Potential Fields Tutorial

Michael A. Goodrich

1 Introduction

Rodney Brooks introduced the concept of *reactive robotics*, often referred to as *behavior-based robotics*, wherein robot architectures are built on *activity-generating* building blocks rather than on centralized representations and deductive logic. You will soon be familiar with these concepts when you read the paper by Brooks, *Intelligence without Representation* which we will discuss next time. In addition to the seminal work by Brooks, two references that I like on reactive robotics are [3, 1]. Much of this tutorial is adapted from these references. (Note that Arkin identifies different behaviors as *schemas*. I don't see much advantage in using this term, but it has been around in cognitive psychology at least since the time of Piaget so maybe Arkin uses the term because it is somewhat grounded in science.)

The basic idea of behavior-based robotics is that robots should be built from the bottom up using ecologically adapted *behaviors*. These behaviors are often simple stimulus-response loops, meaning that the robot senses some environmental stimulus (like light) and generates some action in response (like turn toward the light). In Murphy's words, a behavior is a mapping from sensor input to a pattern of motor action; see Figure 1.

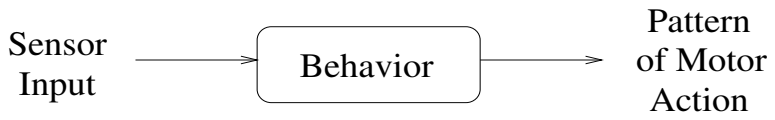

Figure 1: **Graphical definition of a behavior, adapted from Murphy [3].**

There are two approaches to managing multiple behaviors: always-on and sometimes-on. *Always-on* behaviors are always looking at the environment and issuing actuator commands. *Sometimes-on* behaviors only issue actuator commands when some environmental signal triggers their behavior. We will only talk about *always-on* behaviors, but you should know that there is some cool work on *sometimes-on* behaviors too (can you think of a technique for managing when behaviors are triggered?). You should also know that there are techniques (e.g., Brooks's subsumption architecture [2]) where the actuator commands issued by *always-on* behaviors are suppressed or inhibited by other behaviors.

In this class, we will talk about two different *always-on* techniques: *potential fields* and *utilitarian voting*. As we discuss these techniques (in this and another tutorial), we will address how behaviors are represented and how they are combined.

2 Basic Potential Fields Concepts

When you think of potential fields, picture in your mind either a charged particle navigating through a magnetic field or a marble rolling down a hill. The basic idea is that behavior exhibited by the particle/marble will depend on the combination of the shape of the field/hill. Unlike fields/hills where the topology is externally specified by environmental conditions, the topology of the potential fields that a robot experiences are determined by the designer. More specifically, the designer (a) creates multiple behaviors, each assigned a particular task or function, (b) represents each of these behaviors as a potential field, and (c) combines all of the behaviors to produce the robot's motion by combining the potential fields. We will deal with very simple behaviors, show how to represent these simple behaviors using potential fields, and then show how to combine these behaviors.

2.1 Representing Behaviors

The fundamental building block of potential fields is the action vector, which corresponds, roughly, to the speed and orientation of a moving robot. Each behavior outputs a desired output vector. For example, consider a *SeekGoal* behavior that is assigned the task of making the robot head toward an identified goal. The output of the *SeekGoal* behavior is a vector that points the robot toward the goal. If I took the robot on a pretend journey through every point in a two-dimensional world and recorded the output vector at each point, the collection of these vectors would look something like the diagram illustrated in Figure 2. This collection of vectors is called a potential field because it

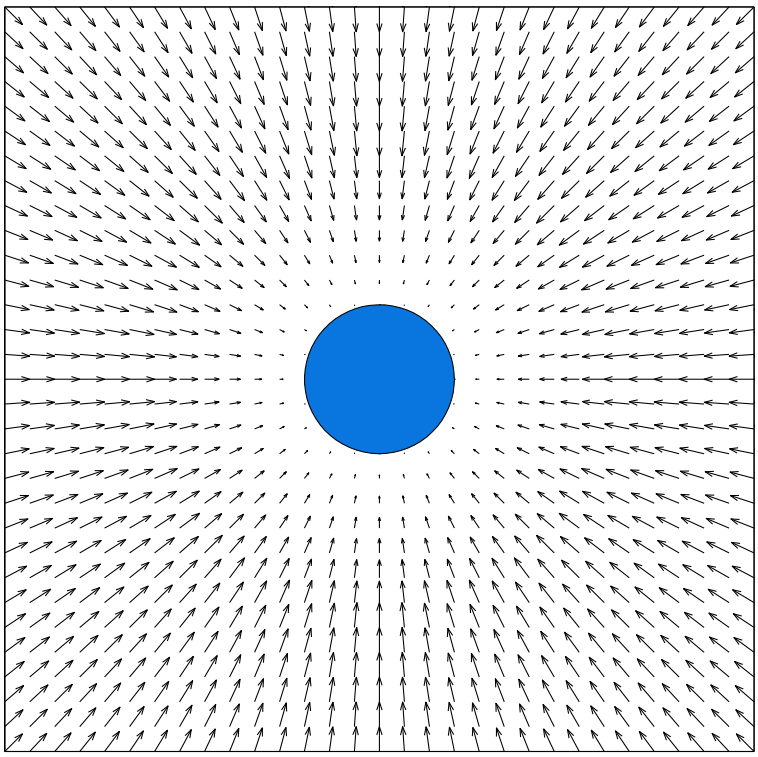

Figure 2: **An attractive potential field, corresponding to the *SeekGoal* behavior.**

represents synthetic energy potentials that the robot will follow. The potential field associated with the *SeekGoal* behavior is an example of an *attractive potential* because the field causes the robot to be attracted to the goal (i.e., all vectors point to the goal). You should note that this whole field is never actually computed. I computed it so that you could get a "god's eye" perspective on what the robot would experience at any point in the space, but the robot only sees that portion of the field that it directly encounters.

You might be interested in how I generated the above potential field. One way to think of a potential field is to think of it as a mapping from one vector into another vector. For the 2-D navigation in the figure, it is the mapping from the vector $\mathbf{v} = [x, y]^T$ into the gradient vector $\Delta = [\Delta x, \Delta y]^T$ (the superscript T represents "transpose" — I use it because I like column vectors better than row vectors). Now, we could find Δ by defining some vector function of $\mathbf{v}$ and then taking the gradient of this function, but I prefer a more direct approach. To generate the above fields, I just defined Δx and Δy in terms of $\mathbf{v}$ as follows:

- Let (x_G, y_G) denote the position of the goal. Let r denote the radius of the goal. Let $\mathbf{v} = [x, y]^T$ denote the (x, y) position of the agent.
- Find the distance between the goal and the agent: $d = \sqrt{(x_G - x)^2 + (y - y_G)^2}$.
- Find the angle between the agent and the goal $\theta = \tan^{-1}\left(\frac{y_G - y}{x_G - x}\right)$. (I use the *atan2* function because it gives you the angle in the correct quadrant.)
- Set Δx and Δy according to the following:

$$\begin{array}{cc} \text{if } d < r & \Delta x = \Delta y = 0 \\ \text{if } r \leq d \leq s + r & \Delta x = \alpha(d - r)\cos(\theta) \text{ and } \Delta y = \alpha(d - r)\sin(\theta) \\ \text{if } d > s + r & \Delta x = \alpha s \cos(\theta) \text{ and } \Delta y = \alpha s \sin(\theta) \end{array}$$

This sets up a goal as a circle with radius r. When the agent reaches the goal no forces from the goal act upon it, whence when $d < r$ both Δx and Δy are set to zero. The field has a spread of s and the agent reaches the extent of this field when $d = s + r$. Outside of this circle of extent, the vector magnitude is set to the maximum possible value. Within this circle of extent but outside of the goal's radius, the vector magnitude is set proportional to the distance between the agent and the goal. I include the constant $\alpha > 0$ so that the strength of the field can be easily scaled.

2.2 Combining Potential Fields

In the world diagrammed in Figure 2, the robot would easily glide to the origin so this is not a very interesting problem. It gets interesting when, in addition to the *SeekGoal* behavior, we have other behaviors such as the *AvoidObstacle* behavior diagrammed in Figure 3. I defined the *AvoidObstacle* potential field as follows:

- Let (x_O, y_O) denote the position of the obstacle. Let r denote the radius of the obstacle. Let $\mathbf{v} = [x, y]^T$ denote the (x, y) position of the agent.
- Find the distance between the obstacle and the agent: $d = \sqrt{(x_O - x)^2 + (y - y_O)^2}$.

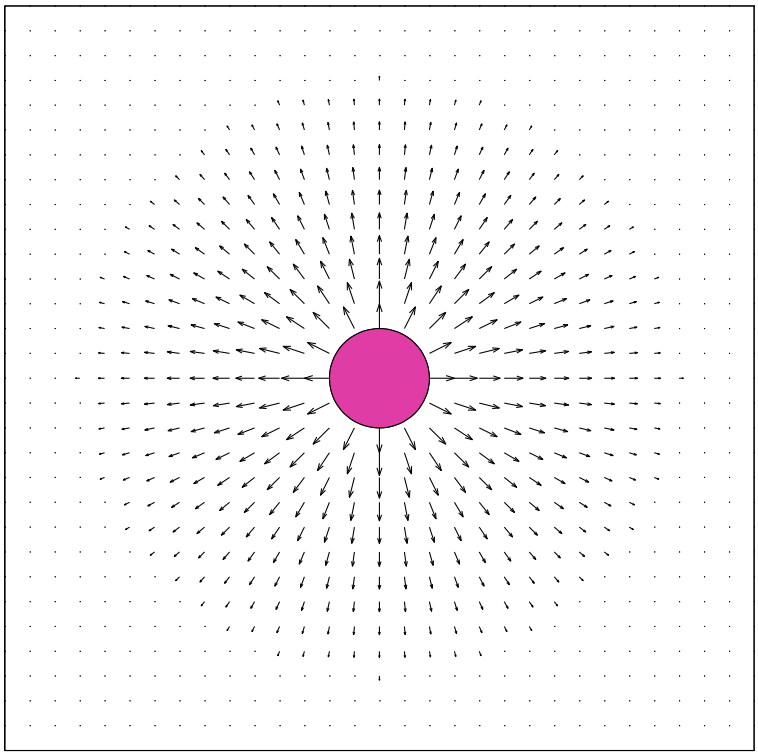

Figure 3: **A reject potential field, corresponding to the *AvoidObstacle* behavior.**

- Find the angle between the agent and the obstacle $\theta = \tan^{-1}\left(\frac{y_O - y}{x_O - x}\right)$. (I use the *atan2* function because it gives you the angle in the correct quadrant.)
- Set Δx and Δy according to the following:

$$\begin{array}{rc} \text{if } d < r & \Delta x = -\text{sign}(\cos(\theta))\infty \text{ and } \Delta y = -\text{sign}(\sin(\theta))\infty \\ \text{if } r \leq d \leq s + r & \Delta x = -\beta(s + r - d)\cos(\theta) \text{ and } \Delta y = -\beta(s + r - d)\sin(\theta) \\ \text{if } d > s + r & \Delta x = \Delta y = 0 \end{array}$$

Within the obstacle, the repulsive potential field is infinite and points out from the center of the obstacle (the *sign* function returns the sign of an argument). Outside of the circle of influence, the repulsive potential field is zero. Within the circle of influence but outside the radius of the obstacle, the magnitude of the vector grows from zero (when $\beta(s + r - d) = 0$ when $d = s + r$ corresponding to the agent being on the edge of the circle of influence) to $\beta(s)$ ($\beta(s + r - d) = \beta(s)$ when $d = r$ corresponding to the agent being on the edge of the obstacle). The constant $\beta > 0$ is given to allow the agent to scale the strength of this field. Notice that the vector points away from the obstacle; this done by introducing the negative sign into the definitions of Δx and Δy.

Since many problems will have both goals (e.g., a flag, in the capture the flag game) and obstacles (e.g., an opponent, in the capture the flag game), we have to figure out how to combine these potential fields. We combine multiple potential fields by adding them together. Doing this for our two-behavior robot in a world with both an obstacle and a goal gives the new potential field diagrammed in Figure 4. This field was generated by first finding $\Delta_G x$ and $\Delta_G y$, the vector

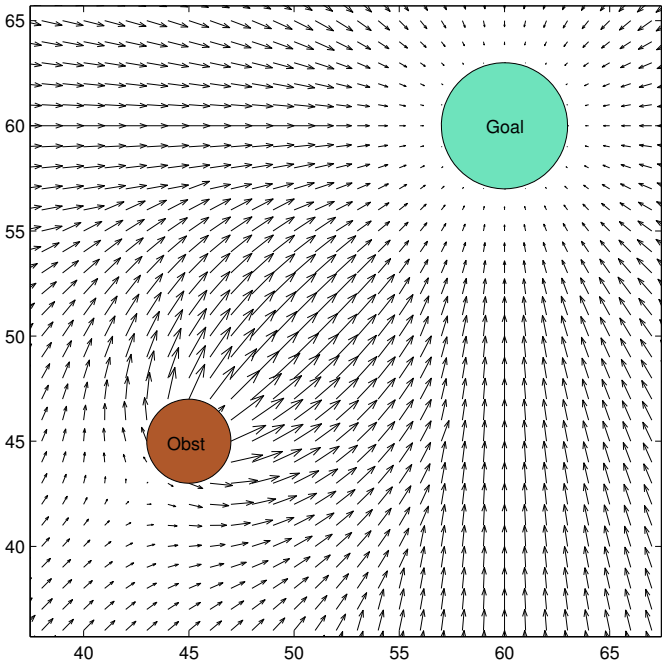

Figure 4: **The potential field generated by our two-behavior robot when there is a goal and an obstacle.**

generated by the attractive goal, second finding $\Delta_O x$ and $\Delta_O y$, the vector generated by the repulsive obstacle, and third adding these vectors together to find

$$\Delta x = \Delta_O x + \Delta_G x \quad (1)$$
$$\Delta y = \Delta_O y + \Delta_G y. \quad (2)$$

2.3 Choosing Actions

Suppose that we put our two-behavior robot near the origin in the world diagrammed in Figure 4. How does the robot choose its behavior? It determines the Δx using Equation (1) of the potential field generated by its two behaviors, determines Δy using Equation (2), finds the velocity $v = \sqrt{(\Delta x^2 + \Delta y^2)}$ and the angle $\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$, and then sets the speed to v and the direction to θ. As it moves through the world, it makes observations of the environment, identifies new action vectors, and chooses new directions/speeds. The resulting trajectory of the robot looks something like the path diagrammed in Figure 5.

3 Types of Potential Fields

We have introduced two types of fields, but there are others that might be useful. I'll let you figure out kinds of worlds where they may might be useful, though I might give you a couple of ideas along the way. Some of the fields are well summarized in [3], and others in [1].

The first field is the *uniform* potential field, and is illustrated in Figure 6. It is obtained by setting $\Delta x = c$ and $\Delta y = 0$ (or other constant values that represent the direction desired). It might

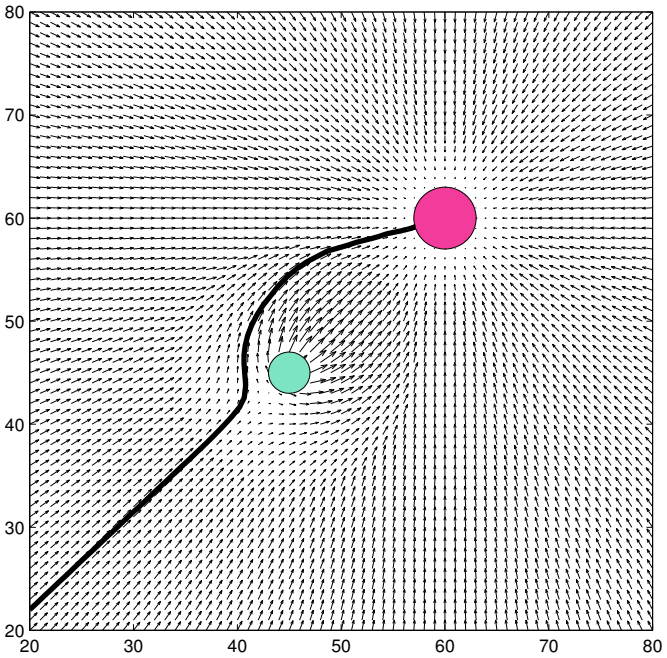

Figure 5: **The trajectory experienced by our two-behavior robot when there is a goal and an obstacle.**

be useful for a *FollowWall* behavior or a *ReturnToTerritory* behavior.

The second field is the *perpendicular* potential field, and is illustrated in Figure 7. It might be useful for a *AvoidWall* or a*AvoidTerritory* behavior. It is obtained by setting $\Delta x = \pm c$ and $\Delta y = 0$ (or other constant values that represent the direction directions).

The fourth field is the *tangential* potential field, and is illustrated in Figure 8. This field is obtained by finding the magnitude and direction in the same way as for the repulsive obstacle. However, θ is modified before Δx and Δy are defined by setting $\theta \leftarrow \theta \pm 90^\circ$ which causes the vector to shift from pointing away from the center of the obstacle to pointing in the direction of the tangent to the circle. The sign of the shift controls whether the tangential field causes a clockwise our counterclockwise spin. It might be useful for a *PatrolFlag* or *CircleOpponent* behavior.

The sixth field is the *Random* potential field. This field is useful because it helps the agent bounce around and avoid getting stuck in local minima that might arise when multiple fields are added together. It is obtained by randomly selecting d from a uniform distribution (i.e., picking any point with equal probability) over the interval $[0, \gamma]$ and θ from a uniform distribution over the interval $[0, 2 * \pi]$.

The final field is the *AvoidPast* potential field. Let me motivate why we might want this field. Consider the world with a box canyon diagrammed in Figure 10. If the robot starts at the bottom of the world, it will be attracted into the box canyon by the goal. As it enters the box canyon, it will begin to experience the forces from the *AvoidWall* behavior which will cause it to center in the canyon. Eventually, it will reach a point where the attractive potential from the goal is insufficient to overcome the repulsive potentials from the walls and the robot will be stuck.

One way to handle this box canyon effect is to introduce an *AvoidPast* behavior which sets ups repelling potential fields for each location that it has visited — more recently visited areas have more

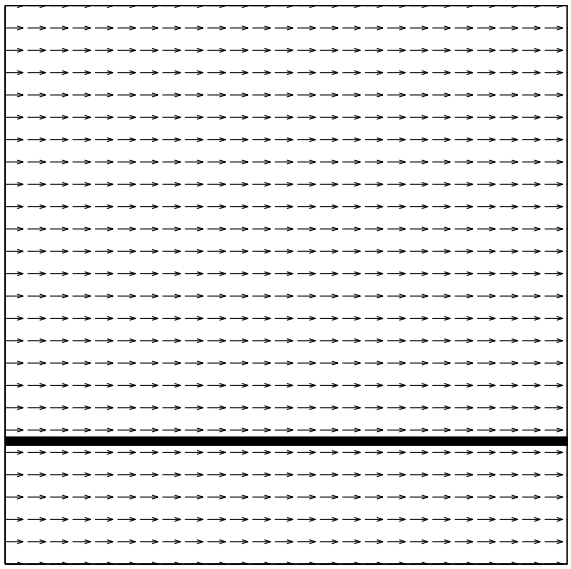

Figure 6: **The uniform potential field.**

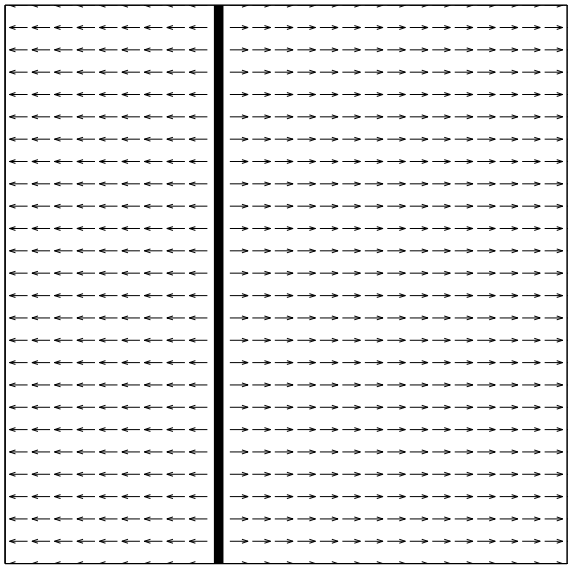

Figure 7: **The perpendicular potential field.**

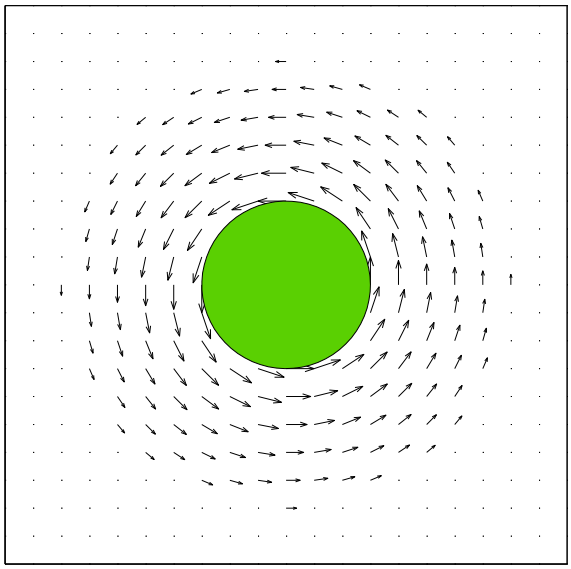

Figure 8: **The tangential potential field.**

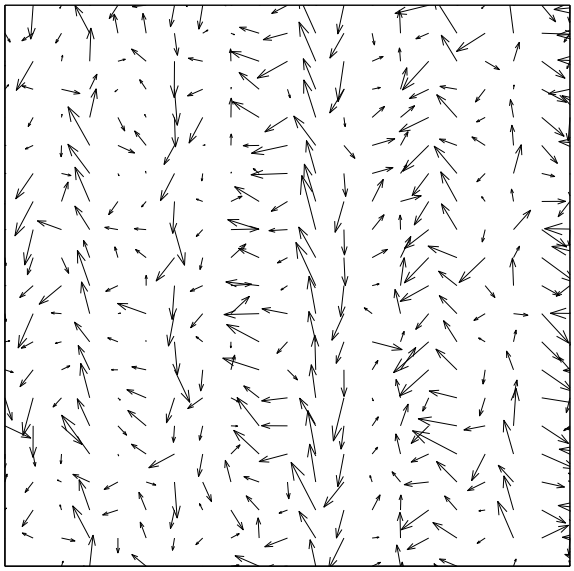

Figure 9: **The random potential field.**

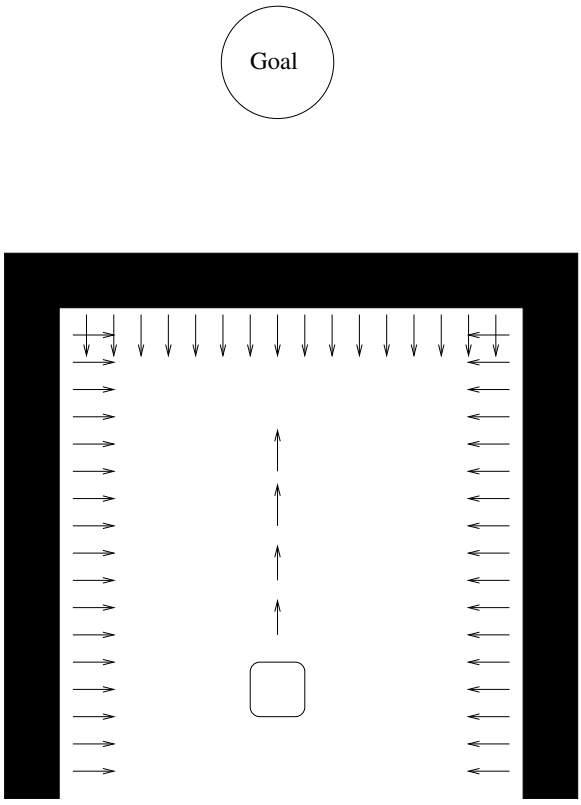

Figure 10: **A world with a box canyon.**

repelling force. If set up right, the robot would explore down the box canyon, retreat as it avoids its past locations, and eventually find its way around the canyon. A very simple implementation is to introduce a series of repulsive potentials with zero radius located at recently visited (x, y) positions. β can be decreased as time progresses so that recently visited locations are more repulsive than positions visited long ago. If you are interested, you can find a paper by Balch and Arkin about avoiding the past on the web.

References

[1] R. C. Arkin. *Behavior-Based Robotics.* MIT Press, Cambridge, Massachusetts, 1998.

[2] R. A. Brooks. *Cambrian Intelligence: The Early History of the New AI.* MIT Press, Cambridge, Massachusetts, 1999.

[3] R. R. Murphy. *Introduction to AI Robotics.* MIT Press, 2000.