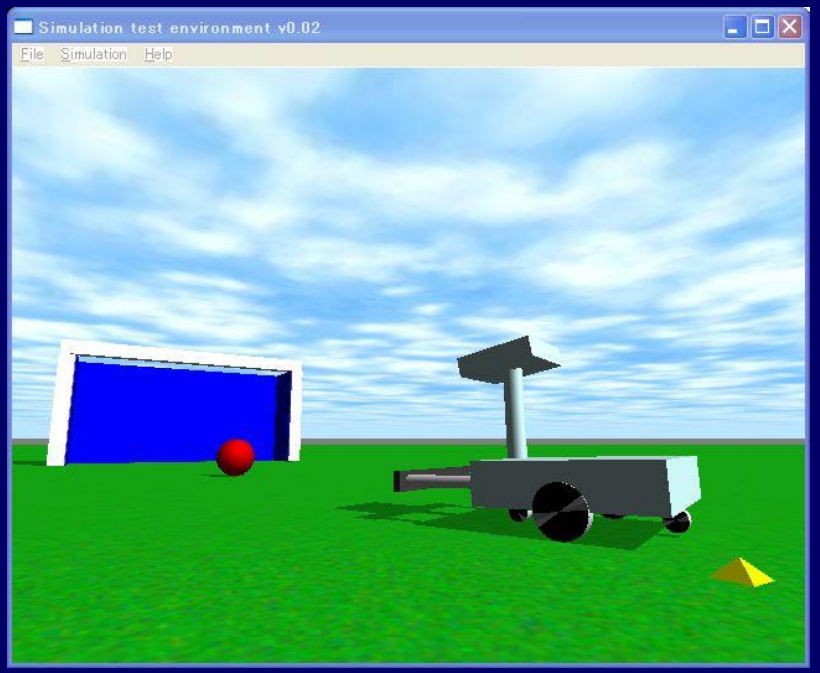
# ロボットシミュレーション

## ODE Dynamics Engineによるロボットプログラミング

## Part2: 車輪型ロボット



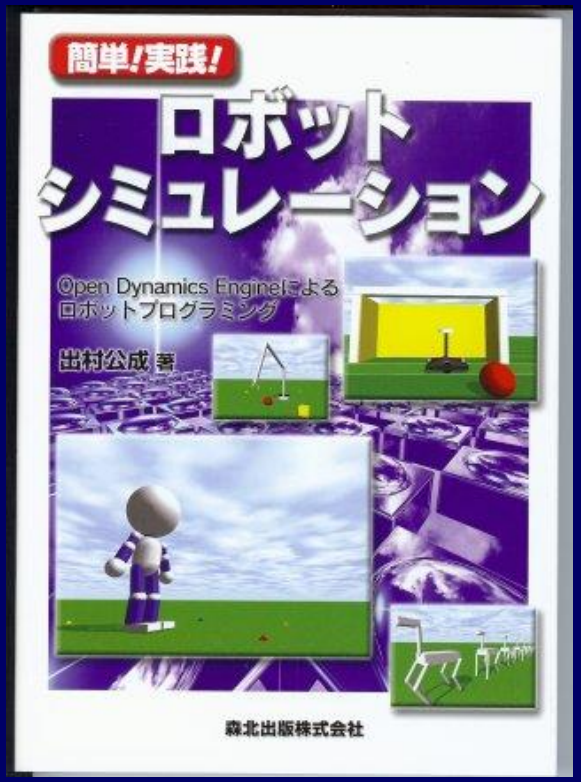

2007-8-23版

出村　公成（でむらこうせい）

[Web] http://demura.net
[Mail] info@demura.net

# 教科書

- 簡単！実践！ロボットシミュレーション
  Open Dynamics Engineによるロボットプログラミング
- 出村公成著
- 森北出版
- 2007年5月
- ISBN-13: 978-4627846913



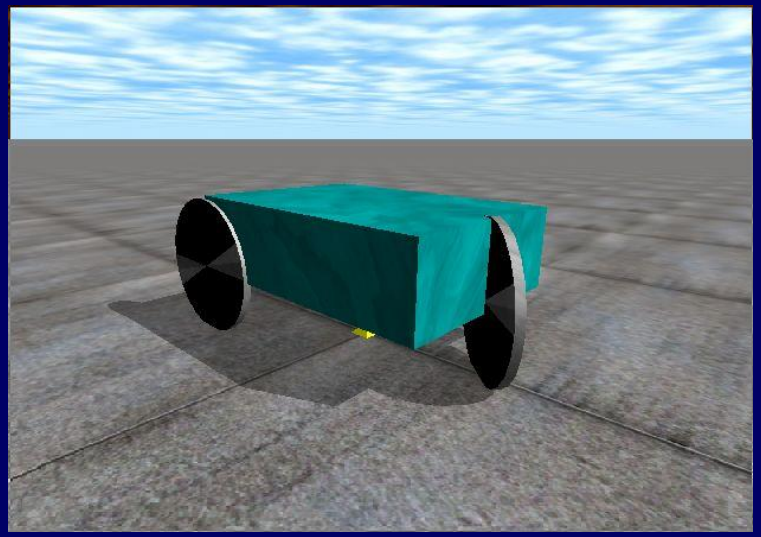

簡単！実践！ロボットシミュレーションの表紙

# 内 容

- 1限目
  - ロボットの種類
  - 座標変換
- 2限目
  - ボールの探索
  - 自己位置同定
  - ナビゲーション
    - ポテンシャル法
- 3限目
  - エクセサイズ

# ロボットの種類

- ステアリング型 (steering drive)
  - 自動車のように車輪の向きを変化させて進行方向を変化
- 作動駆動型 (differential drive)
  - 左右の駆動輪の回転数差により進行方向を変化



ステアリング型

# 座標変換

- 絶対座標系
  - ワールド座標系
  - 全ての物体に共通，動かない
- 相対座標系
  - ローカル座標系
  - ロボット座標系
  - 各物体に固定された座標系，物体が移動，回転するとそれに伴い移動，回転する

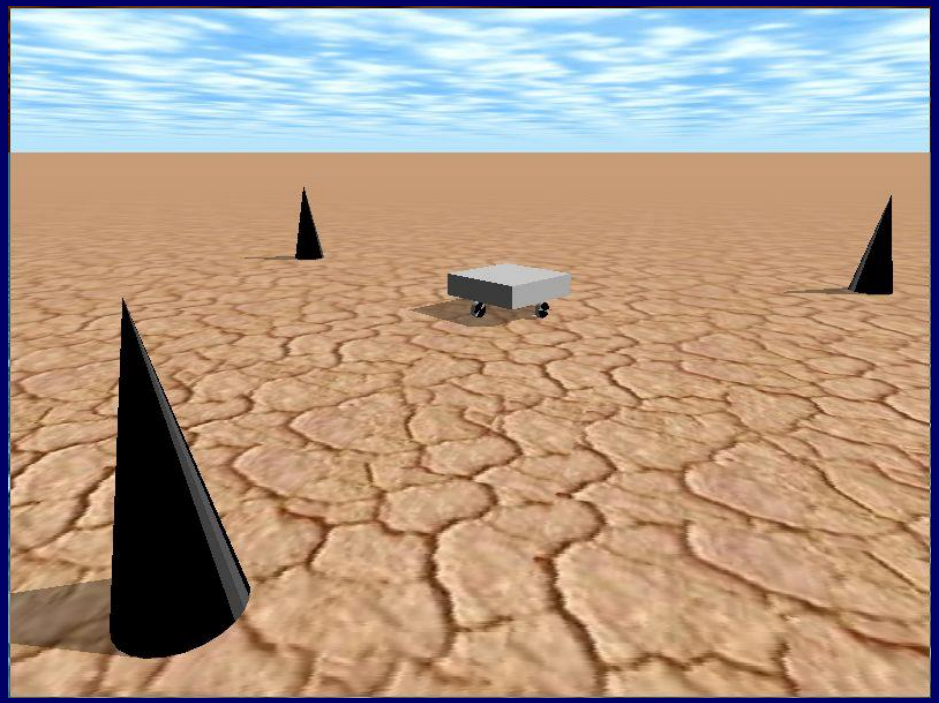
# 位置センサを作るには？

- 絶対座標での位置
- ロボットから見た位置（相対座標系）

絶対座標系

↓

相対座標系



# 位置センサの作り方

ODEなどの物理計算エンジンでは、各物体の絶対座標系での位置や速度の情報などはAPIなどで簡単に取得可能。

- ステップ1：絶対座標系を相対座標系に変換
- ステップ2：直交座標系を極座標系に変換

# 絶対座標と相対座標

- x, y：絶対座標系での物体の位置
- x’, y’: 相対座標系での物体の位置
- $x_r$, $y_r$: 相対座標系の原点
  絶対座標系でのロボットの位置

絶対座標系の位置 ＝ 回転行列 × 相対座標系の位置 ＋ 相対座標系の原点

$$\begin{bmatrix} x \\ y \end{bmatrix} = R(\theta) \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

# 絶対座標と相対座標

- x, y：絶対座標系でのロボットの位置
- x', y': 相対座標系での目標の位置
- $x_r$, $y_r$: 相対座標系の原点

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R^{-1}(\theta) \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} = R^T(\theta) \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix}$$
$$= \begin{bmatrix} (x - x_r)\cos\theta + (y - y_r)\sin\theta \\ -(x - x_r)\sin\theta + (y - y_r)\cos\theta \end{bmatrix} \quad (4.7)$$

相対座標系の位置 ＝ 回転行列の転置
× (絶対座標系の位置 − 相対座標系の原点)

# 直交座標系→極座標系

- 角度と距離
- 注意：角度の取り方が逆（時計まわり）

P117参照

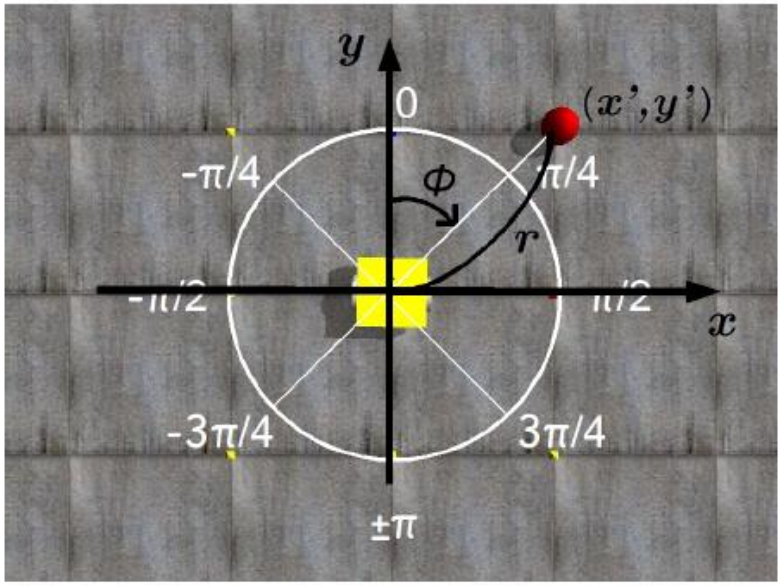



図 4.6　ロボット座標での角度

教科書 P117から転載

# カメラを搭載

- 指向性カメラ
  - 視界が制限
  - 広角でも90度程度
  - 解像度が高い
- 全方位カメラ
  - 360度の視界
  - 解像度が低い

指向性カメラを装備したロボット

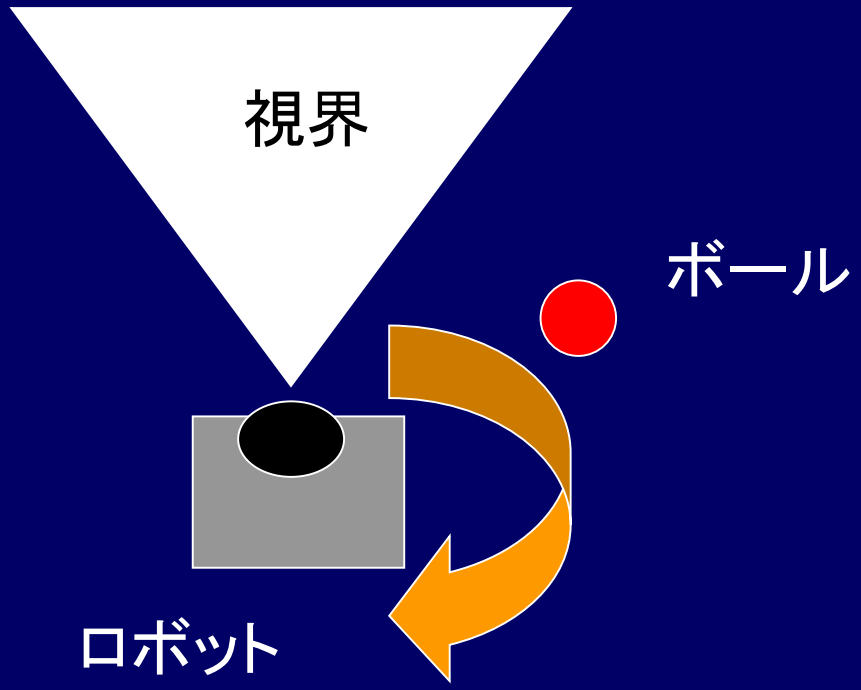
全方位カメラを装備したロボット

# 演　習

- プログラム4.1（P112）のロボットに視野角６０度の指向性カメラを搭載して，ボールを追跡するプログラムを作ろう！

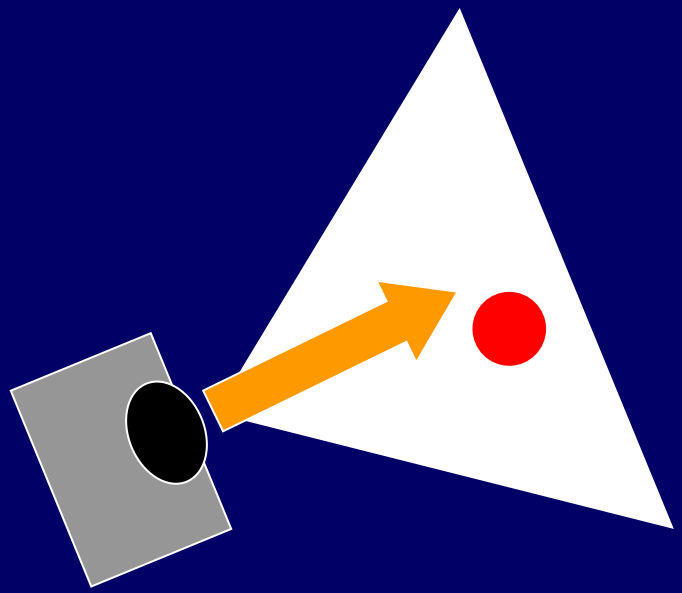
# ボールの探索

- 指向性カメラの場合は視界が制限されているので回転して探索



# ボールの追跡

- ボールが視界内にあれば直進



# ボール追跡のアルゴリズム

STEP1: ボールが見つかるまで、その場回転する。見つかったらSTEP2へ進む。

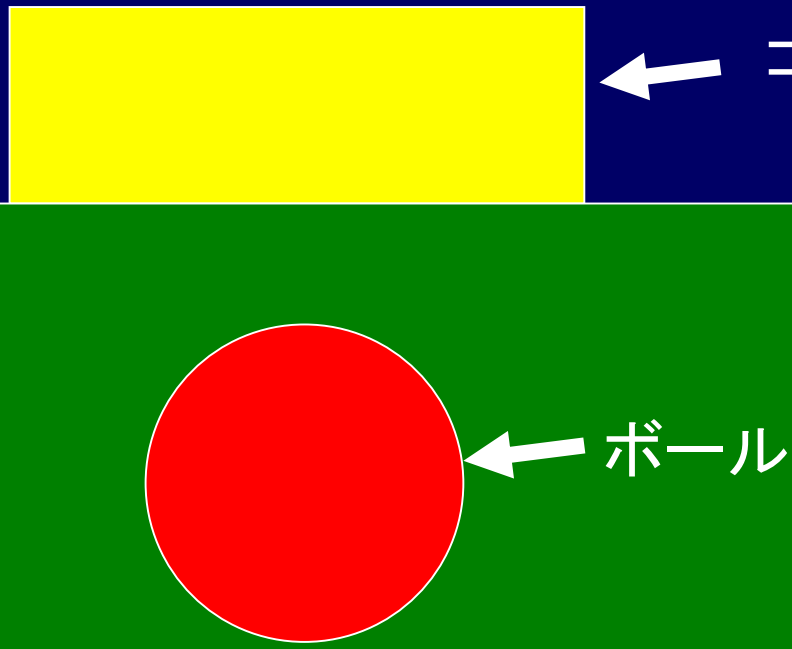
STEP2: 直進する。ボールを見失ったらSTEP1へ戻る。

# ロボカップのロボットをプログラムしよう！

- ボールをゴールまで運ぶプログラムを作ろう
- ボールがゴールの見通し線上にない場合にロボットはどのように移動すれば良いか（回り込み）

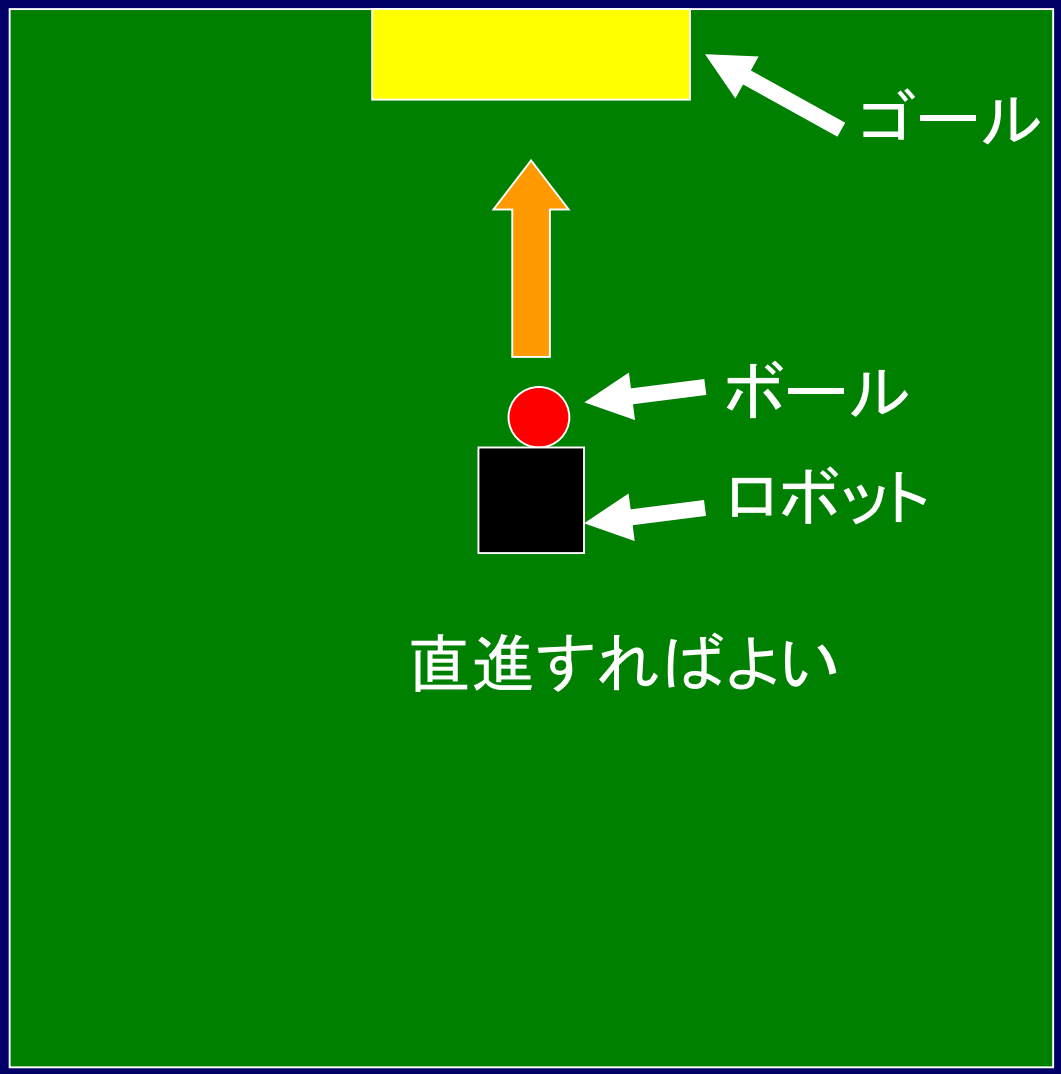

教科書P129から転載

# ボールをゴールへ運ぶには1

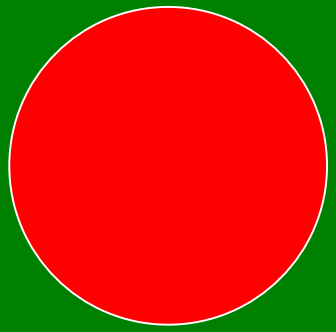



この図はロボットのカメラ画像
ロボットはドリブルしている
ロボットはどの方向に移動すれば良いか？

# ボールをゴールへ運ぶには1

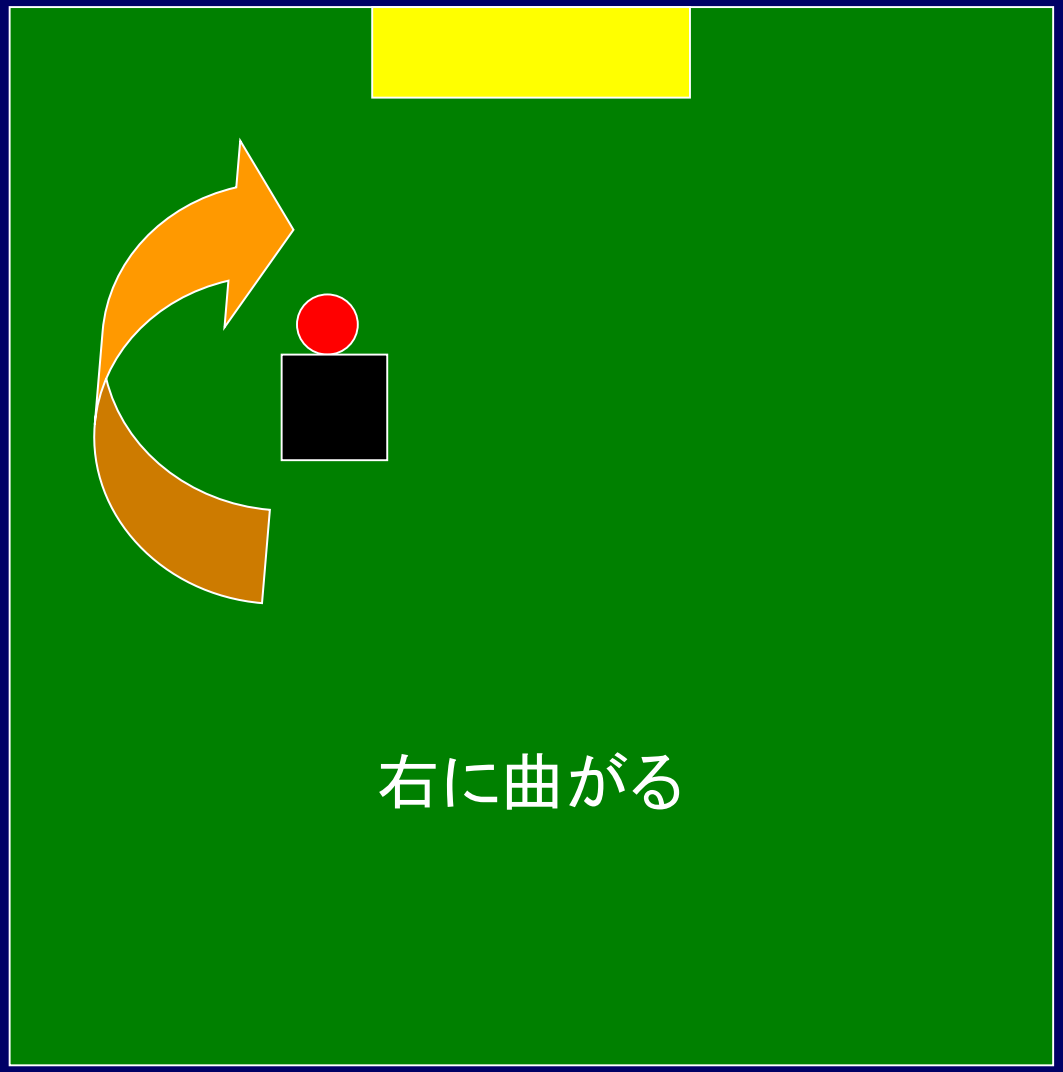


# ボールをゴールへ運ぶには2

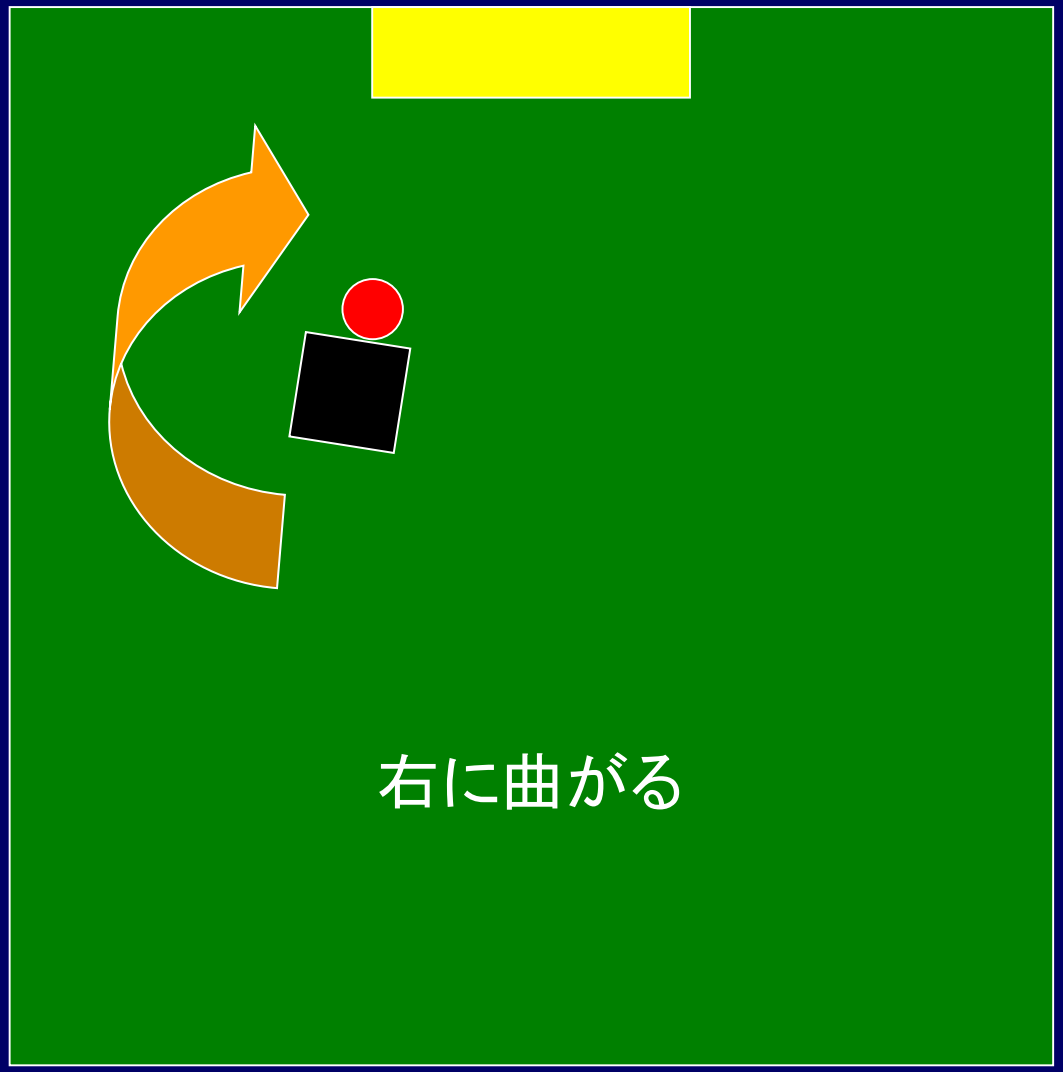


ロボットはドリブルしている
ロボットはどの方向に移動すれば良いか？

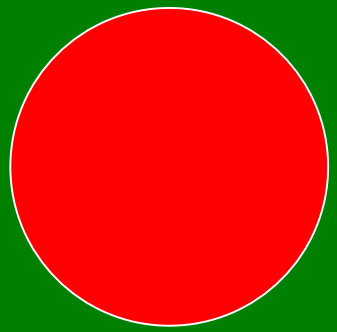
# ボールをゴールへ運ぶには2



# ボールをゴールへ運ぶには2



# ボールをゴールへ運ぶには3



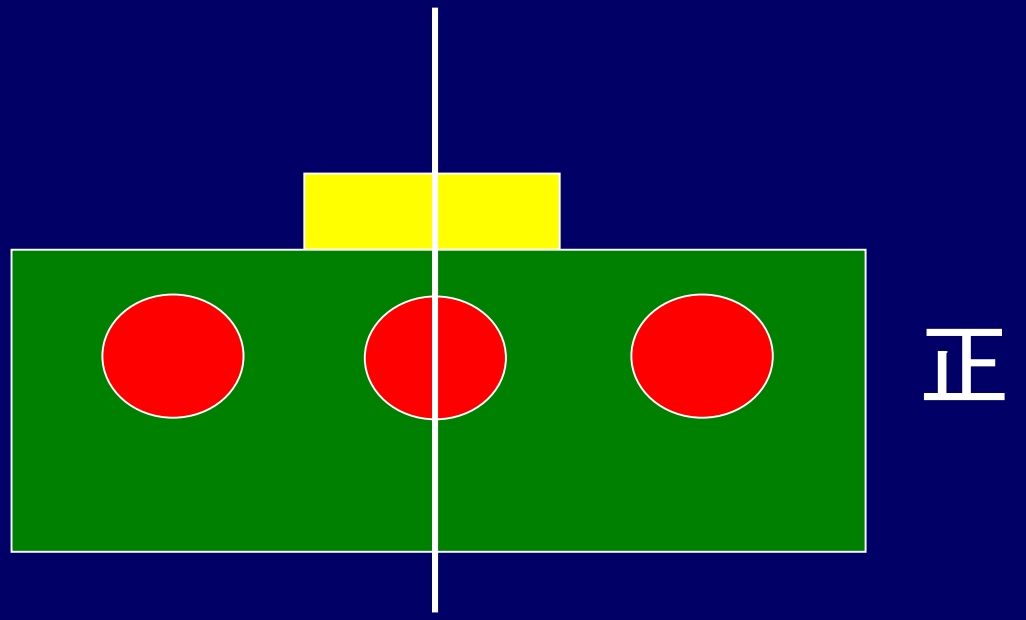
ロボットはドリブルしています.
ロボットはどちらに移動すれば良いか？

# ボールをゴールへ運ぶには3

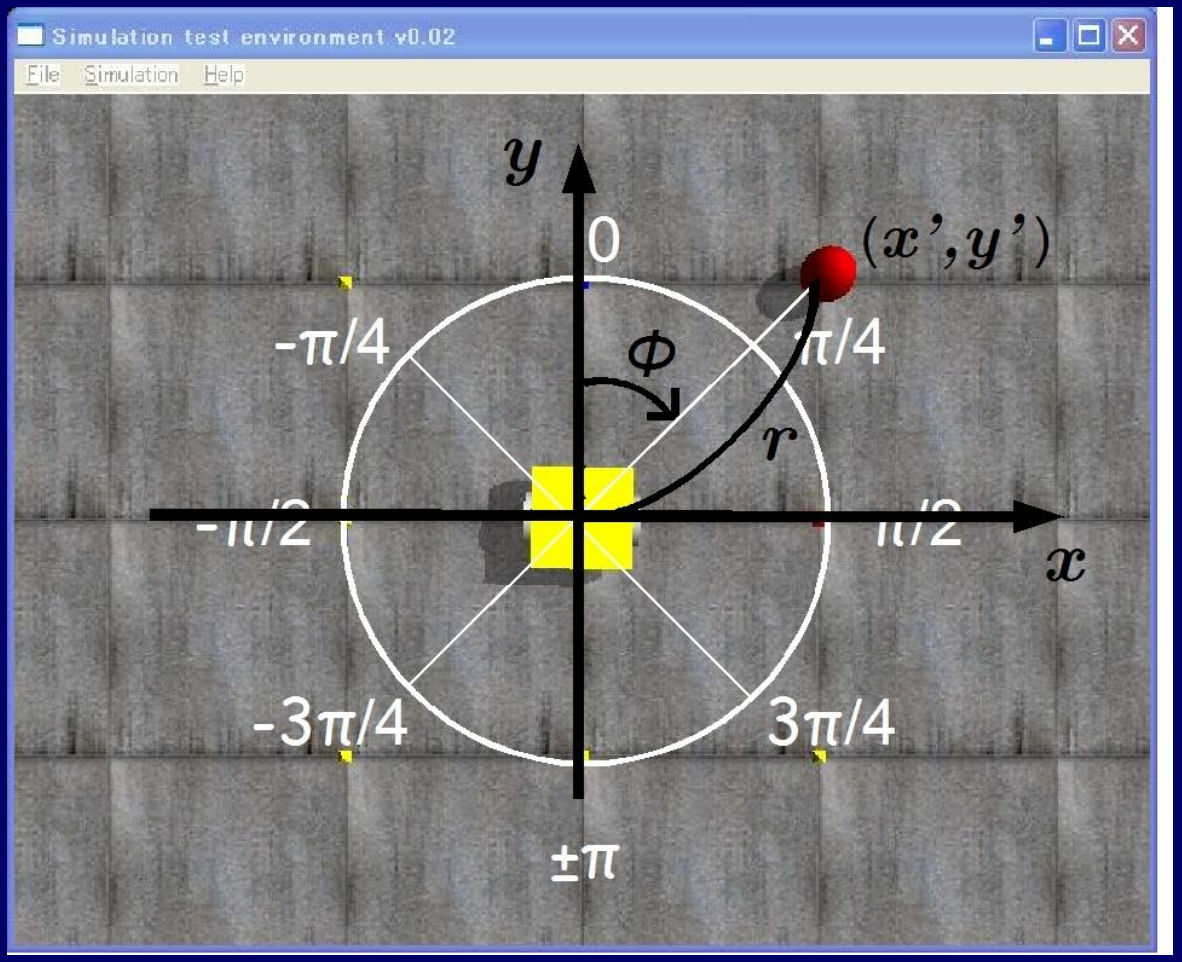
# ボールをゴールへ運ぶには3

# P（比例）制御

- 目標値　－　現在値
- ゴールの重心　－　ボールの重心
  - 正　→　右回転
  - ０　→　直進
  - 負　→　左回転



# ロボットの方位



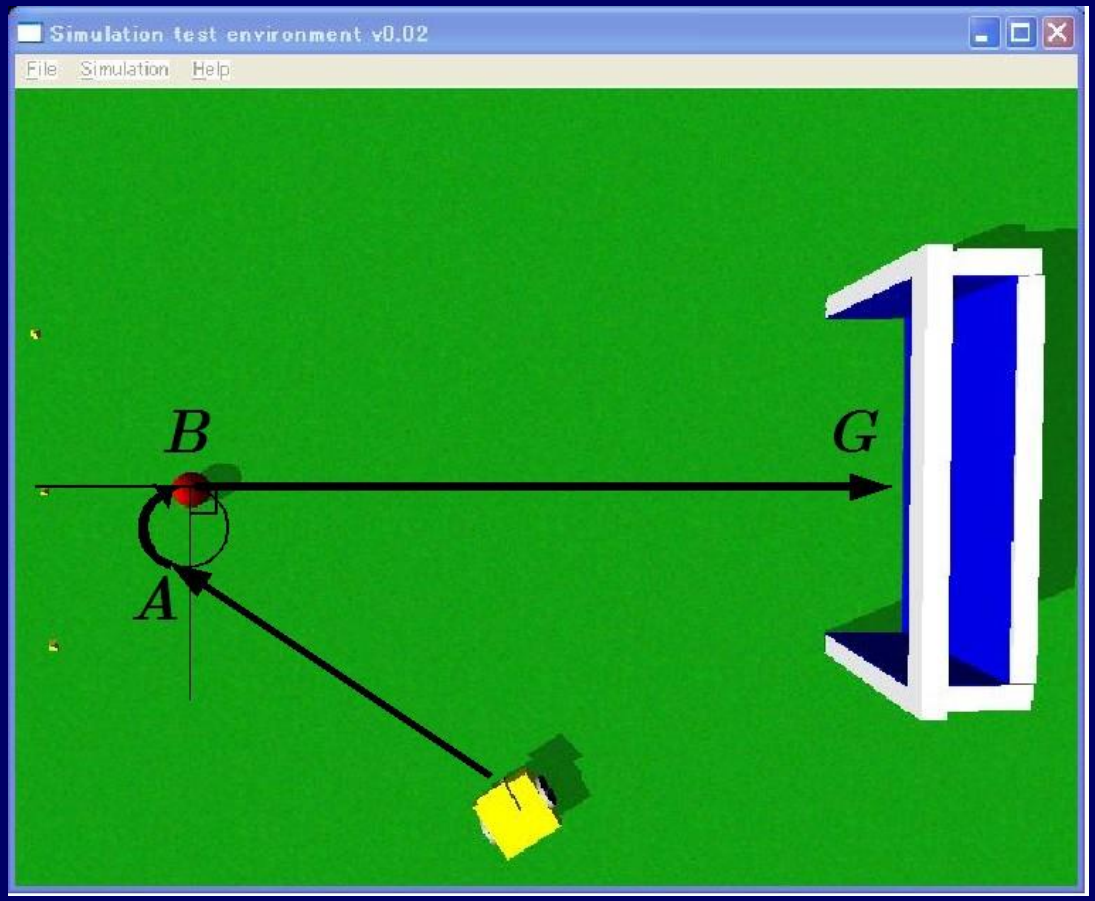

atan2関数（教科書P117から転載）

# 回り込み

- シュートするにはどうするロボット？


教科書P129から転載

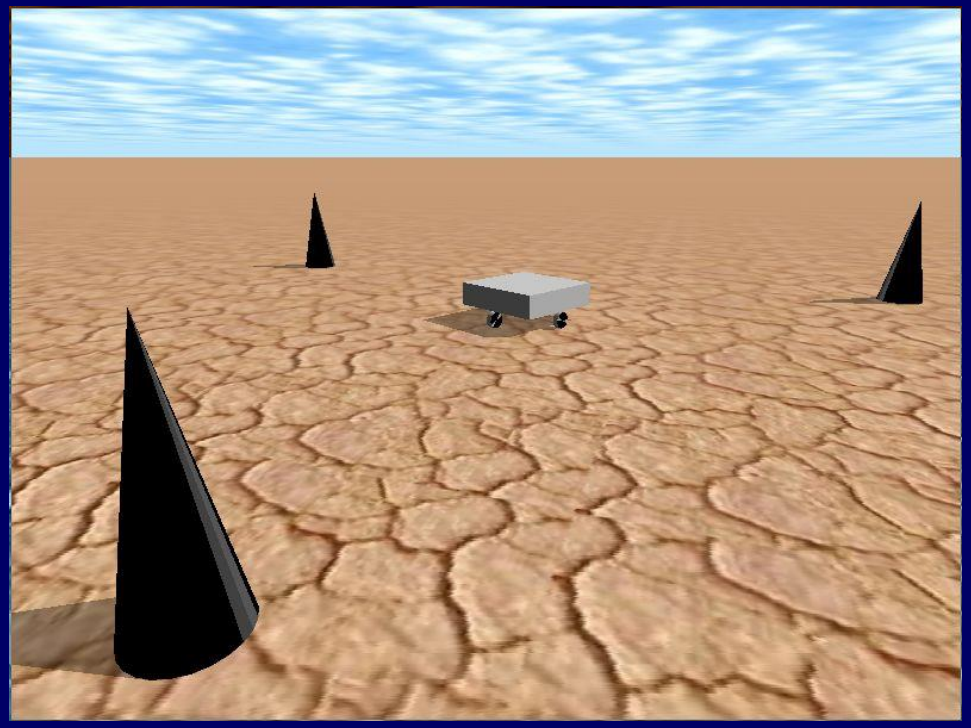
# 回り込み2




A地点でゴールが何度に見えるかを計算

(教科書P130から転載)

# 4.3 自己位置の推定



ランドマークによる自己位置同定

教科書P109から転載

# 用　語

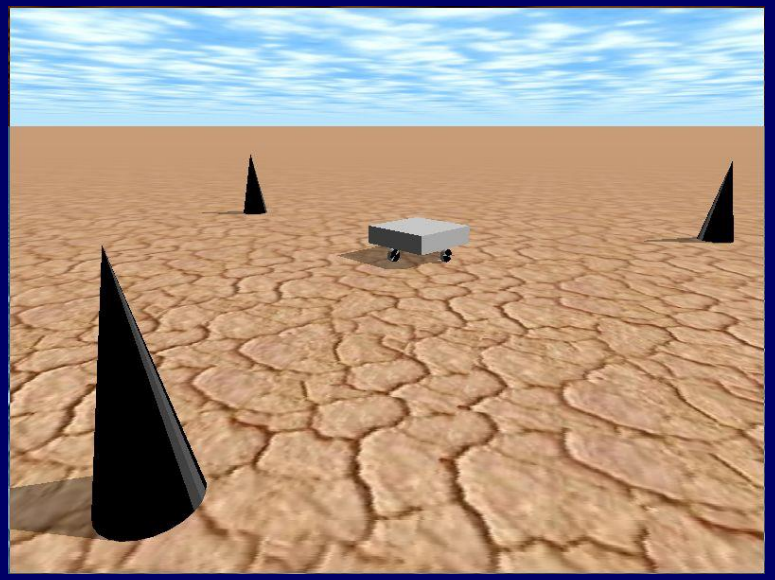
- 自己位置同定
  - 自分の位置を求めること
- ランドマーク （陸標）
  - 自己位置を求めるために，手がかりとなる地上の目印

# 自己位置同定の種類

- ランドマークに基づく方法　(Landmark based method）
  - ある特定のランドマークを用いる
    - 三角測量の原理
    - GPS (Global Positioning System)
  - プログラム簡単，計算量は少ないが，ロバストではない
- 見え方に基づく方法　(View based method ）
  - シーン全体の情報を用いる
    - テンプレートマッチング
    - スキャンマッチング
  - プログラム複雑，計算量は多いが，ロバストである
    - ライブラリの進歩　OpenCV
    - 計算機の進歩

# 自分の位置を知るには？

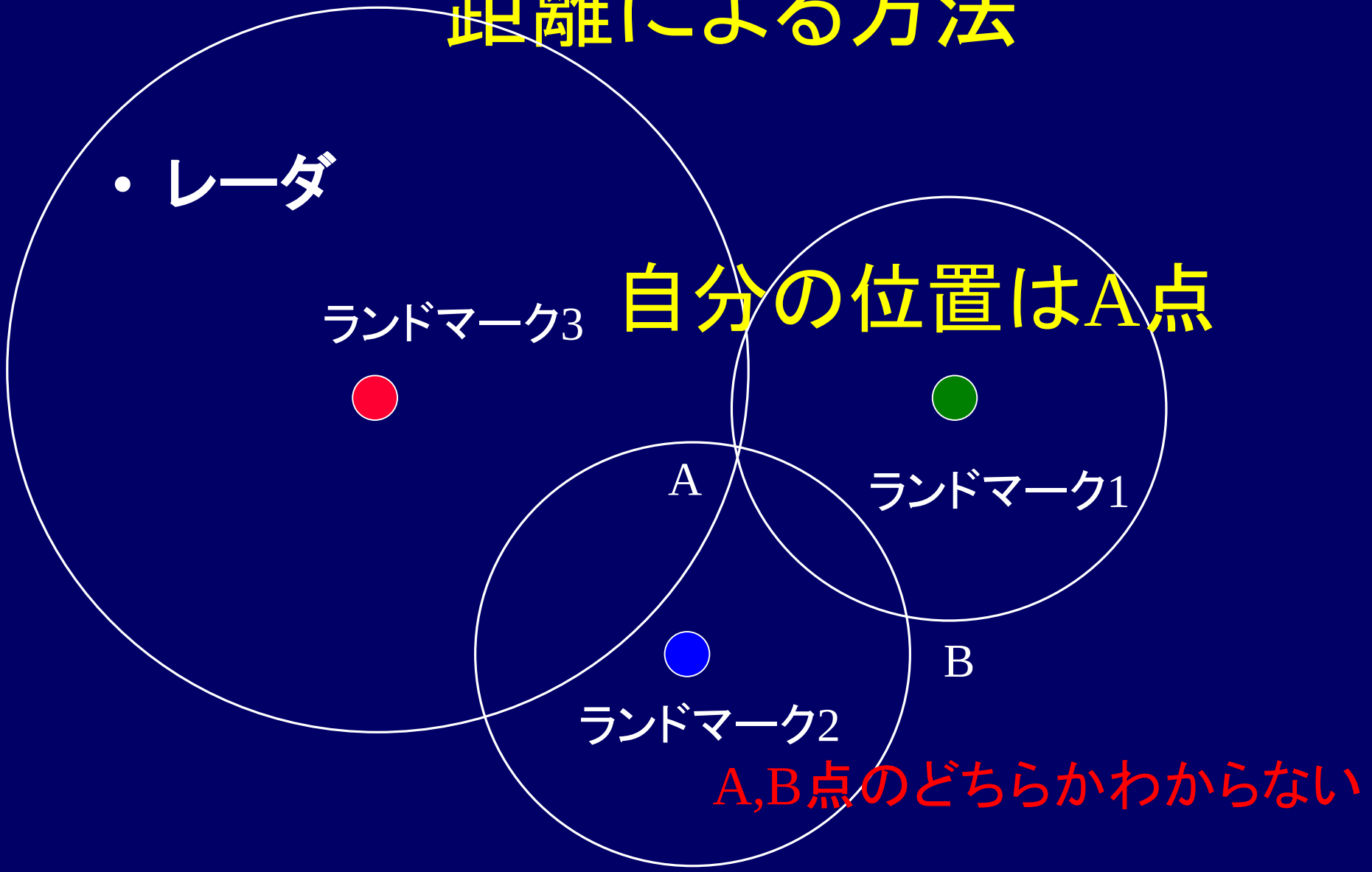
- 外界センサを使う方法
  - 距離による方法
    - レーダ，レーザスキャナ
  - 方位による方法
    - カメラ
- 内界センサを使う方法
  - 車輪の回転数
    - エンコーダ



ランドマークによる自己位置同定

教科書P109から転載

# 距離による方法

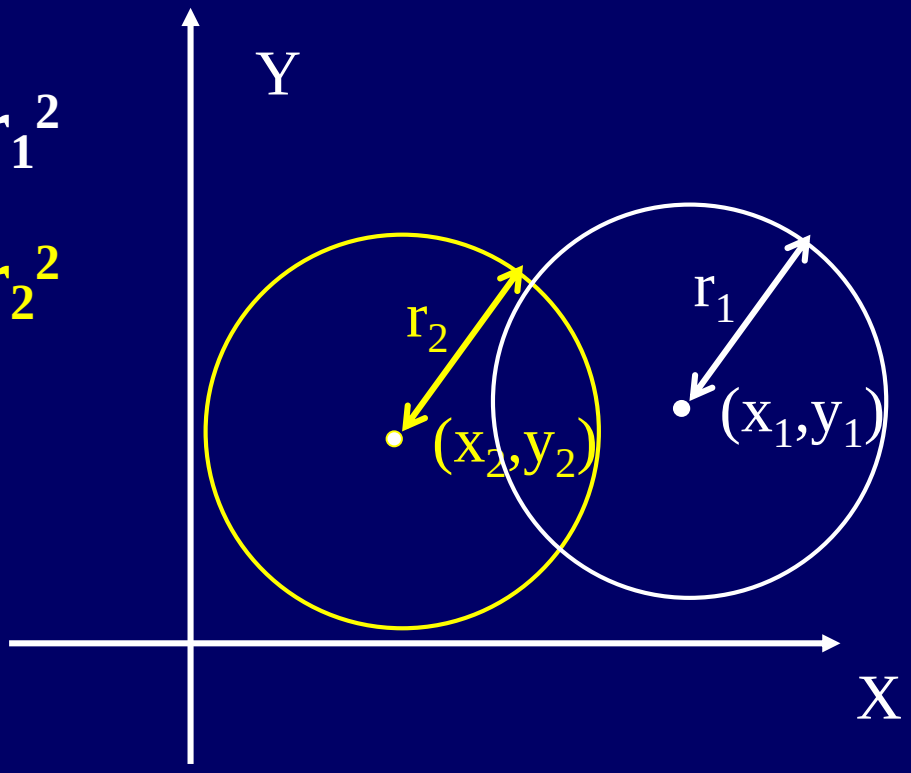


# 円の交点を計算する方法1

- 円の方程式
- 2つの円の交点A, Bを求めよう

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2$$



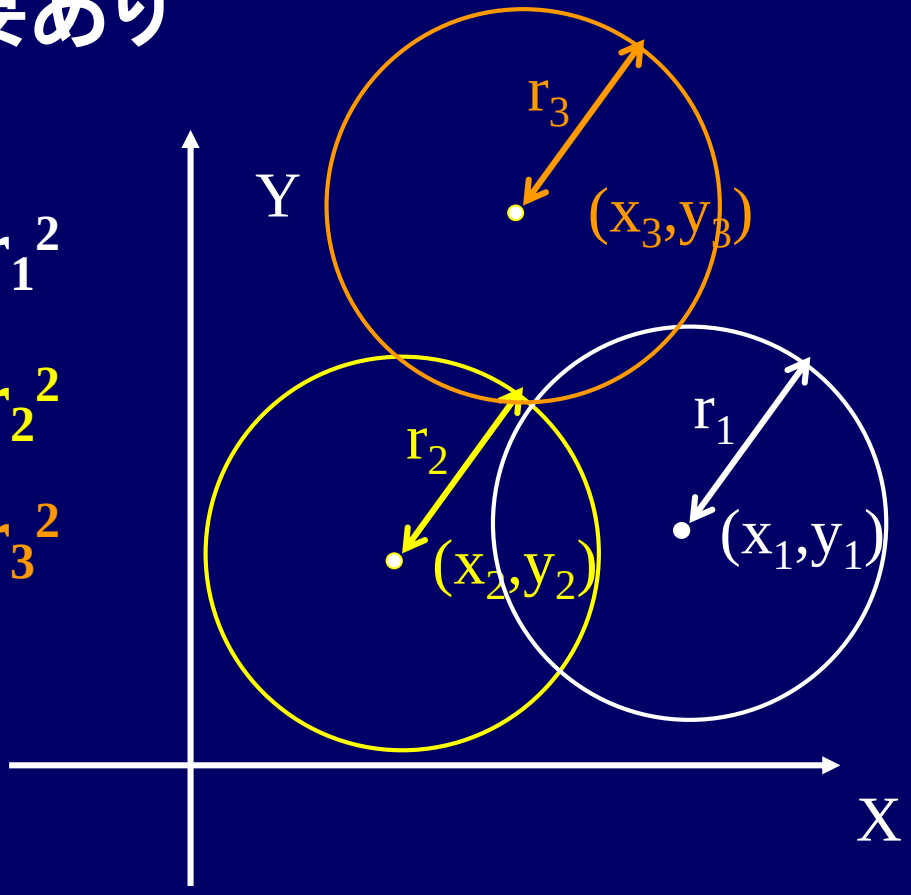
# 円の交点を計算する方法2

- 自分の位置を計算するためには3つの円の交点を求める必要あり
- 計算が複雑

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2$$

$$(x - x_3)^2 + (y - y_3)^2 = r_3^2$$



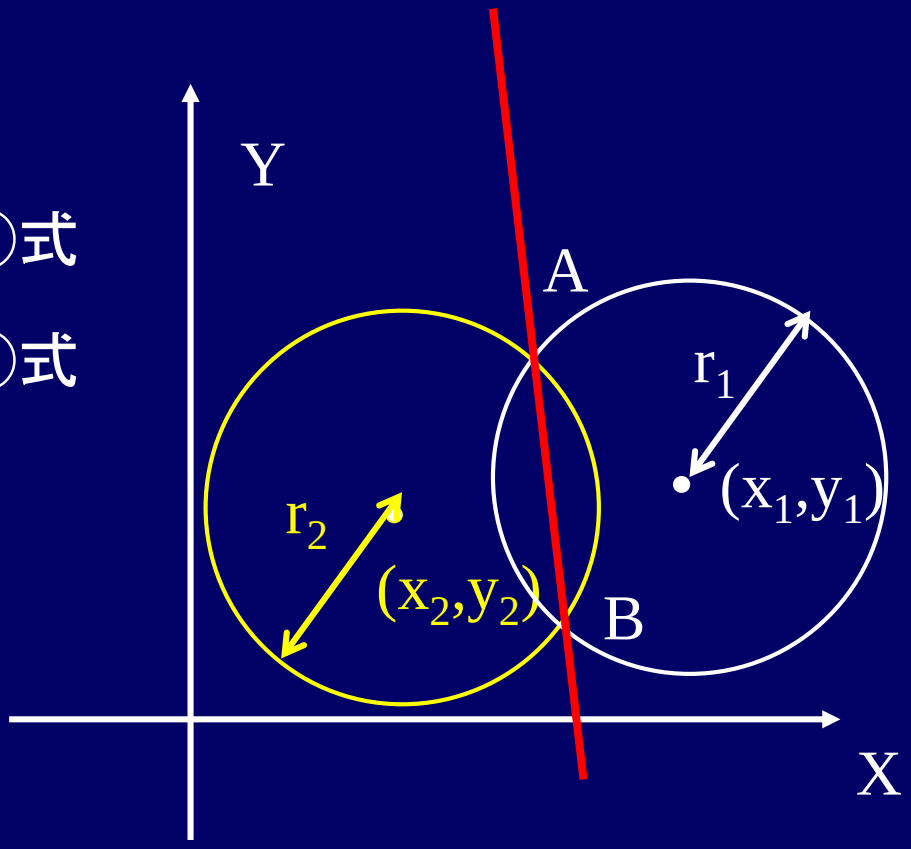
# 交点の直線を使用する方法1

- 円の方程式
- 直線A Bの式は？

$(x - x_1)^2 + (y - y_1)^2 = r_1^2$ ①式

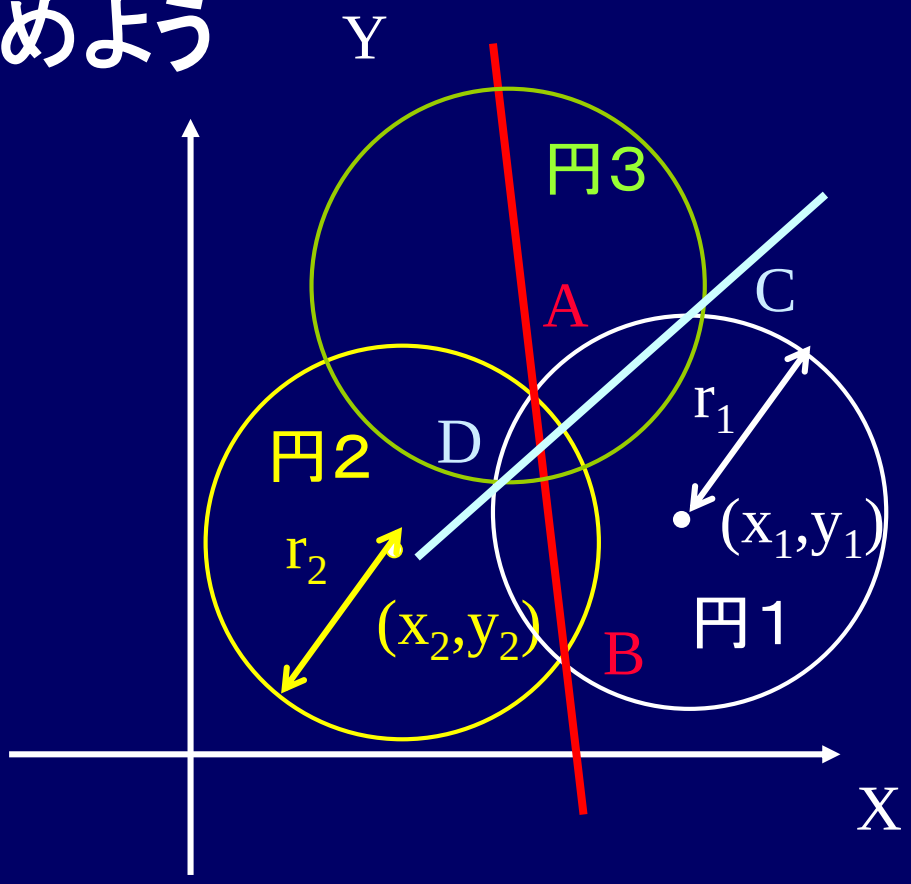
$(x - x_2)^2 + (y - y_2)^2 = r_2^2$ ②式

①式　－　②式



# 交点の直線を使用する方法2

- 円の方程式
- 2つ直線の交点を求めよう



# 方位による方法

- ロボットの位置　(x, y)
- 姿勢角　φ
- ランドマークの位置
  $(x_i, y_i)$
- 姿勢角から測った
  ランドマークの方位　$\beta_i$

$\beta_1 = \mathrm{atan2}(x_1\text{-}x, y_1\text{-}y) - \varphi$

$\beta_2 = \mathrm{atan2}(x_2\text{-}x, y_2\text{-}y) - \varphi$

$\beta_3 = \mathrm{atan2}(x_3\text{-}x, y_3\text{-}y) - \varphi$

$\beta_1 = \mathrm{atan}((x_1\text{-}x)/ (y_1\text{-}y)) - \varphi$

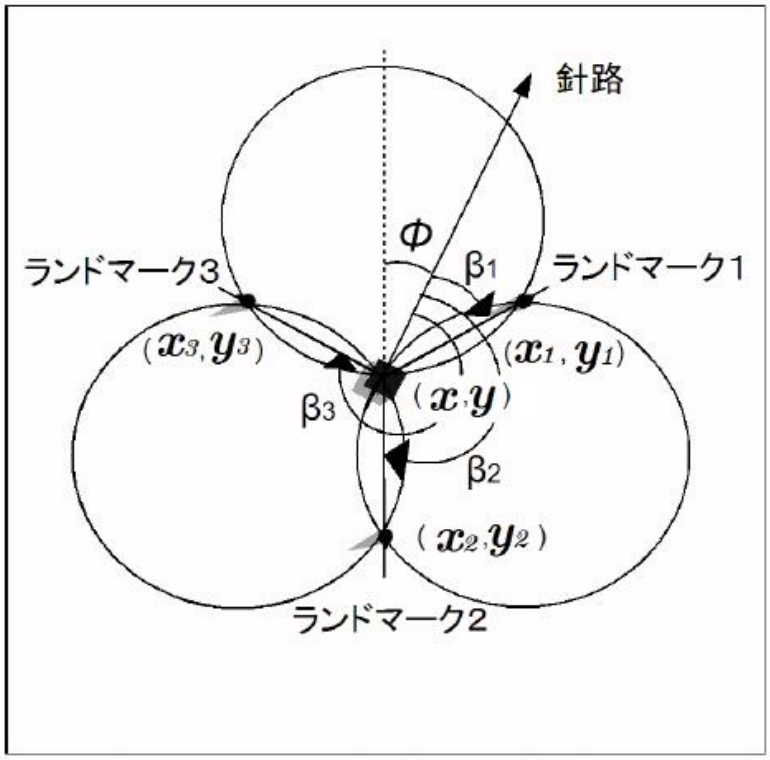



図 4.9　角度を使った自己位置推定 (文献 [26] を改変)

教科書P123から転載

# 方位による方法

$\beta_1 = \mathrm{atan2}(x_1\text{-}x, y_1\text{-}y) - \varphi$ ①

$\beta_2 = \mathrm{atan2}(x_2\text{-}x, y_2\text{-}y) - \varphi$ ②

$\beta_3 = \mathrm{atan2}(x_3\text{-}x, y_3\text{-}y) - \varphi$ ③

- ①と②式から姿勢角φを消去する
- 整理すると円の方程式になる
  - ２つのランドマークの座標
    $(x_1, y_1)$ $(x_2, y_2)$
  - 観測方位　$\beta_1, \beta_2$
- 現在位置(x, y)は右の円周上にある

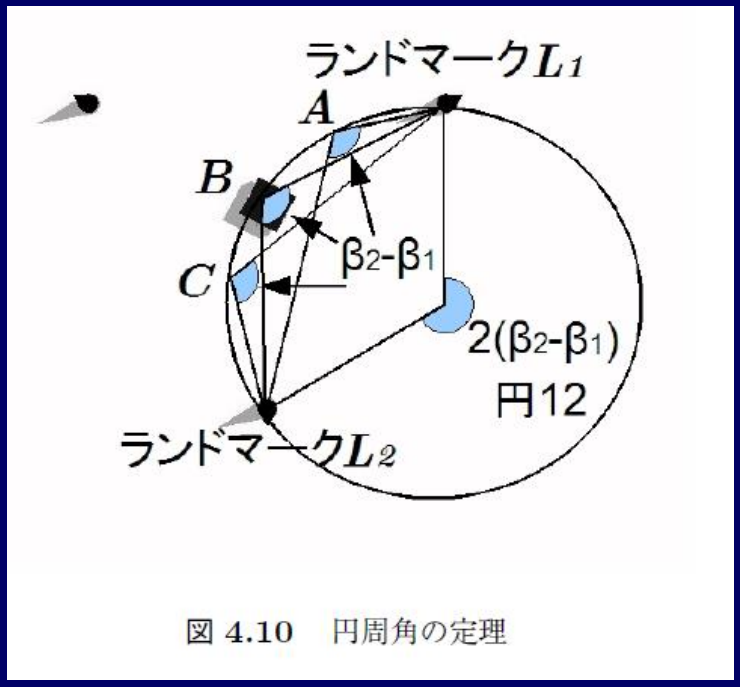


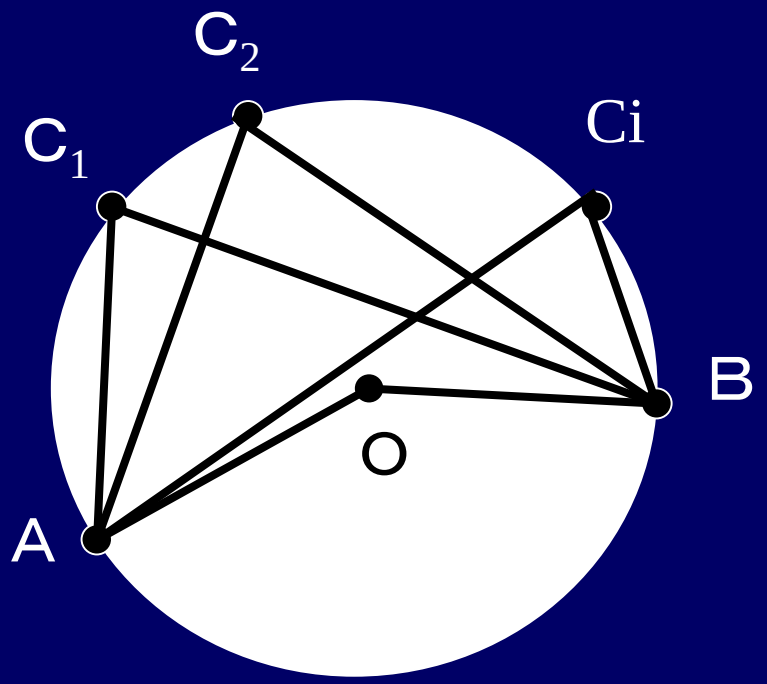

図 4.10　円周角の定理

教科書P124から転載

# 円周角（えんしゅうかく）の定理

円周上の２点から、円弧AB上の任意の点$C_i$に引いた直線のなす角（∠$AC_iB$)は等しく、円弧ABに対する円周角とよぶ。

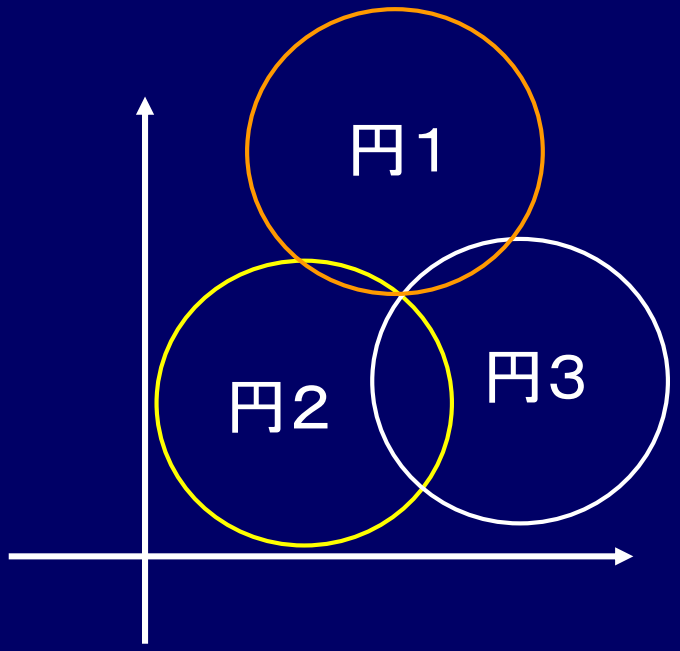
また、円の中心をOとした場合、∠AOB の大きさは円弧ABの円周角∠$AC_iB$の２倍である。



# ランドマークの必要数

- １点：自己位置は円周上
- ２点：自己位置は2点のどちらか
- ３点：自己位置は一意に決定

ランドマークが３点
以上必要となる

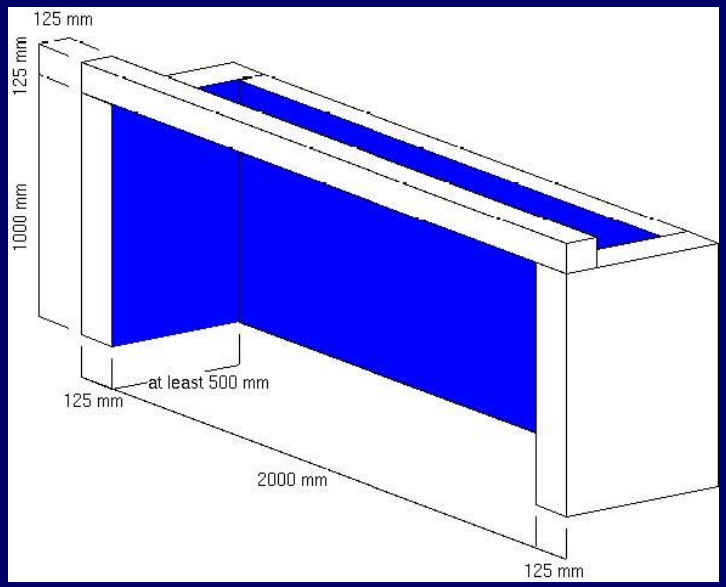


# 宿　題

- 距離による自己位置を2つの方法で求める
  - 3つの円の交点を求める方法
  - よりスマートな方法
    - 2つの円の交点を通る直線を使う方法
- 提出
  - ○月○(○)　授業時
- 方法
  - A4レポート用紙．手書き，鉛筆書きOK

# 演習

- ボールを探索してゴールへ運ぶプログラムを作成
- ステップ1： ボールへ向かうプログラム
- ステップ2： ボールをゴールへ運ぶプログラム




横　　高さ　奥行き
2.0m × 1.0m × 0.5m

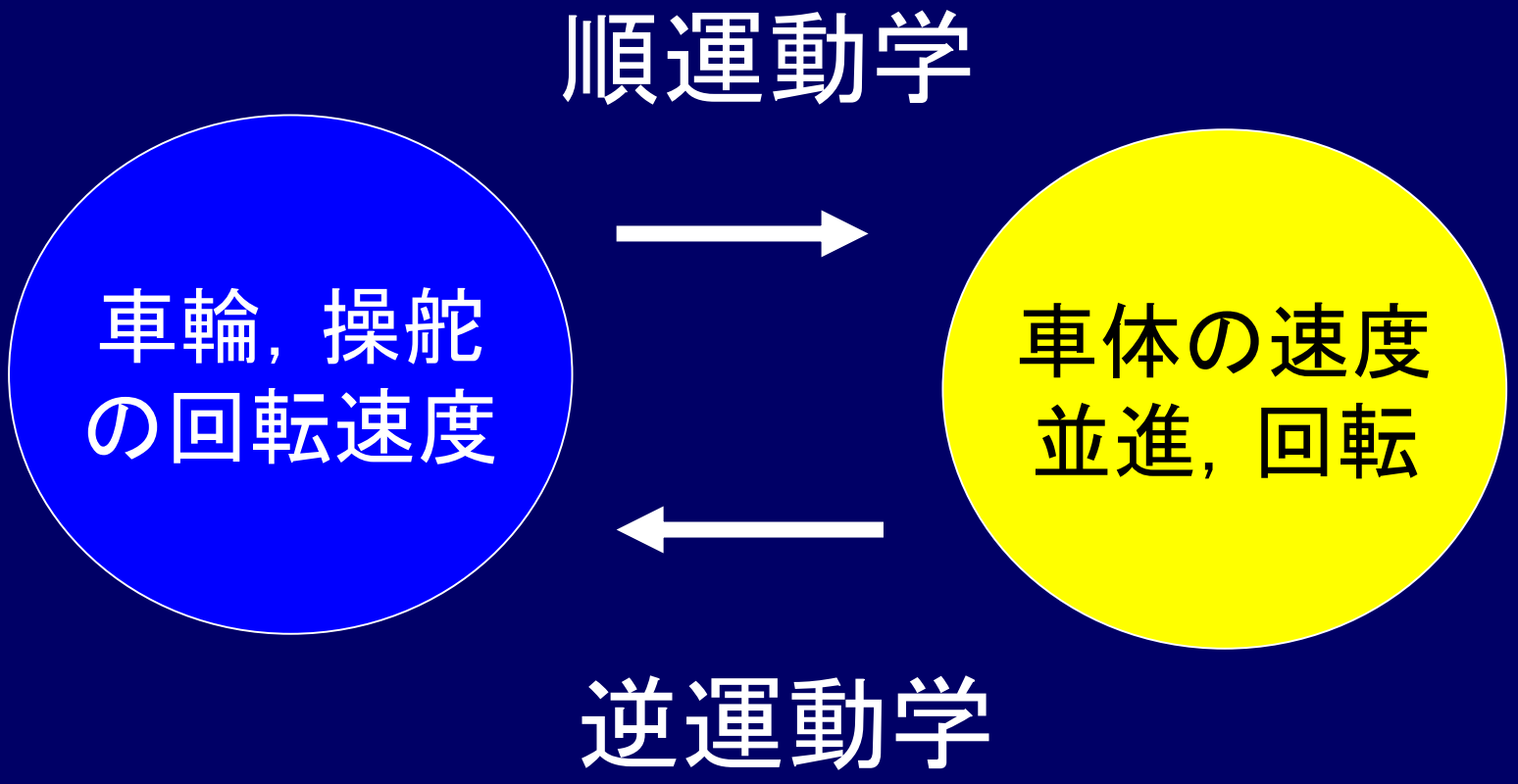
# デッドレコニング（dead reckoning）

- **Reckoning: the act of calculating sth, especially in a way that is not very exact (OXFORT 現代英英辞典から引用）**
- **Star reckoning（天測）**
- **Dead reckoning**

# デッドレコニング（dead reckoning）

- 内界センサにより自己位置を推定する
  - エンコーダ
  - ジャイロ
- 特徴
  - 時間と伴に誤差が蓄積され雪だるま式に増加
  - 大域的位置が分からない
  - 誤差の分散は小さい
  - 外界センサによる自己位置と組み合わせる
    - カルマンフィルタ
    - パーティクルフィルタ

# 車輪型ロボットの運動学



# 運動学

## ロボット重心の位置(x, y)，姿勢θを求めよう

$$\dot{x} = 0.5\, r\, (\omega_r + \omega_l) \sin\theta$$

$$\dot{y} = 0.5\, r\, (\omega_r + \omega_l) \cos\theta$$

$$\dot{\theta} = 0.5\, r\, (\omega_r - \omega_l) / d$$

0.5はロボット重心の速度だから

ここで,
r: タイヤの半径
ωr, ωl: タイヤの角速度　右，左
d: 中心からタイヤの間隔

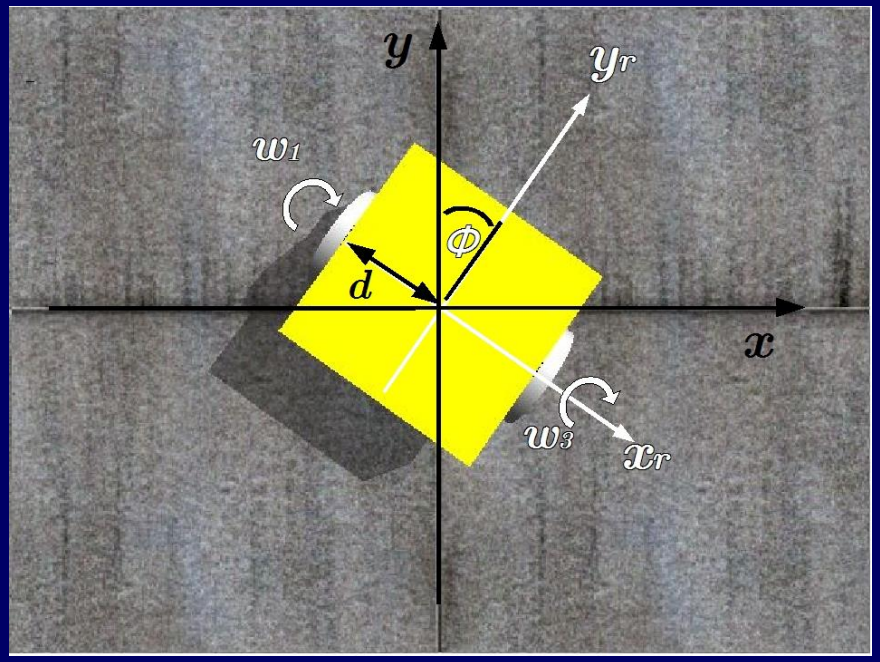
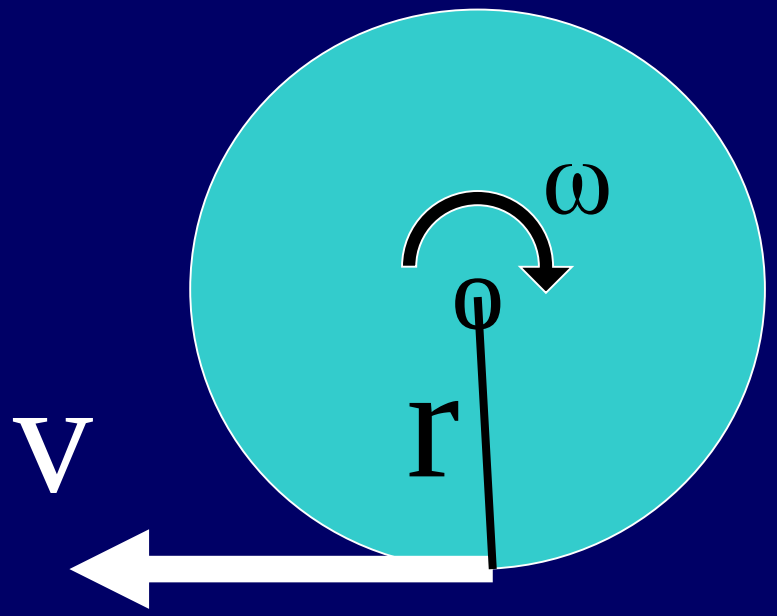



図4.11 差動駆動型ロボットの運動学
教科書P126から転載

# 速度と角速度の関係




半径r　角速度ω　のとき速度vは？

# 移動距離を求める方法

**前式を時間で積分すればよい**

$$x = x0 + 0.5\int_0^t r\,(\omega_r + \omega_l)\sin\theta\,dt$$

$$y = y0 + 0.5\int_0^t r\,(\omega_r + \omega_l)\cos\theta\,dt$$

$$\theta = \theta_0 + 0.5\,r/d\int_0^t (\omega_r - \omega_l)\,dt$$

ここで,
r: タイヤの半径
ωr, ωl: タイヤの角速度　右，左
d:中心からタイヤの間隔

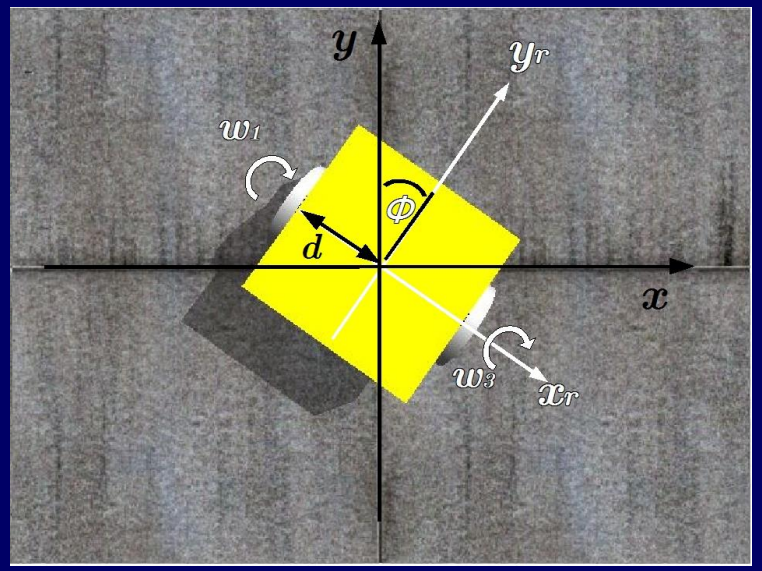
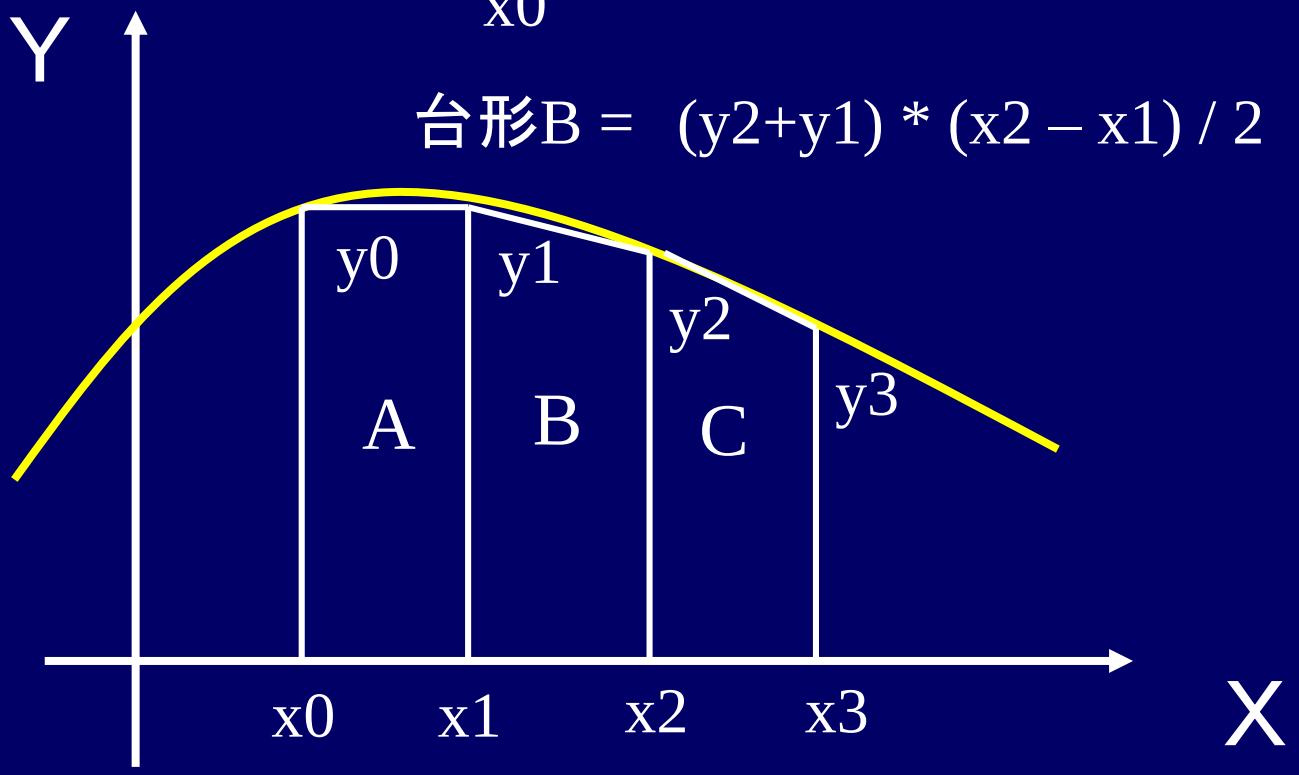



図4.11 差動駆動型ロボットの運動学
教科書P126から転載

# 数値積分　P127

- 台形公式

$$\int_{x0}^{x3} f(x)\,dx = \text{台形A} + \text{B} + \text{C}$$



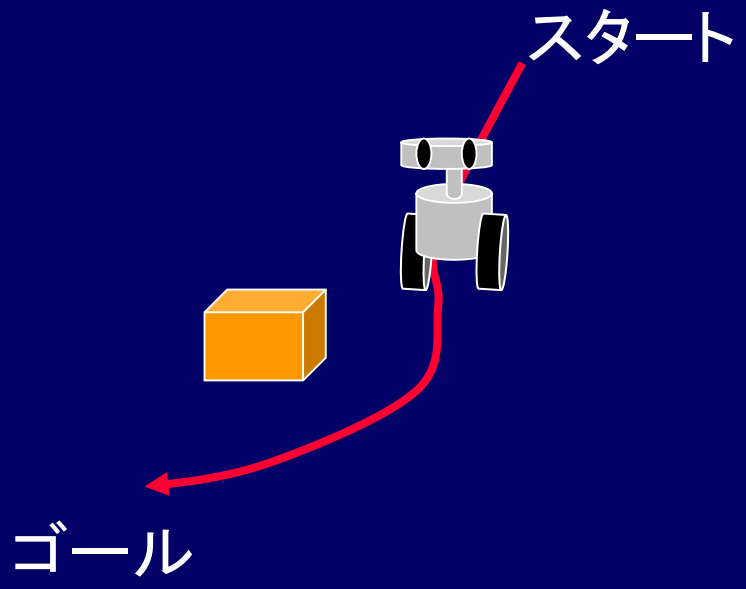

今日の格言：積分は微小区間の足し算である.

# プチプロジェクト

- 全く同じプロジェクトは不可
  - 決まったら私にメール連絡
  - 方法が違えばよい
- 車輪型ロボットなら
  - 自己位置の推定の実装
    - 距離による方法
    - 角度のよる方法
    - その他
  - まわりこみの実装
  - デッドレコニングの実装
  - 逆運動学の実装
  - などなど

# ポテンシャル法

- 障害物回避だけでなくナビゲーションにも利用可能
- 物体や場所にポテンシャルエネルギー場を設定
  - 磁石をイメージ
- 障害物回避
  - 障害物に斥力を設定
- ナビゲーション
  - ゴールに引力
  - 障害物に斥力



# ポテンシャル関数法の特徴

- 利　点
  - １つの関数で追跡と障害物回避が可能
    - ディフェンス
      - キーパーのカバー
    - マンツーマンディフェン
      - 衝突せず，離れず
  - 実装が簡単
- 欠　点
  - 極小値問題
    - 引力と斥力が釣り合う場合
    - ロボット停止
    - 解決法としては慣性の導入
  - パラメータを調整するのが難しい

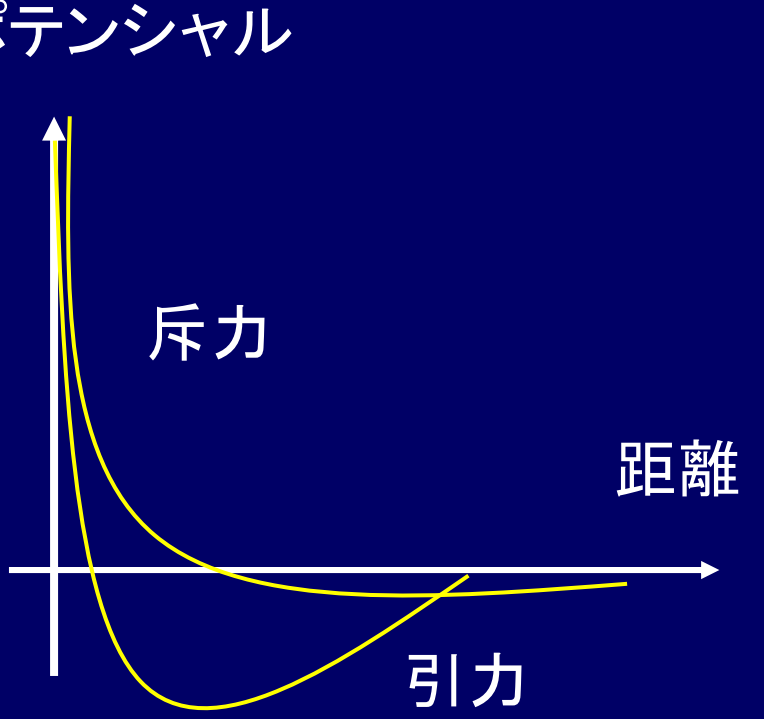
# ポテンシャル関数って何ですか？

- レナードジョーンズポテンシャル関数
  - 分子間の引力と斥力のポテンシャルエネルギーを表す

$$U = - A / r^n + B / r^m$$

引力成分　　斥力成分

U: 分子間のポテンシャルエネルギー
A, B, n, m: パラメータ
r: 分子間距離



# レナードジョーンズポテンシャル関数

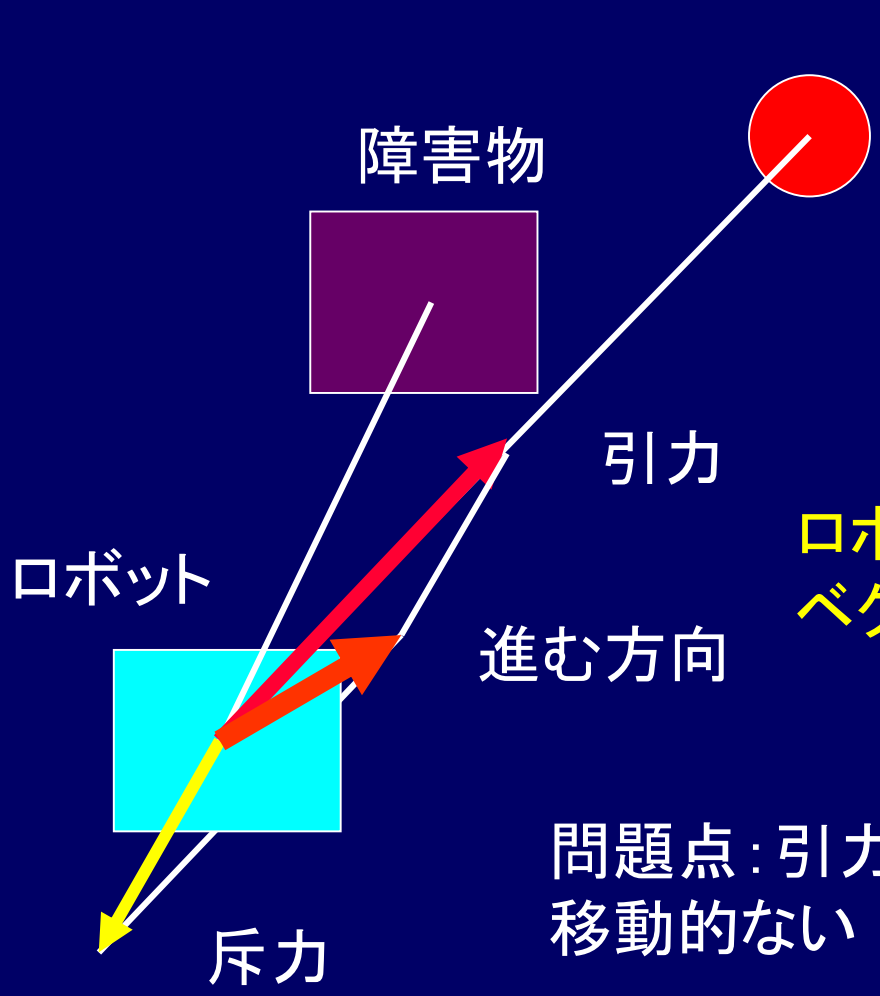
$$U = - A / r^n + B / r^m$$

$$F = - dU/dr = - nA / r^{n+1} + mB / r^{m+1}$$

引力成分　　斥力成分

F: 力 （マイナスは引力, プラスは斥力）
A: 引力の強さ
B: 斥力の強さ
n, m: 曲線の傾き
r: 分子間距離

# 障害物とゴール

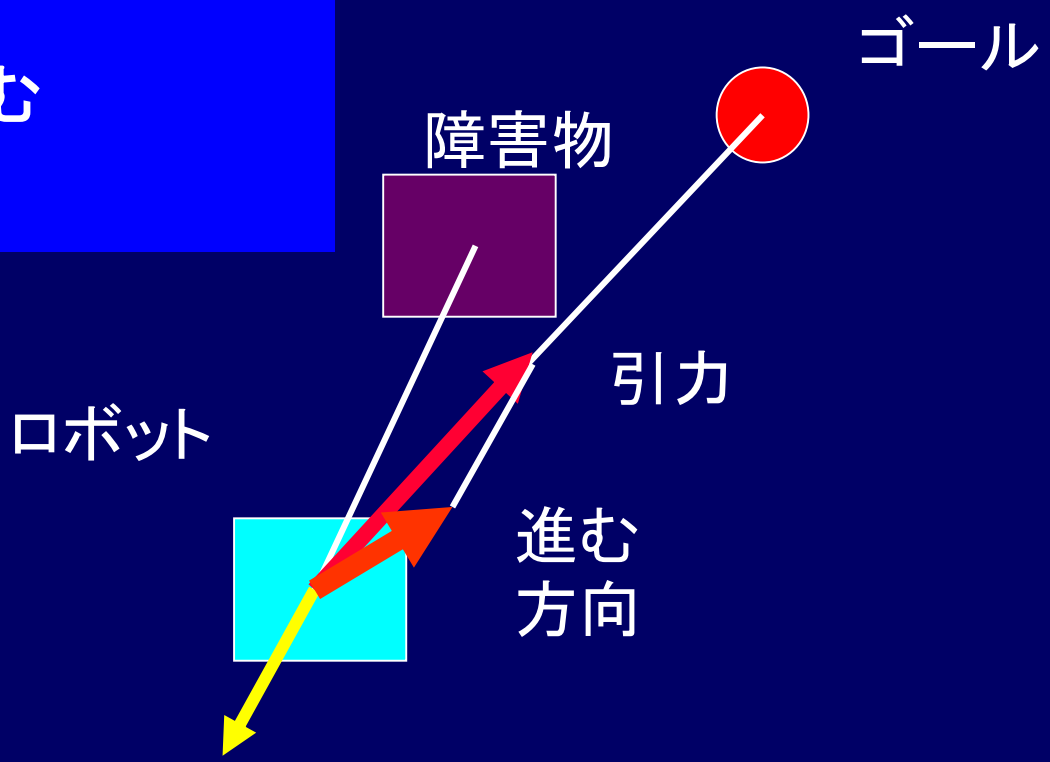



ロボットは引力と斥力の合成ベクトルの方向へ進む

問題点：引力と斥力が釣り合う時は移動的ない

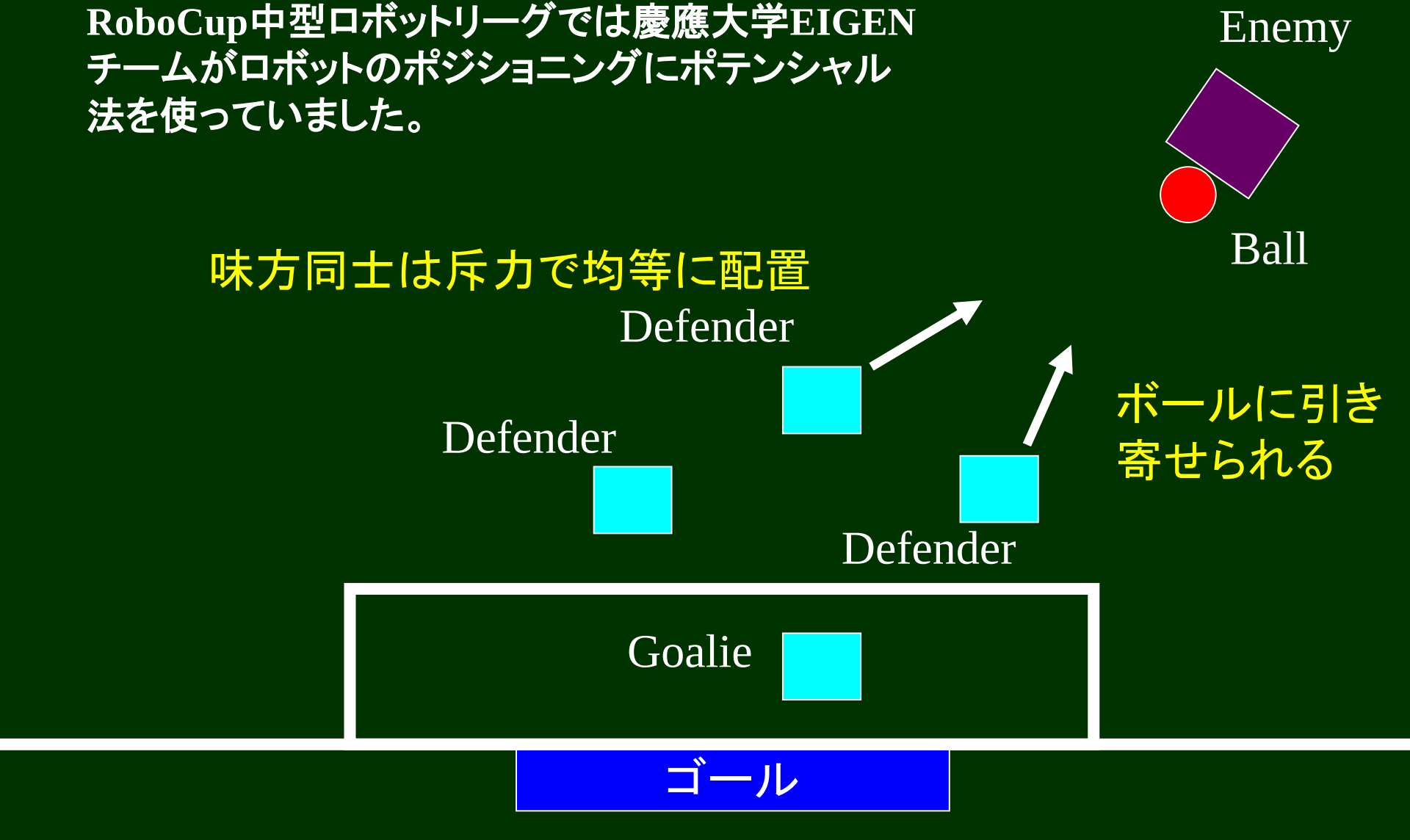
# プログラムのヒント

1. 相対座標系で引力と斥力の合成ベクトルを求める
   - ベクトルの足し算
2. その進路に向かって進む
3. ステップ1に戻る

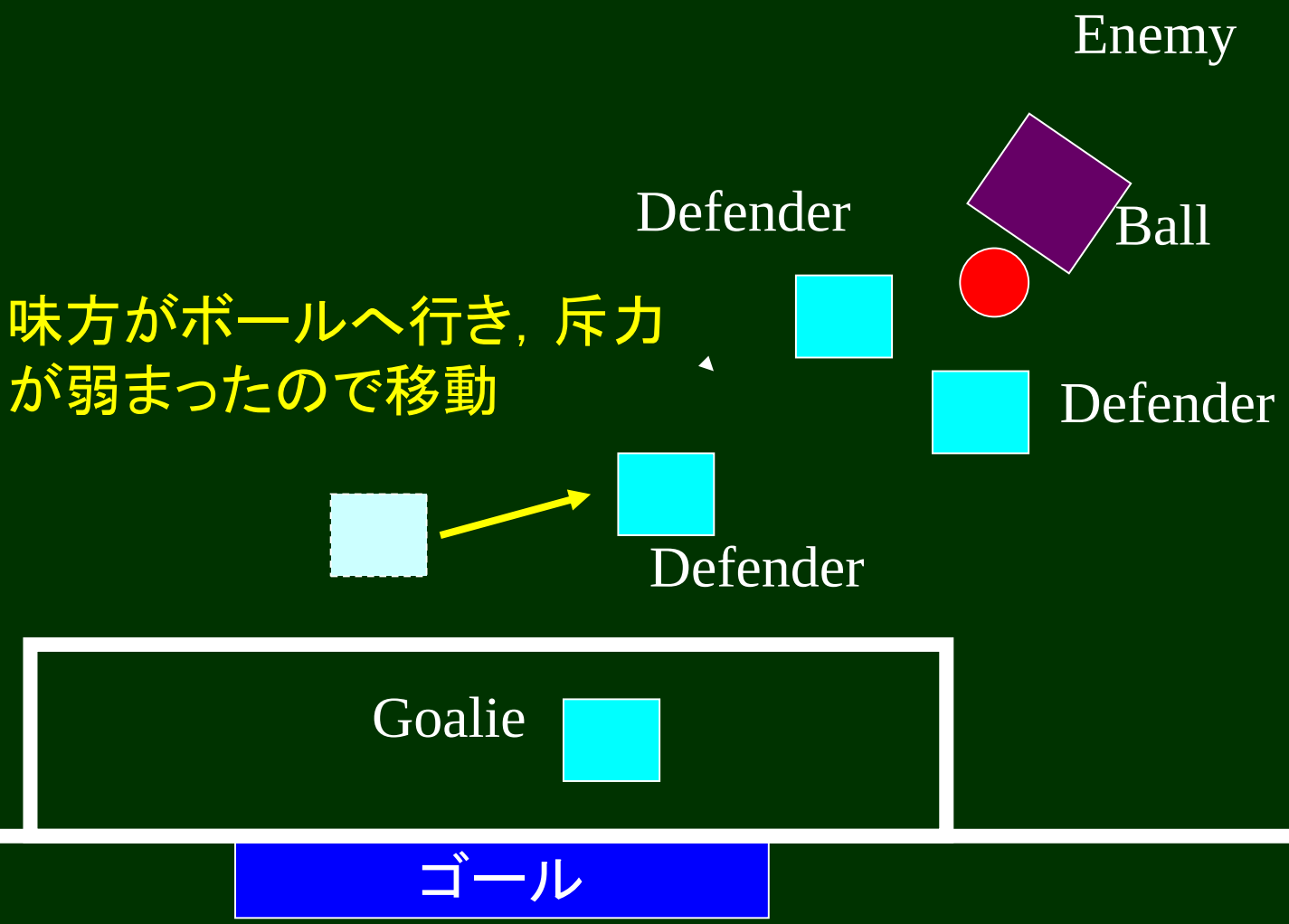


# 応用例　サッカーロボット

**RoboCup中型ロボットリーグでは慶應大学EIGENチームがロボットのポジショニングにポテンシャル法を使っていました。**



# 応用例　サッカーロボット



# 演　習

次のうち，好きなものを1つおやりなさい．

- 提出期日：〇月〇日（〇）
- 1．自己位置同定法の実装
  - P112のプログラム4.1のプログラムにランドマークを3点設定し，角度による方法で自己位置を算出するプログラムを作りなさい．
- 2．ポテンシャル法の実装
- 3．衝突航法の実装
- 4．その他（要相談）

おしまい。