这段 MATLAB 代码是一个用于机器人路径规划的势场算法实现

我将逐行解释代码的主要功能：

```matlab
map=int16(im2bw(imread('map1.bmp'))); %
```

从 bmp 文件中读取输入地图。如果需要新的地图，在这里写文件名。

```
```

这行代码读取一个名为`map1.bmp`的位图文件，并将其转换为二值图像，然后转换为`int16`类型，用于表示地图。

```matlab
source=[50 50]; % 机器人的起始位置，格式为 Y, X。
goal=[450 450]; % 目标位置，格式为 Y, X。
```

```
```

这两行设置了机器人的起始位置和目标位置。

```matlab
robotDirection=pi/8; % 初始朝向方向。
robotSize=[10 10]; % 机器人的长度和宽度。
robotSpeed=10; % 任意单位的速度。
maxRobotSpeed=10; % 任意单位的最大速度。
```

```
```

这些行设置了机器人的初始方向、尺寸、速度和最大速度。

```matlab
S=10; % 安全距离。
distanceThreshold=30; % 一个阈值距离。在此阈值内，点可以被视为相同。
maxAcceleration=10; % 每单位时间内的最大速度变化。
maxTurn=10*pi/180; % 势场输出的转向限制在-60 度和 60 度之间。
```

这些行设置了机器人的安全距离、距离阈值、最大加速度和最大转向角度。

```matlab
k=3; % 计算势的度数。
attractivePotentialScaling=300000; % 吸引势的缩放因子。
repulsivePotentialScaling=300000; % 排斥势的缩放因子。
minAttractivePotential=0.5; % 任何点的最小吸引势。
```

这些行设置了计算势的参数，包括吸引势和排斥势的缩放因子，以及吸引势的最小值。

```matlab
%%%%% 参数在这里结束 %%%%%
currentPosition=source; % 机器人中心的位置。
currentDirection=robotDirection; % 机器人的朝向方向。
```

```
robotHalfDiagonalDistance=((robotSize(1)/2)^2+(robotSize(2)/2)^2)^0.
5; % 用于距离计算。
pathFound=false; % 是否找到路径。
pathCost=0;
```

这些行初始化机器人的当前位置、方向、对角线距离，并设置路径是否找到的标志和路径成本。

```matlab
t=1;
imshow(map==1);
rectangle('position',[1 1 size(map)-1],'edgecolor','k')
```

这些行设置动画的帧数，并显示地图，其中障碍物用黑色矩形表示。

```matlab
pathLength=0;
if
~plotRobot(currentPosition,currentDirection,map,robotHalfDiagonalDis
tance)
    error('source lies on an obstacle or outside map');
end
```

```

这行代码初始化路径长度，并使用`plotRobot`函数检查起始点是否在障碍物上或地图外，如果是，则报错。

```matlab
M(t)=getframe;
t=t+1;
```

这行代码获取当前帧，并增加帧数。

```matlab
if ~feasiblePoint(goal,map), error('goal lies on an obstacle or outside map'); end
```

这行代码使用`feasiblePoint`函数检查目标点是否在障碍物上或地图外，如果是，则报错。

```matlab
tic;
```

这行代码开始计时。
```

```matlab
while ~pathFound
```

这个循环将一直执行，直到找到路径。

接下来的代码块计算机器人前方、左侧、右侧和对角线方向的障碍物距离，并计算到目标的角度和距离。

```matlab
% 计算从目标点的角度和距离
angleGoal=atan2(goal(1)-currentPosition(1),goal(2)-currentPosition(2)
);
distanceGoal=( sqrt(sum((currentPosition-goal).^2)) );
if distanceGoal<distanceThreshold, pathFound=true; end
```

这些行计算从当前位置到目标的角度和距离，并检查是否达到了距离阈值，如果是，则设置路径找到标志。

```matlab
% 计算势
repulsivePotential=...;
attractivePotential=max([(1.0/distanceGoal)^k*attractivePotentialSca
ling minAttractivePotential])*[sin(angleGoal) cos(angleGoal)];
```

totalPotential=attractivePotential-repulsivePotentialScaling*repulsi

vePotential;

```

这些行计算排斥势和吸引势，并计算总势。

```matlab

% 执行转向

preferredSteer=...;

currentDirection=currentDirection+preferredSteer;

```

这些行计算理想的转向角度，并更新当前方向。

```matlab

% 设置速度

preferredSpeed=...;

robotSpeed=max([robotSpeed-maxAcceleration preferredSpeed]);

robotSpeed=min([robotSpeed maxRobotSpeed]);

robotSpeed=max([robotSpeed 0]);

```

这些行根据加速度和速度限制设置理想的速度，并更新机器人速度。

```matlab

```
% 计算新位置
newPosition=currentPosition+robotSpeed*[sin(currentDirection)
cos(currentDirection)];
pathCost=pathCost+distanceCost(newPosition,currentPosition);
currentPosition=newPosition;
```

这些行根据转向和速度计算新位置，并更新路径成本。

```matlab
% 绘制机器人
if
~plotRobot(currentPosition,currentDirection,map,robotHalfDiagonalDis
tance)
    error('collission recorded');
end
M(t)=getframe;t=t+1;
```

这些行绘制机器人，并更新帧。

```matlab
end
```

这个`end`结束了`while`循环。

```matlab
fprintf('processing time=%d \nPath Length=%d \n\n', toc,pathCost);
```

这行代码打印处理时间和路径长度。

请注意，这段代码中有一些函数（如`feasiblePoint`和`plotRobot`）没有给出定义，它们可能是自定义函数，用于检查点的可行性和绘制机器人。

由于代码较长，这里只提供了关键部分的解释。如果需要更详细的逐行解释，请告知。