Relatório:
Missão 4.0
Controle Cinemático

1. Introdução

A missão abordada neste relatório se trata da missão de Controle Cinemático, ela consiste em fazer um um robô solucionar um labirinto por conta própria, no CoppeliaSim.

2. Procedimentos

Eu utilizei o robô “Pioneer P3DX” para a atividade, ele vem com um programa embutido que utiliza sensores para evitar objetos, se posicionando para ir em outra direção quando os encontra, é importante destacar que a princípio ele não considera formatos primitivos como cubóides, objetos, eu precisei utilizar objetos já criados, como paredes, para o robô utilizado considerá-los.

Após descobrir como o robô funcionava, e aprender a fazer ele funcionar, eu criei um labirinto, não tão complicado, mas com vários pontos onde o robô poderia se prender para fazer ele trabalhar um pouco mais pela saída.

Com isto, fiz alguns testes e fui descobrindo qual era a melhor forma de fazer o robô funcionar, e como ele poderia falhar em outros casos.

3. Conclusão

Pode se reparar que pequenas alterações em configurações, como por exemplo, velocidade, distância permitida entre o robô e o objeto, e força do motor, podem fazer o robô falhar ou cumprir a tarefa, muitas vezes onde se altera uma configuração, o robô entra num loop infinito, creio que por conta principalmente do tamanho da “arena” disponibilizada primariamente, onde foi criado o labirinto.

Como resultado, somente com uma das configurações testadas foi possível fazer o robô funcionar, mas funcionou tão bem, que tomei a liberdade de colocar dois robôs no mesmo labirinto e os dois, mesmo próximos um do outros (o que facilita erros) conseguiram achar o caminho para fora.

4. Material de Apoio

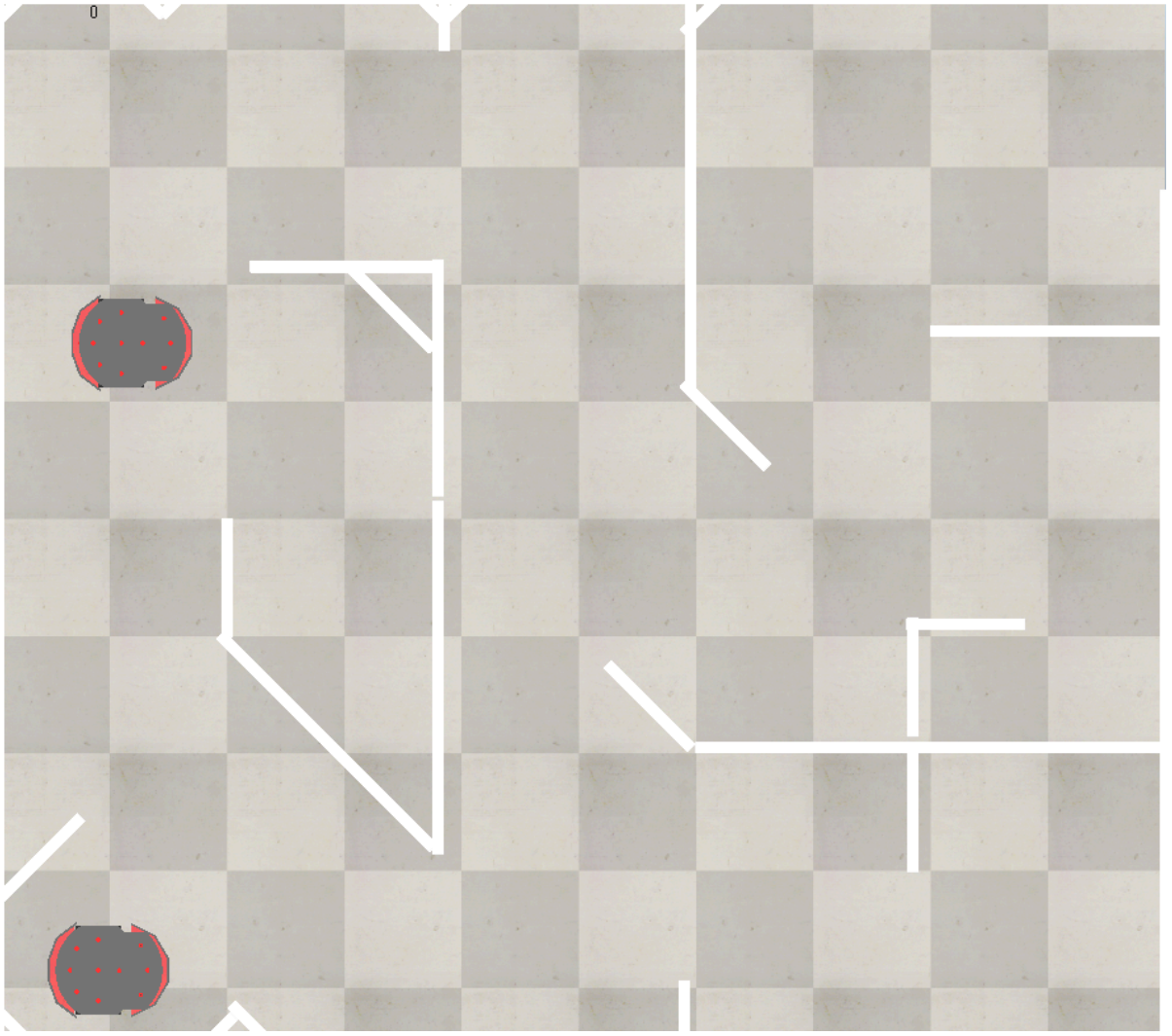
https://www.youtube.com/watch?v=3IRBbZbGK44
https://youtu.be/TGT7KbP7Dfs?si=mqLGD0YnhhPpXwbK
https://youtu.be/xiJZkOoLwP0?si=qDUWEUQAiLgp_WVb
https://youtu.be/waLZeLf0pR0?si=nZnXskkin6S7InfW

5. Resultado

Códigos do robô (em lua):

```
--lua

sim=require'sim'

function sysCall_init()
-- funcoes que indentificam o robo para poder controla-lo
    local robot=sim.getObject('.')
    local obstacles=sim.createCollection(0)
    sim.addItemToCollection(obstacles,sim.handle_all,-1,0)
    sim.addItemToCollection(obstacles,sim.handle_tree,robot,1)
    usensors={} -- sensores
    for i=1,16,1 do -- contador dos sensores
        usensors[i]=sim.getObject("./ultrasonicSensor",{index=i-1})
        sim.setObjectInt32Param(usensors[i],sim.proxintparam_entity_to_detect,obstacles)
    end

    motorLeft=sim.getObject("./leftMotor") -- motor esquerdo
    motorRight=sim.getObject("./rightMotor") -- motor direito
    noDetectionDist=0.5
    maxDetectionDist=0.2 -- Distancia maxima que o robo pode ficar de um objeto.
    detect={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0} -- deteccao dos sensores
    braitenbergL={-0.2,-0.4,-0.6,-0.8,-1,-1.2,-1.4,-1.6, 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0} -- forca aplicada nos sensores esquerdos, de tras
para frente.
    braitenbergR={-1.6,-1.4,-1.2,-1,-0.8,-0.6,-0.4,-0.2, 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0} -- forca aplicada nos sensores direitos, de tras
para frente.
    v0=2 -- Velocidade do robo.

end

function sysCall_cleanup()

end

function sysCall_actuation()
    for i=1,16,1 do -- contador que utiliza os sensores para detectar os objetos e calcular velocidade e distancia.
        res,dist=sim.readProximitySensor(usensors[i])
        if (res>0) and (dist<noDetectionDist) then
            if (dist<maxDetectionDist) then
                dist=maxDetectionDist
            end
            detect[i]=1-((dist-maxDetectionDist)/(noDetectionDist-maxDetectionDist)) -- 1 simboliza que ha objetos na frente.
        else
            detect[i]=0 -- 0 simboliza que nao ha objetos a frente.
        end

    end

    vLeft=v0 -- velocidade da roda esquerda
    vRight=v0 -- velocidade da roda direita

    for i=1,16,1 do -- contador que utiliza os sensores para calcular a velocidade em cada roda
        vLeft=vLeft+braitenbergL[i]*detect[i]
        vRight=vRight+braitenbergR[i]*detect[i]
    end

    sim.setJointTargetVelocity(motorLeft,vLeft) -- Comando que aplica a velocidade ao motor.
    sim.setJointTargetVelocity(motorRight,vRight)
end
```

Arquivo no repositório.



(saída na parte superior direita)