UNISINOS

**PIP/CA** - ***Programa Interdisciplinar de Pós-Graduação***
***Mestrado em Computação Aplicada da UNISINOS***

**2000/2 - 3o. Trimestre - AULA 03 / FSO**

# CONTROLE & ROBÓTICA INTELIGENTE


- **Professor Responsável:**

**Prof. Dr. Fernando Osório**

**E-Mail:** ***osorio@exatas.unisinos.br***

**Web:** ***http://www.inf.unisinos.br/~osorio/robi.html***



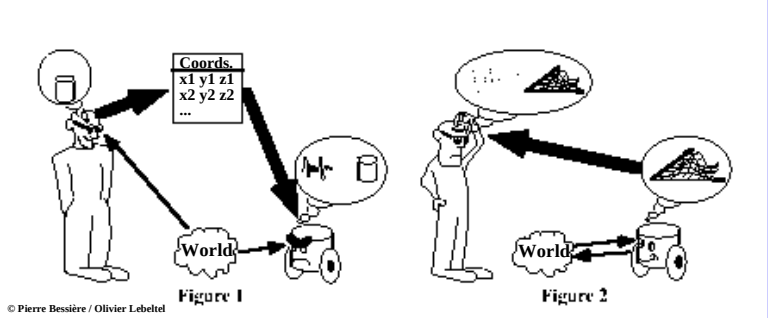

## TEMAS DE ESTUDO: *CONTROLE REATIVO*

***Controle Robótico:***

1. **Deliberativo: Planificação**
2. **Reativo: Sensorial-Motor**
   - **Tratamento da Incerteza e Imprecisão**
   - **Modificações no Ambiente**
   - **Interação com o ambiente: Sense-Act**

**Comportamentos Elementares:**
- **Evitar colisões**
- **Seguir uma parede**
- **Seguir a luz**
- **Empurrar objetos ...**

**Comportamentos Compostos:**
- **Evitar colisões e seguir a luz**
- **Evitar colisões e seguir bússola, ...**



**Controle Reativo:** ***Sensorial-Motor***

- **Informações *tipicamente* usadas no controle deliberativo:**
  **Alto nível = Geometria / Planificação**

- **Informações *tipicamente* usadas no controle reativo:**
  **Baixo nível = Sensorial / Adaptação - Interação**

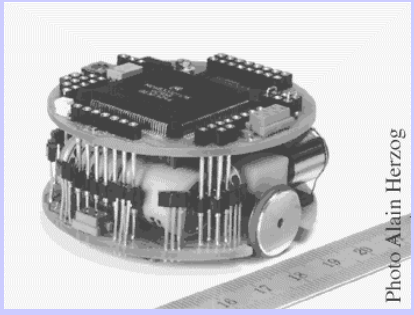

**Visão do Mundo**

**HUMANA** ← → **ROBÓTICA**

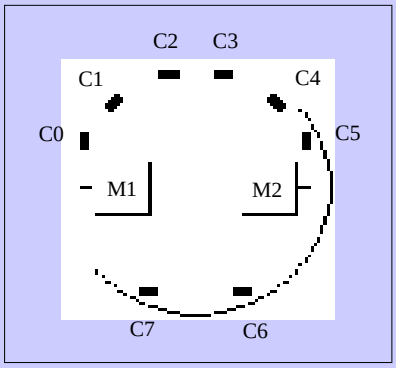


**Controle Reativo:** ***Sensorial-Motor***

- **Experiências de controle reativo -** ***Khepera***




**EPFL - Lausanne / Suiça**




**Robô móvel ⇒ Khepera**

**Controle sensorial-motor:**

**8 sensores / 2 atuadores independentes**

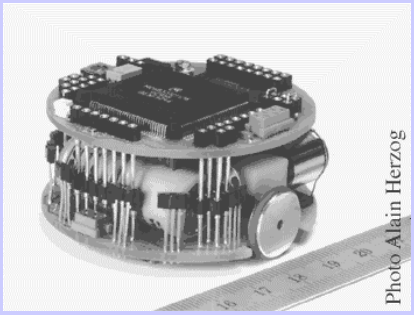
**Experiências típicas de estudo do comportamento reativo**

**- Evitar obstáculos**

**- Seguir muros**

**Controle Reativo:** ***Sensorial-Motor***

• **Experiências de controle reativo -** ***Khepera***




**Robô móvel ⇒ Khepera**

**8 sensores - S0 a S7**

- **Emissor e Sensor de Infra-Vermelho**
  **Escala [0 .. 1023] (+/- 5 a 50mm)**
- **Sensor Luz Visível**
  **Escala [0 .. 500] (+/- 50 a 500mm)**

**2 atuadores independentes**

**Motores de passo com velocidades independentes => -10 .. +10**
**Rotação: +5 (M1) -5 (M2) gira p/ direita**

**Controle:**
**- Via micro-controlador (autônomo)**
**- Via porta serial (semi-autônomo c/** ***cordão umbilical*****)**

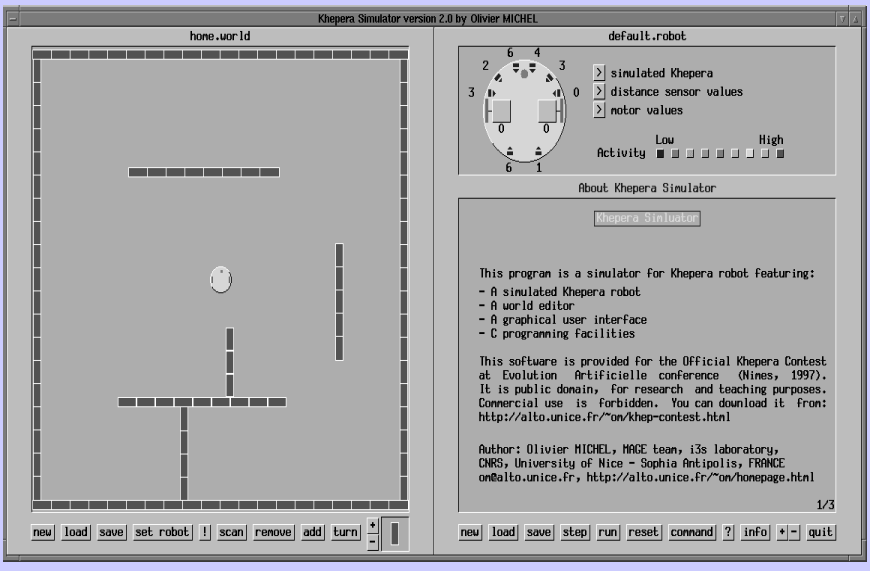
**Software:**
**- Simuladores e compilador (cross-compiler)**



**Controle Reativo:** ***Sensorial-Motor***

***Software Gratuito e Aberto***

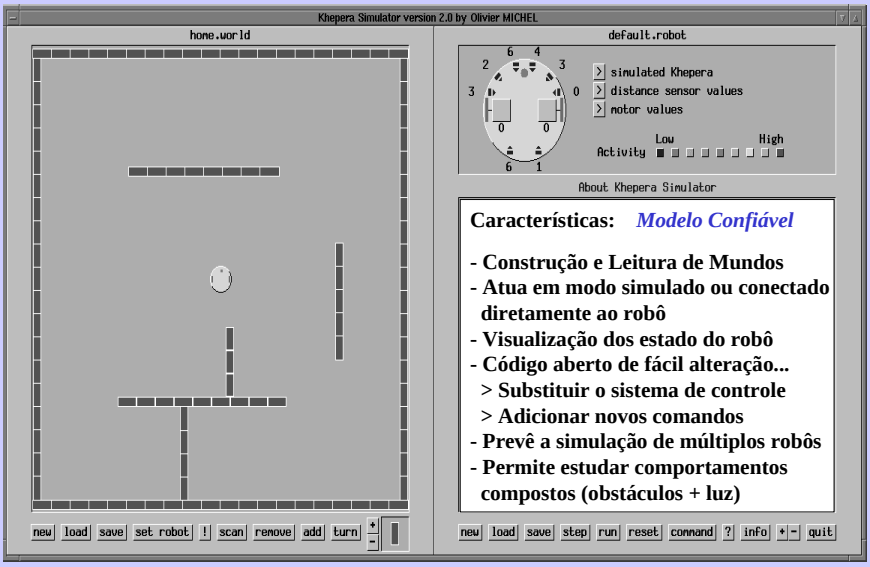
• **Simulador do** ***Khepera*** **/ SIM 2.0 Unix / Olivier Mitchell / INRIA Sophia Antipolis**



## Controle Reativo: *Sensorial-Motor*

***Software Gratuito e Aberto***

• **Simulador do *Khepera* / SIM 2.0 Unix / Olivier Mitchell / INRIA Sophia Antipolis**



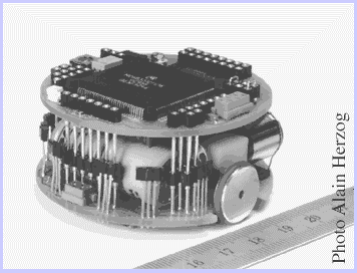

**Características:** ***Modelo Confiável***

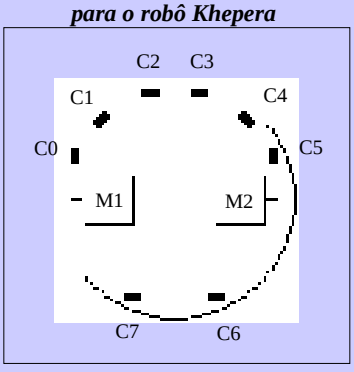
- **Construção e Leitura de Mundos**
- **Atua em modo simulado ou conectado diretamente ao robô**
- **Visualização dos estado do robô**
- **Código aberto de fácil alteração...**
  - **> Substituir o sistema de controle**
  - **> Adicionar novos comandos**
- **Prevê a simulação de múltiplos robôs**
- **Permite estudar comportamentos compostos (obstáculos + luz)**



## Controle Reativo: *Sensorial-Motor*

***Criando um Sistema de Controle para o robô Khepera***




**Sensores: 8 - S0 à S7**

**Comandos: 3 ações (L=Turn Left, F=Forward, R=Turn Right)**

| *S0* | *S1* | *S2* | *S3* | *S4* | *S5* | *S6* | *S7* | *L* | *F* | *R* |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 3 | 4 | 6 | 3 | 0 | 4 | 0 | 1 | 0 |
| 1 | 3 | 6 | 3 | 6 | 6 | 6 | 6 | 0 | 1 | 0 |
| 6 | 3 | 104 | 234 | 772 | 96 | 3 | 6 | 1 | 0 | 0 |
| 2 | 6 | 104 | 229 | 724 | 107 | 3 | 4 | 1 | 0 | 0 |
| 563 | 1023 | 57 | 6 | 6 | 0 | 3 | 1 | 0 | 0 | 1 |
| 544 | 1023 | 1 | 3 | 4 | 0 | 5 | 1 | 0 | 0 | 1 |

**IF S1 < Limite *and* S2 < Limite *and***
**S3 > Limite *and* S4 > Limite**
**THEN Action(Turn_Left)**

**IF S2 > Limite *and***
**S3 > Limite *and***
**S2 > S3 *and***
**S1 > S4**
**THEN Action(Turn_Right)**

## Controle Reativo: *Sensorial-Motor*

### *Criando um Sistema de Controle para o robô Khepera*

**Programa de Controle: SIM0**
**- Acompanha o Simulador**
**- Objetivo: NENHUM - Sem Controle**
**Apenas o cabeçalho das funções...**

```
EXAMPLE 0 - README file

This example contains the minimal almost empty user files necessary for
compiling Khepera simulator. If you want to write your own controller for the
Khepera robot, you can fill in the empty areas with your own C code. If you
run example 0 without modifying it, the robot will remain stuck because it has
no controller ! However, everything else should work...
```

```
/*****************************************************************************/
/* File:        user.c (Khepera Simulator)                                   */
/* Author:      Olivier MICHEL <om@alto.unice.fr>                            */
/* Date:        Thu Sep 21 14:39:05 1995                                     */
/* Description: example of user.c file                                       */
/*                                                                           */
/* Copyright (c) 1995                                                        */
/* Olivier MICHEL                                                            */
/* MAGE team, i3S laboratory,                                                */
/* CNRS, University of Nice - Sophia Antipolis, FRANCE                       */
/*                                                                           */
/* Permission is hereby granted to copy this package for free distribution.  */
/* The author's name and this copyright notice must be included in any copy. */
/* Commercial use is forbidden.                                              */
/*****************************************************************************/

#include "../SRC/include.h"
#include "user_info.h"
#include "user.h"
```

```
void UserInit(struct Robot *robot)
{
}

void UserClose(struct Robot *robot)
{
}

void NewRobot(struct Robot *robot)
{
}

void LoadRobot(struct Robot *robot,FILE *file)
{
}

void SaveRobot(struct Robot *robot,FILE *file)
{
}

void RunRobotStart(struct Robot *robot)
{
}

void RunRobotStop(struct Robot *robot)
{
}

boolean StepRobot(struct Robot *robot)
{
  return(TRUE);
}

void FastStepRobot(struct Robot *robot)
{
}

void ResetRobot(struct Robot *robot)
{
}

void UserCommand(struct Robot *robot,char *text)
{
}

void DrawUserInfo(struct Robot *robot,u_char info,u_char page)
{
}
```



## Controle Reativo: *Sensorial-Motor*

### *Criando um Sistema de Controle para o robô Khepera*

**Programa de Controle: SIM1**
**- Acompanha o Simulador**
**- Objetivo: Evitar Obstáculos... E nada mais!**

```
/*****************************************************************/
/* File:        user.c (Khepera Simulator)                       */
/* Author:      Olivier MICHEL <om@alto.unice.fr>                */
/* Date:        Thu Sep 21 14:39:05 1995                         */
/* Description: example of user.c file                           */
/*                                                               */
/* Copyright (c) 1995                                            */
/* Olivier MICHEL                                                */
/* MAGE team, i3S laboratory,                                    */
/* CNRS, University of Nice - Sophia Antipolis, FRANCE           */
/*                                                               */
/* Permission is hereby granted to copy this package for free    */
/* distribution. The author's name and this copyright notice must */
/* be included in any copy. Commercial use is forbidden.         */
/*****************************************************************/

#include "../SRC/include.h"
#include "user_info.h"
#include "user.h"

#define FORWARD_SPEED   5       /* normal (slow) forward speed*/
#define TURN_SPEED      4       /* normal (slow) turn speed */
#define COLLISION_TH    900     /* value of IR sensors to be
                                   considered as collision */

int pas=0;

void DrawStep()
{
  char text[256];

  sprintf(text,"step = %d",pas);
  Color(GREY);
  UndrawText(200,100,"step = 500");
  Color(BLUE);
  DrawText(200,100,text);
}

void UserInit(struct Robot *robot) { }

void UserClose(struct Robot *robot) { }
```

```
void NewRobot(struct Robot *robot) { pas = 0; }

void LoadRobot(struct Robot *robot,FILE *file) { }

void SaveRobot(struct Robot *robot,FILE *file) { }

void RunRobotStart(struct Robot *robot) { ShowUserInfo(2,1); }

void RunRobotStop(struct Robot *robot)  { ShowUserInfo(1,1); }

boolean StepRobot(struct Robot *robot)
{
  pas++;
  DrawStep();
  if ((robot->IRSensor[0].DistanceValue > COLLISION_TH) ||
      (robot->IRSensor[1].DistanceValue > COLLISION_TH) ||
      (robot->IRSensor[2].DistanceValue > COLLISION_TH))
                                  /* if there is a collision on the
                                     left side of the robot */
  {
    robot->Motor[LEFT].Value  =  TURN_SPEED;
    robot->Motor[RIGHT].Value = -TURN_SPEED;       /* turn right */
  }
  else if ((robot->IRSensor[3].DistanceValue > COLLISION_TH) ||
           (robot->IRSensor[4].DistanceValue > COLLISION_TH) ||
           (robot->IRSensor[5].DistanceValue > COLLISION_TH))
                                 /* if there is a collision on the
                                    right side of the robot */
  {
    robot->Motor[LEFT].Value  = -TURN_SPEED;
    robot->Motor[RIGHT].Value =  TURN_SPEED;      /* turn left */
  }
  else
  {
    robot->Motor[LEFT].Value  = FORWARD_SPEED;
    robot->Motor[RIGHT].Value = FORWARD_SPEED;
    /* else go forward (default) */
  }

  if ((robot->IRSensor[6].DistanceValue > COLLISION_TH)||
      (robot->IRSensor[7].DistanceValue > COLLISION_TH))
   return(FALSE);            /* collision in the back */
  else
   return(TRUE);
}
```

## Controle Reativo: *Sensorial-Motor*

### *Criando um Sistema de Controle para o robô Khepera*

**Programa de Controle: SIM1**
**- Acompanha o Simulador**
**- Objetivo: Evitar Obstáculos... E nada mais!**

```
void FastStepRobot(struct Robot *robot) { }

void ResetRobot(struct Robot *robot) { pas = 0; }

void UserCommand(struct Robot *robot,char *text)
{
  WriteComment("unknown command"); /* no commands */
}

void DrawUserInfo(struct Robot *robot,u_char info,u_char page)
{
  char text[256];

  switch(info)
  {
    case 1:
      switch(page)
      {
        case 1: Color(MAGENTA);
                FillRectangle(0,0,40,40);
                Color(BLUE);
                DrawLine(100,100,160,180);
                Color(WHITE);
                DrawPoint(200,200);
                Color(YELLOW);
                DrawRectangle(240,100,80,40);
                Color(GREEN);
                DrawText(240,230,"hello world");
                break;
        case 2: Color(RED);
                DrawArc(200,50,100,100,0,360*64);
                Color(YELLOW);
                FillArc(225,75,50,50,0,360*64);
                Color(BLACK);
                DrawText(140,170,"This is the brain of the robot");
      }
      break;
    case 2:     DrawStep();
  }
}
```

```
void NewRobot(struct Robot *robot) { pas = 0; }

void LoadRobot(struct Robot *robot,FILE *file) { }

void SaveRobot(struct Robot *robot,FILE *file) { }

void RunRobotStart(struct Robot *robot) { ShowUserInfo(2,1); }

void RunRobotStop(struct Robot *robot)  { ShowUserInfo(1,1); }

boolean StepRobot(struct Robot *robot)
{
  pas++;
  DrawStep();
  if ((robot->IRSensor[0].DistanceValue > COLLISION_TH) ||
      (robot->IRSensor[1].DistanceValue > COLLISION_TH) ||
      (robot->IRSensor[2].DistanceValue > COLLISION_TH))
                                  /* if there is a collision on the
                                     left side of the robot */
  {
    robot->Motor[LEFT].Value  =  TURN_SPEED;
    robot->Motor[RIGHT].Value = -TURN_SPEED;       /* turn right */
  }
  else if ((robot->IRSensor[3].DistanceValue > COLLISION_TH) ||
           (robot->IRSensor[4].DistanceValue > COLLISION_TH) ||
           (robot->IRSensor[5].DistanceValue > COLLISION_TH))
                                  /* if there is a collision on the
                                     right side of the robot */
  {
    robot->Motor[LEFT].Value  = -TURN_SPEED;
    robot->Motor[RIGHT].Value =  TURN_SPEED;      /* turn left */
  }
  else
  {
    robot->Motor[LEFT].Value  = FORWARD_SPEED;
    robot->Motor[RIGHT].Value = FORWARD_SPEED;
    /* else go forward (default) */
  }

  if ((robot->IRSensor[6].DistanceValue > COLLISION_TH)||
      (robot->IRSensor[7].DistanceValue > COLLISION_TH))
   return(FALSE);            /* collision in the back */
  else
   return(TRUE);
}
```



## Controle Reativo: *Sensorial-Motor*

### *Criando um Sistema de Controle para o robô Khepera*

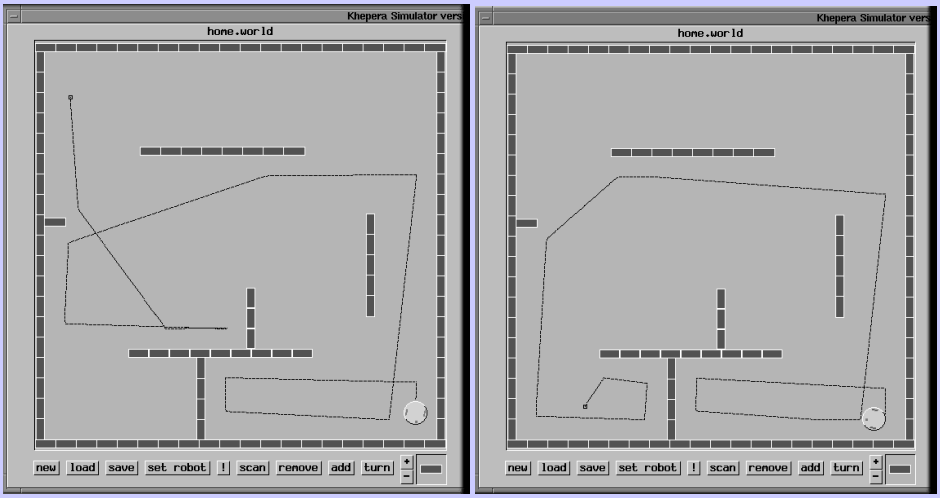
**Programa de Controle: Desenvolvido por F.S.Osório**
**- Objetivo: Evitar Obstáculos... E nada mais! (Usa regras do NeuComp / INSS)**

```
#define DETECTWALL 900
#define SENSIB     100

% VERY SIMPLE Autonomous Robot Controller - by Osorio, Lab. Leibniz

$Features:
SSL : range : [0.0 , 1024.0];  % Sensor 0            2   3
SL  : range : [0.0 , 1024.0];  % Sensor 1       1    SFL SFR    4
SFL : range : [0.0 , 1024.0];  % Sensor 2         SL          SR
SFR : range : [0.0 , 1024.0];  % Sensor 3     SSL               SSR
SR  : range : [0.0 , 1024.0];  % Sensor 4      0                 5
SSR : range : [0.0 , 1024.0];  % Sensor 5
SBR : range : [0.0 , 1024.0];  % Sensor 6            SBL SBR
SBL : range : [0.0 , 1024.0].  % Sensor 7             7   6
$End_Feat.

$Rules:

right <= GT (SSL, DETECTWALL, SENSIB);
right <= GT (SL,  DETECTWALL, SENSIB);
right <= GT (SFL, DETECTWALL, SENSIB);

left  <= Not(right), GT (SFR, DETECTWALL, SENSIB);
left  <= Not(right), GT (SR,  DETECTWALL, SENSIB);
left  <= Not(right), GT (SSR, DETECTWALL, SENSIB);

forward <=  Not(right), Not(left).

$End_rules.

$End.
```

**Regras: Transformadas em uma Rede Neural**

**Controle Reativo:** ***Sensorial-Motor*** ***Criando um Sistema de Controle para o robô Khepera***

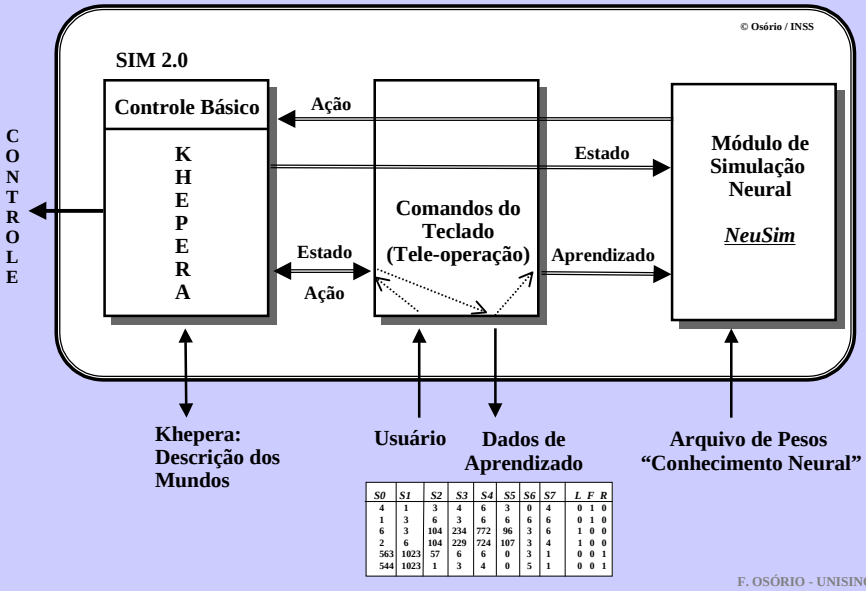
**Programa de Controle: Desenvolvido por F.S.Osório**
**- Objetivo: Evitar Obstáculos... E nada mais! (Usa regras do NeuComp / INSS)**






**Controle Reativo:** ***Sensorial-Motor*** ***Criando um Sistema de Controle para o robô Khepera***

**Alterações / Personalizações feitas no simulador...**




| S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | L | F | R |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 3 | 4 | 6 | 3 | 0 | 4 | 0 | 1 | 0 |
| 1 | 3 | 6 | 3 | 6 | 6 | 6 | 6 | 0 | 1 | 0 |
| 6 | 3 | 104 | 234 | 772 | 96 | 3 | 6 | 1 | 0 | 0 |
| 2 | 6 | 104 | 229 | 724 | 107 | 3 | 4 | 1 | 0 | 0 |
| 563 | 1023 | 57 | 6 | 6 | 0 | 3 | 1 | 0 | 0 | 1 |
| 544 | 1023 | 1 | 3 | 4 | 0 | 5 | 1 | 0 | 0 | 1 |

**Controle Reativo:** *Sensorial-Motor* — *Criando um Sistema de Controle para o robô Khepera*

## Alterações / Personalizações feitas no simulador...

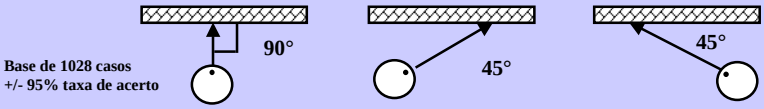
```
MYEXAMPLE: Comando do módulo user.c implementado por F.S. Osório

Modo "Command"

  set angle xx   (xx = 0 to 360)
  file name      (file name with no extension - used extensions: *.dat, *.rpl)
  net  name      (file name with no extension - used extensions: *.cfg, *.wts)
  hyc  0/1       (Hybrid Control ON (1) or OFF (0))
  act  xx        (Activation threshold = xx)
  binp xx        (Binary inputs/Sensors - Binarization threshold = xx)
  rplt xx        (Replay file type)
                 (Parameter: Type 0 = initial position + commands  "replay mode")
                             Type 1 = position x,y, angle          "trace  mode")
                             Type 2 = sensors + position x,y,angle "conv_robdt")
  ntyp xx       (ANNetwork type - Inputs specification)
                 (Parameter: Type 0 = 8 inputs - sensors)
                             Type 1 =   ??                        )
                             Type 2 =   ??                        )

Modo "Run"

  'o' = Open output files (*.dat - sensor/action data, *.rep - replay data)
  'm' = Output replay file type = X,Y,Angle (type 1)
  'w' = Write ANN trace on replay file on/off
  'c' = Close output files
  'r' = Replay mode on/off
  't' = Print robot trajectory on screen (replay file)
  'n' = ANN Robot control on/off (step by step - using space)
  'a' = ANN Robot control on
  ' ' = Repeat one replay or ANN step (' ' = space)
   ^  = Go forward (upper arrow key)
  <-  = Turn left  (left  arrow key)
   -> = Turn right (right arrow key)
  'q' = Quit interactive mode (finish run)

  'h' = SEE THIS TEXT!
```

```
Output File Description
***********************

Replay Type 0 - *.rpl
"""""""""""""""""""""
0
x y angle
F/L/R
F/L/R...

Replay Type 1 - *.rpl
"""""""""""""""""""""""
1
x y angle
x y angle
...

Replay Type 2 - *.rpl
"""""""""""""""""""""""
2
s0 s1 s2 s3 s4 s5 s6 s7 x y angle [F/L/R]
s0 s1 s2 s3 s4 s5 s6 s7 x y angle [F/L/R]
...

Data File - *.dat
"""""""""""""""""

s0 s1 s2 s3 s4 s5 s6 s7 - [F/L/R]  [B?!?]
...

F=Forward, L=Left, R=Right, B=Backward
```

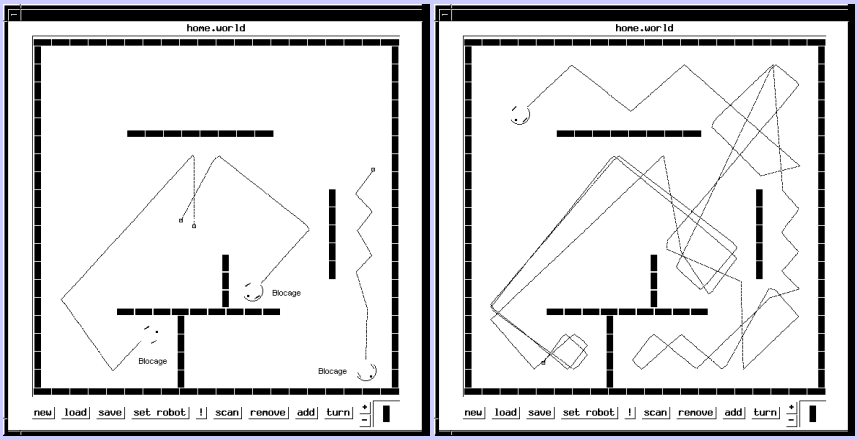


**Controle Reativo:** *Sensorial-Motor* — *Criando um Sistema de Controle para o robô Khepera*

## Aprendizado... Evitar Obstáculos




**Base de 1028 casos**
**+/- 95% taxa de acerto**

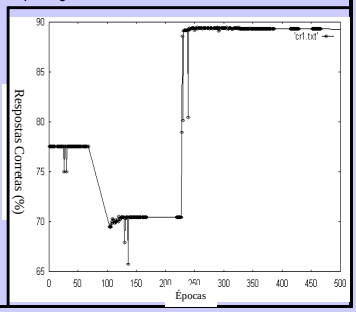



*Evitar Obstáculos - Aprendizado Supervisionado Incremental*

**Controle Reativo:** *Sensorial-Motor* *Criando um Sistema de Controle para o robô Khepera*

**Aprendizado... Integrando regras e Exemplos**

```
#define DETECTWALL 900
#define SENSIB     100

% VERY SIMPLE Autonomous Robot Controller - by Osorio, Lab. Leibniz

$Features:
SSL : range : [0.0 , 1024.0];  % Sensor 0            2   3
SL  : range : [0.0 , 1024.0];  % Sensor 1       1    SFL SFR    4
SFL : range : [0.0 , 1024.0];  % Sensor 2         SL          SR
SFR : range : [0.0 , 1024.0];  % Sensor 3     SSL               SSR
SR  : range : [0.0 , 1024.0];  % Sensor 4      0                 5
SSR : range : [0.0 , 1024.0];  % Sensor 5
SBR : range : [0.0 , 1024.0];  % Sensor 6           SBL SBR
SBL : range : [0.0 , 1024.0].  % Sensor 7            7   6
$End_Feat.

$Rules:

right <= GT (SSL, DETECTWALL, SENSIB);
right <= GT (SL,  DETECTWALL, SENSIB);
right <= GT (SFL, DETECTWALL, SENSIB);

left  <= Not(right), GT (SFR, DETECTWALL, SENSIB);
left  <= Not(right), GT (SR,  DETECTWALL, SENSIB);
left  <= Not(right), GT (SSR, DETECTWALL, SENSIB);

forward <=  Not(right), Not(left).

$End_rules.

$End.
```

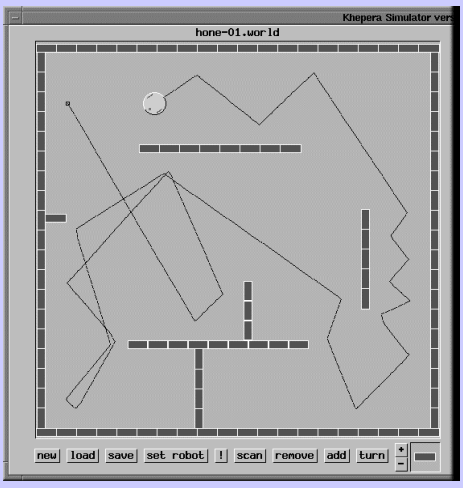


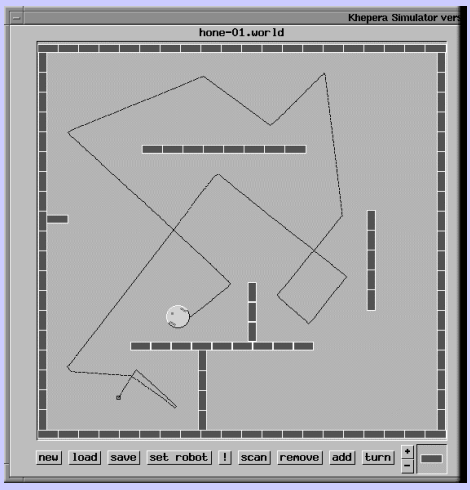

***Evitar Obstáculos - Híbrido Neuro-Simbólico***



**Controle Reativo:** *Sensorial-Motor* *Criando um Sistema de Controle para o robô Khepera*

**Aprendizado... Integrando regras e Exemplos**



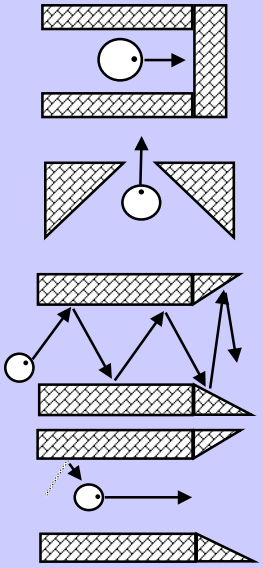



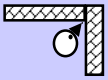

***Evitar Obstáculos - Híbrido Neuro-Simbólico***

F. OSÓRIO - UNISINOS 2000

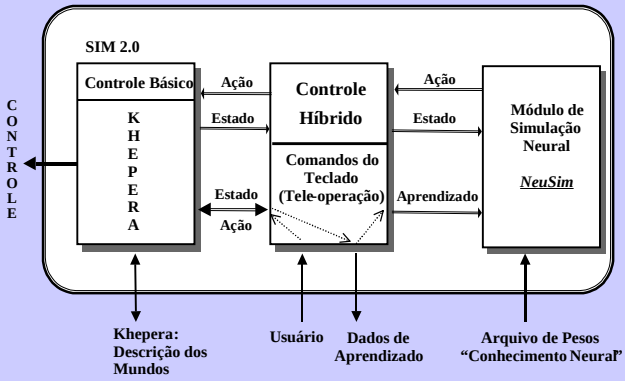
**Controle Reativo**: *Sensorial-Motor* *Criando um Sistema de Controle para o robô Khepera*

**Aprendizado...**
**Evitar Obstáculos: PROBLEMAS!**





**Alternância:**
**- Esq, Dir, Esq, Dir, Esq, Dir, ...**



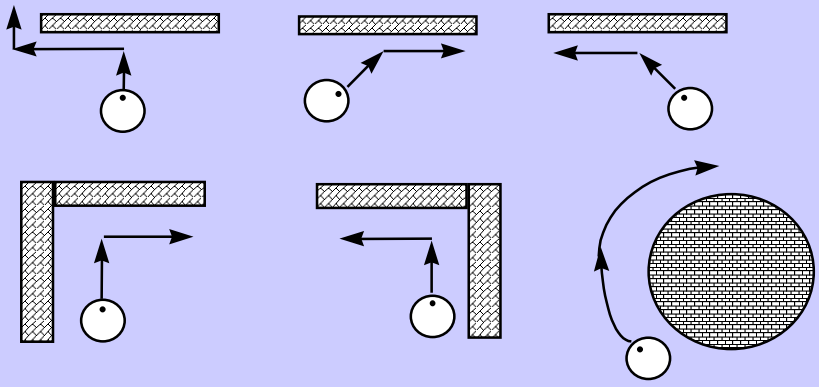

**Tarefas:**
- **Evitar as paredes**
- **Seguir as paredes**

**Problemas:**
* **Tempo / Seqüência**
* **Controle "bem comportado"**

F. OSÓRIO - UNISINOS 2000

**Controle Reativo**: *Sensorial-Motor* *Criando um Sistema de Controle para o robô Khepera*

**Aprendizado... Seguir Paredes**
**Situações previstas na base de dados de aprendizado.**

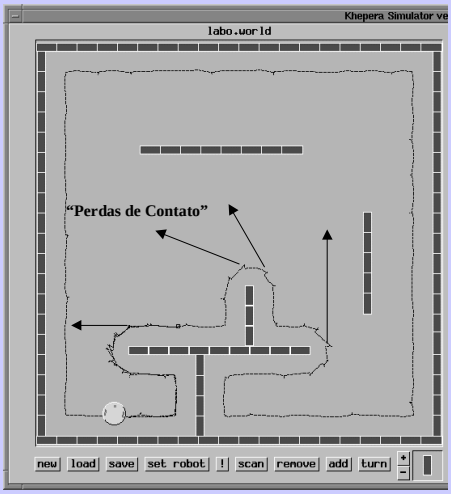


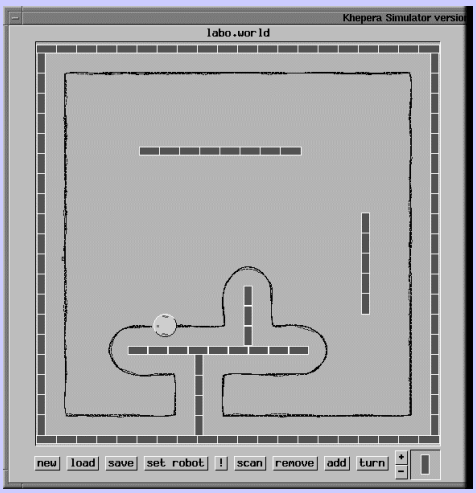
F. OSÓRIO - UNISINOS 2000

**Controle Reativo**: *Sensorial-Motor*

*Criando um Sistema de Controle para o robô Khepera*

**Aprendizado... Seguir Paredes**
**Problema: "perda de contato com a parede"**



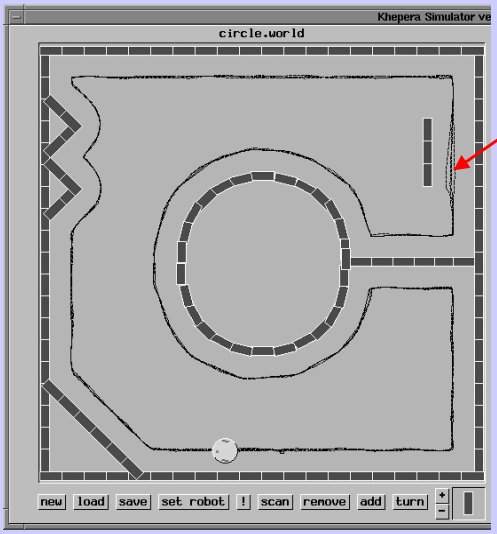






**Controle Reativo**: *Sensorial-Motor*

*Criando um Sistema de Controle para o robô Khepera*

**Aprendizado... Seguir Paredes**
**Problema: "perda de contato com a parede"**

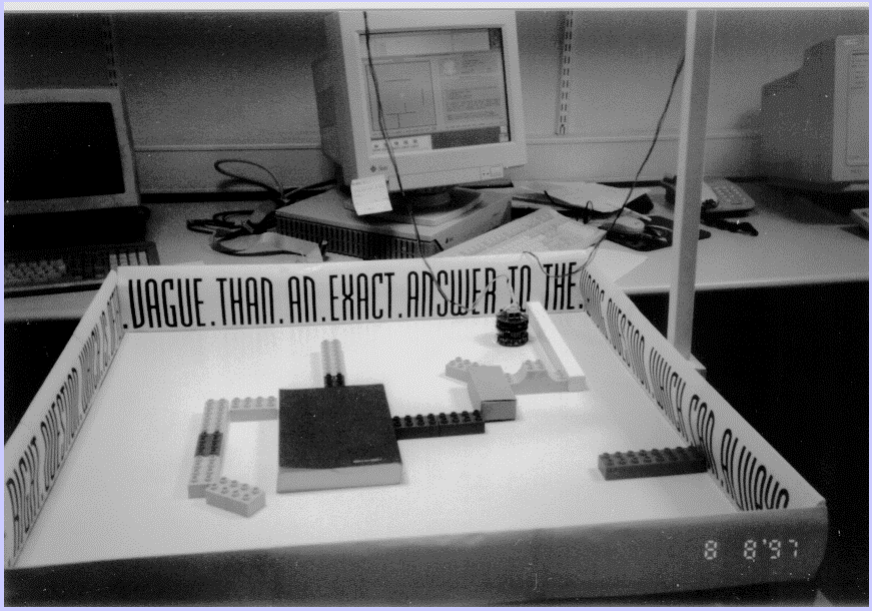



**Situação nova:**
**Boa generalização!**

**Problema: "Perda de Contato"**
**Soluções para o problema:**
**- Memória**
**- Contexto**

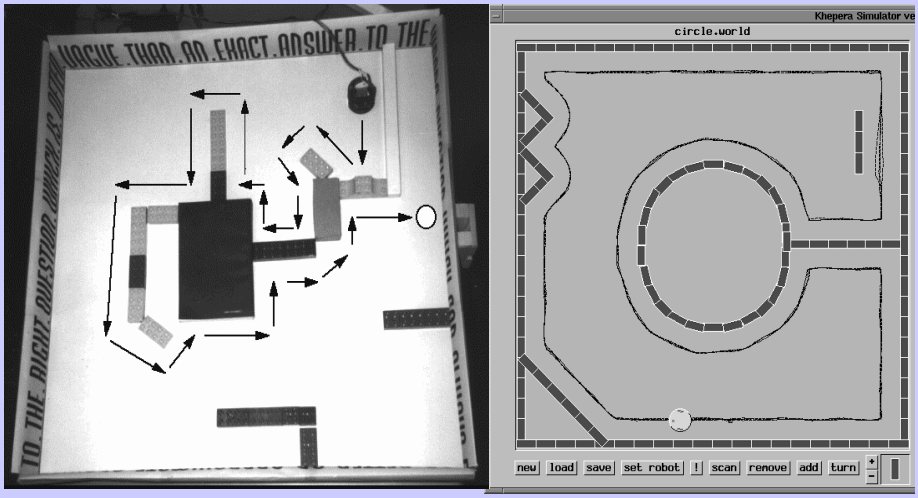
F. OSÓRIO - UNISINOS 2000

**Controle Reativo:** ***Sensorial-Motor - Experiência usando o robô Khepera real***






**Controle Reativo:** ***Sensorial-Motor - Experiência usando o robô Khepera real***

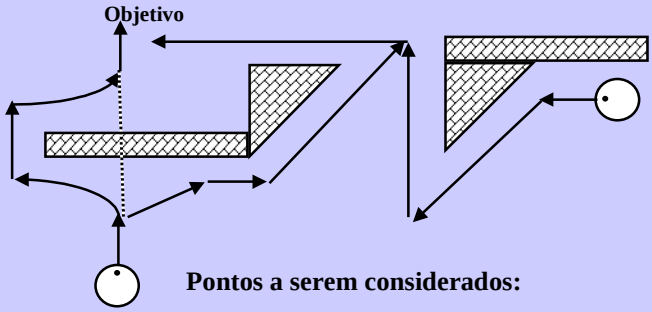



***INSS - Seguir um Muro - Aprendizado Incremental com Contexto***
***Contexto = S0 a S7 (tempo t) + S0 a S7 (tempo t-1)***

**Controle Reativo:** ***Sensorial-Motor - Experiências usando o robô Khepera***

**Discussão sobre os estudos realizados...**

⇒ Seguir uma trajetória e, ao mesmo tempo, evitar os obstáculos




**Pontos a serem considerados:**

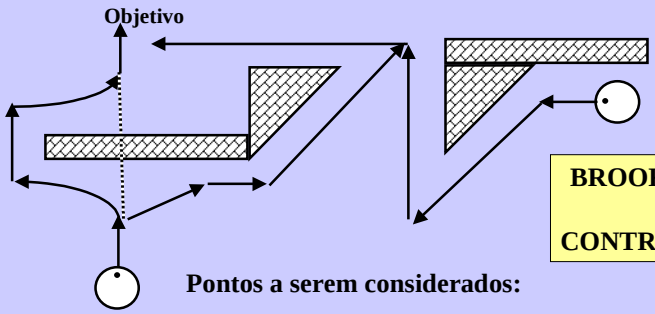
*** "Visão" local - Cego com a bengala (Global x Local)**
*** Armadilhas - Solução Local (Mínimos locais)**
*** Problema do posicionamento em um mapa global**
*** Modularidade: evitar, seguir, passar, ...**
*** Conhecimentos de alto nível: estratégias**



**Controle Reativo:** ***Sensorial-Motor - Experiências usando o robô Khepera***

**Discussão sobre os estudos realizados...**

⇒ Seguir uma trajetória e, ao mesmo tempo, evitar os obstáculos




**BROOKS / Hierárquico**
**&**
**CONTROLE HÍBRIDOS**

**Pontos a serem considerados:**

*** "Visão" local - Cego com a bengala (Global x Local)**
*** Armadilhas - Solução Local (Mínimos locais)**
*** Problema do posicionamento em um mapa global**
*** Modularidade: evitar, seguir, passar, ...**
*** Conhecimentos de alto nível: estratégias**

F. OSÓRIO - UNISINOS 2000

## ROBÓTICA AUTÔNOMA : Controle Reativo

*** Controle Reativo:**

- **Vantagens:**
  - **Capacidade de se adaptar (sentir) ao ambiente no qual está inserido.**
  - **Trata: Imprecisão, incerteza e alterações do ambiente.**
  - **Pode resolver problemas impostos por novas situações não previstas.**
- **Desvantagens:**
  - ***Mínimos locais*!**
  - ***Tarefas complexas*: "fazer algo além de ficar andando sem rumo".**
  - **Não trabalha com a noção de posicionamento do robô.**
  - **Não aproveita os conhecimentos disponíveis sobre o ambiente.**

**Vide:** **Brooks HomePage - http://www.ai.mit.edu/people/brooks/papers.html**
**Rodney A. Brooks - Cambrian Intelligence: The Early History of the New AI**
**Ronald C. Arkin - Behaviour Based Robotics**
**Richard Sutton & Andrew Barto - Reinforcement Learning: An Introduction**

**Fernando Osório. INSS: Un Système Hybride Neuro-Symbolique pour l'Apprentissage Automatique Constructif. Thèse de Doctorat. INPG / IMAG, Grenoble. 1998 (cap. 5)**
**Web: http://www.inf.unisinos.br/~osorio/these/**

**Khepera: http://diwww.epfl.ch/lami/team/michel/khep-sim/**
**LAMI - EPFL: http://dmtwww.epfl.ch/isr/asl/**