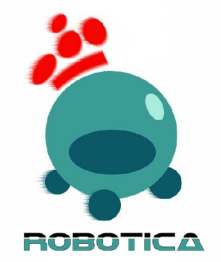
Programación de robots y arquitecturas software, casos prácticos

José María Cañas Plaza

http://gsyc.es/jmplaza



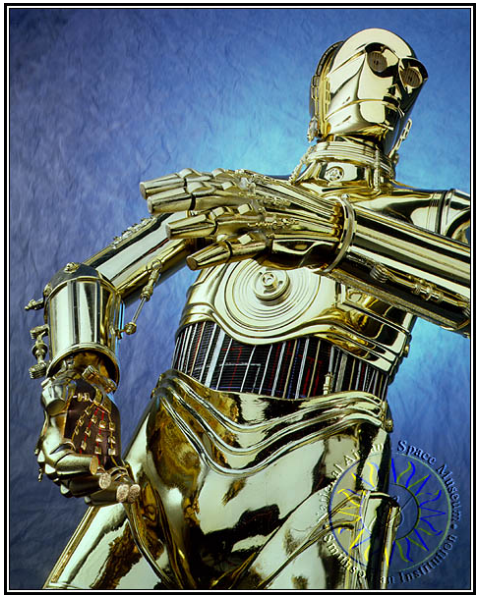
SICFIMA, 26 marzo 2007

Contenidos

- Introducción
- Arquitecturas cognitivas. JDE
- Plataforma software jdec
- Ejemplos de uso
- Conclusiones

Introducción

Robótica ficción vs Robótica real



¿Qué es un robot?

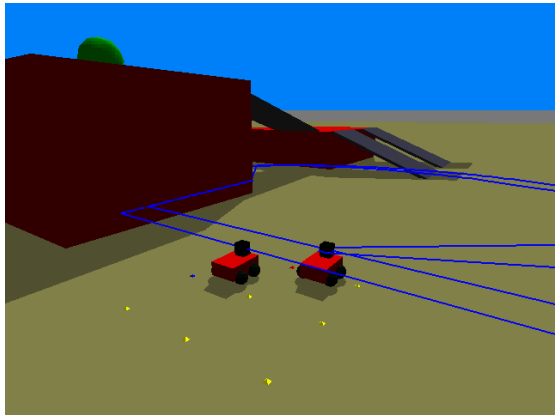
Sistema informático con:

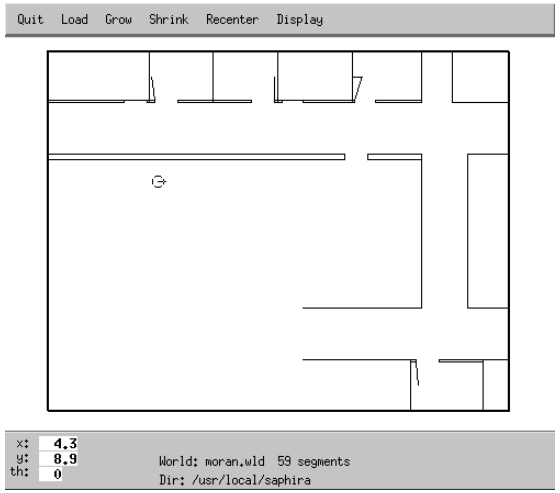
- Sensores
- Actuadores
- Computador

Hay que programarlo para que consiga sus objetivos y sea sensible a la situación

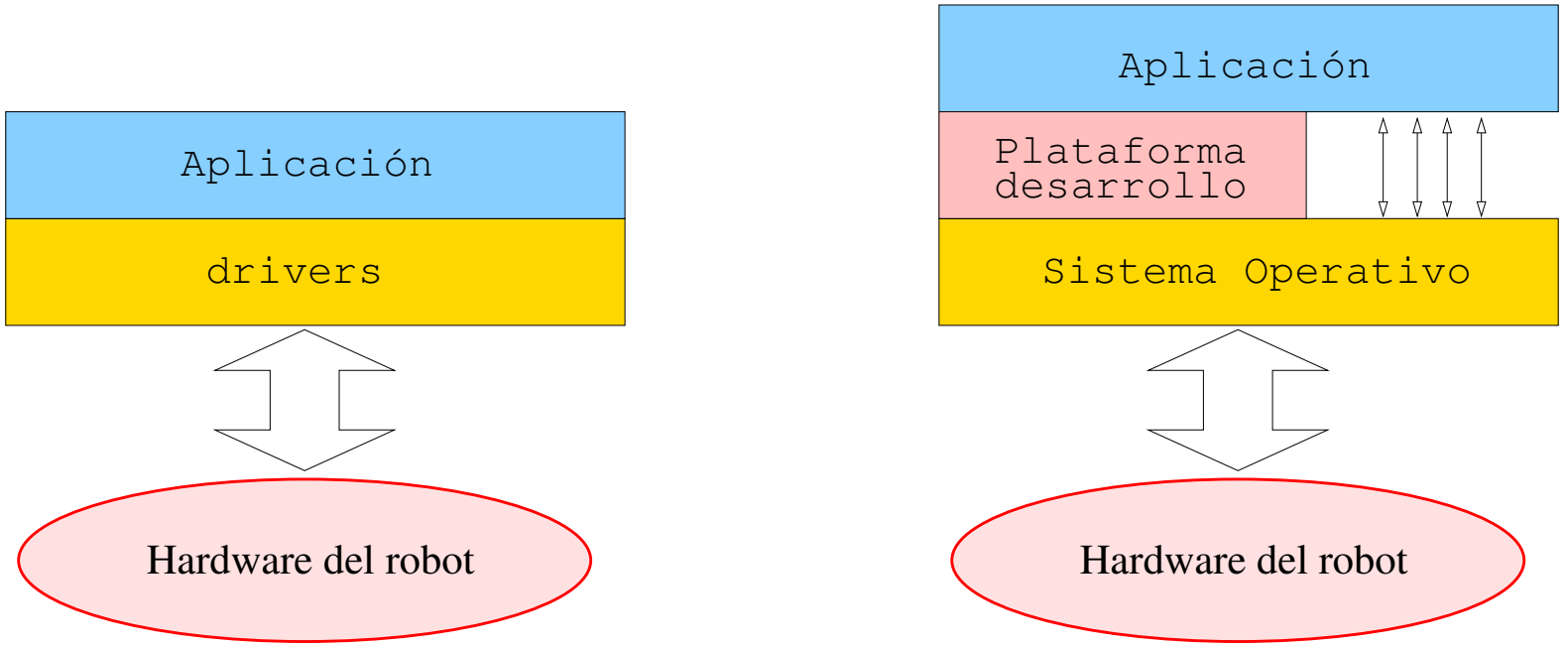
Software para robots

- Requisitos específicos
  - Vivacidad, agilidad
  - Multitarea
  - Distribuido, comunicaciones
  - Interfaz gráfica, depuración
  - Expandible
- Simuladores
- Lenguajes: ensamblador, C, C++
- Sistemas operativos: dedicados o generalistas
- Heterogeneidad





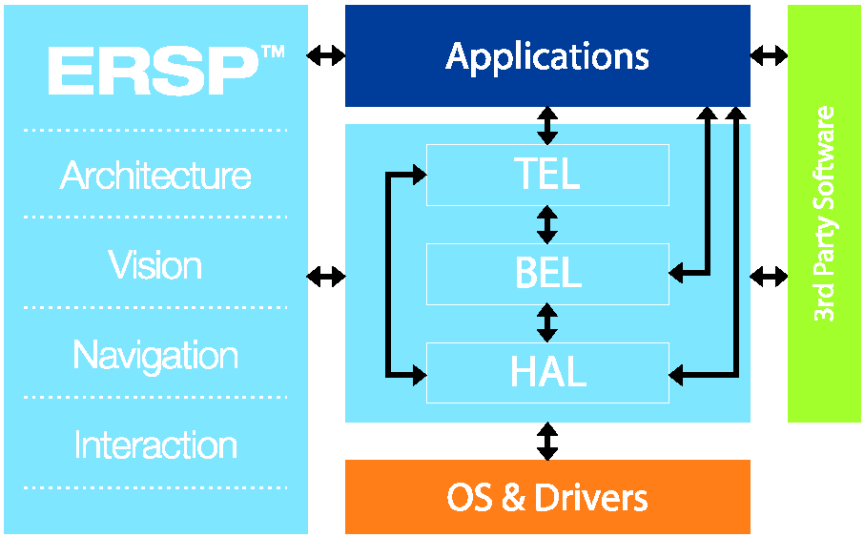
Programación de robots



- Robots pequeños: procesadores empotrados.
- Robots medianos-grandes: el PC como computador principal
- *Middleware* para simplificar la creación de aplicaciones robóticas

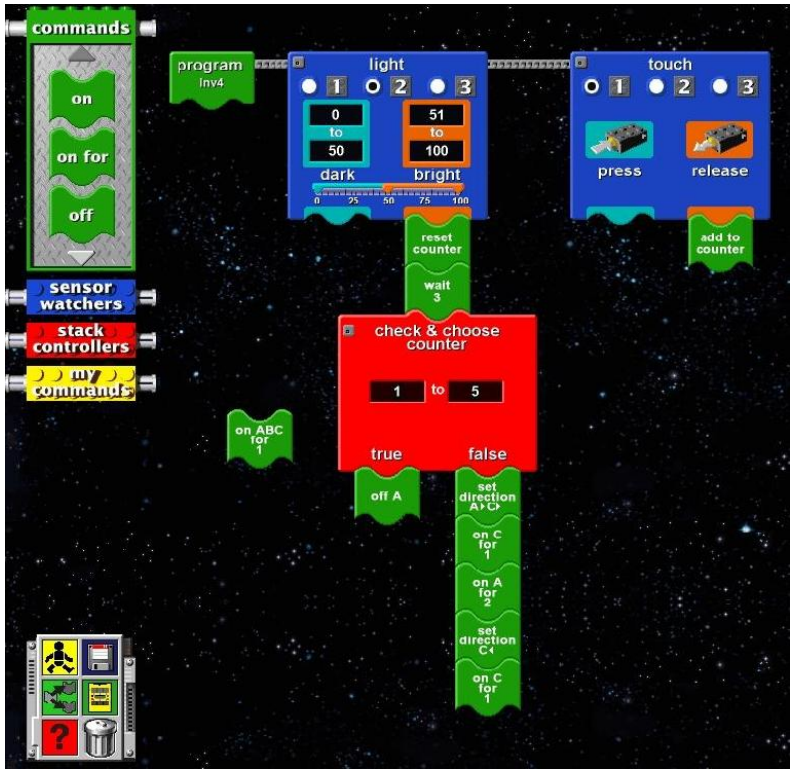
¿Qué proporciona una plataforma sw para robots?

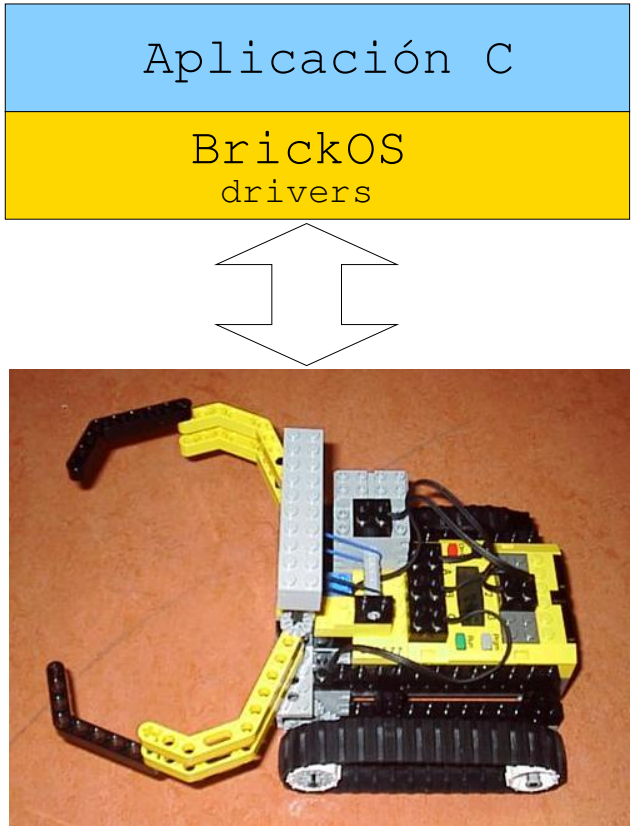
- Abstracción del hardware
- Arquitectura software
- Funcionalidades de uso común
- Arquitectura cognitiva



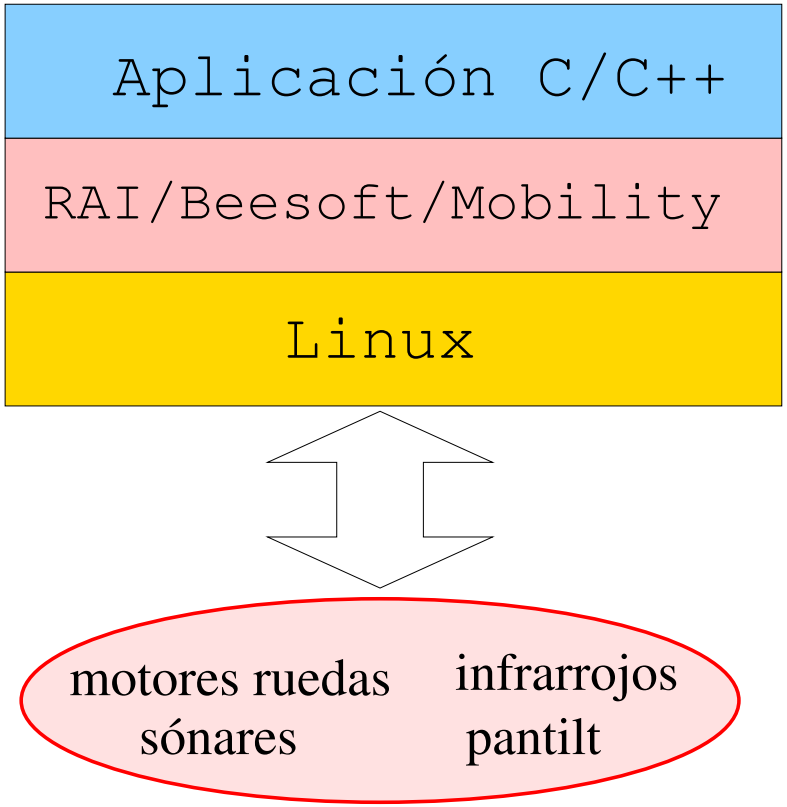
- Comerciales, investigación, software libre
- Ingeniería software: orientación a objetos, distribución
- Carmen, OROCOS, ERSP, Player/Stage, Miro, etc.

Programación del robot LEGO RCX

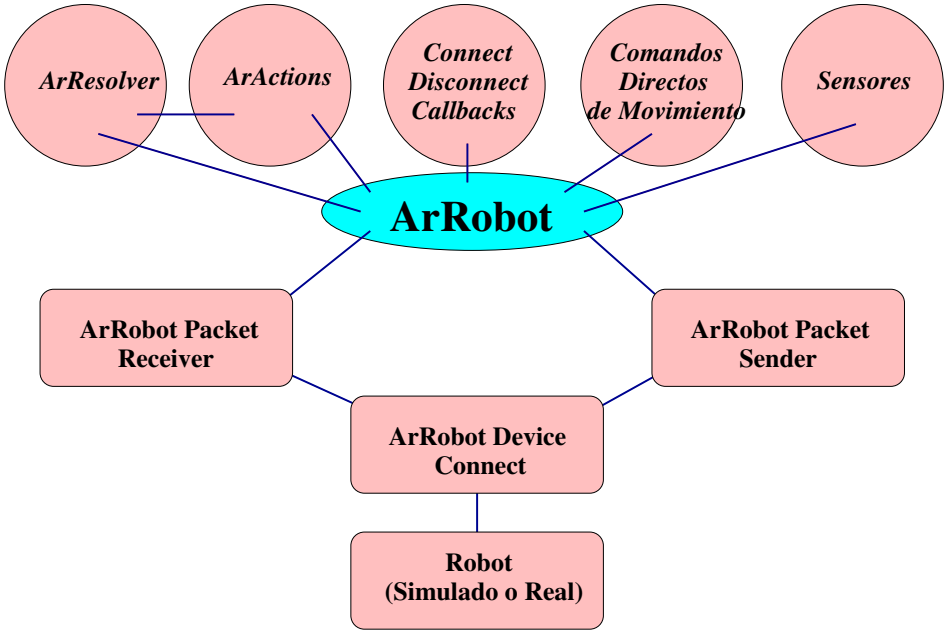


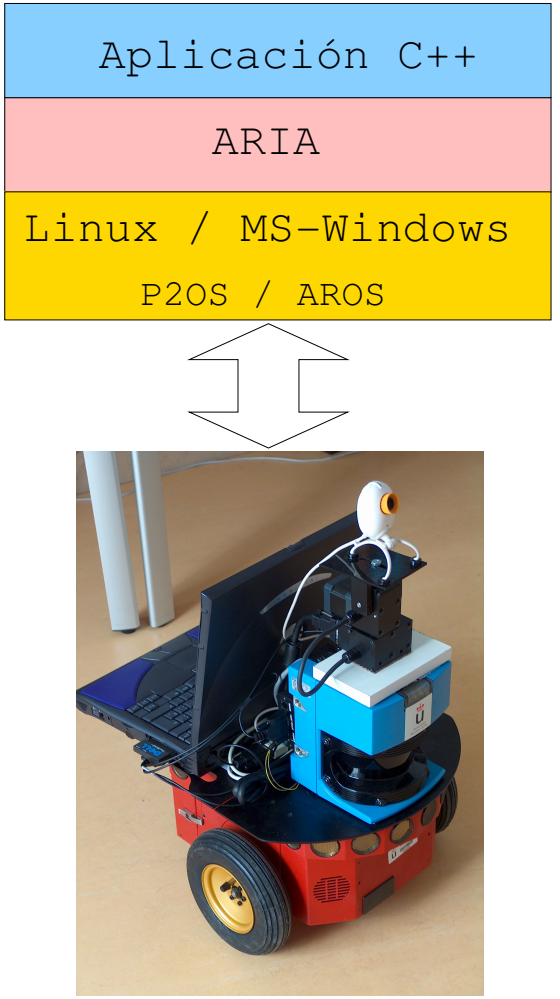


Programación del robot B21



Programación del robot Pioneer

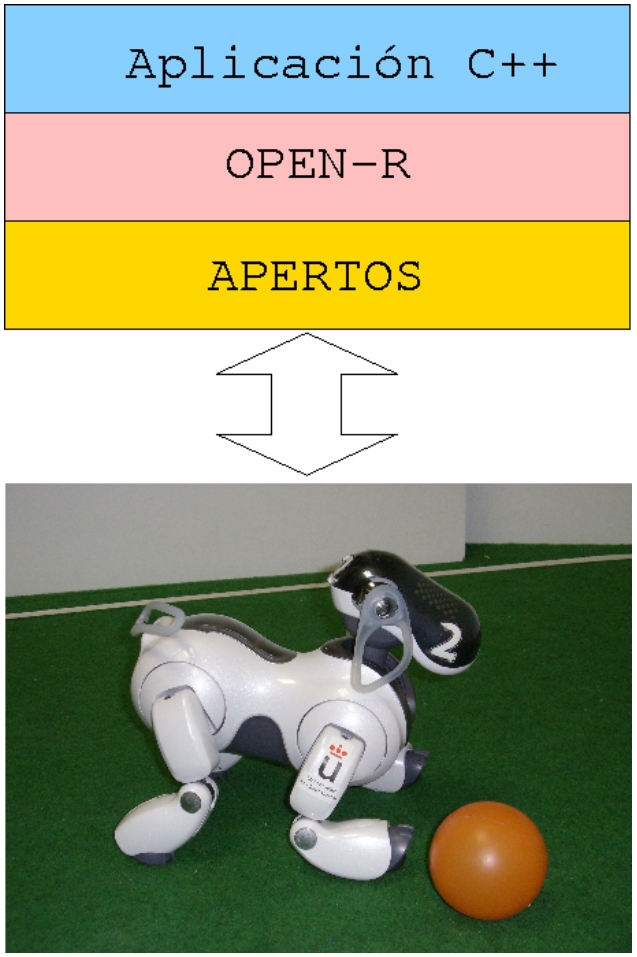




- Acceso a los sensores y actuadores
- Librerías, objetos
- ActivMedia: Saphira, ARIA (C++)

Programación del robot Aibo

- Objetos monohilo
- Se comunican vía mensajes
- Objetos básicos
  - OVirtualRobotComm: Effector, Sensor, OFbkImageSensor
  - OVirtualAudioComm: Mic, Speaker
  - ANT Aibo Network Tool



Arquitecturas cognitivas. JDE

¿Por qué no tenemos robots que hagan las tareas domésticas?

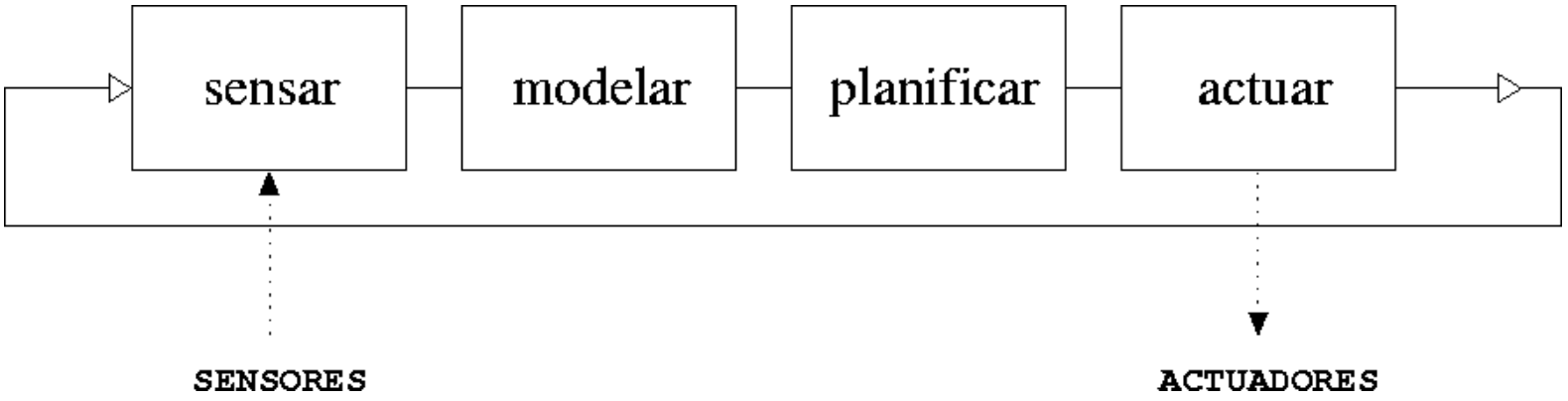
- Generar comportamiento autónomo
- Falta flexibilidad
- Tareas complejas
- ¿Problema tecnológico o teórico?
- No es sólo un problema de programación

Arquitectura cognitiva

La arquitectura de un robot es la organización de sus capacidades sensoriales, de procesamiento y de acción para conseguir un repertorio de comportamientos inteligentes interactuando con cierto entorno

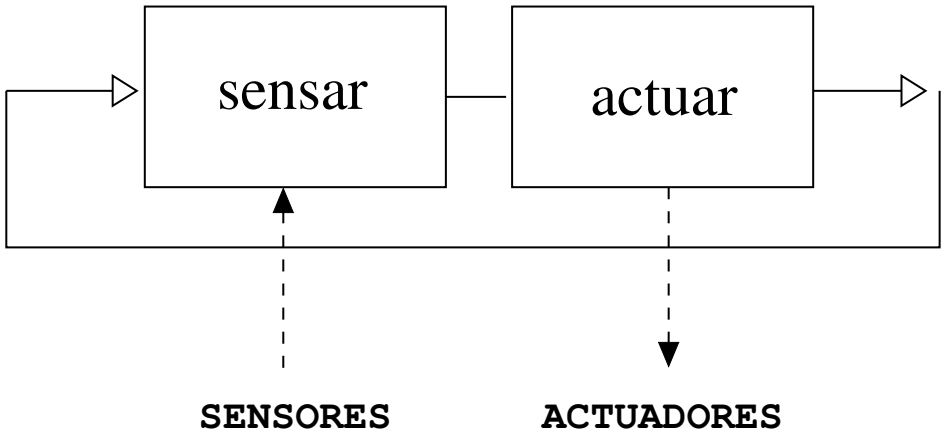
- La arquitectura determina el comportamiento observable
- Repertorio de comportamientos
- Información desbordante, incierta
- Selección de acción, atención

Escuela deliberativa



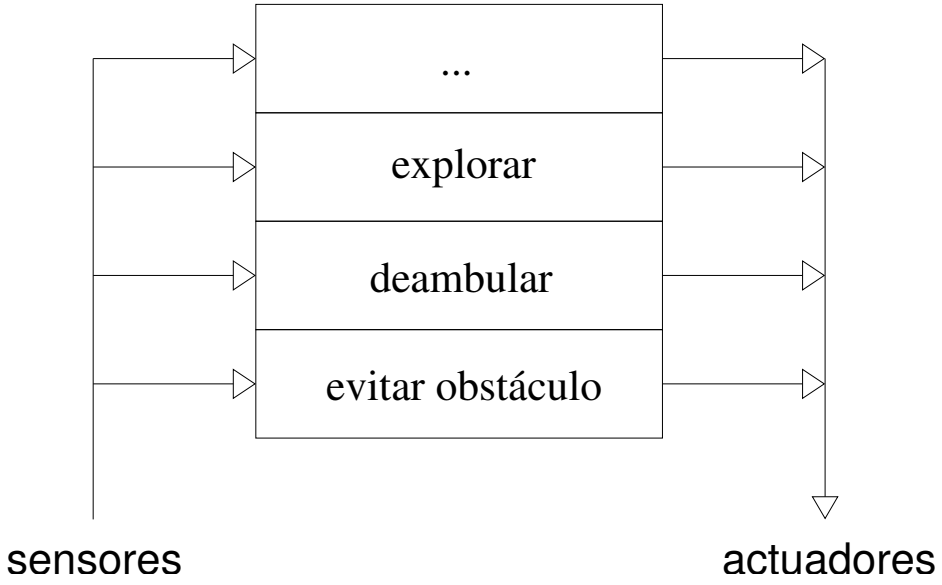
- Curso específico de acción: planificación y ejecución
- Percepción objetiva y modelado del mundo
- Abstracción funcional
- Shakey *[Nilsson,1969]*, Hilare *[Giralt et al,1983]*, RCS *[Albus,1993]*
- Frágil y lento

Escuela reactiva



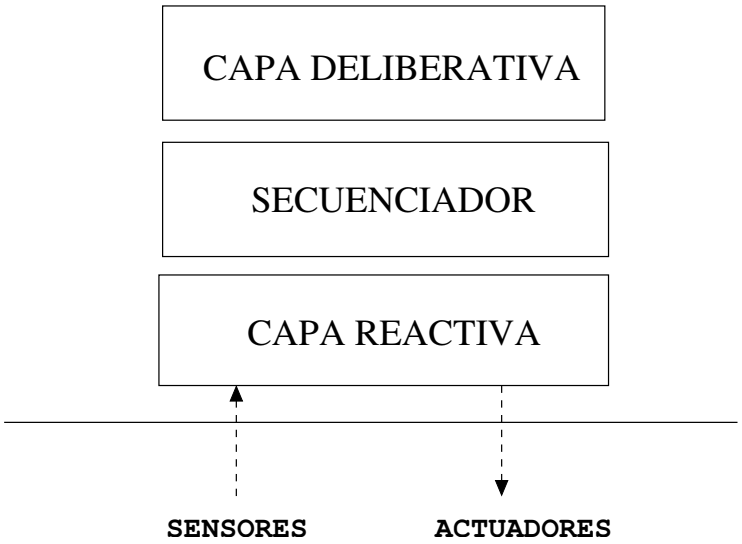
- Interacción continua con el entorno, acción situada
- Percepción subsimbólica
- Autómatas, Pengi *[Agre,1987]*, *[Payton,1990]*
- No escala

Sistemas basados en comportamientos



- Reactividad
- Subsunción: suprimir salidas y suplantar entradas
- *[Brooks,1986]*, *[Arkin,1989]*, *[Maes,1989]*
- No escala bien

Sistemas híbridos



- Capa deliberativa: razonamiento simbólico
- Capa reactiva: procesos que se activan discrecionalmente
- 3T *[Bonasso et al,1997]*, TCA *[Simmons,1994]*, Saphira *[Konolige,1998]*
- Niveles fijos

Arquitectura JDE

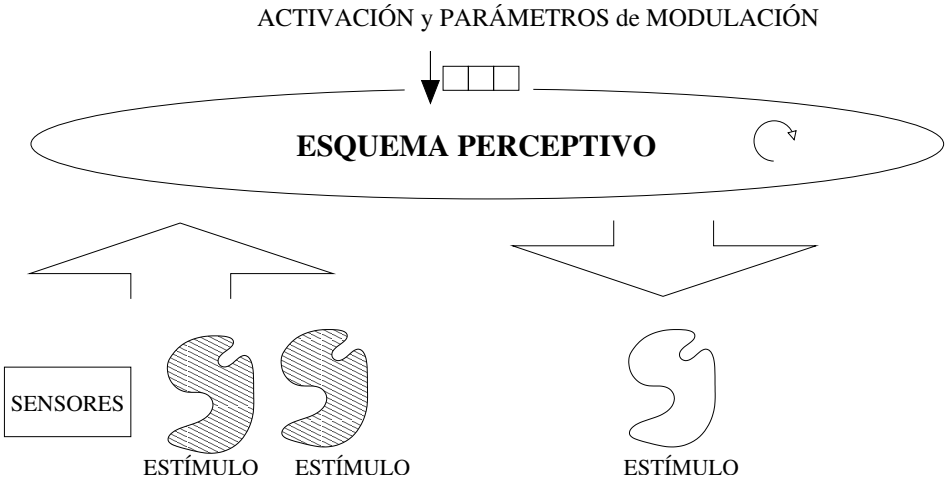
- Comportamiento = percepción y control
- Fragmentación en unidades (esquemas) asíncronas concurrentes
  - de percepción elaboran estímulos
  - de actuación toman decisiones
- La colección de esquemas se organiza en jerarquía
- Jerarquía Dinámica de Esquemas
- Sigue el paradigma basado en comportamientos

Esquemas

Un esquema es un flujo de ejecución independiente, con un objetivo

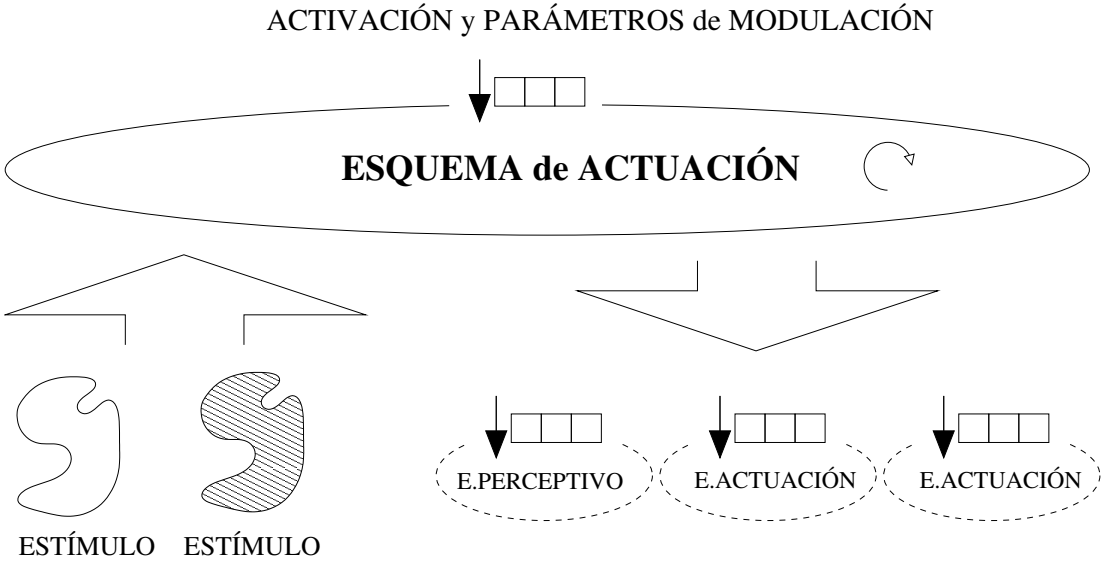
- Iterativo
- Se puede activar y desactivar a voluntad
- Modulable a través de parámetros
- Esquemas perceptivos y de actuación
- Su funcionalidad se usa despertándolo y modulándolo

Esquemas perceptivos



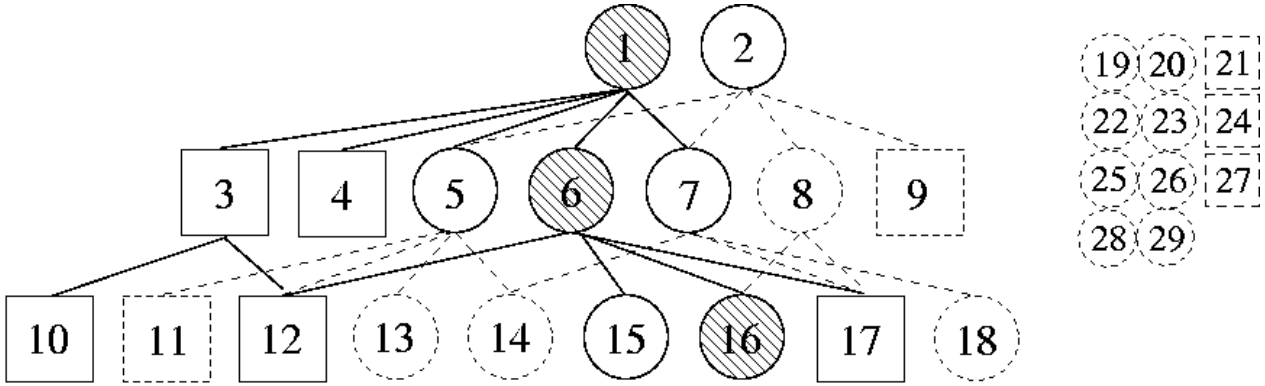
- Un estímulo es *una pieza de información que al menos un esquema de actuación necesita para tomar sus decisiones*
- Producen estímulos y los mantienen actualizados (anclados)
- Lecturas sensoriales, transformaciones más elaboradas
- Estados: DORMIDO o ACTIVO

Esquemas de actuación



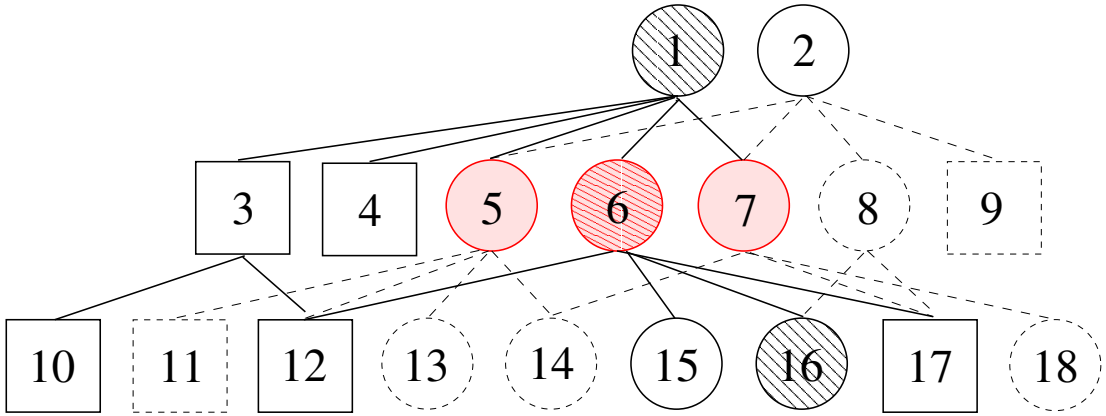
- Toman decisiones de actuación para conseguir o mantener un objetivo
- Su salida pueden ser comandos a los actuadores o la activación y modulación de otros esquemas hijos
- Estados: DORMIDO, ALERTA, PREPARADO o ACTIVO

Jerarquía

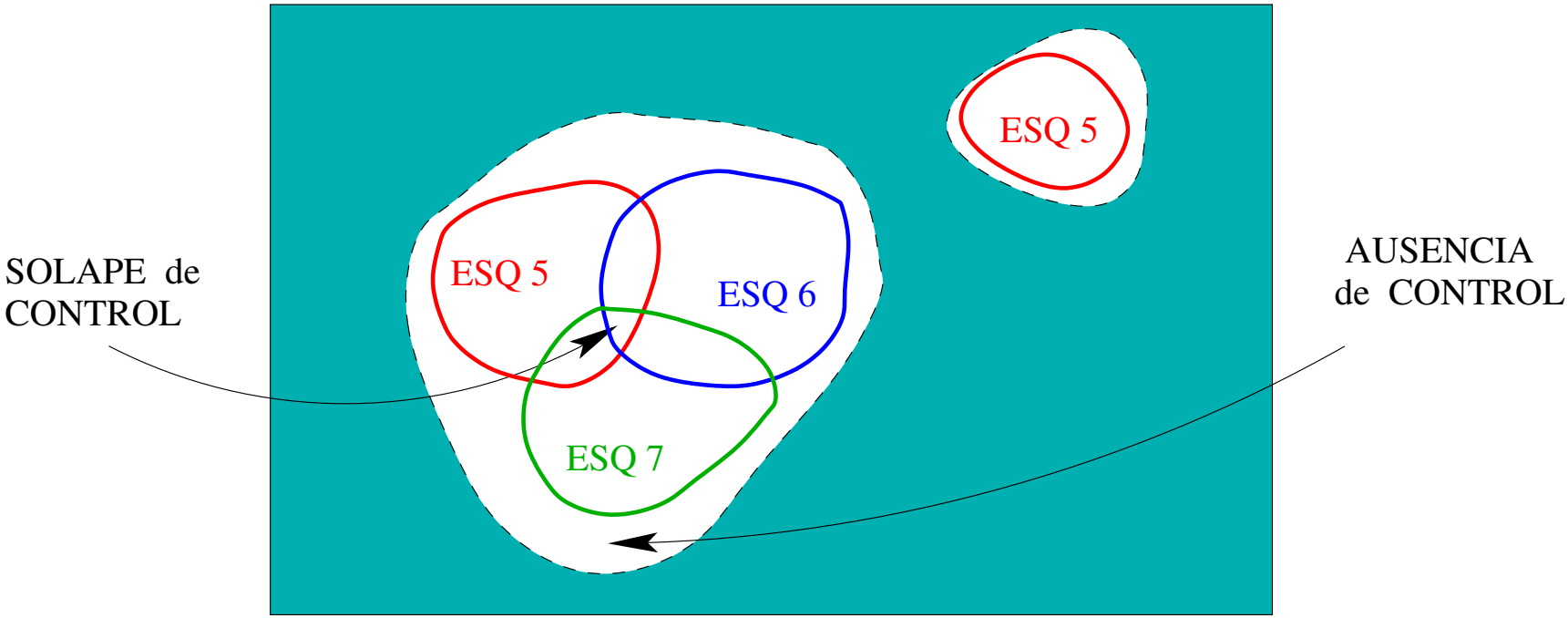


- Los esquemas despiertos se ejecutan asíncrona y concurrentemente
- Los esquemas se estructuran en jerarquía, donde hay padres e hijos
- Se construye y reconfigura dinámicamente
- Es específica de cada comportamiento
- El número de niveles depende de la complejidad de la tarea

Selección de acción

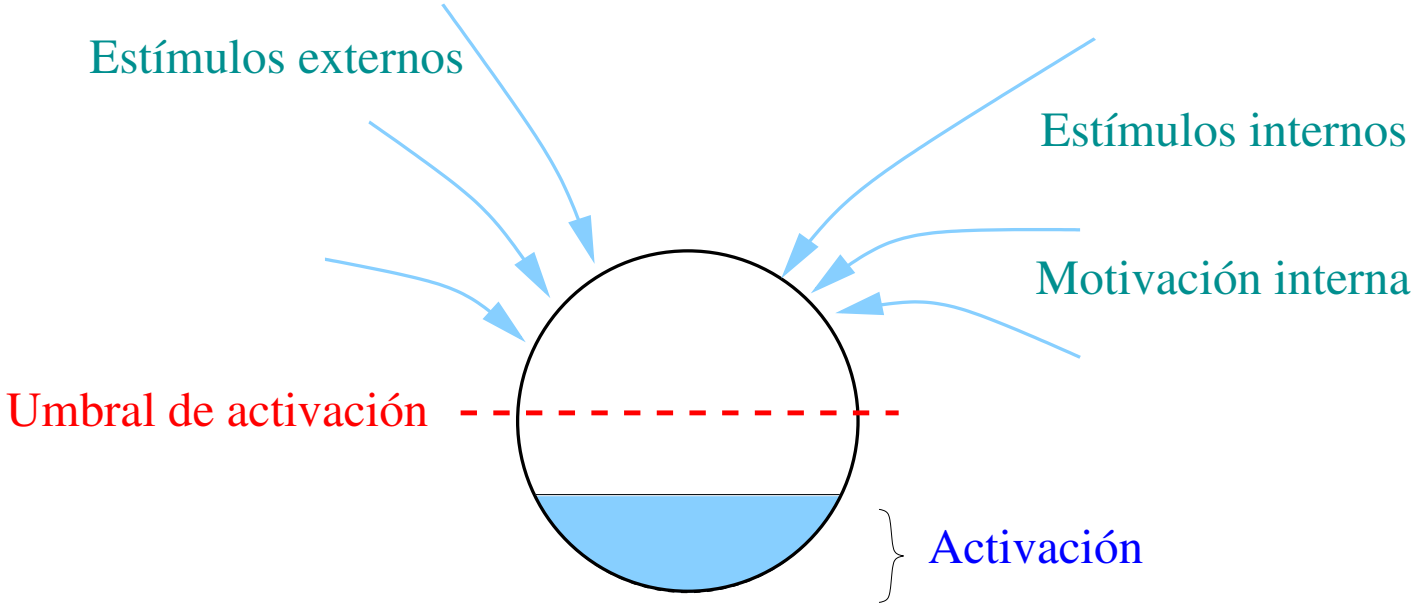


- Distribución: una competición por nivel
- El padre determina el paso de DORMIDO a ALERTA
- La situación determina el paso de ALERTA a PREPARADO
- De PREPARADO a ACTIVO se resuelve con un arbitraje



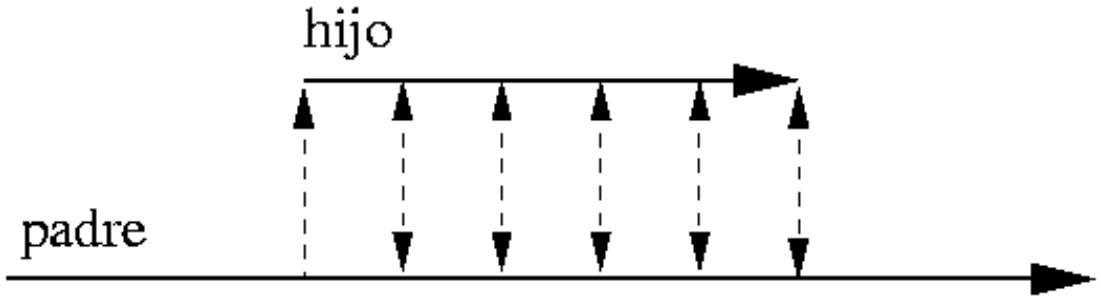
- Precondiciones configuran regiones de activación
- Arbitraje explícito en colisiones y vacíos de control

Combinación aditiva de estímulos



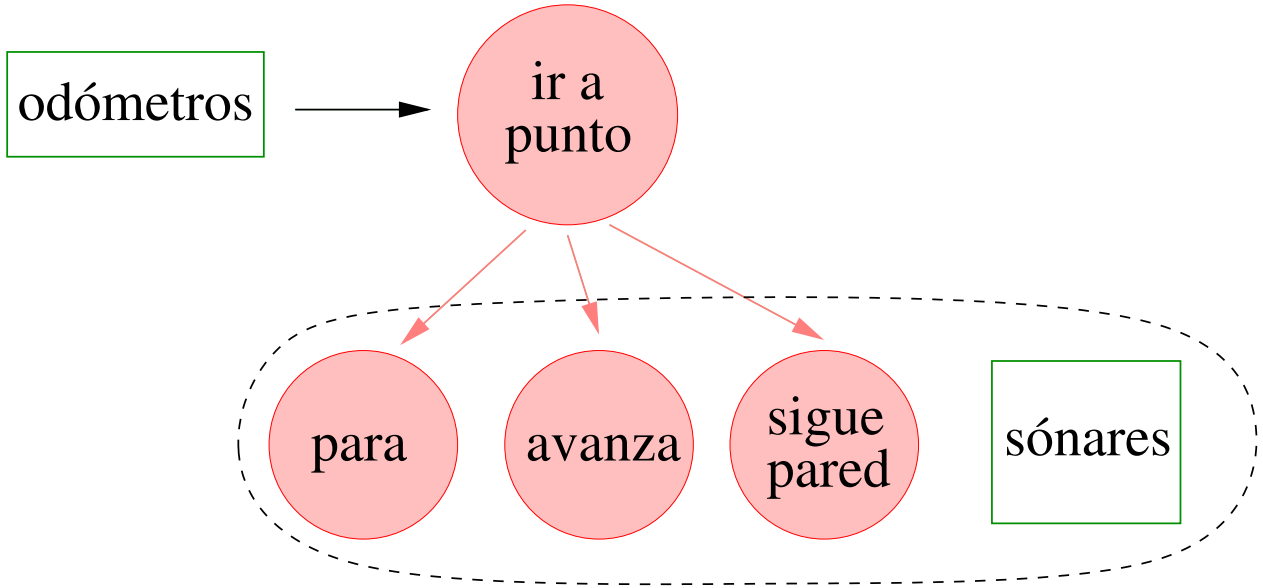
- Varios estímulos pueden disparar las precondiciones
- Inspiración etológica: percepción gestáltica

Monitorización continua



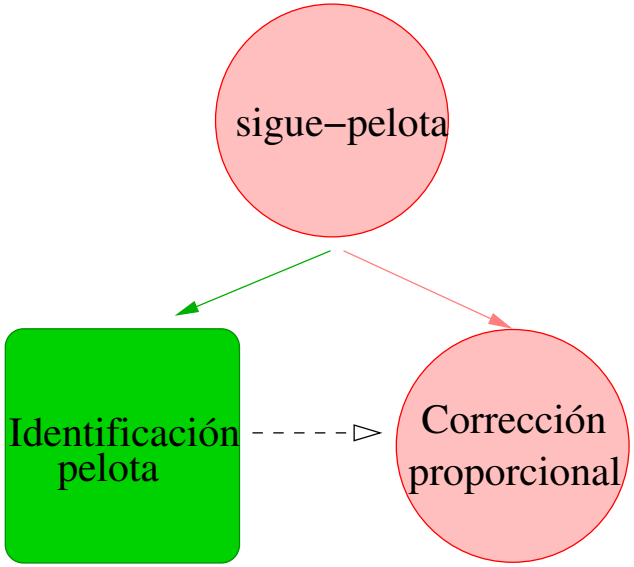
- El padre monitoriza iterativamente los resultados de sus hijos
- Puede cambiar su modulación o cambiar de hijos cuando le convenga
- Modulación continua con parámetros
- No descomposición funcional

Jerarquía como predisposición



- ALERTA $\neq$ ACTIVO
- Influencia de etología: estímulos clave, reflejos encadenados...
- Secuencias flexibles

Percepción en JDE



- Colección dinámica de estímulos (organizados en jerarquía)
- Carácter subsidiario de la percepción
- Coordinación con actuación: percepción situada, ¿cuándo?
- Sólo se buscan los estímulos que interesan (atención)

Plataforma software jdec

- Implementación en C de la arquitectura cognitiva JDE
- Nace con una tesis doctoral y la mantiene un grupo de desarrolladores
- ¿Donde conseguirla? `https://trac.robotica-urjc.es/jde/`
- Soporte amplio:
  - hardware del pioneer, láser, cuello mecánico
  - cámaras video4linux, cámaras firewire, imágenes fichero
  - simulador SRIsim, Stage
- Prácticas de varias asignaturas de informática en la URJC

Programación con esquemas

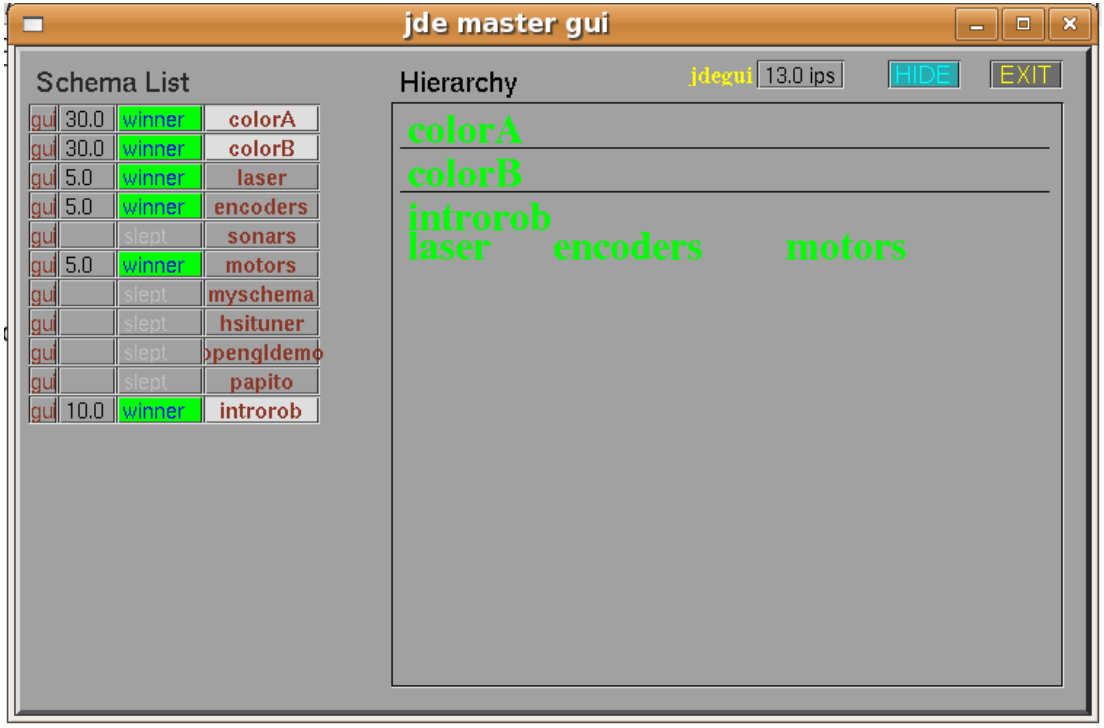
- Una aplicación robótica es un conjunto de esquemas concurrentes
- Cada esquema se implementa como hebra (`pthreads`)
- Iteraciones periódicas de control de ritmo controlado
- Comunicación por memoria compartida, variables
- Importación y exportación de variables (censo)
- Leer y escribir variables

Plugins

- Cada esquema se compila por separado, es un *plugin*
- La plataforma ofrece drivers, que también son *plugins*
- Fichero de configuración (jde.conf):
  - Cuál es la fuente de cada dispositivo (local o remota)
  - Qué drivers se van a usar, y qué esquemas básicos carga cada uno
  - Qué esquemas del usuario se van a cargar
  - Cuáles arrancar al principio
- Shell de texto para activar/desactivar esquemas e interfaces gráficas

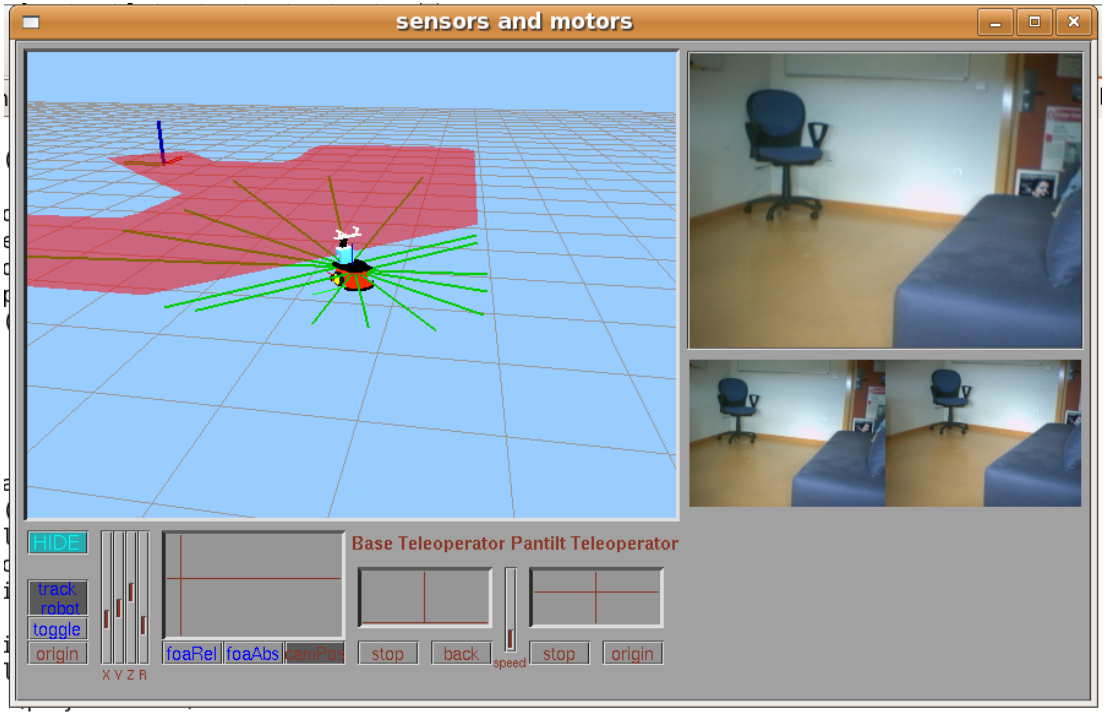
MasterGui

- Permite activar/desactivar esquemas y su interfaz gráfico
- Muestra el ritmo de cada esquema



SensorsMotorsGui

- Visualiza los sensores, cámaras, etc.
- Permite teleoperar los motores (base y cuello)

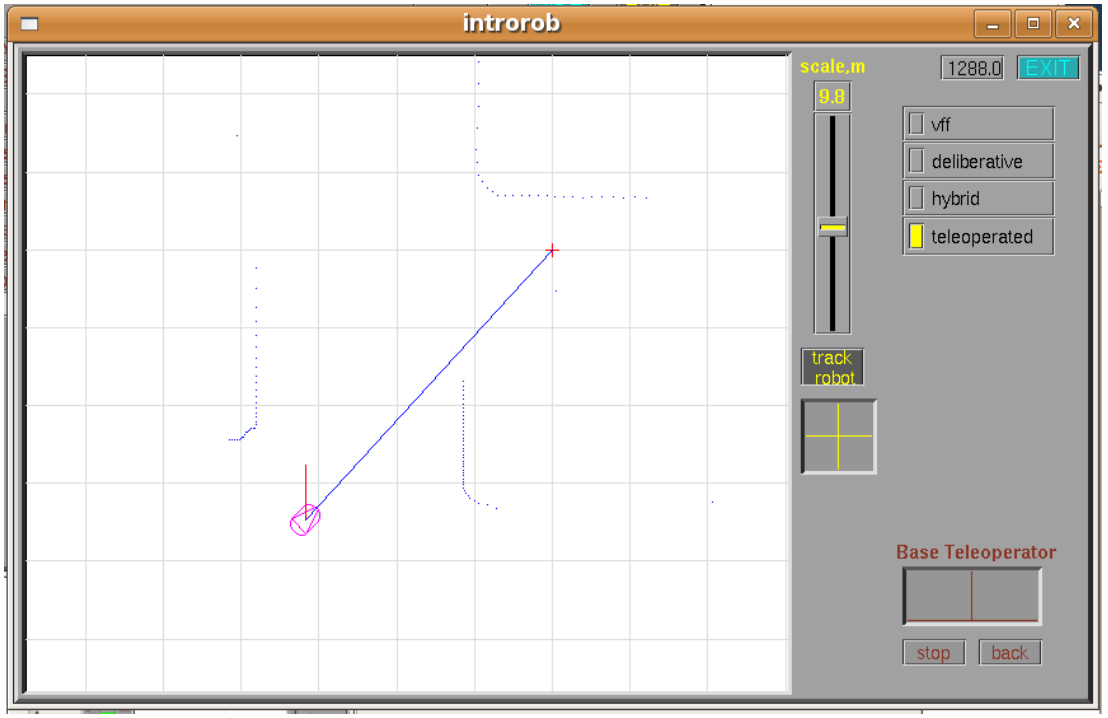


Esqueleto de un esquema

- Cada esquema mantiene su espacio de nombres
- schema_startup()
- schema_suspend()
- schema_resume(int *brothers, arbitration fn)
- schema_guisuspend()
- schema_guiresume()
- schema_cycle, ms

Interfaz gráfico de cada esquema

- Cada esquema se compila por separado, es un *plugin*
- Cada esquema mantiene su propia (y opcional) interfaz gráfica
- Visualización centralizada en ejecución pero distribuida en código



Hardware de referencia

- Base: motores, sónares, odometría
- Láser
- Cuello mecánico
- Webcam USB, firewire
- Ordenador portátil
- Puerto serie

Acceso al hardware

- Drivers son plugins que cargan esquemas básicos
- Variables sensoriales:

  laser

  us

  x, y, theta

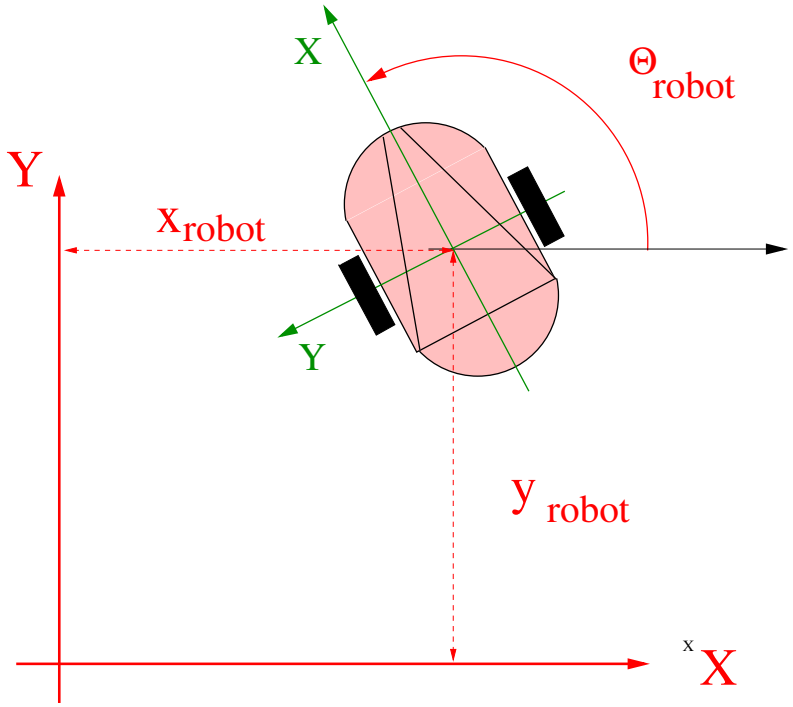
  colorA, colorB, colorC, colorD

  pan, tilt
- Variables motoras:

  v,w

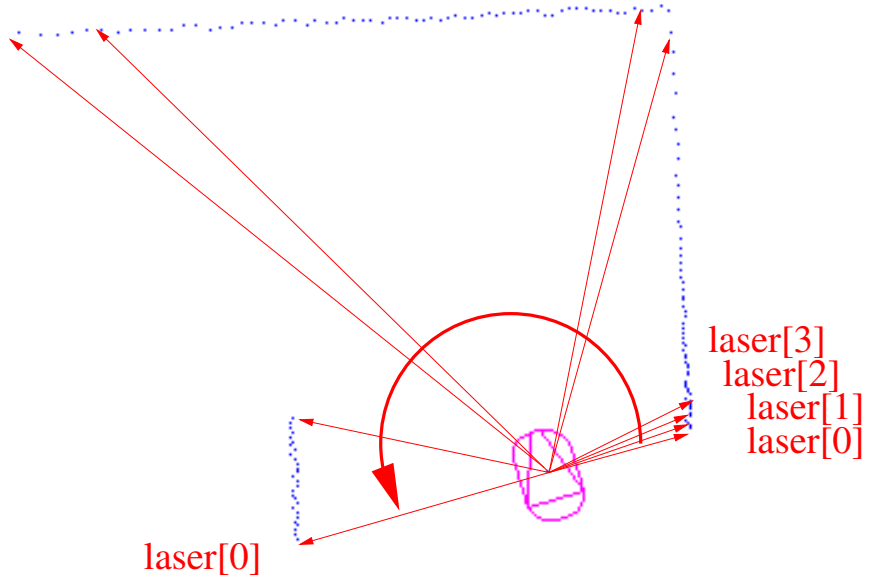
  latitude, longitude

Posición

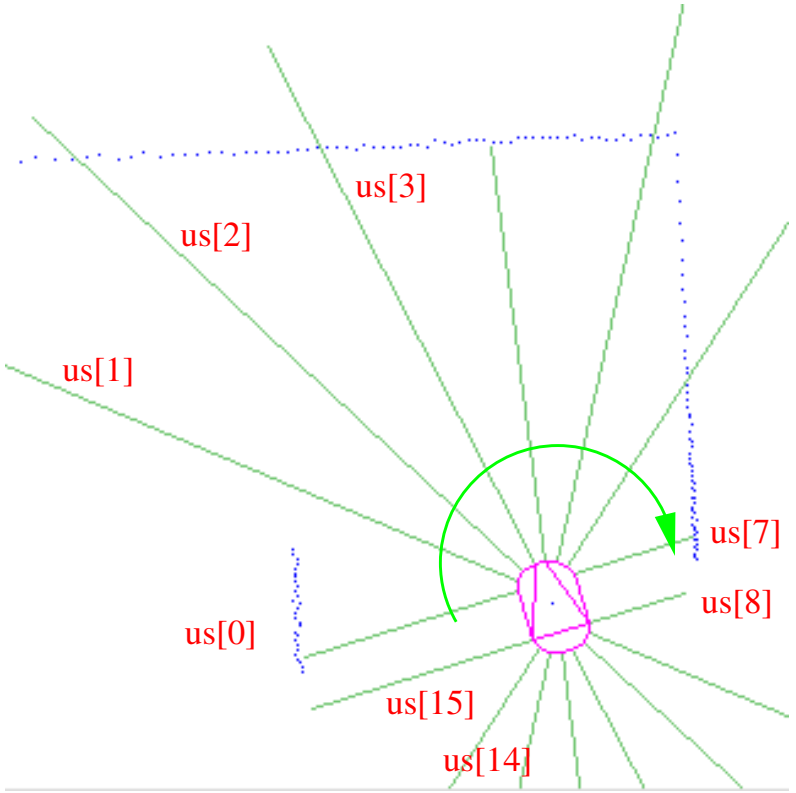


- x, y, theta (mm,radianes)
- pan, tilt (grados)

Obstáculos



laser



us

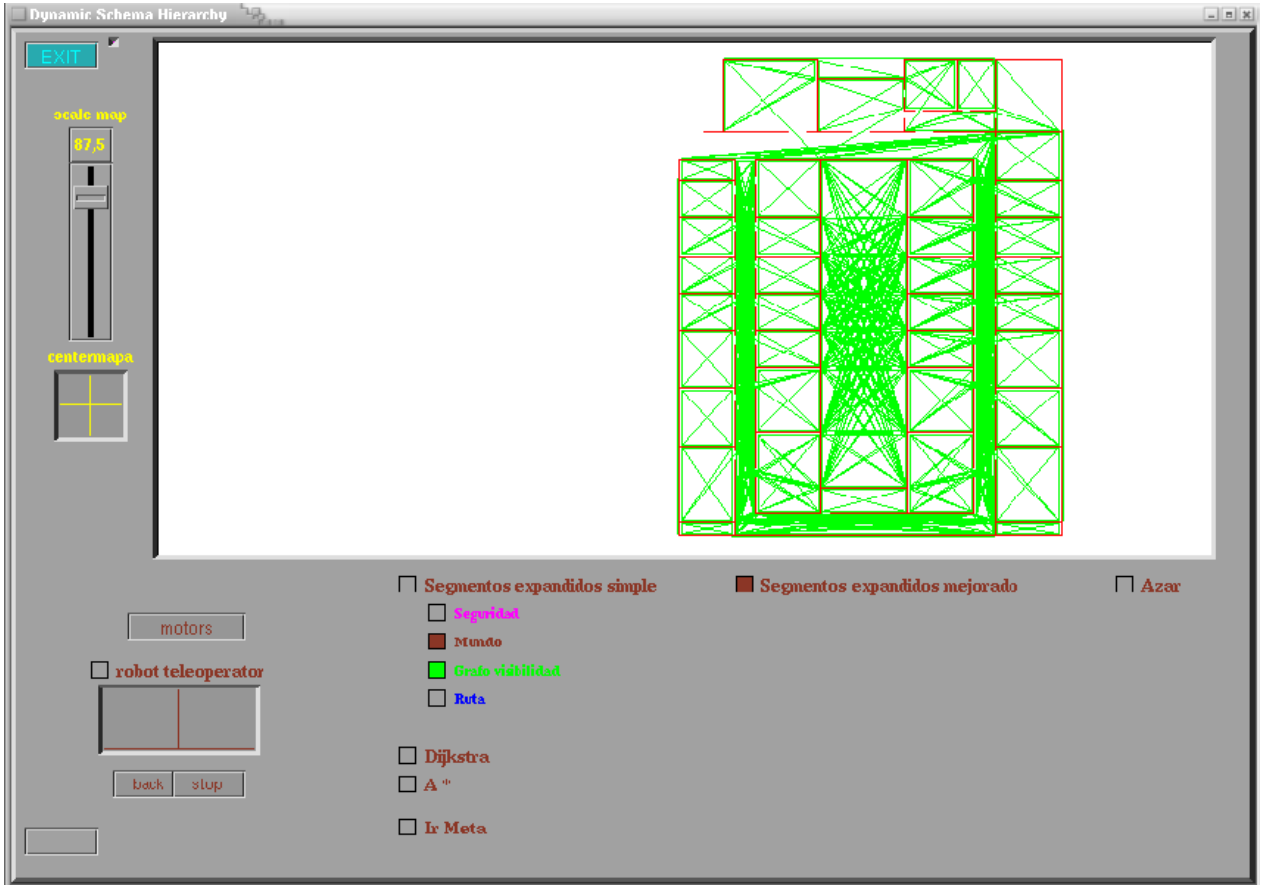
Imágenes

- colorA, colorB, colorC, colorD
- soporte local: file, video4linux, firewire
- soporte remoto oculo: llega una y pide la siguiente

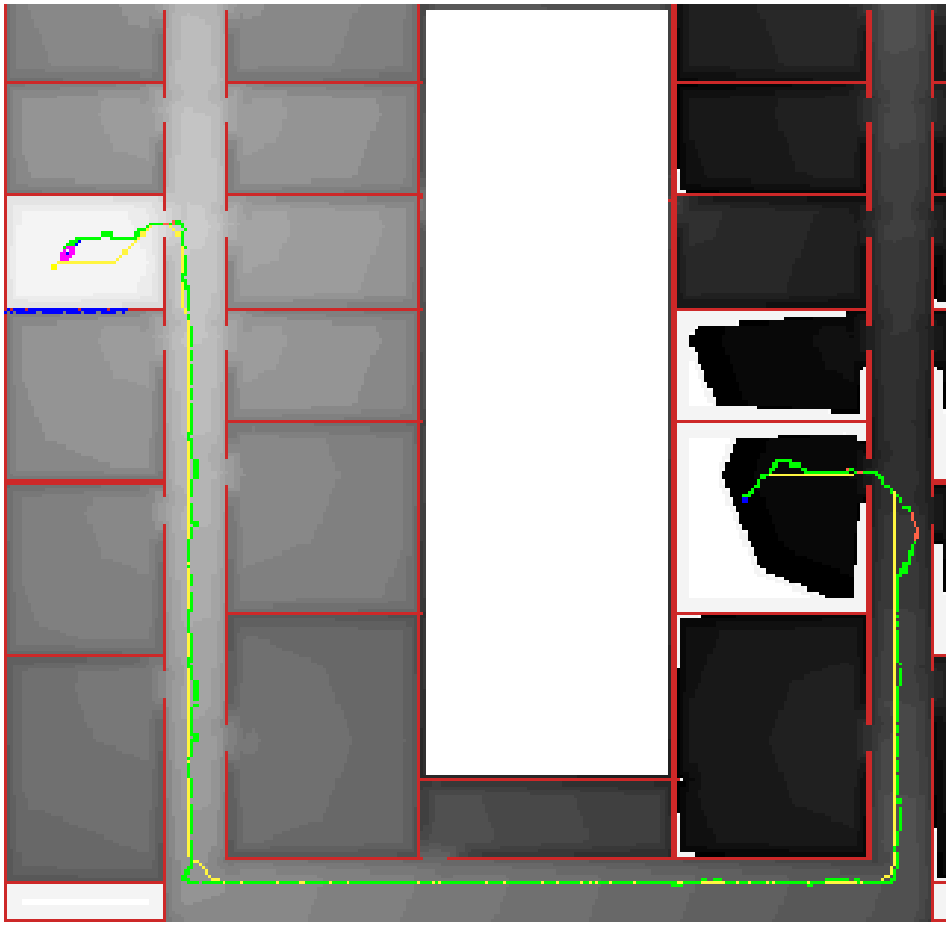
Ejemplos de uso

- Navegación
- Localización
- Taxias
- Comportamientos de seguimiento
- Esquemas perceptivos

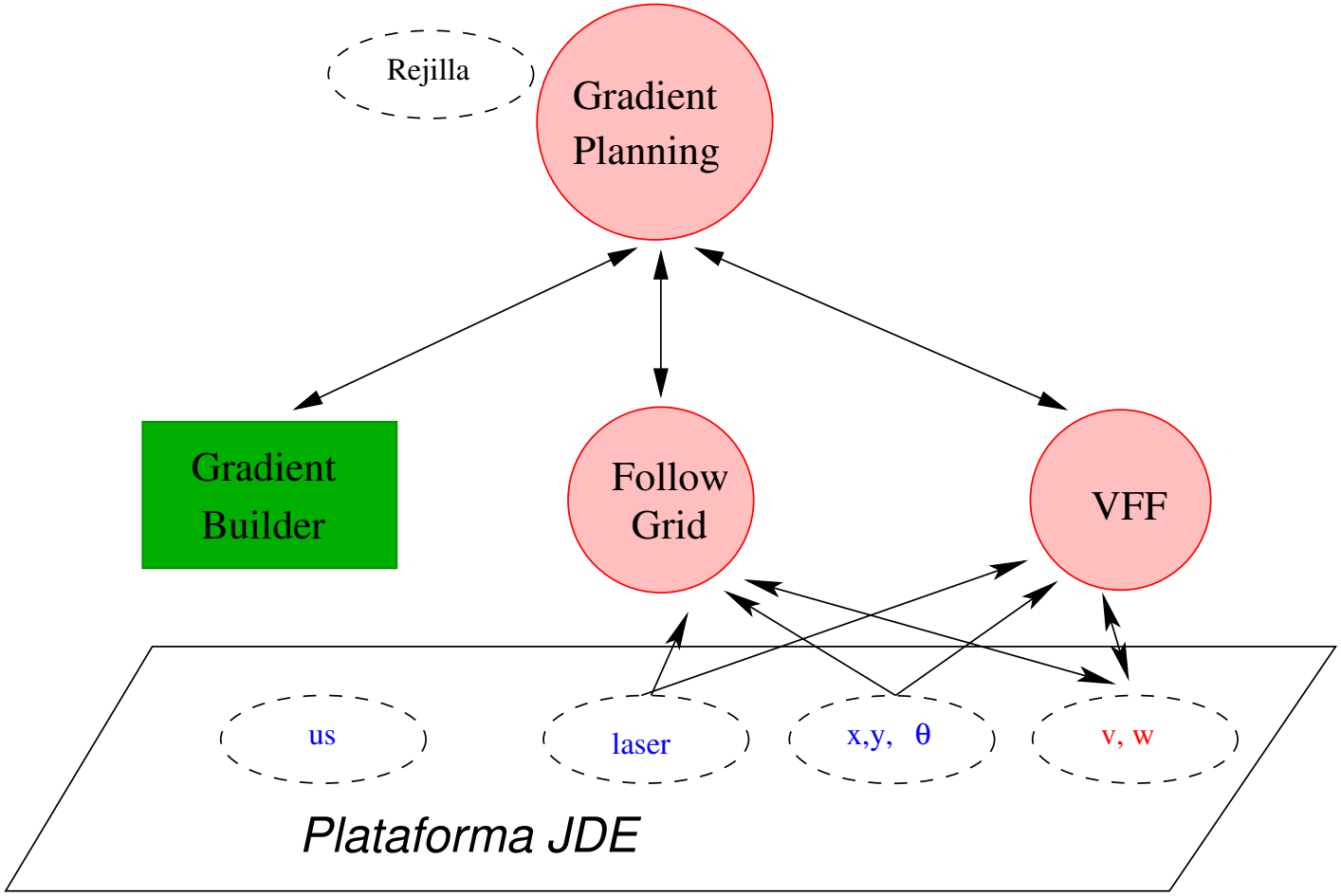
Navegación global por grafos de visibilidad



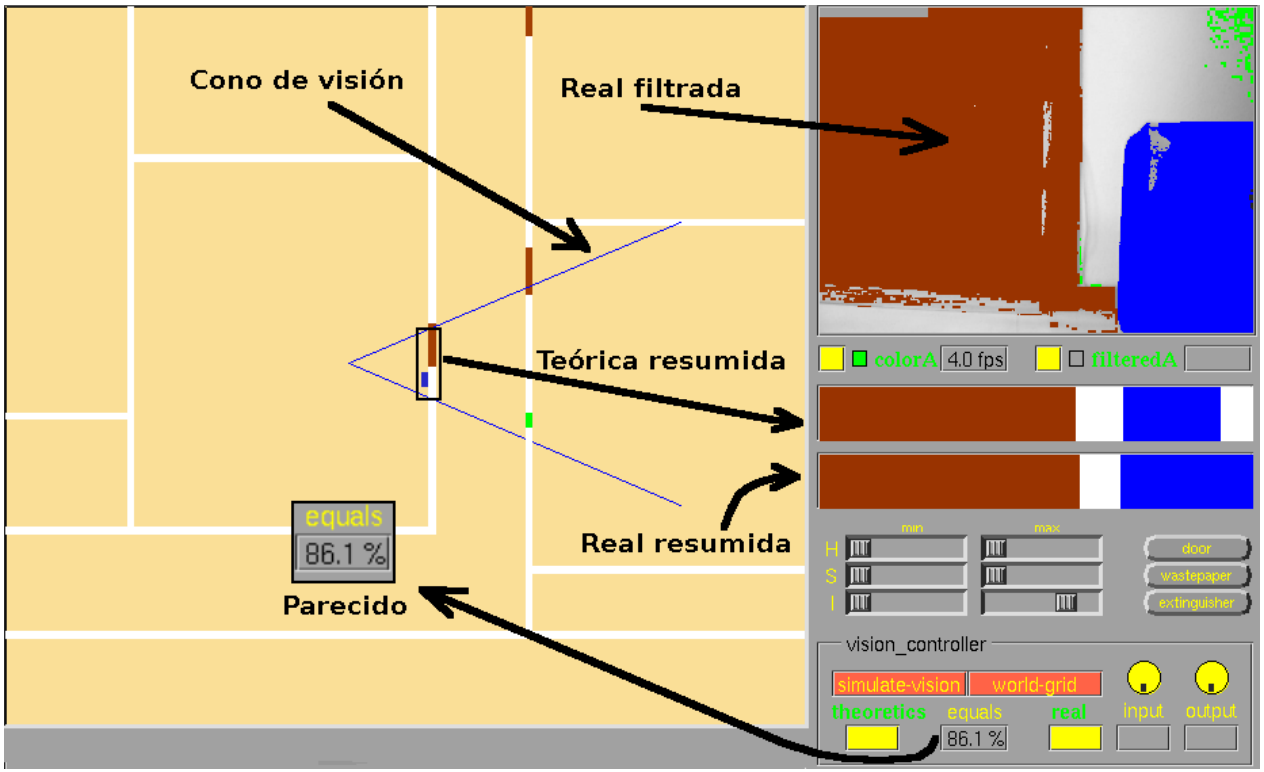
Navegación híbrida: local y global



¿Cómo?


Rejilla
Gradient Planning
Gradient Builder
Follow Grid
VFF
us
laser
x,y, θ
v, w
Plataforma JDE

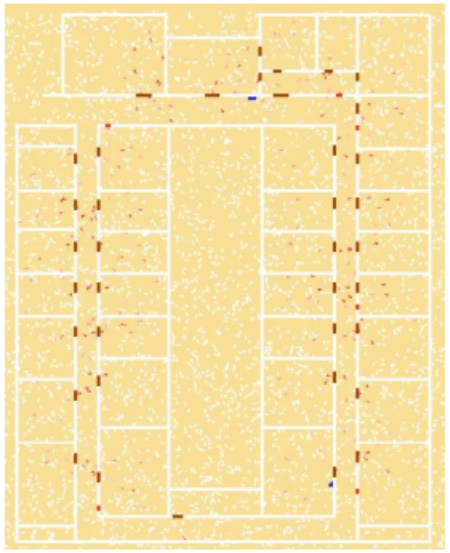
Localización visual



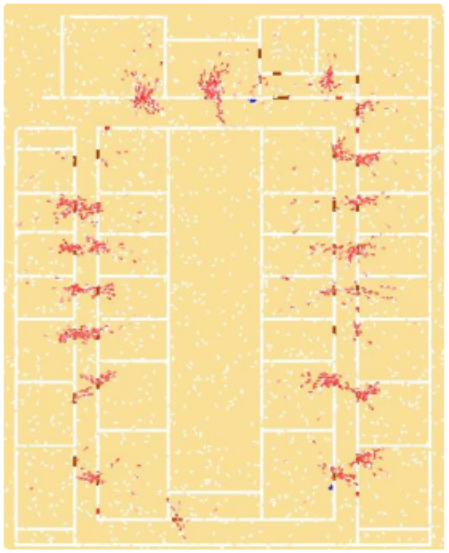
- Filtro de particulas: modelo observación, movimiento y remuestreo

Observación 1

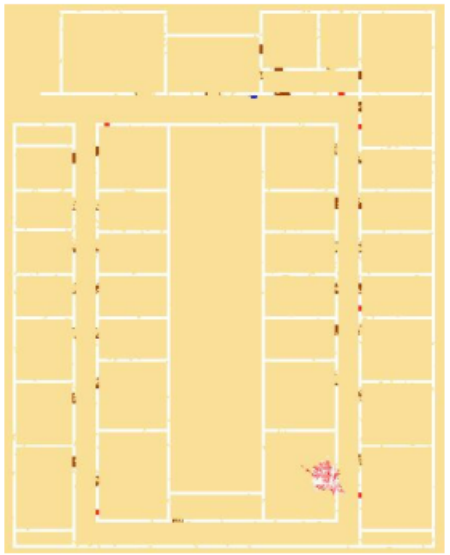
Observación 2



Distribución inicial
(uniforme)

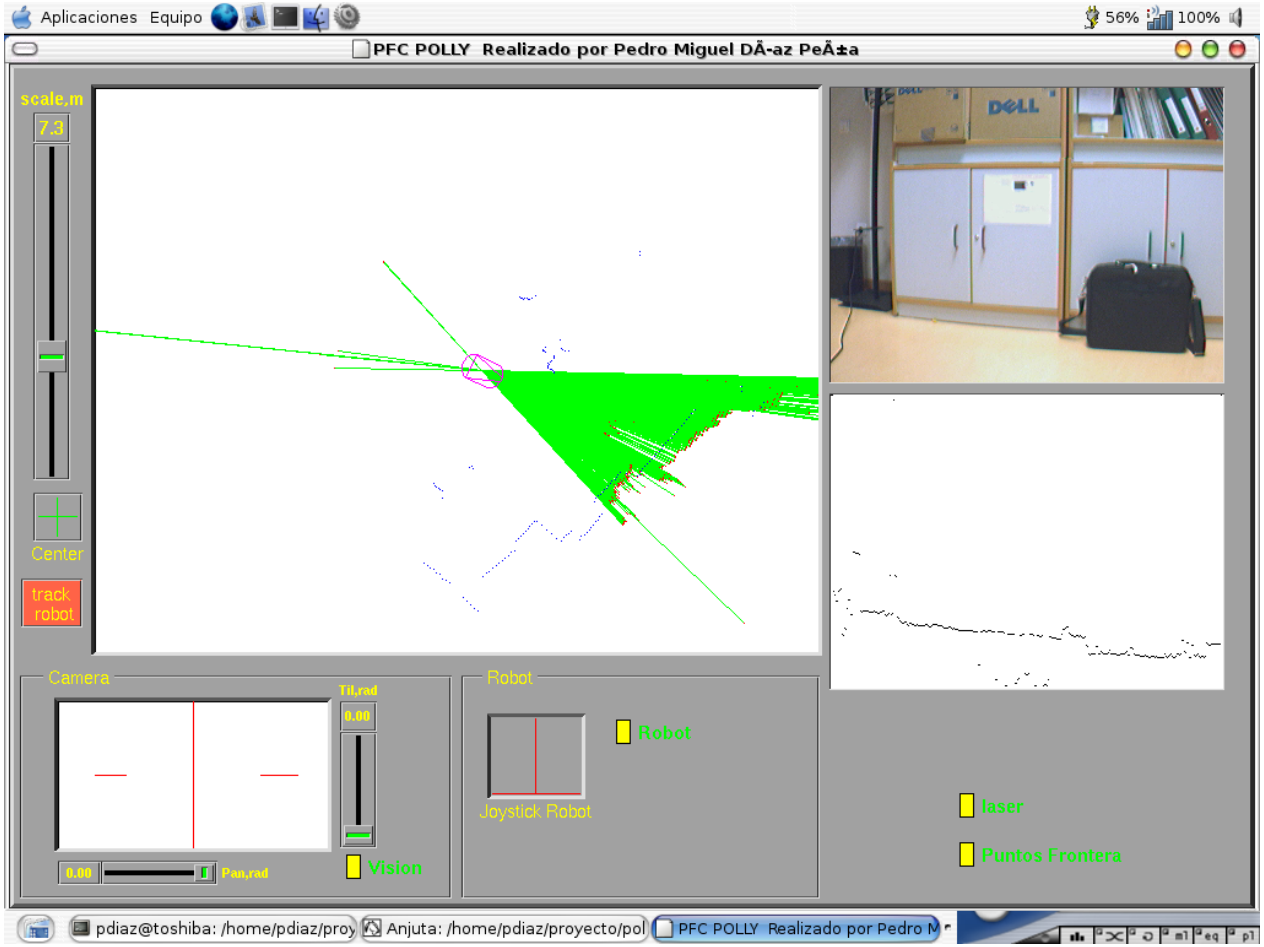


Acumulación en
zonas probables

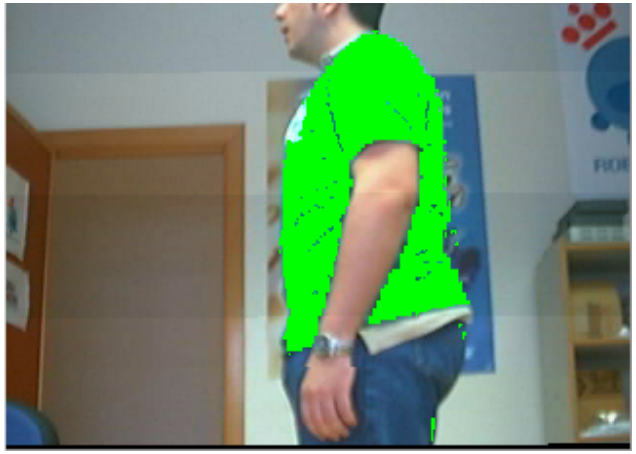


Localización final

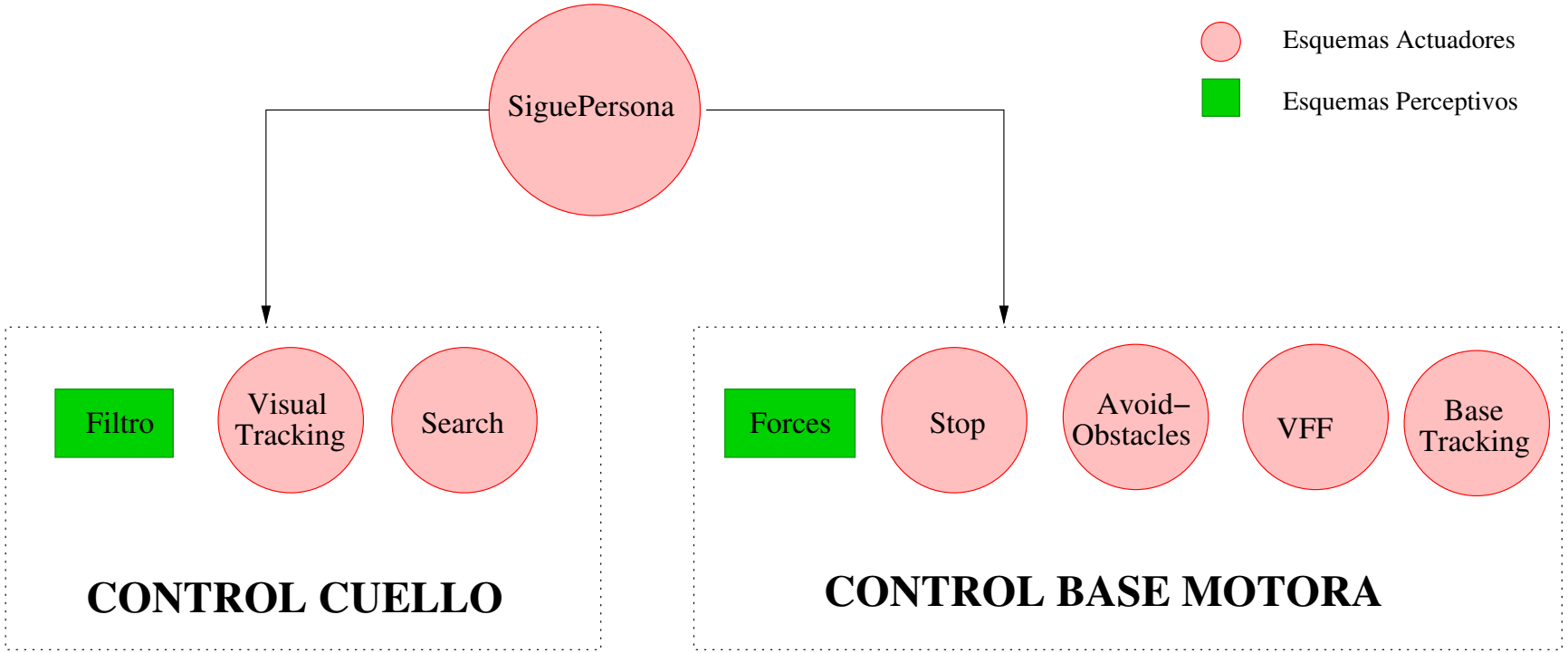
Navegación local con visión monocular



Comportamiento sigue-persona con visión

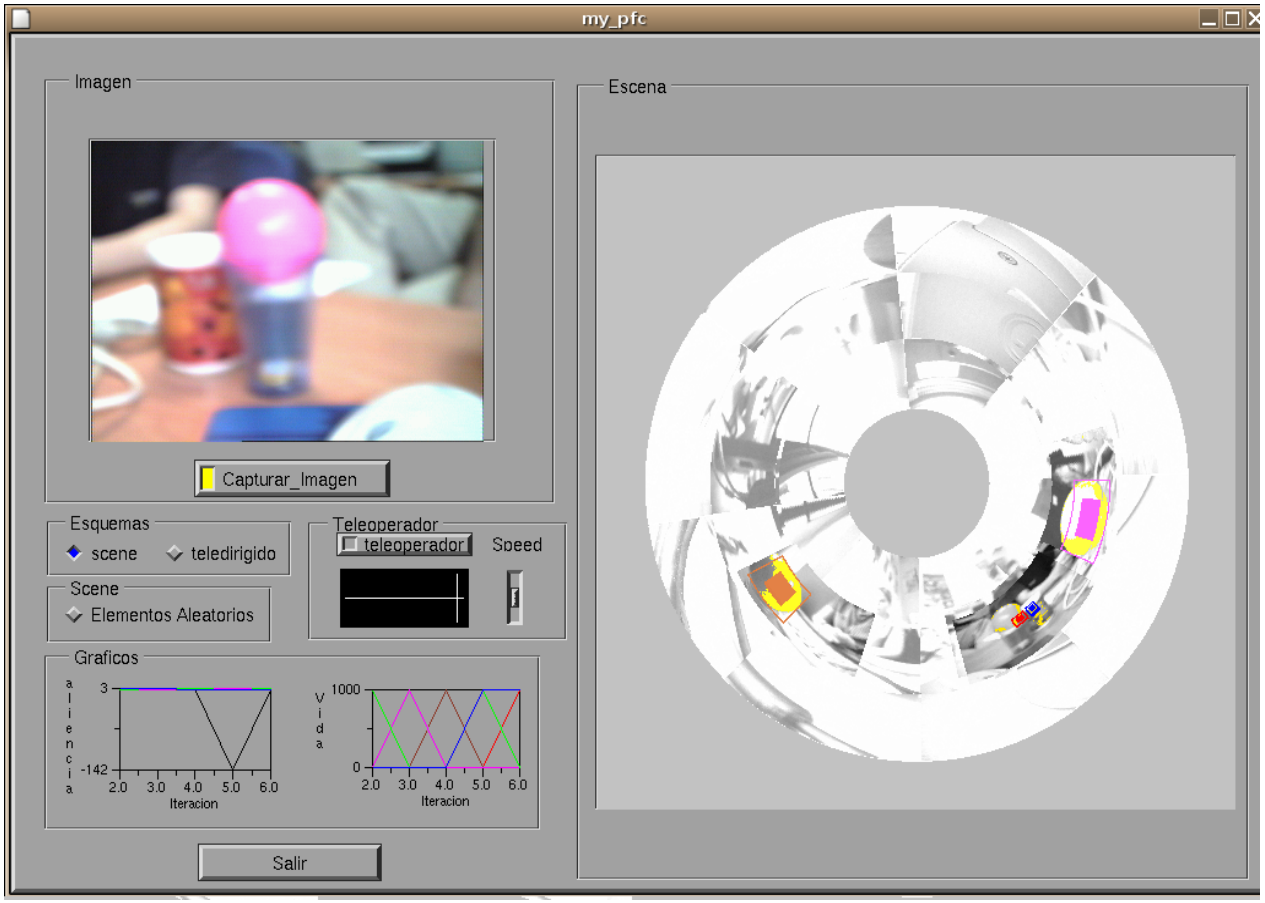


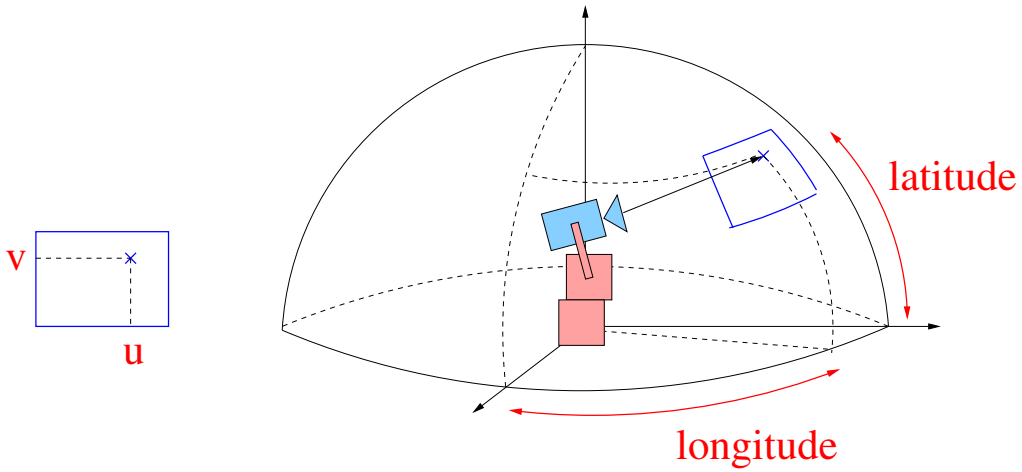
¿Cómo?


SiguePersona
Esquemas Actuadores
Esquemas Perceptivos
Filtro
Visual Tracking
Search
CONTROL CUELLO
Forces
Stop
Avoid–Obstacles
VFF
Base Tracking
CONTROL BASE MOTORA

Esquema perceptivo visualdiff

Atención visual



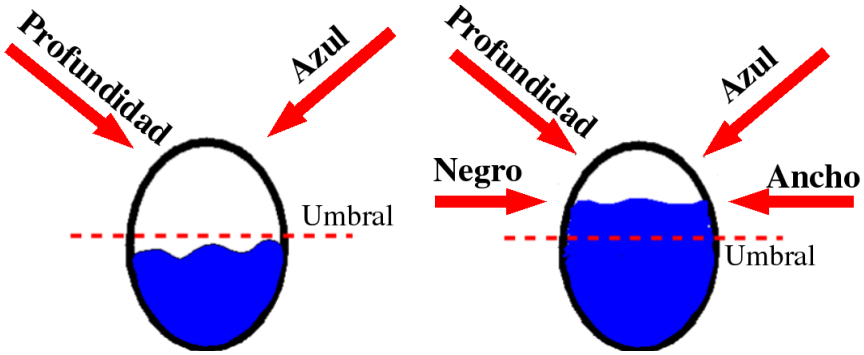


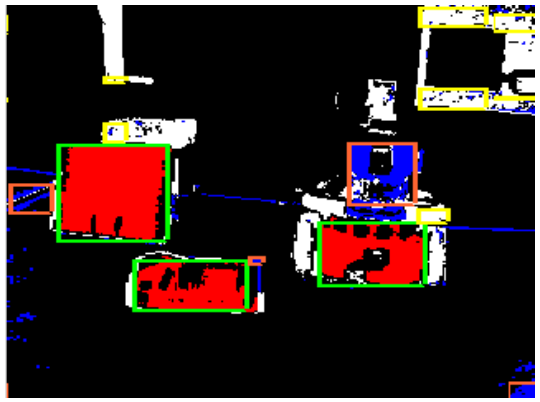
- La imagen de escena es más rica que la monocular
- Percepción activa: hay que mover el cuello mecánico
- Filtro para los colores relevantes
- Dinámicas temporales de saliencia y de vida
  - $liv(object, t) = liv(object, t-1) - \Delta L_{time}$
  - $liv(object, t) = liv(object, t-1) + \Delta L_{observation}$
  - $sal(fixp, t) = sal(fixp, t-1) + \Delta S_{time}$
  - $sal(fixp, t) = 0$

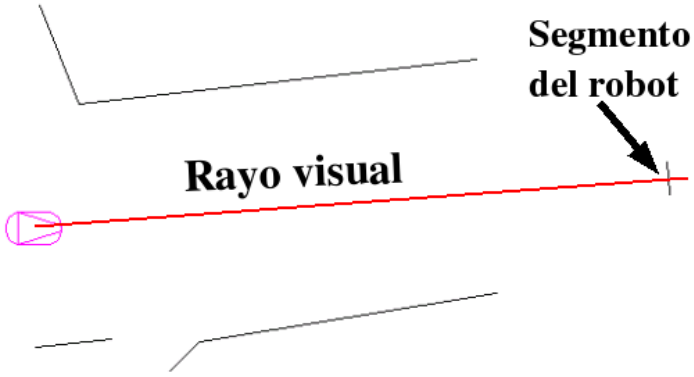
Reconocimiento de congénere

Suma heterogénea de subestímulos

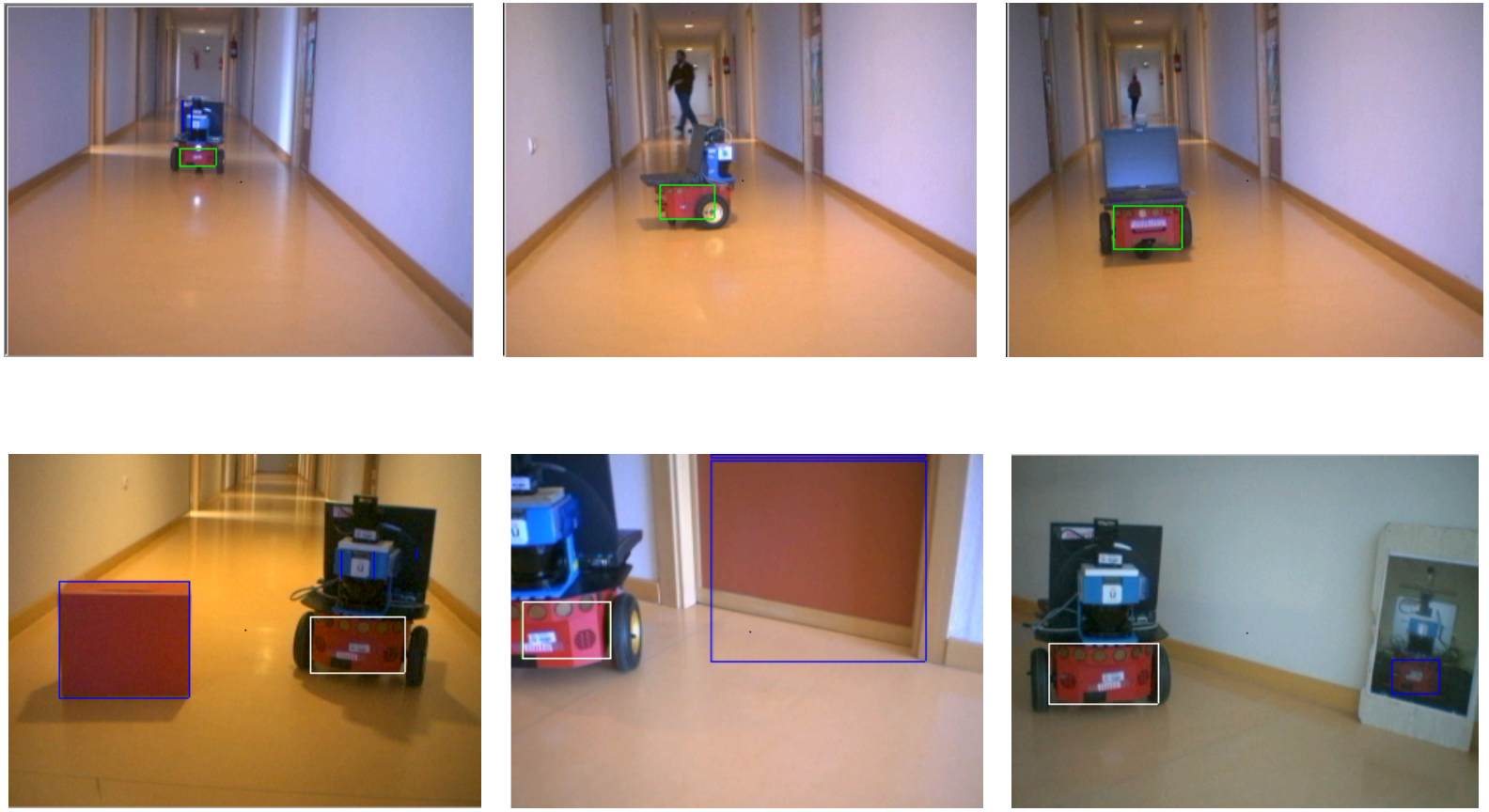
- mancha visual roja
- más ancho que alto
- mancha negra o azul encima
- sensación de profundidad



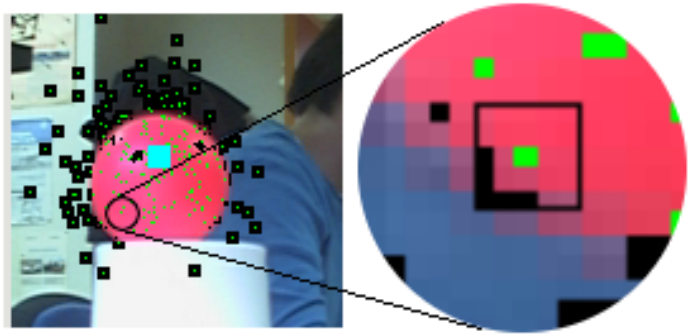




Es muy robusto y discriminante

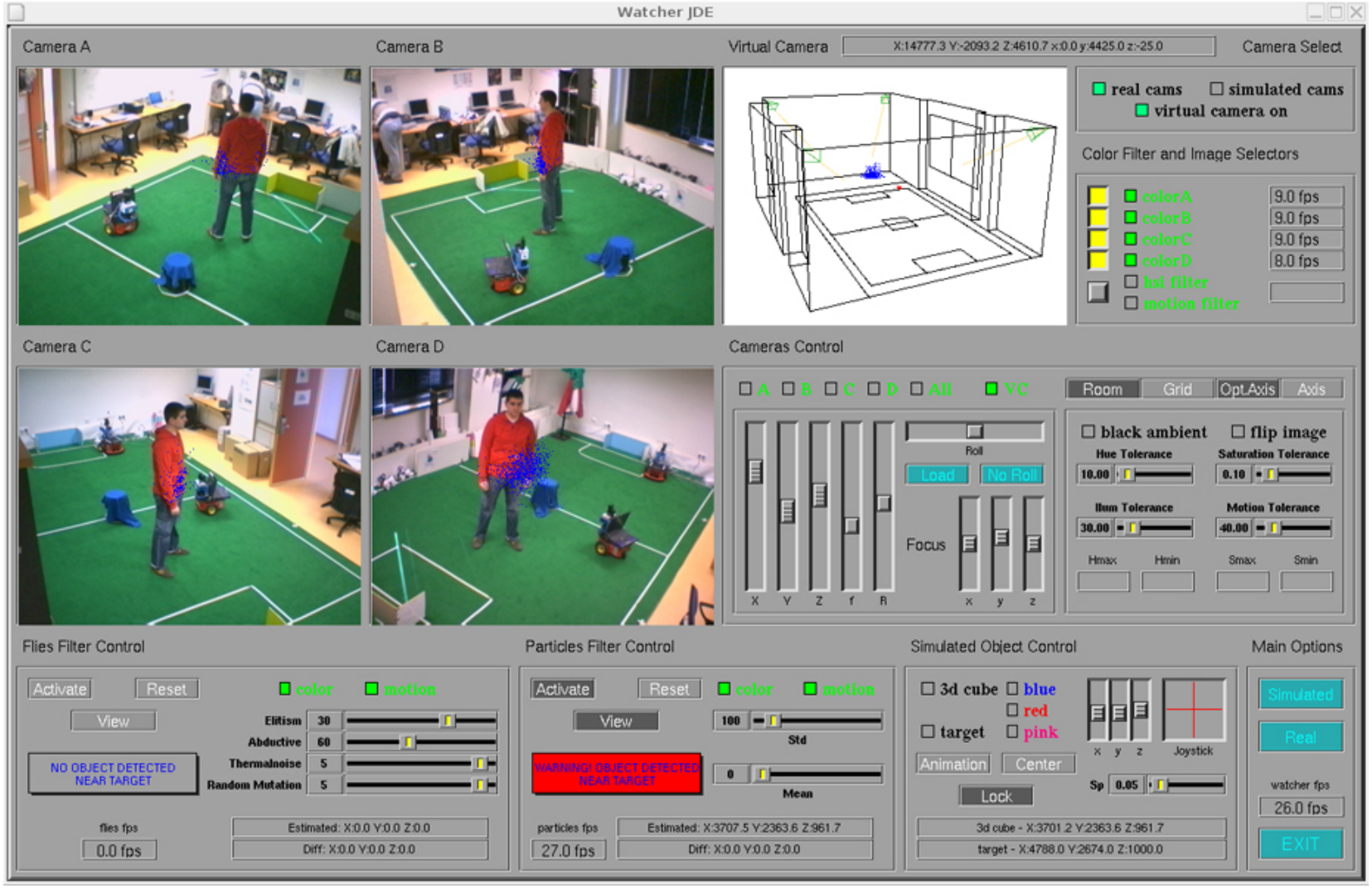


Seguimiento 3D con filtro de partículas



- Filtro de partículas: modelo observación, movimiento y remuestreo
- Población de partículas se mueve por el espacio de estados
- Cada particula tiene un peso
- Iterativo, eficiente
- Sólo se explota el color

Aplicación de seguridad



Conclusiones

- Generar comportamiento autónomo en robots móviles es complicado.
- El software da la inteligencia a los robots móviles.
- Las plataformas software simplifican la programación de robots.
- Las arquitecturas cognitivas guían la organización y permiten llegar a comportamientos más complejos.
- jdec es una plataforma solida y flexible basada en JDE.
  - plugins
  - drivers
  - censo de variables

- Líneas futuras: más esquemas, uso de mecanismos jerárquicos, GTK.