FUNDAÇÃO EDUCACIONAL INACIANA "PE. SABÓIA DE MEDEIROS" (FEI)

Felipe Orlando Lanzara

João Vitor Governatore

PROJETO DE ROBÓTICA

Relatório sobre o projeto Webots

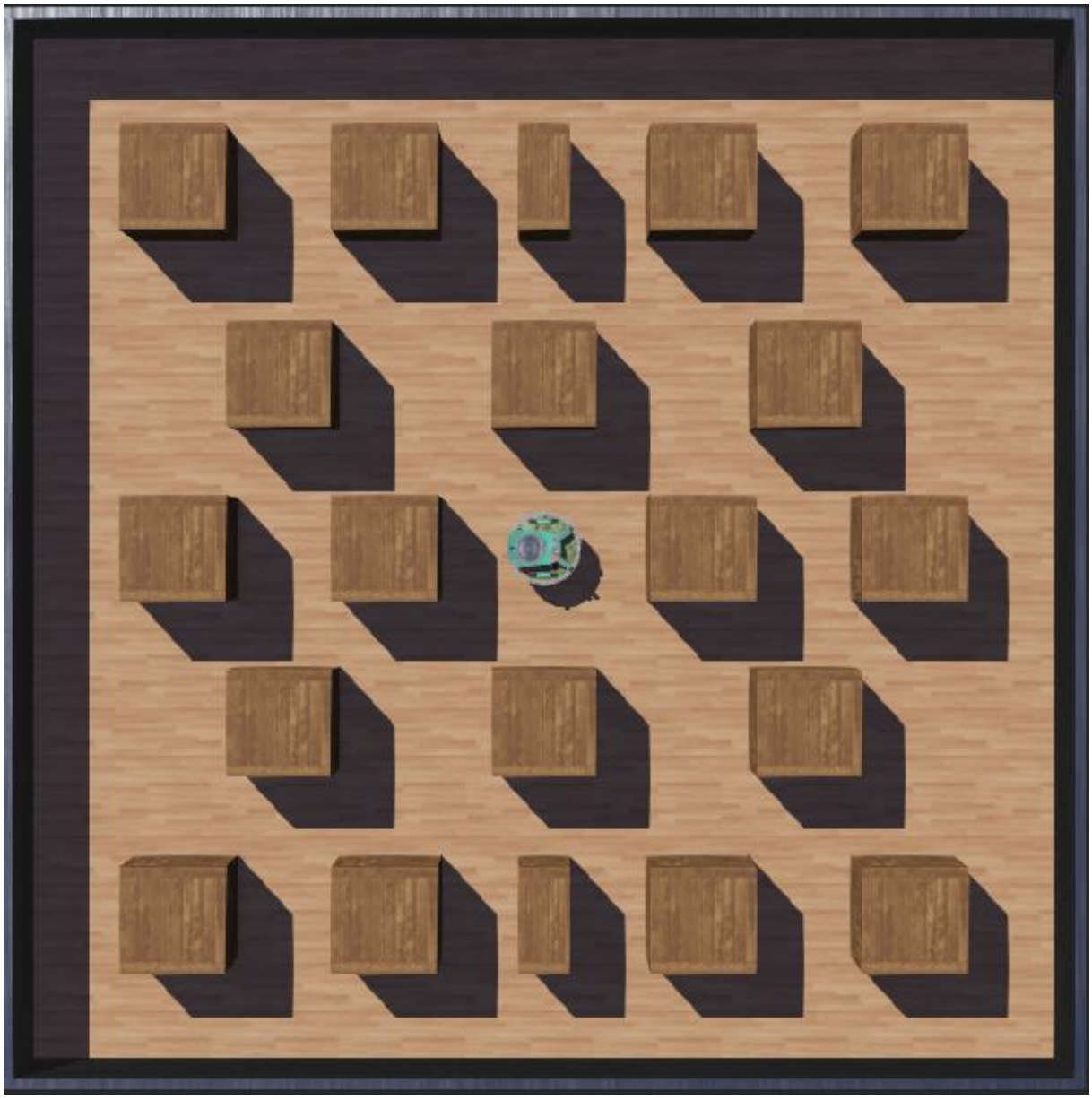
São Bernardo do Campo

2025

Introdução

O objetivo deste projeto é desenvolver, em C, um controlador para o robô e-puck no simulador Webots, capaz de patrulhar automaticamente um ambiente composto por várias caixas dispostas em grade, das quais apenas uma se movimenta. Ao identificar essa caixa em movimento, o e-puck interrompe seu deslocamento e passa a girar sobre o próprio eixo, sinalizando a detecção do objeto deslocado.

Abaixo está a imagem do mapa, mostrando as caixas e o robô em sua posição inicial:



Inicializações e Configuração de LEDs

Inicialização

Configura o controlador do Webots e a semente aleatória para decisões de giro.

```
wb_robot_init();                    // inicializa o robô
srand(time(NULL));                  // semente para rand()
```

Configuração de LEDs

Armazena em leds[] as tags dos 8 LEDs do e-puck, usados depois para sinalizar a detecção de caixa deslocada.

```
// --- Inicializa LEDs ---
WbDeviceTag leds[QtddLeds];
char led_name[8];
for (int i = 0; i < QtddLeds; i++) {
  sprintf(led_name, "led%d", i);          // monta o nome "led0"..."led7"
  leds[i] = wb_robot_get_device(led_name); // obtém a tag do LED
}
```

Movimentação do Robô

Detecção de Obstáculos

A cada ciclo de simulação, o controlador varre todos os sensores de proximidade (ps0 a ps7) e, sempre que qualquer leitura ultrapassa o limiar definido por OBST_THRESHOLD, a variável *obstacle* é imediatamente setada para 1 e o laço interrompido. Enquanto *obstacle* permanecer 0, o caminho é considerado livre de colisões e o e-puck prossegue sem alterar sua trajetória.

```
// 1) Detecta obstáculo
int obstacle = 0;
for (int i = 0; i < N_SENSORS; i++) {
  if (wb_distance_sensor_get_value(ps[i]) > OBST_THRESHOLD) {
    obstacle = 1;         // encontrou obstáculo
    break;
  }
}
```

Vagueio em Linha Reta

Sempre que nenhum obstáculo é detectado, ambos os motores recebem a mesma velocidade máxima (MAX_VELOCITY), fazendo com que o robô avance em linha reta. Esse comportamento contínuo garante que o e-puck percorra de forma uniforme os corredores formados pelas caixas, mantendo uma exploração sistemática do ambiente até encontrar um obstáculo.

```
if (!obstacle) {
  wb_motor_set_velocity(left_motor,  MAX_VELOCITY);   // vai em frente
  wb_motor_set_velocity(right_motor, MAX_VELOCITY);
}
```

Giro Aleatório e Fase de “Escape”

Ao identificar um obstáculo, o e-puck escolhe aleatoriamente uma direção de giro e mantém esse movimento por um tempo também randômico, garantindo que o padrão de evasão nunca seja previsível. Logo em seguida, executa uma breve fase de “escape” avançando por 0,2 segundos, o que evita que ele fique preso no próprio obstáculo e retoma automaticamente o vagueio em linha reta. Essa combinação de giro estocástico e fuga rápida cria trajetórias variadas, fazendo com que o tempo até encontrar a caixa móvel seja diferente a cada execução.

```
else {
  int dir = (rand() % 2) ? +1 : -1;                    // escolhe direção
  double duration = 0.5 + ((double)rand() / RAND_MAX); // duração aleatória
  double start_time = wb_robot_get_time();
  // gira até completar 'duration'
  while (wb_robot_step(TIME_STEP) != -1 &&
         wb_robot_get_time() - start_time < duration) {
    wb_motor_set_velocity(left_motor,  dir * VELOCITY_TURN);
    wb_motor_set_velocity(right_motor, -dir * VELOCITY_TURN);
  }
  // escape: avança um pouco
  double escape_start = wb_robot_get_time();
  while (wb_robot_step(TIME_STEP) != -1 &&
         wb_robot_get_time() - escape_start < ESCAPE_TIME) {
    wb_motor_set_velocity(left_motor,  MAX_VELOCITY);
    wb_motor_set_velocity(right_motor, MAX_VELOCITY);
  }
}
```

Supervisor e Detecção de Caixa Movimentada

Utilização do Supervisor para Leitura das Posições

Na primeira etapa o controlador recorre ao nó Supervisor do Webots para obter, em tempo de execução, referências a cada caixa definida em cena (DEF “CAIXA00” a “CAIXA17”). Com *wb_supervisor_node_get_from_def()* armazenamos essas referências em um vetor, e a cada ciclo de simulação usamos *wb_supervisor_node_get_position()* para ler as coordenadas atuais x, y, z de cada

caixa diretamente do mundo virtual, sem interromper o comportamento reativo de navegação do robô.

```
// --- Carrega referências das caixas pelo DEF ---
WbNodeRef caixa[QtddCaixa];
char nomeCaixa[10];
for (int i = 0; i < QtddCaixa; i++) {
  sprintf(nomeCaixa, "CAIXA%02d", i);              // DEF: CAIXA00, CAIXA01...
  caixa[i] = wb_supervisor_node_get_from_def(nomeCaixa);
  if (caixa[i] != NULL) {
    printf("%2d. %s encontrada em %p\n", i, nomeCaixa, (void*)caixa[i]);
  } else {
    printf("Falha ao carregar a posição da %s\n", nomeCaixa);
  }
}
printf("\n== CAIXAS OK ==\n\n");
```

Estrutura de Dados para Armazenar as Posições Originais/Iniciais

Na segunda etapa definimos uma *struct PosicaoCaixas* com três campos — x, y e z, e criamos um array *originais[]* capaz de guardar as posições iniciais de cada uma das 18 caixas. Logo após carregar todas as referências, lemos uma vez *pos[0]*, *pos[1]*, *pos[2]* de cada nó e as atribuímos a *originais[i]*, garantindo um registro imutável do estado inicial do ambiente.

```
typedef struct {     // armazena a posição XYZ de cada caixa
  double x;
  double y;
  double z;
} PosicaoCaixas;
```

```
// --- Armazena posições iniciais ---
PosicaoCaixas originais[QtddCaixa];
printf("POSIÇÃO ORIGINAL DAS CAIXAS:\n");
for (int i = 0; i < QtddCaixa; i++) {
  const double *pos = wb_supervisor_node_get_position(caixa[i]); // lê posição
  originais[i].x = pos[0];
  originais[i].y = pos[1];
  originais[i].z = pos[2];
  printf("CAIXA%02d: %5.2f, %5.2f, %5.2f\n",
         i, originais[i].x, originais[i].y, originais[i].z);
}
```

Detecção de Movimento Comparando com o Valor Original

Na terceira etapa, a cada passo de simulação voltamos a chamar *wb_supervisor_node_get_position()* e calculamos o desvio absoluto (fabs) entre a posição atual e a original em cada eixo. Utilizando um pequeno epsilon (1e-6) para filtrar ruídos numéricos, disparamos o alerta quando qualquer uma dessas diferenças ultrapasse o limite, sinalizando que a caixa foi efetivamente movimentada.

Quando o e-puck encosta na caixa que foi deslocada, seu controlador imediatamente zera as velocidades dos dois motores para parar o deslocamento, acende todos os LEDs do chassi como sinal visual de alerta e passa a girar continuamente sobre o próprio eixo, indicando que identificou a movimentação não autorizada da caixa.

```
  // 3) Verifica se alguma caixa mudou de posição
  for (int i = 0; i < QtddCaixa; i++) {
    if (caixa[i] == NULL) {
      continue;        // pula se nó inválido
    }
    const double *pos = wb_supervisor_node_get_position(caixa[i]);
    double dx = fabs(pos[0] - originais[i].x);
    double dy = fabs(pos[1] - originais[i].y);
    double dz = fabs(pos[2] - originais[i].z);

    if (dx > 1e-6 || dy > 1e-6 || dz > 1e-6) {
      // detectou movimento real da caixa
      printf("CAIXA%02d mudou!   antiga=(%.5f,%.5f,%.5f)   nova=(%.5f,%.5f,%.5f)\n",
             i,
             originais[i].x, originais[i].y, originais[i].z,
             pos[0], pos[1], pos[2]);

      // para o robô
      wb_motor_set_velocity(left_motor,  0.0);
      wb_motor_set_velocity(right_motor, 0.0);

      // acende todos os LEDs
      for (int j = 0; j < QtddLeds; j++) {
        wb_led_set(leds[j], 1);
      }

      // gira indefinidamente
      while (wb_robot_step(TIME_STEP) != -1) {
        wb_motor_set_velocity(left_motor,   VELOCITY_TURN);
        wb_motor_set_velocity(right_motor, -VELOCITY_TURN);
      }
    }
  }
}
```

Observação para a execução e teste do projeto

Após realizar o git clone do repositório, navegue até o diretório ProjetoIA/worlds. Em seguida, abra o arquivo TesteProjeto.wbt utilizando o Webots. Se, ao iniciar a simulação, o robô não começar a executar automaticamente, abra o mundo disponibilizado no moodle, selecione o robô, crie um novo controller em C, cole nele o código do controller que disponibilizei no repositório, caminho: ProjetoIA/controllers/my_controller/my_controller.c e execute novamente a simulação.

Considerações Finais

Variabilidade no Tempo de Detecção

Por se tratar de um procedimento de exploração puramente aleatória, o intervalo até que o e-puck encontre a caixa móvel pode oscilar amplamente entre diferentes execuções. Em alguns casos o robô encontra o objeto deslocado em poucos minutos, enquanto em outros percorre grande parte do mapa antes de acioná-lo.

CAIXA00 não Utilizada

No mapa de teste disponibilizado, a referência CAIXA00 foi mantida apenas para preservar a sequência dos identificadores, mas não está presente como obstáculo físico. Todas as verificações ocorrem sobre as caixas numeradas de 01 a 17.

Vídeo Demonstrativo e Repositório do GitHub

Para acompanhar o sistema em funcionamento, assista ao vídeo demonstrativo em: https://www.youtube.com/watch?v=tNnDUJCGiAY.

Além disso, é possível consultar o código-fonte completo no GitHub: https://github.com/jvgoverna/CC7711-Inteligencia-Artificial-e-Robotica/tree/main/ProjetoIA