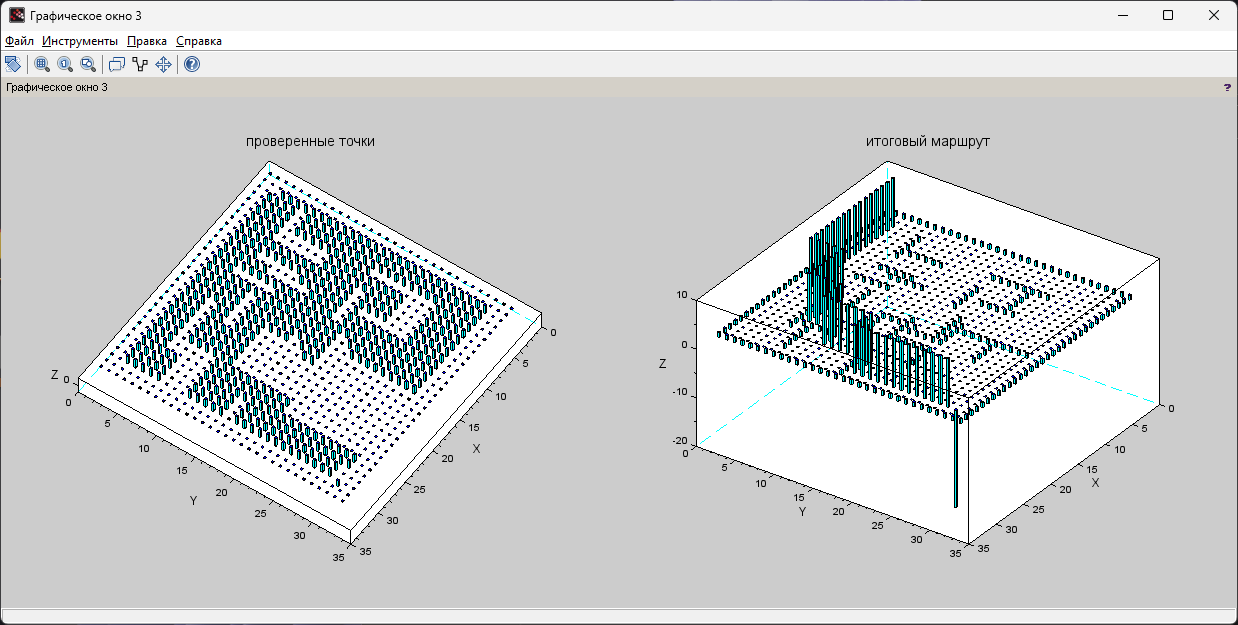


Рис.2. Найденный путь при kg=0.5 kh=0.5.

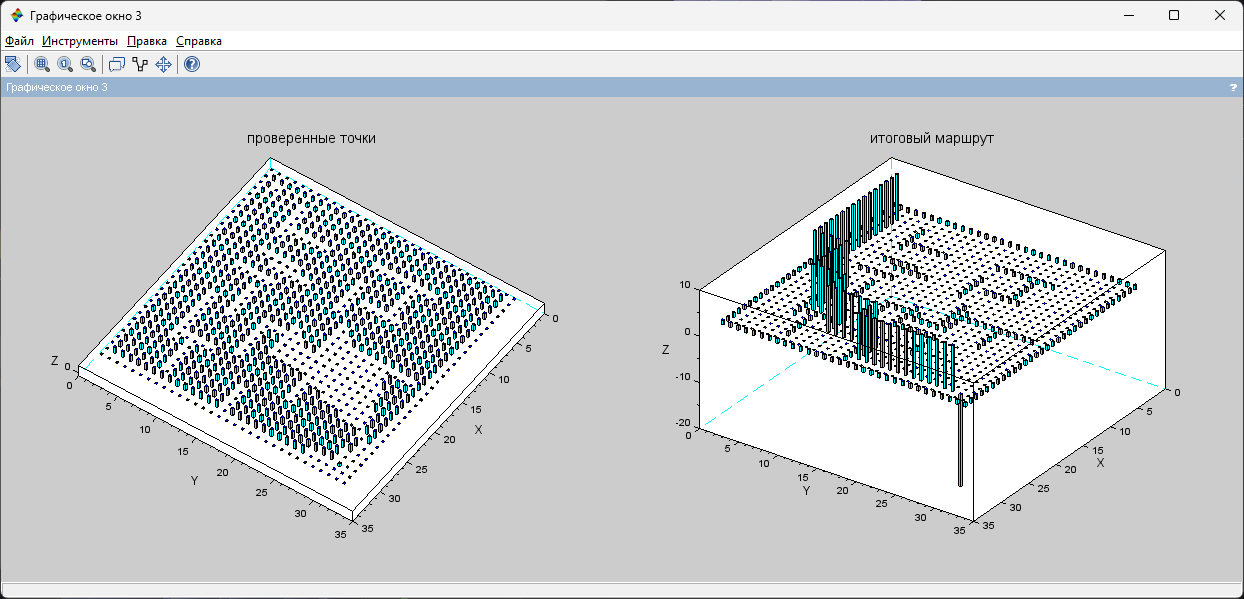


Рис.3. Найденный путь при kg=1 kh=0.5.

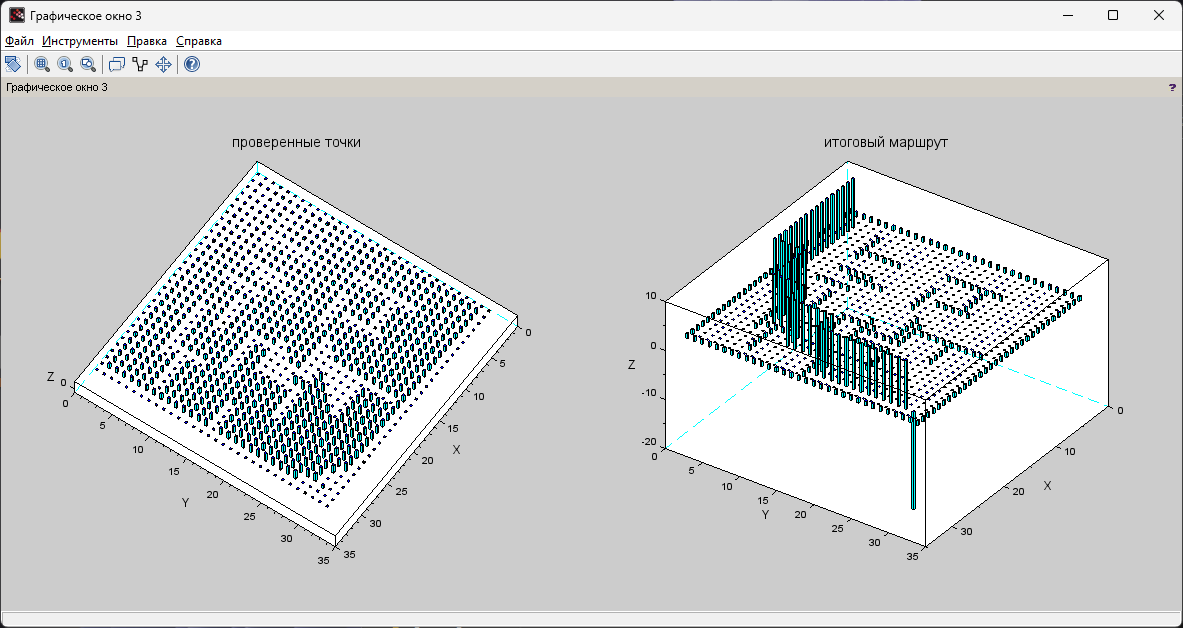


Рис.4. Найденный путь при kg=1 kh=0.1.

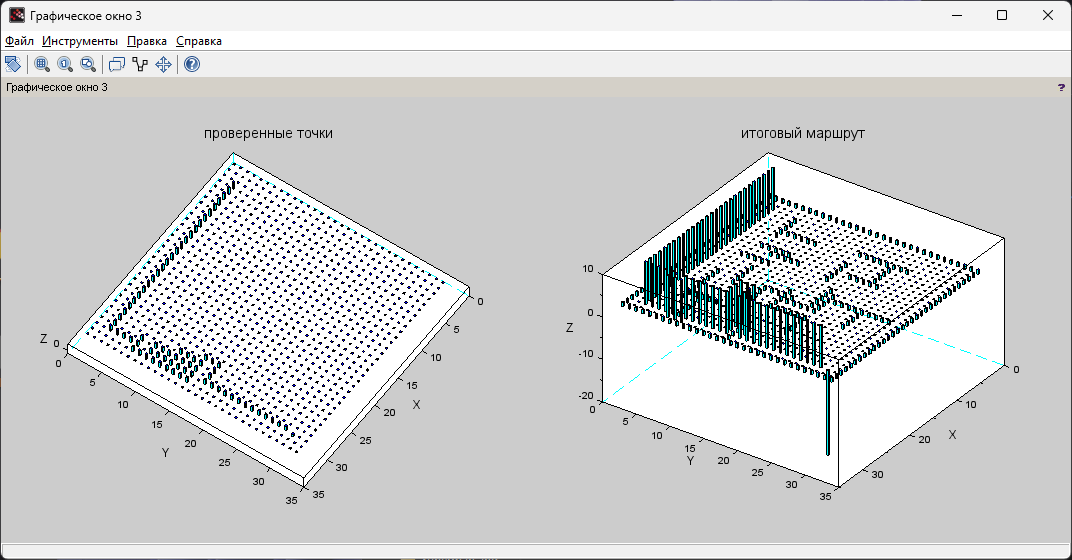


Рис.5. Найденный путь при kg=0.5 kh=1.

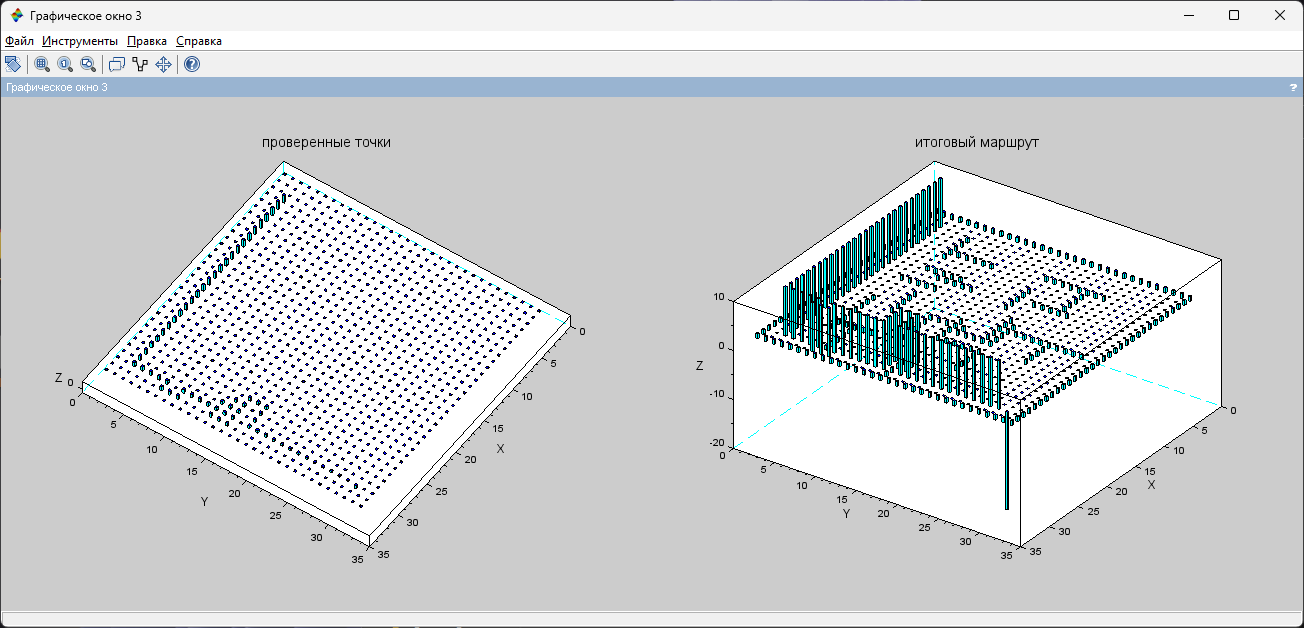


Рис.6. Найденный путь при kg=0.1 kh=1.

**Вывод:** в данной лабораторной работе мы научились реализовывать алгоритм А\* с помощью SCILAB и исследовали преимущества и недостатки алгоритма А\* и его модификаций.