UNICAMP

Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

IA903 – Introdução à Robótica Móvel 24 de novembro de 2024

Docente: Eric Rohmer

Discente:

- Gabriel Toffanetto França da Rocha – 289320

Exercise 4: Line-based Extended Kalman Filter for Robot Localization

---

Sumário

---

1 Transição de estados

Sendo o vetor de estados do robô dado por (1), e o vetor de entradas por (2), têm-se que a odometria, dado pelo modelo dinâmico incremental do sistema é dada por (3).

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & zt \end{bmatrix}^T \tag{1}$$

$$\mathbf{u}_t = \begin{bmatrix} \Delta s_l & \Delta s_r \end{bmatrix}^T \tag{2}$$

$$\widehat{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) = \mathbf{x}_{t-1} + \begin{bmatrix} \dfrac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta_{t-1} + \dfrac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \\ \dfrac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta_{t-1} + \dfrac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \\ \dfrac{\Delta s_r - \Delta s_l}{l} \end{bmatrix} \tag{3}$$

Realizando a derivada parcial da odometria em relação aos estados do sistema, obtém-se o jacobiano em relação aos estados, dado por (4). Fazendo o mesmo processo em relação ao vetor de entradas, obtém-se o jacobiano a associado às entradas, (5), com as derivadas parciais listadas de (6) à (11).

$$\begin{aligned} \mathbf{F_x} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x} & \dfrac{\partial \mathbf{f}}{\partial y} & \dfrac{\partial \mathbf{f}}{\partial z} \end{bmatrix} &= \begin{bmatrix} \dfrac{\partial f_x}{\partial x} & \dfrac{\partial f_x}{\partial y} & \dfrac{\partial f_x}{\partial z} \\ \dfrac{\partial f_y}{\partial x} & \dfrac{\partial f_y}{\partial y} & \dfrac{\partial f_y}{\partial z} \\ \dfrac{\partial f_z}{\partial x} & \dfrac{\partial f_z}{\partial y} & \dfrac{\partial f_z}{\partial z} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & \dfrac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta_{t-1} + \dfrac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \\ 0 & 1 & \dfrac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta_{t-1} + \dfrac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4}$$

$$\mathbf{F_u} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial \Delta s_l} & \dfrac{\partial \mathbf{f}}{\partial \Delta s_r} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_x}{\partial \Delta s_l} & \dfrac{\partial f_x}{\partial \Delta s_r} \\ \dfrac{\partial f_y}{\partial \Delta s_l} & \dfrac{\partial f_y}{\partial \Delta s_r} \\ \dfrac{\partial f_z}{\partial \Delta s_l} & \dfrac{\partial f_z}{\partial \Delta s_r} \end{bmatrix} \tag{5}$$

$$\frac{\partial f_x}{\partial \Delta s_l} = \frac{1}{2} \cos\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) + \frac{\Delta s_r + \Delta s_l}{2 \cdot 2 \cdot l} \sin\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \tag{6}$$

$$\frac{\partial f_x}{\partial \Delta s_r} = \frac{1}{2} \cos\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) - \frac{\Delta s_r + \Delta s_l}{2 \cdot 2 \cdot l} \sin\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \tag{7}$$

$$\frac{\partial f_y}{\partial \Delta s_l} = \frac{1}{2}\sin\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) - \frac{\Delta s_r + \Delta s_l}{2 \cdot 2 \cdot l}\cos\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \tag{8}$$

$$\frac{\partial f_y}{\partial \Delta s_r} = \frac{1}{2}\sin\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) + \frac{\Delta s_r + \Delta s_l}{2 \cdot 2 \cdot l}\cos\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2 \cdot l}\right) \tag{9}$$

$$\frac{\partial f_y}{\partial \Delta s_l} = -\frac{1}{l} \tag{10}$$

$$\frac{\partial f_y}{\partial \Delta s_r} = \frac{1}{l} \tag{11}$$

Aplicando a odometria pelo modelo incremental, observa-se na Figura 1 que rapidamente a posição obtida por meio da integração dos estados diverge da posição real, sendo uma informação que não confiável, sendo necessário a fusão da mesma com outros sensores para a obtenção de uma localização precisa.

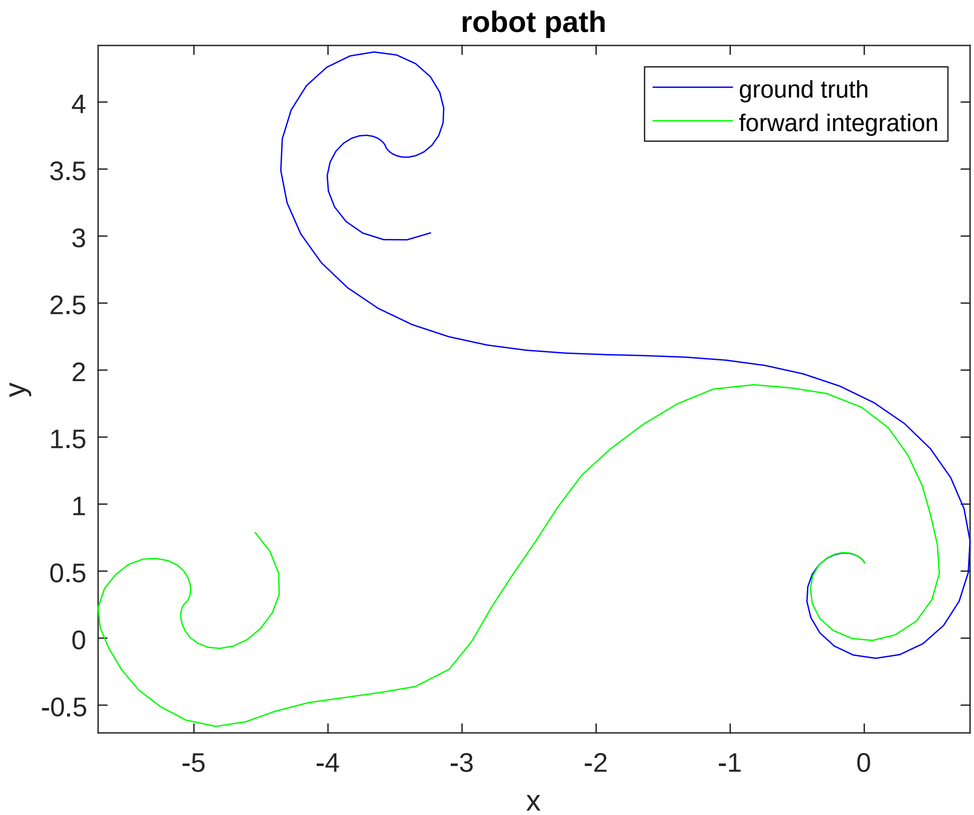

Figura 1: Caminho percorrido pelo robô × odometria obtida pela integração do modelo cinemático incremental direto.

A implementação da função de transição dos estados é dada abaixo:

```matlab
function [f, F_x, F_u] = transitionFunction(x,u, l)
% [f, F_x, F_u] = transitionFunction(x,u,l) predicts the state x at time t given
% the state at time t-1 and the input u at time t. F_x denotes the Jacobian
% of the state transition function with respect to the state evaluated at
% the state and input provided. F_u denotes the Jacobian of the state
% transition function with respect to the input evaluated at the state and
% input provided.
% State and input are defined according to "Introduction to Autonomous Mobile Robots", pp. 337

%STARTRM

f = x +  [(u(1) + u(2))/2*cos(x(3) + ((u(2)-u(1))/(2*l))); ...
(u(1) + u(2))/2*sin(x(3) + ((u(2)-u(1))/(2*l))); ...
(u(2) - u(1))/l ];

F_x = [1, 0, -(u(1) + u(2))/2*sin(x(3)+(u(2)-u(1))/(2*l)); ...
0, 1, (u(1) + u(2))/2*cos(x(3)+(u(2)-u(1))/(2*l)); ...
0, 0, 1];

F_u = [(cos(x(3) + (u(2)-u(1))/(2*l))/2 + (u(1) + u(2))/(2*2*l)*sin(x(3) + (u(2)-u(1))/(2*l))), ...
(cos(x(3) + (u(2)-u(1))/(2*l))/2 - (u(1) + u(2))/(2*2*l)*sin(x(3) + (u(2)-u(1))/(2*l))); ...
(sin(x(3) + (u(2)-u(1))/(2*l))/2 - (u(1) + u(2))/(2*2*l)*cos(x(3) + (u(2)-u(1))/(2*l))), ...
(sin(x(3) + (u(2)-u(1))/(2*l))/2 + (u(1) + u(2))/(2*2*l)*cos(x(3) + (u(2)-u(1))/(2*l))); ...
-1/l, 1/l];


%ENDRM
```

2 Atualização do estado

2.1 Função de medição

Dada a refência de uma linha dada pelo mapa, pelo vetor $\begin{bmatrix} \alpha_t^j & r_t^j \end{bmatrix}^T$, no referencial inercial do mundo, porém, para utilizar esses dados, é preciso saber qual a posição das linhas em relação ao robô, pois assim o mapa estará no mesmo referencial dos sensores que serão utilizados na percepção. Com isso, realiza-se a transformação para o referencial do robô no estado atual por meio de (12). O jacobiano dessa transformação pode ser obtido aplicando as derivadas parciais em relação aos estados, como feito em (13).

$$\mathbf{h}(\widehat{\mathbf{x}}_t, \mathbf{m}^j) = \begin{bmatrix} \alpha_t^j - \widehat{\theta}_t \\ r_t^j - \widehat{x}_t \cos(\alpha_t^j) - \widehat{y}_t \sin(\alpha_t^j) \end{bmatrix} \tag{12}$$

$$\mathbf{H}^J = \begin{bmatrix} \dfrac{\partial \alpha_t^j}{\partial \widehat{x}} & \dfrac{\partial \alpha_t^j}{\partial \widehat{y}} & \dfrac{\partial \alpha_t^j}{\partial \widehat{z}} \\ \dfrac{\partial r_t^j}{\partial \widehat{x}} & \dfrac{\partial r_t^j}{\partial \widehat{y}} & \dfrac{\partial r_t^j}{\partial \widehat{z}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ -\cos(\alpha_t^j) & -\sin(\alpha_t^j) & 0 \end{bmatrix} \tag{13}$$

A implementação da função de medição é dada abaixo:

```
function [h, H_x] = measurementFunction(x, m)
% [h, H_x] = measurementFunction(x, m) returns the predicted measurement
% given a state x and a single map entry m. H_x denotes the Jacobian of the
% measurement function with respect to the state evaluated at the state
% provided.
% Map entry and state are defined according to "Introduction to Autonomous
    Mobile Robots" pp. 337

%STARTRM
h = [m(1) - x(3);
m(2) - x(1)*cos(m(1)) - x(2)*sin(m(1))];

H_x = [0, 0, -1;
-cos(m(1)), -sin(m(1)), 0];

%ENDRM

[h(1), h(2), isRNegated] = normalizeLineParameters(h(1), h(2));

if isRNegated
H_x(2, :) = - H_x(2, :);
end
```

2.2 Filtro

Para aplicação da correção utilizando filtro de Kalman estendido, é necessário inicialmente realizar a etapa de predição, feita anteriormente pela função de transição de estados. Além disso, é preciso saber a covariância do erro *a priori* ($\widehat{\mathbf{P}}_t$), que é obtida com base na matriz de covariância $\mathbf{Q}$, dada por (14).

$$\mathbf{Q} = \begin{bmatrix} k|\Delta s_r| & 0 \\ 0 & k|\Delta s_l| \end{bmatrix} \tag{14}$$

Sendo assim, a covariância do erro *a priori* é dada por (15), utilizando os jacobianos obtidos em (4) e (5).

$$\widehat{\mathbf{P}}_t = \mathbf{F_x}\mathbf{P}\mathbf{F_x}^T + \mathbf{F_u}\mathbf{Q}\mathbf{F_u}^T \tag{15}$$

Realizada a predição, é feita a etapa de correção, onde uma vez feito o *matching* do estado predito com a leitura dos sensores exteroceptivos, é obtida a inovação ($\mathbf{S}$), que permite a obtenção do ganho de Kalman ($\mathbf{K}$) que é utilizado para realização da correção propriamente dita.

Em (16) é obtida a inovação com base na covariância da predição e o jacobiano do sensor, obtido em (13), além da matriz de covariância do sensor $\mathbf{R}$. Por fim, o ganho de Kalman é conforme (17), e a correção é feita em (18), obtendo a predição *a posteriori*.

$$\mathbf{S}_t = \mathbf{H}\widehat{\mathbf{P}}_t\mathbf{H}^T + \mathbf{R}_t \tag{16}$$

$$\mathbf{K}_t = \widehat{\mathbf{P}}_t\mathbf{H}^T\mathbf{S}^{-1} \tag{17}$$

$$\widehat{\mathbf{x}}_{t|t} = \widehat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\widetilde{\mathbf{y}}_t \tag{18}$$

Por meio da filtragem da odometria utilizando filtro de Kalman extendido, fundindo os dados juntamente à medida da percepção do robô, observa-se na Figura 2 que a localização do robô se torna precisa, acompanhando a trajetória do mesmo com fidelidade.

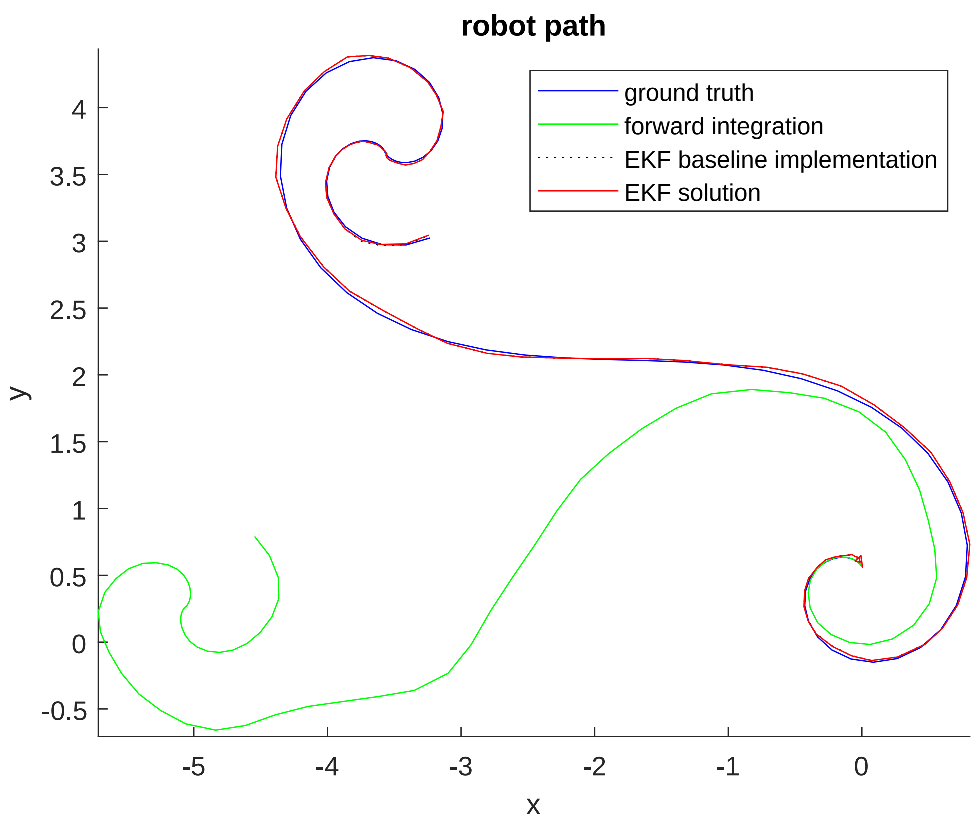

Figura 2: Correção da odometria por meio do filtro de Kalman extendido.

```
function [x_posteriori, P_posteriori] = filterStep(x,P,u,Z,R,M,k,g,l)
Q = [k*abs(u(2)) 0; 0 k*abs(u(1))];

[x_priori, F_x, F_u] = transitionFunction(x, u, l);
P_priori = F_x*P*F_x' + F_u*Q*F_u';

if size(Z,2) == 0
x_posteriori = x_priori;
P_posteriori = P_priori;
return;
end

[v, H, R] = associateMeasurements(x_priori, P_priori, Z, R, M, g);

y = reshape(v, [], 1);
H = reshape(permute(H, [1,3,2]), [], 3);
R = blockDiagonal(R);

S = H*P_priori*H'+R; %S is the innovation covariance matrix
K = P_priori*H'/S;

x_posteriori = x_priori + K * y;
P_posteriori = (eye(size(P_priori)) - K*H) * P_priori;
```

3 Experimento V-REP

Ao executar a simulação de localização incremental no CoppeliaSim, utilizando a fusão dos dados de odometria com os dados obtidos com o LiDAR comparados ao mapa, obtém-se uma localização precisa, como visto na Figura 4, onde o robô *ghost* está posicionado de forma coincidente ao robô real. Na Figura 3, observba-se o *match* feito entre as retas presentes no mapa, transformadas para o referencial do robô, e as retas obtidas por meio da percepção. Mesmo com o casamento de apenas algumas retas, a correção da localização é poderosa.

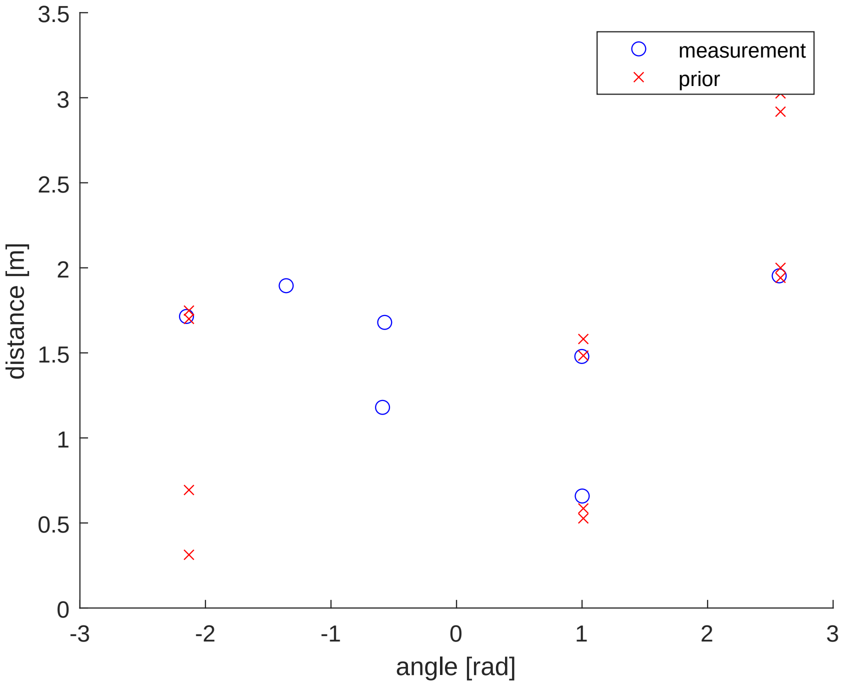

Figura 3: Representação das retas do mapa (prior) e as retas obtidas pelo LiDAR no espaço de Hough.

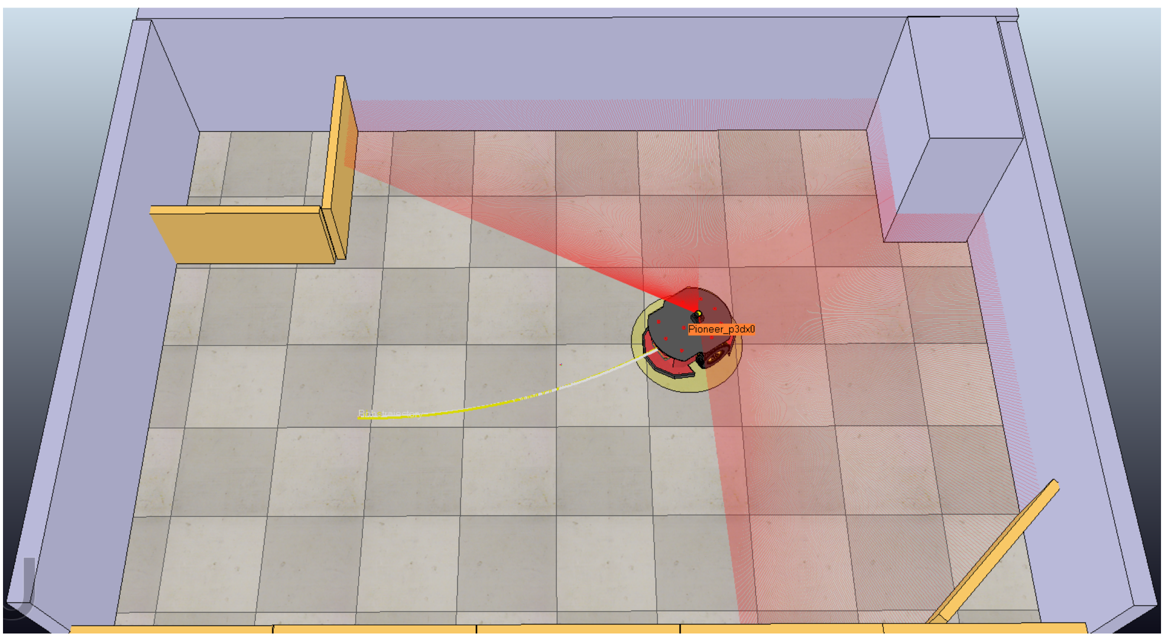

Figura 4: Estado final do robô na simulação.

A implementação da função de localização incremental utilizando filtro de Kalman estendido é dada abaixo:

```
function [x_posterori, P_posterori] = incrementalLocalization(x, P, u, S, M,
    params, k, g, l)
% [x_posterori, P_posterori] = incrementalLocalization(x, P, u, S, R, M,
% k, l, g) returns the a posterori estimate of the state and its covariance,
% given the previous state estimate, control inputs, laser measurements and
% the map

C_TR = diag([repmat(0.1^2, 1, size(S, 2)) repmat(0.1^2, 1, size(S, 2))]);
[z, R, ans] = extractLinesPolar(S(1,:), S(2,:), C_TR, params);


%STARTRM
figure(2), cla, hold on;

%compute z_prior
z_prior = zeros(size(M));

for j = 1:size(M, 2)
[z_prior(:,j) H] = measurementFunction(x, M(:,j));
end

plot(z(1,:), z(2,:),'bo');
plot(z_prior(1,:), z_prior(2,:),'rx');
xlabel('angle [rad]'); ylabel('distance [m]')
legend('measurement','prior')
drawnow

% estimate robot pose
[x_posterori, P_posterori] = filterStep(x, P, u, z, R, M, k, g, l); %hint:
    you just coded this function

%ENDRM
```

Referências

SIEGWART, R.; NOURBAKHSH, I. *Introduction to Autonomous Mobile Robots*. [S.l.]: Bradford Book, 2004. (A Bradford book). ISBN 9780262195027.