Implantación y Comparación de diversos Métodos para Control de Robots basados en Visión

Juan de Mata Domingo Esteve

9 de octubre de 1993

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

D Físicas

29 cms.

IMPLANTACIÓN Y COMPARACIÓN DE DIVERSOS MÉTODOS PARA CONTROL DE ROBOTS BASADOS EN VISIÓN

MEMORIA PARA OPTAR AL GRADO DE DOCTOR EN CC. FÍSICAS
PRESENTADA AL DPTO. DE INFORMÁTICA Y ELECTRÓNICA
FACULTAD DE CIENCIAS FÍSICAS
UNIVERSIDAD DE VALENCIA

Codirectores:

Dr. Marcelino Vicens Lorente

Dr. Joan Pelechano Fabregat

9 de octubre de 1993

Agradecimientos

Muchas son las personas e instituciones a las que debo reconocimiento por su ayuda o colaboración, directa o indirecta. Entre ellas debo citar:

- A mi director de tesis, Dr. Marcelino Vicens, y a mi codirector, Dr. Joan Pelechano, por su apoyo y consejo todos estos años. Igualmente a los miembros del LISITT y al resto del Dpto. de Informática y Electrónica de la Facultad de Física por su ánimo, ayuda y sugerencias. En particular, a los Drs. Jesús Albert, Francisco Ferri y Vicente Arnau.

- Al supervisor de mi trabajo en Edimburgo, Mr. Chris Malcolm, por su ingenio y sugerencias. Al Dpto. de Inteligencia Artificial de la Universidad de Edimburgo, por enseñarme gran parte de la poca Robótica que sé, y por abrirme los horizontes a nuevas vías de investigación.

- A la Comisión Interministerial de Ciencia y Tecnología (CICYT), por subvencionarme con una beca durante el tiempo de ejecución de este trabajo, tanto en España como en Escocia, y por subvencionar los proyectos ROB89-0285 y ROB91-0383, que se usaron en parte para financiar el material empleado.

- A D. Francisco Rosich, que ha mantenido los sistemas informáticos en los que he trabajado durante los últimos meses en el mejor estado de funcionamiento que los errores en el sistema de HP le permitían, y ha atendido amablemente mis numerosas peticiones.

- Al Dr. Salvador Moreno, por algunos comentarios y sugerencias muy interesantes sobre los métodos que estaba usando.

- Al Dr. Antonio Galbis, por dirigirme hacia la bibliografía adecuada del método de Newton modificado, y por su amistad.

- A todas las personas que, desinteresadamente, ponen al servicio de los demás investigadores sus conocimientos e información a través de los grupos de noticias de la red académica. Y más aún a los que producen programas de excelente calidad para aplicaciones tanto científicas como generales. En particular, quisiera mencionar a la Free Software Fundation, y a los autores de *GNUPLOT, xfig, TEX,* LaTeX, *xdvi, ghostview,* y sobre todo, *SNNS.*

- Un agradecimiento muy particular al Dr. Guillermo Ayala, que como Estadístico me indicó qué tratamientos debía dar a los datos, así como la fundamentación teórica de los mismos, y que como gran amigo me animó y orientó como el que más.

- A mi tío Salvador por su apoyo, su consejo y sus oraciones. Por la preocupación que ha tenido por mí desde siempre.
- A todo el resto de mi familia, que me ha apoyado y entendido.
- Y el último agradecimiento, aunque de ningún modo el menor, a mi madre, que ha cuidado de mí con cariño todos estos años, tanto aquí como en Edimburgo, permitiéndome así dedicar prácticamente todo mi tiempo a mi trabajo.

El autor de este trabajo ha sido subvencionado por una beca de la Comisión Interministerial de Ciencia y Tecnología (CICYT) en el Area de Robótica y Automatización Avanzada durante los años 1988 a 1992. La misma comisión concedió al grupo investigador en que se integra las ayudas para el desarrollo de los proyectos ROB89-0285 y ROB91-0306, las cuales han financiado la inmensa mayoría del material empleado.

A mi madre, con cariño.

A mi padre, aunque no pueda estar aquí para verlo.

Resumen

Esta tesis tiene por objeto implantar sobre un robot real varios métodos de control basados en visión y compararlos. El resultado fundamental consiste en la demostración de que transformaciones generales entre coordenadas visuales, tal como vienen dadas por las cámaras, y coordenadas motoras, tal como son enviadas a las articulaciones del robot, es posible sin el uso de sistemas de coordenadas diferentes (p. ej., cartesianas), lo cual elimina la necesidad de calibración de las cámaras (que basta con fijar en posiciones arbitrarias), reduciendo por tanto una fuente importante de errores. Hasta ahora, son escasísimas las implantaciones de estos métodos en robots reales; tales aplicaciones plantean problemas muy específicos que no aparecen en simulación, los cuales son discutidos, aportando posibles soluciones.

Los métodos que se han implantado son un método simple de regresión, que dará una pauta para la comparación de métodos clásicos con los otros, más modernos; la Teoría de la Red de Tensores, basada en los estudios sobre el funcionamiento del reflejo vestíbulo-ocular en el cerebelo humano de Pellionisz y Llinás; el Modelo de Computador aritmético Cerebelar (CMAC) de Albus, y varios modelos de redes neurales, entre ellos la Red Neural Autoorganizativa de Martinetz, Ritter y Schulten. Algunos de estos métodos poseen capacidad de aprendizaje, y otros no, así como posibilidad de realimentación en la señal visual de error.

El dispositivo experimental consiste en un robot semi-educacional (RT100) de poca precisión y repetibilidad aceptable (0.5 mm) y dos cámaras provistas de lentes de gran apertura (105°) para abarcar desde poca distancia toda el área de trabajo. El robot se controla por la puerta serie desde un ordenador tipo AT que contiene una placa digital de imagen que captura y procesa las imágenes de ambas cámaras.

Los experimentos consisten en la toma de medidas de pares de puntos del espacio de entrada (las coordenadas, en pixels, de ambas cámaras) y del espacio de salida (las cuentas para las articulaciones del robot), bien en puntos seleccionados, bien en gran número de puntos aleatorios que se usan para el aprendizaje de la función buscada en aquellos métodos que gozan de esta capacidad.

El resultado esencial es que es posible sobre un robot real, y por supuesto útil, una aproximación que, prescindiendo de coordenadas cartesianas y de todos los errores introducidos por el cálculo y la simulación, subsuma las transformaciones cámaras-cartesianas y cartesianas-cuentas (cinemática inversa) almacenandolas con aceptable precisión en cantidades razonables de memoria (no superiores a 2 Mb). Otro resultado, esta vez negativo, es la ausencia de convergencia en ninguno de los métodos de red neural intentados, las causas de lo cual se comentan y discuten.

Había, además, un objetivo científico en este trabajo, que es entender cómo funcionan en seres vivos los procesos de coordinación entre visión y control a través de sus implantaciones en un robot. A este respecto es importante haber conseguido, en particular en la Teoría de la Red de Tensores, una realización aquivalente a nivel de proceso de la información (no anatómico ni fisiológico) de un reflejo realmente existente.

Indice General

Apéndices:

Indice de Figuras

Indice de Tablas

Capítulo 1

Introducción.
Objetivos, resumen y guía de lectura.

Esta tesis versa sobre la implantación en un robot real de diversos métodos que puedan relacionar al nivel más bajo posible la visión y el control. Infinidad de trabajos se han publicado sobre aplicaciones robóticas de sistemas de visión que tratan de extraer información de más o menos alto nivel de la escena a partir de las imágenes tomadas por una o más cámaras, e igualmente una multitud sobre cómo la información generalmente geométrica provista por los sistemas de visión puede ser usada para guiar el robot y hacer que ejecute una secuencia de acciones, fija o descubierta por el propio programa (planificadores de tareas). No se cita aquí bibliografía, porque de ellos se dará cuenta, al menos sucinta, en el capítulo destinado a describir el estado de la investigación.

Lo que aquí se desea apuntar por el momento es que este trabajo no pretende una aproximación de este tipo, sino que trata de relacionar la información de más bajo nivel suministrada por un proceso visual muy elemental con la información motora, también a bajo nivel, que controlará la posición de un brazo robot. La idea que subyace aquí es que el módulo de sotware más hardware construído, juntamente con su aplicación en un entorno dado, debería ser una capa inferior (no un eslabón) que ejecutase su tarea de modo eficiente cuando fuese requerido por otros módulos o por estímulos apropiados del ambiente captados por los sensores. Tal tarea es llevar el brazo robot al punto donde otros módulos deciden que las cámaras deben *verlo*, independientemente de que conozcamos o no su posición en coordenadas cartesianas (o en algún otro sistema exterocéntrico).

Demostrar que tal módulo puede funcionar sobre un robot real, que existen varios métodos para implantarlo, y comparar estos métodos entre sí es el objetivo inmediato de esta tesis. Este objetivo responde a otros dos, más esenciales: uno de tipo científico y otro práctico.

El objetivo científico es entender hasta donde sea posible una de las capacidades de varios seres vivos consistente en coordinar los movimientos de ciertas partes de su cuerpo (cabeza o extremidades) de acuerdo a la información visual que reciben. Este es uno de los reflejos más necesarios para el correcto funcionamiento y la supervivencia de los seres, y es por tanto uno de los primeros que deberá programarse

en cualquier robot autónomo.

Por otra parte, el objetivo práctico consiste en preparar el sistema para una posible aplicación futura de estas técnicas a robots de ensamblado, que les permitan ejecutar las tareas más básicas de posicionado del brazo en el punto correcto respecto a una pieza, y de movimiento por el espacio libre evitando la colisión con piezas. Para ser precisos, no deberíamos hablar aquí de piezas, sino de aspectos visuales relevantes de las mismas (esquinas, bordes o similares), que en cualquier caso deberán ser detectados por el sistema de visión. Lo importante es que no necesariamente tendría que ejecutarse todo el proceso de modelización geométrica de la escena para llevar a cabo tales tareas.

Respecto al objetivo científico mencionado, el método que ha pretendido usarse, al menos en la implantación de la Teoría de la Red de Tensores, es el conocido como "ingeniería inversa" (reverse engineering) que consiste en observar un sistema que realice la tarea que deseamos ejecutar, funcionando correctamente. A continuación averiguar cómo la ejecuta, y por último reproducir dicha tarea. Esta es también, en último extremo, la metodología usada en las redes neurales, con la salvedad de que muchas veces los programadores, armados de muy escasos conocimientos de anatomía y neurofisiología, extrapolamos indebidamente los resultados, creyendo ver analogías de organización y funcionamiento donde sólo hay una reproducción, casi siempre parcial y limitada, del comportamiento del sistema biológico real. Sería mucho más apropiado hablar de que nuestras redes funcionan a nivel computacional (es decir, vistas como dispositivos de proceso de la información) de modo funcionalmente similar a los sistemas reales (es decir, dan salidas análogas ante entradas análogas, provocando, por tanto, un comportamiento similar). Sobre la complejidad real de los sistemas de neuronas puede encontrarse una estimación para el sistema visual humano en [Mir93]

Respecto al objetivo práctico, la idea es que es resultado final sea un módulo fiable, implantable en robots diferentes sin cambios drásticos, que reciba sus entradas directamente del proceso visual elemental destinado a localizar el punto terminal del brazo, y de otros módulos que le indiquen dónde (en el espacio de cámaras) debe ser visto, y que mande sus salidas directamente al sistema de control de cada motor del robot. En principio, y fundamentalmente por problemas técnicos y de tiempo, hemos decidido usar como nivel mínimo las cuentas de las articulaciones, dejando que los controladores PID incorporados en cada una se aseguren de que el motor se sitúa en la posición correspondiente al número de cuentas que se le envían, lo cual siempre ocurre, en más o menos tiempo. A los efectos de demostración de métodos que nos interesan, esto no tiene ninguna influencia, pero si dispusiésemos de hardware de visión capaz de localizar la posición del brazo en tiempo real cabría pensar en enviar las señales de los torques para cada motor, si los controladores PID resultasen demasiado lentos. Esta posibilidad se comentará en el último capítulo. Es interesante hacer notar que a este nivel de cuentas eliminamos la necesidad de que el robot sea preciso, en el sentido en que se le suele entender hoy: que dada una posición

cartesiana, el punto terminal del brazo se sitúe en ella con error menor que cierto valor garantizado por el fabricante. Esto se sustituye por el requisito de repetibilidad: que la misma señal (en cuentas) lleve el brazo al mismo punto, independiemtemente de la posición anterior o de la carga. En robots baratos, como el que usamos, la precisión es muy pobre, pero sin embargo la repetibilidad es razonable.[1]

Una vez sentados la filosofía y objetivos de la tesis daré ahora una visión de cada una de las partes de la misma.

En el capítulo 2 se exponen algunas ideas nuevas sobre qué ha sido la Robótica hasta ahora y cómo se relaciona con la Inteligencia Artificial clásica: fundamentalmente, la construcción de modelos y el ciclo recoger información-pensar-actuar, que lleva a una descomposición funcional y vertical de las tareas. Esto se contrapone a otra tendencia reciente (la Robótica Comportamental), que partiendo de una visión diferente de la inteligencia, pretende obtener descomposiciones horizontales de las tareas mediante la relación al nivel más bajo posible de percepción y acción. Este capítulo está íntimamente ligado a la inspiración de la tesis, y tiene consecuencias en su ejecución y en la elección de métodos y forma de programar, pero no es imprescindible desde un punto de vista técnico; ha sido incluído también para difundir al máximo una investigación todavía minoritaria.

El capítulo 3 define con detalle cuál es la tarea que se desea realizar, primero al nivel computacional (viéndolo como un sistema de proceso de la información), lo cual permite definir cuál es la información de entrada, cuál la de salida, qué transformación se efectúa y qué restricciones aparecen. Después hay un análisis del nivel algorítmico (cómo se puede ejecutar la computación definida en el nivel superior) en donde aparecen las diferencias entre los métodos que propongo, cada uno de los cuales cumple sólo parcialmente las especificaciones formuladas. Y finalmente una descripición del nivel inferior (el de implantación) en el que se describen los dispositivos físicos utilizados y por qué se han seleccionado (es decir, por qué son apropiados para implantar sobre ellos los algoritmos escogidos). En el capítulo 10 se propondrá una extensión a ello, explicando posibles mejoras en la implantación con dispositivos diferentes que permitiesen, p. ej., una paralelización de alguno de los algoritmos.

El capítulo 4 es una exposición de varios trabajos realizados hasta ahora en el campo del control basado en visión. Se comentarán a nivel general los trabajos de la aproximación clásica, y mucho más en particular los más modernos entre los que plantean exactamente el mismo problema: cómo conectar visión y control en un sistema compuesto por un robot y una o dos cámaras. Se observan en ellos varios planteamientos ligeramente diferentes, pero con una filosofía común, en sus aciertos y errores. Bastantes de ellos trabajan sobre robot simulado, lo cual, como veremos, obvia algunos problemas cruciales que aparecen en robots reales, y prácticamente

[1] no peor de 0.5 mm, según el fabricante. He comprobado en la experimentación que esto es cierto

todos ellos usan coordenadas cartesianas como paso intermedio.

Los capítulos 5,6 y 7 describen con detalle cada método. Indican cuál es su fundamento matemático o biológico, cuál es su algoritmo, cómo se han programado, cuántos recursos emplean (esencialmente, memoria y coste computacional) y sobre todo, qué capacidades tienen: si el conocimiento es introducido o adquirido por aprendizaje, si tienen o no la posibilidad de realimentación en la señal visual, etc. Estas características, además de los resultados que se obtengan sobre el robot real, serán también importantes a la hora de la comparación.

El capítulo 8 explica las pruebas con diferentes métodos de redes neurales, en particular explica con detenimiento la Red autoorganizativa de Ritter y Martinetz, y muchos otros métodos que se emplearon sin éxito. Se piensa que ello es debido a que el tamaño y la complejidad del problema son excesivos, no para ser abordados por una red determinada, sino para encontrar dicha red.

El capítulo 9 presenta cuál ha sido la eficacia de cada método cuando es usado para mover el robot real llevándolo a cada uno de los puntos de un conjunto seleccionado, distribuído regularmente por todo el espacio de trabajo. Se da un análisis estadístico que explique qué método resulta mejor global y localmente.

El capítulo 10 extrae las conclusiones, la principal de las cuales es demostrar, cómo dijimos en la introducción, que es posible implantar sobre un robot real este tipo de métodos, que presentan algunas ventajas sobre los métodos clásicos de calibración de cámaras más cinemática del brazo, y que dan un modelo plausible de ciertos fenómenos biológicos. También se subrayan los inconvenientes de cada uno de los métodos, como el largo tiempo de aprendizaje, o la falta de precisión sin realimentación. Finalmente, se comentan ciertas mejoras que, bien por exceder el alcance de lo previsto para esta tesis (como la conexión con otros módulos de un sistema de visión), bien por no disponer de material adecuado (como placas para visión en tiempo real), no han podido programarse, pero que son una línea de trabajo abierta para el futuro.

El lector interesado en conocer las nuevas aproximaciones a la Robótica, y cómo estas influyen en una tarea práctica, deberá comenzar por el capítulo 2; en caso contrario, puede omitirlo. El capítulo 3 está destinado a aquellos que hayan efectuado experimentos similares, o quieran reproducir éste. El capítulo 4 puede ser omitido por aquellos lectores que tengan conocimiento del estado de la cuestión, o por los que busquen métodos generales para implantar transformaciones entre espacios de entrada y salida (construcción de funciones desconocidas) independientemente de su aplicación concreta. Los capítulos 5, 6, 7 y 8 pueden ser leídos por separado por los lectores interesados sólo en alguno o algunos de los métodos, quizá con destino a su uso en otras aplicaciones. El capítulo 9 es útil para los roboticistas con intereses similares, o para los que piensen en usar en alguna aplicación robótica real alguno de estos métodos, de cara a una elección del más apropiado para su caso. El capítulo 10 de conclusiones es, obviamente, importante y deberá ser leído en cualquier caso.

Capítulo 2

Robótica y su relación con la Inteligencia Artificial.

En este capítulo se describirá brevemente el estado de las tendencias en la investigación Robótica general, y en los robots de ensamblado en particular. Esto será útil para entender por qué este proyecto fue escogido, y para qué puede ser útil.

Los robots se han usado hasta ahora en instalaciones industriales esencialmente como robots de montaje, soldadura o pintura de maquinaria (coches, etc.). Su característica es la repetición de las acciones preprogramadas sin variación, o a lo sumo con el uso de sensores cuya información detiene el robot en caso de colisión, o ajusta la fuerza o la inclinación del brazo. Una serie de problemas importantes relacionados con el control de bajo nivel (teoría de control de sistemas dinámicos, identificación, modelización, estabilidad, etc.) han sido formulados y resueltos para su uso en estos sistemas, y ello ha permitido una mejora técnica importante. Véanse ejemplos en [Bea82]. No obstante, tal clase de robots carece por completo de cualquier comportamiento que podamos llamar inteligente, y en este sentido se acercan más a las máquinas-herramienta que a la moderna concepción de un robot.

Un avance sobre ellos lo representan los sistemas para la clasificación o el ensamblado de piezas en las que éstas llegan al entorno de trabajo en posiciones u orientaciones variables, o con defectos. Aqui ya tenemos un cierto comportamiento inteligente, si bien la mayoría de las veces preprogramado, no adaptativo. (es decir, no mejora su eficacia con la práctica). En la experimentación y aplicación real de estos sistemas surgen problemas extraordinariamente difíciles relacionados, entre otras cosas, con la incertidumbre y el ruido. Las piezas para ensamblar no son perfectas, de modo que se debe tener en cuenta cierta tolerancia cuando el brazo robot tiene que asirlas y ensamblar una con otra. Por otra parte, el ruido, inevitablemente unido a los datos de cualquier medida tomada por cualquier sensor, e inherente a tales datos, provoca también indeterminación. Las técnicas de ensamblado deberían ser suficientemente robustas como para enfrentarse a estas fuentes de incertidumbre y ejecutar un ensamblado correcto a pesar de ellas.

Es obvio que un sistema del tipo apuntado en el apartado anterior requiere un cierto grado de inteligencia, y es por ello el momento de clarificar qué entendemos por inteligencia, y qué queremos decir con la expresión "un cierto grado". Fre-

cuentemente se atribuye a los humanos el monopolio de la inteligencia, y esto es cierto si sólo consideramos como tal al razonamiento de alto nivel, el uso de lenguaje simbólico, y tareas similares. Pero no deberíamos olvidar (y la Inteligencia Artificial clásica lo ha hecho frecuentemente) que todas estas capacidades se asientan en, y necesitan de, facultades inferiores, como el proceso de la información visual (necesario para el establecimiento de relaciones espaciales), el sentido del equilibrio (necesario para la navegación en terreno irregular) o el tacto (para el ajuste de la fuerza en operaciones de prensión). Por eso, en opinión de bastantes psicólogos y etólogos, debería considerarse inteligencia tanto al razonamiento como al conocimiento de sentido común.

Como dijimos en el capítulo introductorio, dos aproximaciones muy diferentes a la Robótica han sido propuestas, y es mi intención dar una breve descripción de los principios fundamentales de cada una de ellas para hacer ver cómo los métodos que propongo emergen de la aproximación comportamental y son prioritariamente guiados por ella.

El primer paradigma, el más antiguo y extendido en la investigación Robótica, es el clásico, derivado del modelo de la mente que la Inteligencia Artificial había construído en los años sesenta y principios de los setenta, basada en el razonamiento simbólico sobre modelos del mundo, y conocida a veces (el nombre es de Haugeland) como GOFAI[1] La suposición subyacente a esta visión (o al menos, a la parte más pura de ella) se enuncia en la hipótesis del Sistema Físico de Símbolos (PSSH) de Newell y Simon ([NS76])

> Un sistema físico de símbolos posee los medios necesarios y suficientes para producir acción inteligente general

Esto conduce a la noción de Inteligencia Artificial como

> El estudio de los sistemas de símbolos con el propósito de entender e implantar en ellos una búsqueda inteligente ([LS89])

Lo anterior significa que, si construímos un sistema de símbolos (proposiciones, u otros) que modelicen suficientemente bien un aspecto de la realidad, podremos razonar sobre los modelos, de tal modo que el resultado nos indique el comportamiento del sistema real, y como influirá sobre él cualquiera de nuestras actuaciones. En el caso general de un modelo del mundo físico (o incluso no físico) en su conjunto, podríamos obtener inteligencia general (en el sentido humano del término).

La consecuencia fundamental en Robótica es la necesidad de construir una representación abstracta (un conjunto de símbolos) que sea propuesto como modelo del mundo real en el que el robot trata de operar. Es lógico pensar, al menos en un primer momento, que un modelo geométrico puede satisfacernos, dado que la mayoría

[1] Good Old Fashioned Artificial Intelligence, (la buena y vieja Inteligencia Artificial)

de las cosas manipuladas por robots son objetos construídos por el hombre. Por otra parte, si un modelo geométrico de las superficies y/o volúmenes de los objetos, junto con su posición en el espacio, es llenado con los datos obtenidos, entonces se podrían diseñar métodos apropiados para detectar colisiones o llevar el brazo a un punto seguro para asir un objeto. Tales métodos estarán basados en geometría analítica.

El problema esencial que ahora emerge es, pues, como ir desde los datos adquiridos por los sensores a una descripción geométrica (o incluso más abstracta). Hemos usado en el párrafo anterior el verbo "llenar". Ello es porque algo (o quizá mucho) del conocimiento acerca del mundo es codificado en un modelo general donde los huecos son completados con objetos (o, para ser precisos, con símbolos representando los objetos y los valores de sus propiedades físicas como peso, color, posición, orientación, etc.). Se supone que un planificador de alto nivel trabajará con estos datos y tendrá que obtener una secuencia de acciones, usualmente dada en forma de movimientos del robot, que describirá las subtareas ordenadas que constituyen la tarea completa. Un paso adicional convertiría estas especificaciones de la tarea en movimientos del robot. El esquema se puede ver en la figura 2.1. Esta cadena de procesos constituye el llamado ciclo de recoger información-pensar-actuar y este tipo de descomposición vertical es conocido como descomposición funcional.

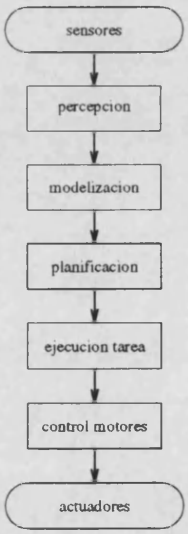


Figura 2.1: Descomposición funcional y vertical

Los inconvenientes de la construcción de un modelo simbólico para estos casos deberían ser evidentes simplemente observando la longitud y complejidad del proceso mencionado. En primer lugar, los datos llegan con ruido, tanto el propio de la digitalización, como el debido a sombras, imperfecciones en las piezas, ocultaciones, puntos de vista anómalos que provocan coincidencias de ejes o bordes sin correspondencia real, falta de contraste, etc. Lógicamente, datos imprecisos crean descripciones simbólicas erróneas, y es difícil encajarlas en los modelos almacenados. Para evitar

esto se puede dotar al modelo de más grados de libertad y tolerancias en los valores de las variables, pero ello implica que una misma descripción puede entonces encajar en varios modelos, creando ambigüedad. Esta ambigüedad se intenta resolver tomando más aspectos (datos) del objeto, como color, o textura de la superficie; pero ello implica más cálculo para su extracción y modelos más complicados (dado que, por supuesto, estos nuevos aspectos también están afectados de un error, y sus valores en cierto rango de tolerancias).

Otro inconveniente es que cuando alguna de las descripciones proporcionadas por el bajo nivel no encaja en ningún modelo hay que volver a tomar los datos y analizarlos, orientando el proceso a la resolución de los aspectos que resultaron ambiguos.

Y por último, una descomposición secuencial implica que cada paso no puede empezar hasta que el anterior haya terminado, y que el fallo de uno sólo de ellos hace fracasar el proceso completo: una cadena es tan fuerte como el más débil de sus eslabones. Por ejemplo, existen ingeniosos métodos para la interpretación de dibujos de líneas ([Tur74, Kan81]) pero ninguno realmente efectivo para la construcción de tales dibujos a partir de imágenes reales tomadas por cámaras.

Esta ha sido una visión breve y un tanto personal sobre la Robótica clásica y los problemas con que se enfrenta; hasta ahora, nadie ha sido capaz de dar una buena solución general a ellos, pero el paradigma es hoy el más vivo en la investigación Robótica y se sigue desarrollando. De hecho, la mayor parte de los investigadores piensa que los problemas serán resueltos construyendo mejores modelos y planificadores más flexibles. Por otra parte, otros investigadores hace no mucho tiempo discreparon de ellos y pensaron que una aproximación casi completamente nueva era necesaria para abordar el ruido y la incertidumbre, así como otros muchos aspectos. Sus ideas, basadas en otra actitud hacia la Inteligencia Artificial, surgen de la interacción entre ésta y la Robótica que se ha revitalizado en los últimos diez años. Esta postura y sus implicaciones en mi trabajo serán comentadas ahora.

Recientemente, aunque la idea ya aparece de modo minoritario e incluso un tanto herético casi desde el principio de la Inteligencia Artificial, se ha desarrollado un modelo de los procesos inteligentes, que llamaremos sintético por oposición al clásico, analítico, que establece que las facultades mentales son el resultado de comportamientos, que se entienden como los actos que pueden conseguir la ejecución correcta de una determinada acción, resultando tal ejecución de la interacción entre el robot o ser vivo y el entorno. Estos comportamientos pueden ser directamente implantados en el sistema como partes de él, que llamaremos módulos comportamentales, o pueden resultar de la acción concurrente de varios de tales módulos.

Como es obvio, este punto de vista concede una importancia excepcional a la relación con el exterior, que se realiza a traves de los órganos de los sentidos. Casi todos admitiríamos sin discusión que la inteligencia no puede desarrollarse sin tal relación, pero nos cuesta más admitir que no pueda sostenerse sin ella. Sin embargo, los modelos que hacemos del mundo están fuertemente influídos por las percepciones

sensoriales; esto puede comprobarse analizando la forma en que aprenden a ver los raros casos de invidentes de nacimiento que mediante cirugía o tratamientos recuperan la visión ya adultos.

En el aspecto práctico, a la hora de construir un robot, la Robótica comportamental prescindiría de programarle explicitamente un comportamiento orientado a un propósito. Mas bien, programaría reflejos o comportamientos muy simples, pero eficaces, que en conexión directa con los sensores y los actuadores, generasen mediante su interacción el comportamiento requerido. Los ejemplos más espectaculares hasta ahora son los trabajos de Rodney Brooks, tanto sus robots insectoides como un robot móvil para recoger vasos vacíos ([BCN88]).

Para el caso particular de robots de ensamblado, directamente relacionado con nuestro objetivo, los programas serían construídos usando módulos comportamentales que deberían ser no simplemente piezas de programa, sino programa, mas su soporte físico (ordenador, robot, etc.) usando y cambiando el entorno, que de este modo es también parte del módulo. Sus entradas serían no sólo parámetros, sino también lecturas de los sensores. Y, similarmente, sus salidas serían señales para los actuadores y cambios en el mundo. Estos cambios eventualmente provocarán diferentes lecturas en algunos sensores, y consecuentemente diferentes comportamientos de otros módulos (o incluso la activación de unos en vez de otros). El resultado final sería (como debe ser en cualquier ensamblado) un estado diferente del mundo, pero no en el sentido clásico de estado (algo cuya interpretación encaja en el paso final de los almacenados por el modelo) sino en el sentido usual (algo en el mundo es diferente, o está puesto en un lugar diferente ahora que cuando el ensamblado comenzó). La idea clave es que una tarea de ensamblado podría ser lógicamente (¡no cronológicamente!) descompuesta en tareas que deberían ejecutarse eficientemente cuando fuesen llamadas, bien por otras, bien por el estado del mundo. Tales tareas son del tipo 'lleva brazo sobre bloque','captura bloque' o similares, posiblemente basadas en comportamientos más simples, como 'orienta mano','empuja bloque', 'mira hacia el bloque', etc. Los dos últimos ejemplos de tareas son interesantes porque sugieren algo no muy usual en Robótica clásica: movimientos de las partes sin ser capturadas ("constrained motions") y movimientos del sistema visual (" active vision").

La mayor parte de las veces el sistema puede incluso ser perfectamente ignorante sobre lo que ha conseguido, del mismo modo que una termita (o incluso la comunidad de termitas completa) ignora la forma del gran termitero que han construído. El único requerimiento importante es que cada módulo sea eficiente, y tan autocontenido como sea posible. Es dentro del módulo donde la fusión de datos de diferentes sensores, si los hay, y la contención del ruido son efectuados, y esperamos ser capaces de hacer esto bien (o mejor, esperamos que el módulo sea capaz de hacerlo bien) porque su tarea es muy simple, y consecuentemente podemos programarlo más fácilmente que si tuvieramos que preocuparnos en cada momento de las consecuencias sobre todo el resto del sistema. A la arquitectura de control distribuído que conecta

los módulos se la llama "arquitectura de subsumción" ([Bro91]), en el sentido de que cada módulo en muchas ocasiones subsume los comportamientos del anterior, de modo que si alguno falla, es posible que el robot sea todavía capaz de ejecutar algo de su comportamiento, o al menos no pararse completamente; esto se conoce como degradación suave ("graceful degradation"). El lector interesado puede encontrar más información en [Mal91].

En el aspecto de la construcción de modelos, hay diferentes posiciones más o menos extremistas entre los partidarios de la Robótica comportamental. Desde los que dicen, como Brooks, que deben ser evitados al máximo, y que la mayoría de las tareas usuales en un robot podrían ser ejecutadas sin ellos, (vease [Bro91]) hasta los que transigen en su uso, especialmente si se trata de modelos mínimos (es decir, construídos con un grado de detalle no demasiado fino y capacer de instanciar una clase amplia de objetos con razonable seguridad). Vease de nuevo [Mal91].

Por todo lo dicho, se entiende que la Robótica comportamental busque habitualmente su fuente de inspiración en la biología. Sus proponentes piensan que es mejor estudiar primero criaturas completas intentando reproducir cada una de sus capacidades senso-motoras para hacer que trabajen separadamente, si es posible, o bien en conjunto. Cuando digo "si es posible" quiero decir que a veces incluso el reflejo más simple no puede ser entendido aisladamente de su entorno, en el que todos los demás reflejos participan. Esto explica el interés en la búsqueda de sistemas biológicos que hayan resuelto problemas cruciales para la Robótica. Veremos un ejemplo de esto en el capítulo 6.

Un esquema de cómo organiza la Robótica comportamental una aplicación típica (un robot móvil) puede verse en la figura 2.2.

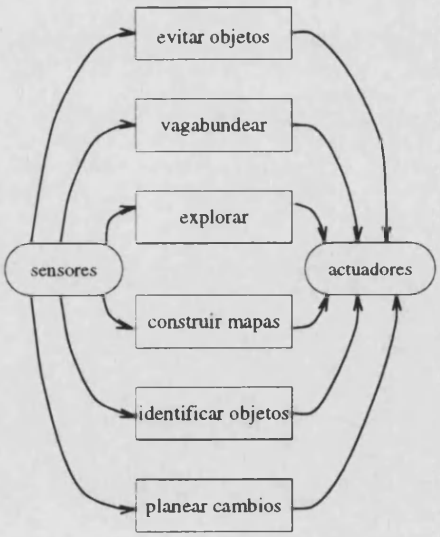


Figura 2.2: Descomposición modular horizontal

La idea central que orienta la inspiración biológica del trabajo toma como base la

visión. Es sabido que una gran parte (aproximadamente el 80%) de la información que los humanos recibimos del entorno nos llega a través de la vista, e igualmente a algunos animales. La variedad de sistemas visuales en la naturaleza es enorme, y el humano es probablemente el más flexible. Usamos la vista fundamentalmente para organizar el entorno físico en forma de relaciones espaciales entre las cosas, como arriba de, abajo de, detrás o delante de, etc. También asignamos tamaños a los objetos, lo cual nos sirve para determinar si algo (incluídos nosotros mismos) podemos pasar por o encajar en algún espacio vacío. Es interesante notar que casi nunca formulamos estas relaciones en términos analíticos (referidas a ningún sistema de coordenadas externo) ni hacemos ningún tipo de planificación consciente de las trayectorias. Sin embargo, nuestra coordinación y precisión es usualmente muy buena, incluso en la realización de movimientos balísticos (ejecutados en respuesta a una señal del sistema nervioso, sin esperar ninguna otra, y por tanto sin realimentación). Es tiempo de preguntarse si los robots que construimos no deberían intentar seguir también estos métodos, aunque fuese en tareas más simples.

De acuerdo con todo ello, el objetivo de la tesis puede, pues, enunciarse como la construcción de un módulo comportamental que, en un sistema formado por dos cámaras fijas apuntando hacia el espacio de trabajo de un brazo robot, lleve este brazo al lugar donde queremos verlo, en el espacio de medidas de tales cámaras. La descripción concreta del sistema usado para materializar este propósito será el objeto del próximo capítulo.

Capítulo 3

Planteamiento teórico y requerimientos del sistema experimental.

En este capítulo se describirá con detalle cuál es el problema concreto que deseamos abordar y cuáles son las restricciones que le van a ser impuestas, bien por su propia esencia, o bien por factores externos como disponibilidad de material, o limitaciones voluntariamente aceptadas para demostrar que es abordable aun con ellas.

El hecho de que el problema real sea presentado aquí, y no al principio, como debiera haber sido si se hubiese seguido un orden cronológico es consecuencia de nuestra intención dar una explicación lógicamente ordenada del trabajo. El problema práctico es la primera cosa que llama la atención de un investigador y motiva su reflexión acerca de métodos para resolverlo y principios generales que deberán aplicarse; pero es nuestra opinión que la memoria gana claridad usando este orden de exposición.

Un planteamiento general para todos los métodos de control que proponemos consiste en verlos como transformaciones de un espacio de entrada (cuyas coordenadas son los pixels medidos por un par de cámaras que miran al área de trabajo) y un espacio de salida (cuyas coordenadas son las cuentas para los motores del robot). Estos valores son, efectivamente, coordenadas, dado que cualquier cuaterna de valores de pixels en que encontramos un determinado punto real visto por dos cámaras no alineadas determinan de forma unívoca tal punto; e igualmente cualquier combinación de valores de los circuitos de control de los motores (en adelante, joints) lleva el brazo a cierto punto.

En el caso de las coordenadas de las cámaras, la razón de que deban ser al menos tres con al menos dos cámaras es obvia; pero, inversamente, no hay razón por la que no puedan ser más de tres. En este caso es claro que no todas serán independientes, pero eso no es obstáculo para su uso, antes al contrario, añade robustez, dado que al ir todos los valores afectados de un error (ruido) aleatorio, que además puede normalmente considerarse que sigue una distribución normal, el valor medio de las coordenadas reales del punto dadas por cada una de las combinaciones independientes es un buen estimador del valor real. Esta y otras técnicas son ampliamente usadas para la localización cartesiana de aspectos a partir de cámaras calibradas (véase, p. ej. [Tsa87]). No obstante, es necesario hacer notar que la dimensión real del

espacio de entrada será 3, y que al usar más coordenadas, si se trata de medidas tomadas en puntos reales, como es nuestro caso, los valores aparecerán confinados a un subespacio (una variedad) de dimensión 3 del espacio de entrada. Esta es una de las restricciones debidas a la naturaleza del problema que antes dijimos que aparecían, y que tendrá algunas consecuencias en ciertos métodos.

En el caso de los joints, la univocidad sólo es cierta en un sentido: una combinación dada de valores lleva el brazo a cierto punto, pero la recíproca no suele ser cierta. En muchos robots hay puntos que pueden ser alcanzados por dos configuraciones diferentes del brazo, e incluso por un número infinito de ellas (las singularidades). Este hecho deberá también ser tenido en cuenta a la hora de declarar en qué puntos del espacio cada método puede ser válidamente aplicado, y cómo se decide entre distintos valores para un mismo punto.

Es necesario hacer notar que tanto unas como otras coordenadas no son lineales; es decir, no puede llegarse por transformaciones lineales (rotaciones o traslaciones) desde las coordenadas cartesianas hasta las de las cámaras o las del robot, ni viceversa. Esto implica que los ejes de coordenadas de estos sistemas vistos desde un sistema cartesiano, serían curvos. No obstante, si las transformaciones son contínuas y localmente diferenciables, pueden, siempre de modo local, ser aproximadas por transformaciones lineales. Usaremos este hecho particularmente para la Teoría de la Red de Tensores. Estas suposiciones de continuidad y diferenciabilidad han de ser explicadas con cuidado. En el caso de las transformaciones cámara-cartesianas, desconocemos cuál es la función analítica, y nuestra confianza sólo puede estar basada en la observación del hecho físico de que un sistema óptico como los comunmente usados en las cámaras lleve puntos próximos en el espacio real a puntos próximos en su proyección. En el caso de las cartesianas-joints, conocemos la función analítica, al menos aproximadamente, y en nuestro caso resulta ser contínua, y diferenciable, salvo en las singularidades.

La figura que muestra una idea (sin detalles, por el momento) de la disposición del sistema es la figura 3.1.

La misión del proceso visual elemental sería únicamente la detección del punto terminal del brazo,y su salida, las coordenadas de dicho punto, en pixels, según son vistas por las cámaras izquierda (X_L, Y_L) y derecha (X_R, Y_R). La caja etiquetada "Algoritmo de mapeado" las transformará en las coordenadas para los joints, que hemos denotado *zed, shoulder, elbow, yaw*, por referencia al brazo SCARA[1] de la figura, pero en general sería válido con las coordenadas de cualquier brazo. El problema ahora estriba en ver qué se pone dentro de esta caja, por el momento negra. Una u otra elección es la diferencia entre los métodos que planteamos.

Impondremos ahora algunas restricciones al sistema, en parte arbitrarias, y en parte forzadas por los medios de que disponemos. En primer lugar, las cámaras no deberán calibrarse, es decir, no conoceremos sus posiciones reales en el espacio,

[1] Selective Compliance Remote Center Arm

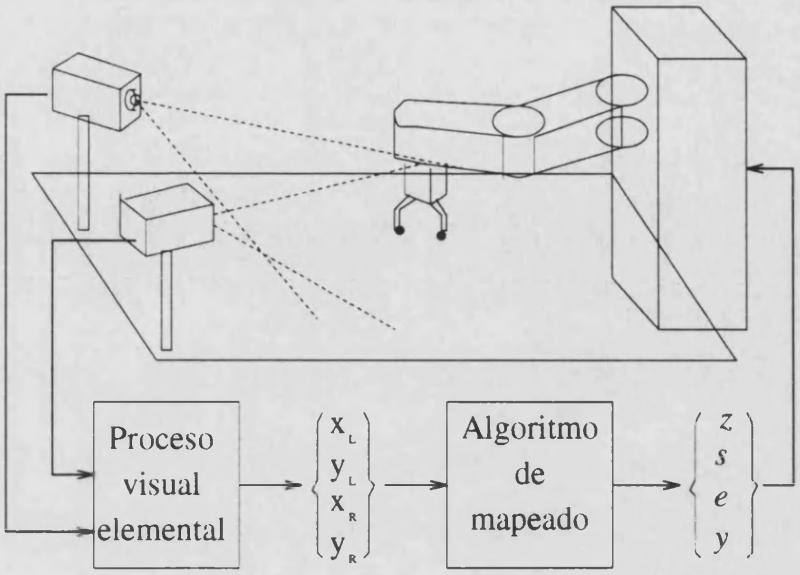


Figura 3.1: Esquema general del sistema

expresadas en coordenadas cartesianas (x, y, z), y renunciamos también a conocer la expresión analítica de la transformación que relacione las (X_L, Y_L, X_R, Y_R) antedichas con (x, y, z). Esta transformación estará implícita en datos numéricos que se almacenarán como resultado de los experimentos pero no será calculada usando conocimiento sobre la posición de las cámaras, la distancia focal de sus lentes o cosas similares. Sin embargo, vamos a obligar a las cámaras a que estén fijas, para forzar la consistencia entre las medidas tomadas en el momento de los experimentos y un posicionado que subsecuentemente las use.

Además de esto, el brazo no será llevado a una posición cartesiana, sino a una posición en el espacio de cámaras; en otras palabras, el brazo irá al punto donde las cámaras quieren verlo. Esto implica necesariamente que el procedimiento que diseñemos para relacionar unas y otras coordenadas tendrá que tener una parte de toma de medidas, y otra en la que se usen dichas medidas. Cabe incluso la posibilidad de que el sistema decida cuándo ha terminado la primera, o cuando debe ser invocada de nuevo si la información que almacenó no resulta válida por cualesquiera razones.

Las últimas restricciones establecerán que el preprocesado de la señal visual deberá ser el mínimo posible, e igualmente el prostprocesado de las señales para los motores, eliminando incluso el cálculo de la cinemática directa e inversa del brazo. Esto es una forma de cumplir al máximo el objetivo de la Robótica Comportamental consistente en relacionar percepción y acción al nivel más bajo posible. El cálculo de la cinemática inversa representó en muchos sistemas robóticos hasta ahora un problema dado el número de operaciones que requiere y la frecuencia con la que se ha de ejecutar; actualmente el aumento en la velocidad de proceso de los ordenadores lo ha paliado grandemente, por lo que la restricción de no usar la cinemática no

debería ser tan drásticamente mantenida en un sistema para uso real, siempre que los cálculos y las características del robot permitan alcanzar la precisión deseada, y que se ejecute un número limitado de veces.

A continuación explicaremos cómo hemos ideado y dispuesto una célula de trabajo que nos permita realizar los experimentos y cumplir todas las condiciones.

El único robot del que disponemos es el modelo RT100, fabricado por UMI Inc.. Es un brazo tipo SCARA, con seis grados de libertad más una pinza. Sus dimensiones y área de trabajo se muestran en la figura 3.2 y en la figura 3.3.

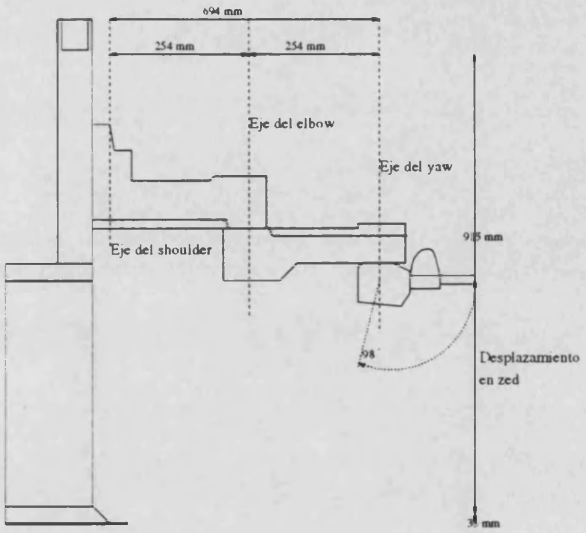


Figura 3.2: Dimensiones y área de trabajo del RT100

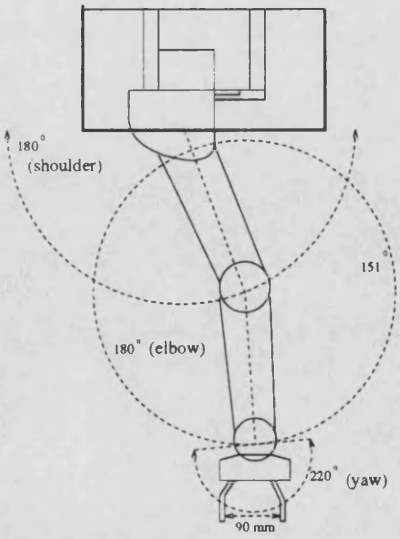


Figura 3.3: Vista superior

Se controla a través del conector RS-232 (la puerta serie) de un ordenador tipo PC o de una estación de trabajo; el protocolo enviado está al nivel de las cuentas para cada una de las articulaciones, dado que todas ellas disponen de un codificador óptico compuesto por dos diodos que detectan el paso de los vanos en un disco de plástico que gira con el motor, produciendo dos ondas cuadradas desfasadas 90^0 (salida en cuadratura) que permiten contar el número de pulsos en cualquier intervalo de tiempo, conociendo así la posición (a partir de la anterior), velocidad y sentido de giro. Esta señal pasa a controladores PID que son circuitos tipo Intel 8031 montados sobre placas de circuito impreso internas al robot. Su salida es una señal cuadrada con pulso modulado en anchura que controla los circuitos que suministran corriente a los motores (tipo Signetics L293E). La fuerza viene dada por la proporción entre la duración del pulso alto y el bajo. El funcionamiento de los codificadores y de la señal de salida puede verse en la figura 3.4

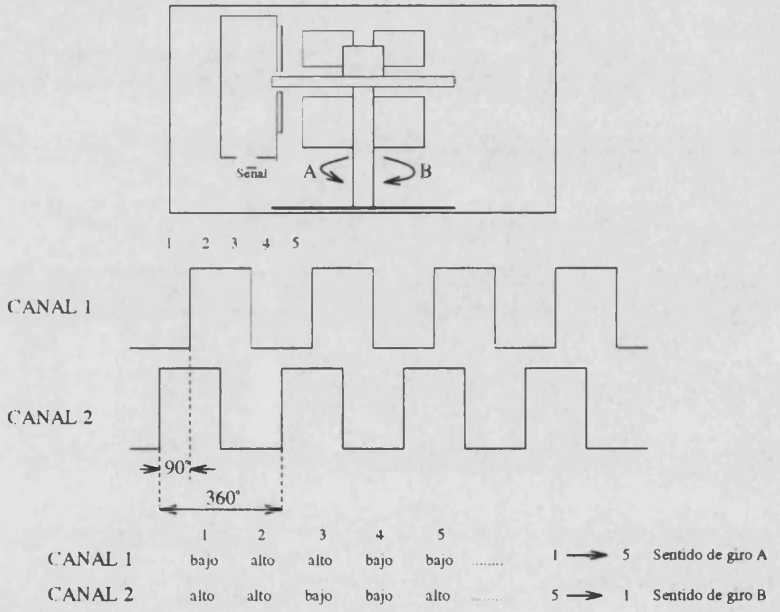


Figura 3.4: Codificadores y señal para los motores en el RT100

Para controlar los motores al nivel de torques sería necesario interceptar estos circuitos de control, construyendo físicamente otros que actuasen sobre los L293E; en principio, hay una entrada en la placa para hacer esto, y el fabricante podría suministrar información, pero consideramos que el nivel de cuentas era suficientemente bajo para nuestros propósitos, y que la dificultad de construcción y programación de tales circuitos no compensaba los resultados que se podrían obtener con ellos. No obstante, se comentará más sobre esto en el capítulo 10.

El control por la puerta serie se realiza simplemente enviando a través de ella determinados códigos que el manual explica para inicializar el brazo, fijar la velocidad (cambio en el número de cuentas de un motor por unidad de tiempo), cambiar los

parámetros de los controladores PID, y tareas similares. Situar estos párametros que se reciben por la puerta serie en los registros concretos de los circuitos de control es tarea de un programa residente en una ROM de la placa de control.

En cuanto al área de trabajo, resulta, según se puede apreciar en la gráfica de dimensiones del robot, bastante pequeña: un semicírculo de radio 694 mm. con la mano horizontal, y 508 con ella vertical. En aplicaciones de ensamblado real esto puede ser un serio inconveniente, pero en el tipo de experimentos que planteamos no altera en nada las conclusiones, excepto en lo que se refiere al uso de la memoria; como se verá en capítulos posteriores, en todos los métodos (excepto la regresión, cuyo resultado es una fórmula) se usa una memoria proporcional al volúmen del área de trabajo, de modo que áreas de trabajo reales (un volúmen quizá hasta unas ocho o diez veces el que tenemos) ocuparían una memoria de hasta un órden de magnitud superior, lo cual, como se verá luego, no es un problema grave.

La precisión de este robot es bastante reducida, en el sentido de que, programada adecuadamente la cinemática inversa según las dimensiones dadas por el manual ([Oxf90]), puntos cercanos al límite del área de trabajo se alcanzan con errores máximos de aproximadamente 7mm, y a ello se une un error adicional cuando la mano está vertical (pitch de -90^o) de otros 6 mm., aprox., probablemente debido a que los ejes del yaw, el pitch y el roll, que según el manual se intersectan en un punto, no lo hacen exactamente. [2] En cambio, la repetibilidad resultó correcta: si se le enviaba al brazo la misma señal en cuentas en instantes muy diferentes (en días diferentes, después de varios calibrados) o viniendo de posiciones anteriores diferentes, se llegaba a posiciones apenas distinguibles entre sí, no separadas más de aproximadamente 0.5 mm. Esta es otra de las razones para intentar evitar el uso de la cinemática en esta clase de robots. Pero incluso en robots de gran calidad pequeñas dilataciones de las piezas del brazo pueden provocar variaciones sensibles en la posición del punto terminal, que no pueden corregirse por la realimentación a los motores, ya que no tienen nada que ver con ella, y para las que se deberá necesariamente recurrir a la realimentación sobre la señal de sensores externos.

Respecto a la ambiguedad en la cinemática inversa, que en brazos SCARA permite alcanzar la misma posición en el espacio con dos configuraciones diferentes (derecha o izquierda, ver figura 3.5), hemos elegido usar la izquierda por tener en este brazo acceso a un área de trabajo ligeramente mayor, debido a la construcción asimétrica de la torre de sostiene el brazo, que impone limites distintos al recorrido de la articulación del shoulder en uno y otro sentido. Lo idóneo en una aplicación real sería usar ambas configuraciones para acceder al máximo de espacio posible. En el capítulo 10 se comentará la forma en que podría hacerse esto en cada método.

Por su parte, el sistema de visión consta de dos cámaras CCD iguales (aunque para ninguno de los métodos es este un requisito imprescindible) orientadas de tal

[2] La seguridad de estas observaciones se basa en que otra persona (Dr. Litkourgos Petropoulakis) programó independientemente de mí la cinemática, obteniendo prácticamente los mismos resultados

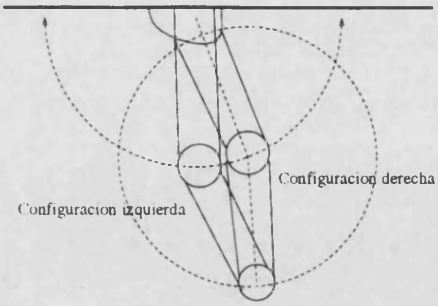


Figura 3.5: Configuraciones derecha e izquierda en brazo SCARA

modo que su área de vision abarque lo más posible del espacio de trabajo. La colocación de las cámaras y la elección de su óptica planteó ciertos problemas. Si se escogía una óptica más o menos usual (lentes con una focal de 16 mm.) era necesario situar las cámaras a una distancia no inferior a 2 m del área de trabajo para recogerla entera, pero ello implicaba que también entraba en el campo de visión un área inútil, debido a la poca apertura, y además la distancia real que representaba un pixel era demasiado grande. Nótese que, como el error experimental mínimo que vamos a poder obtener es de un pixel, nos interesa que éste represente la menor distancia posible, dado que la resolución de la imágen es fija (512x512 pixels). Hay dos soluciones a esto: usar lentes de mayor distancia focal (teleobjetivo) colocadas lejos, o lentes de gran apertura (gran angular) colocadas cerca. Optamos por esta segunda solución, por razones de precio de la óptica, y además porque es una solución raramente usada, dada la gran deformación que introducen en la parte de la imágen más alejada del centro óptico, y una de las cosas que pretendemos probar es precisamente que nuestros métodos pueden soportar fuertes alinealidades tanto como disposiciones más corrientes. El ángulo de apertura se tomó, pues, de 105°. La célula de trabajo queda, pues, como se muestra en la figura 3.6, vista desde arriba.

Las imágenes tomadas por ambas cámaras son capturadas y digitalizadas por una placa modelo MVP-AT de Matrox Inc., que dispone de cuatro cuadrantes para el almacenamiento de hasta cuatro imágenes de 512x512 pixels de resolución, en 256 niveles de gris (8 bits), además de algunos circuitos para la realización de ciertas operaciones elementales en tiempo casi real (umbralización y operaciones aritméticas). El procesado de imágen debía ser, como comentamos al principio, lo más simple posible. Como el objetivo era la detección del punto terminal del brazo, decidimos añadir los dispositivos necesarios para simplificar esta tarea al máximo, haciéndola enteramente robusta ante el ruido. Para ello lo más sencillo fue situar en el punto terminal del brazo dos diodos luminiscentes (LEDs), uno en cada pinza de la mano, que deberá estar un poco abierta (unos 30 mm). Pudimos conectar con dos transistores estos diodos a una salida auxiliar que el robot posee para propósitos como éste, y que consta de varios contactos que funcionando en nivel TTL pueden ponerse a 0 ó 1 mandandole al robot las señales de control oportunas, que encenderán

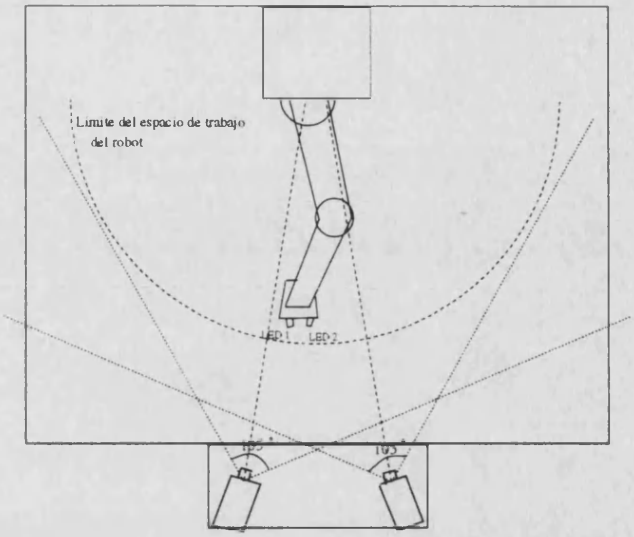


Figura 3.6: Vista superior del área de trabajo

y apagarán los LEDs. El brazo fue situado con la mano vertical (pitch -90^{o}) lo cual hace los diodos absolutamente visibles por cualquiera de las cámaras en cualquier lugar del área de trabajo evitando autoocultaciones. Además, esto resultará óptimo en un futuro para asir objetos desde su parte superior sin que los diodos estorben, y de modo que el punto de prensión coincida prácticamente con aquel en que ellos están situados. Por otra parte, permite un cálculo de la orientación de la pinza (en el espacio de cámaras) que también puede servir como señal visual sobre la que realimentar. El aspecto de la célula del robot vista por cada una de las cámaras puede apreciarse en la figura 3.7, al final de este capítulo.

El procedimiento que se encontró más sencillo para el procesado consistió en ejecutar para cada cámara los siguientes pasos:

- Tomar una imagen con los diodos encendidos.
- Tomar la misma imagen, pero con éstos apagados.
- Restar ambas imágenes, con lo cual todo lo que no ha variado (es decir, todo menos los diodos) se elimina, y sólo quedan dos puntos no negros (o sea, con valor de gris no nulo) en las posiciones de los diodos, aparte de ruido.
- Umbralizar esta imagen, de modo que queden dos manchas blancas sobre fondo negro
- Encontrar tales manchas, y hallar el centro (en pixels, referido a la esquina superior izquierda de la imagen) de cada una.
- Dar como posición en pixels del punto terminal la media de las dos anteriores

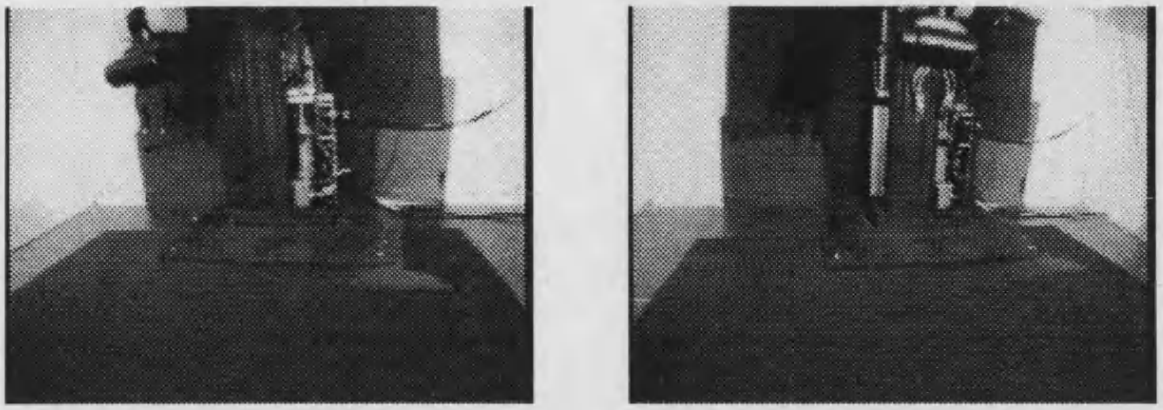
Aunque este procedimiento parezca algo farragoso, no lo es para esta placa en particular, dado que la diferencia de ambas imágenes se ejecuta con los circuitos de la placa, así como la umbralización. Encontrar tales manchas es, sin embargo lo más lento del proceso, dado que se ha de recorrer toda la imágen desde su esquina superior izquierda hasta encontrar un punto no negro. Una vez encontrado, se hallará su centro no como el centro de masas, sino como el centro del rectángulo que inscribe a su perímetro, el cual ha sido hallado recorriendo dicho perímetro desde el punto inicial. Esto tiene la ventaja de ser mucho más rápido, dado que el tiempo de cómputo es lineal con el perímetro, no con el área, como en el caso del cálculo del centro de masas, y no introduce error significativo, dado que las manchas son convexas y relativamente pequeñas (perímetro del órden de 8 a 10 píxels). Sobre el algoritmo usado para seguir el perímetro, véase [RK88]. El proceso seguido se puede ver en las imágenes que son sus pasos más importantes, mostradas en la figura 3.8, al final de este capítulo. En esta figura la imágen (a) es la original, con los diodos apagados; la (b) es la diferencia entre las imágenes con diodos apagados y encendidos. Obsérvese que aparece cierto ruido, especialmente en los bordes de los objetos; esto es debido al tamaño finito de las celdillas de la cámara CCD, que provocan errores de muestreado; los niveles de gris de la imagen han sido artificialmente elevados para poder apreciar este ruido, pero una umbralización simple a valor aproximadamente 50 (no es crítico) genera las dos manchas mostradas en la imagen (c). Finalmente, la (d) muestra una cruz sobre el punto que ha sido reconocido como posición correcta, superpuesta a la imagen original.

Ahora describiremos el procedimiento de fijación de las cámaras. Aun cuando, según dijimos, desconocemos su posición cartesiana, es necesario que estén siempre en la misma posición y orientación. Para ello se escogieron puntos relevantes del espacio de trabajo, señalados como marcas en un papel que se sitúa sobre el suelo, en posición también marcada. A continuación el usuario emplea la placa digital de imagen para hacer coincidir dos cruces dibujadas en la memoria de imagen con dos marcas en el papel; esto se hace para cada cámara, y se toma nota de las posiciones de las cruces. El hecho de que sean al menos dos (pueden ser más) es debido a que nuestras cámaras están sobre un soporte atornillado al suelo, que les permite variar sólo su orientación (es decir, dos grados de libertad). Para usar el procedimiento, normalmente antes de cada toma de medidas, la placa captura imagen en vivo, y situa dos cruces en las posiciones en pixels en que deben estar; esta vez lo que se mueve no son las cruces, sino las cámaras, hasta que la posición correspondiente de las cruces coincide con las marcas sobre las que se situaron la primera vez. Este procedimiento tiene la ventaja de ser rápido, preciso hasta el límite de resolución que la cuantización permite, y puede ser ejecutado por personas no especializadas. Tiene el inconveniente de que requiere una placa de imagen que pueda superponer gráficos (las cruces) con imagen tomada en vivo, mediante el uso de que los fabricantes de estas placas llaman " overlays" (recubrimiento). Las cruces sobre las imágenes de ambas cámaras, más una línea horizontal adicional que se hizo coincidir con el borde de la tarima, para aumentar la robustez del procedimiento, pueden verse en

la figura 3.9, al final.

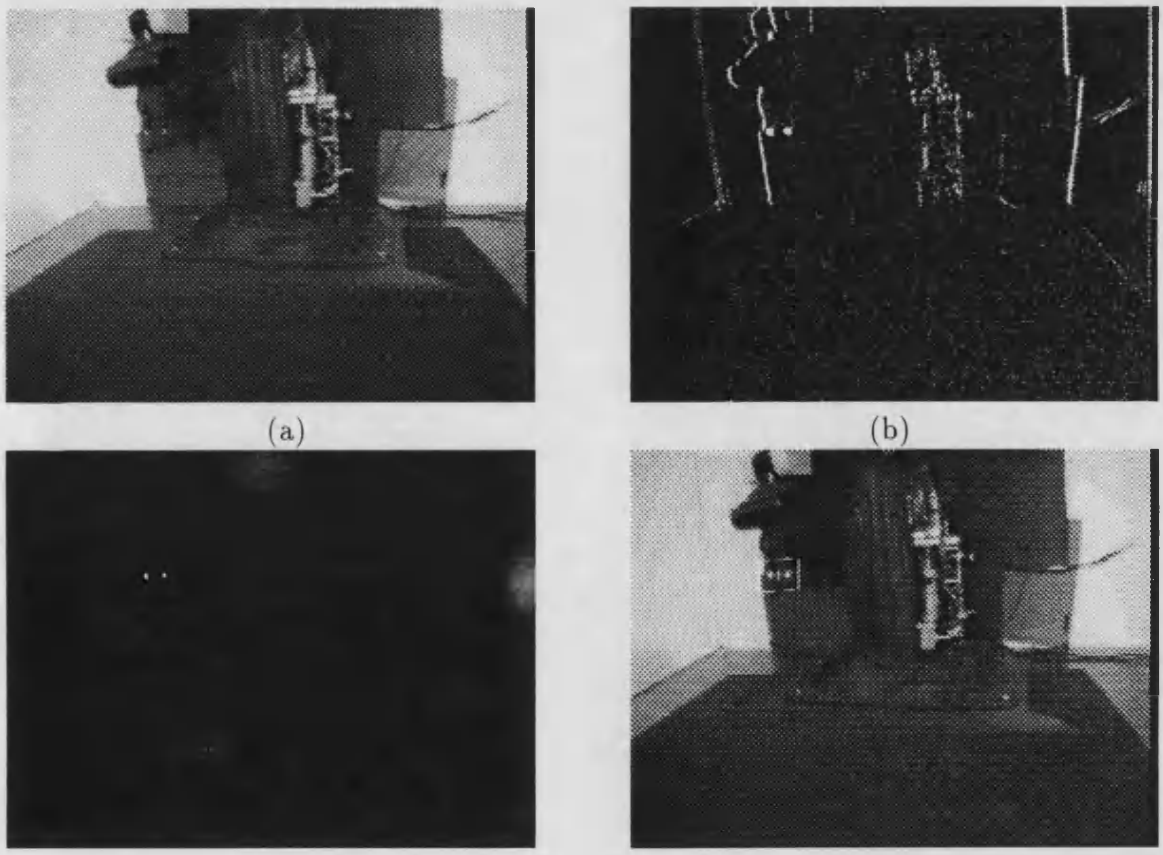
Por último, un aspecto importantísimo es el tiempo de medición. Como veremos en sucesivos capítulos, el coste computacional de los métodos que propondremos, en términos de operaciones a realizar con las posiciones en pixels del punto terminal para generar las cuentas de los joints, es mínimo, y por tanto los cuellos de botella del sistema estarán en la localización de dicho punto terminal, y en el control de los motores. Respecto a lo segundo ya se comentó algo en este mismo capítulo sobre el nivel elegido; respecto a lo primero, hay unos tiempos mínimos que tienen que ver con el encendido de los diodos (50 ms) y su apagado (otros 50), lo cual, para las dos cámaras da 200. La propia toma de las cuatro imágenes necesarias (dos por cámara), es de 50 ms por imagen, por tanto, 200 ms. El tiempo para hallar su diferencia y binarizarla es, en nuestra placa, de 220 ms por pareja, lo cual implica 440 ms más. El tiempo para seguir el perímetro de las manchas blancas y dar la posición central en pixels es despreciable, menor de 1 ms por imagen. Pero hasta que se localiza el primer punto blanco, para usarlo como inicio del perímetro, hay que leer secuencialmente los pixels anteriores, lo cual, dada la lenta velocidad de transferencia entre la memoria de la placa y los registros del PC, lleva unos 550 ms por imagen, dando 1100 más. Esto da un total de poco más de 1900 ms, 1.9 segundos, lo cual hace impracticable un control en tiempo real. Para acelerar este proceso debería disponerse de hardware que detectase en el tiempo más breve posible la posición (NO el valor, eso ya lo hay) del punto más brillante de la imágen; o, alternativamente, disponer de placas digitalizadoras instaladas sobre el bus de estaciones de trabajo, que mapeen directamente su imagen a la memoria de éstas, las cuáles sí resultan suficientemente rápidas, incluso en búsqueda secuencial, dado que no hay transferencias que hacer; esta es la solución usada en [HC93].

Una vez expuestas las restricciones, material, disposición del mismo y limitaciones, es tiempo de escoger los métodos y planificar experimentos concretos. Para ello, buscaremos en la bibliografía trabajos que, directa o indirectamente, acometan este objetivo o similares intentando aprovechar sus aciertos y evitar sus errores; tal es el contenido del siguiente capítulo.



(a) (b)

Figura 3.7: El área de trabajo vista por cada cámara



(a) (b)

(c) (d)

Figura 3.8: Detección visual del punto terminal

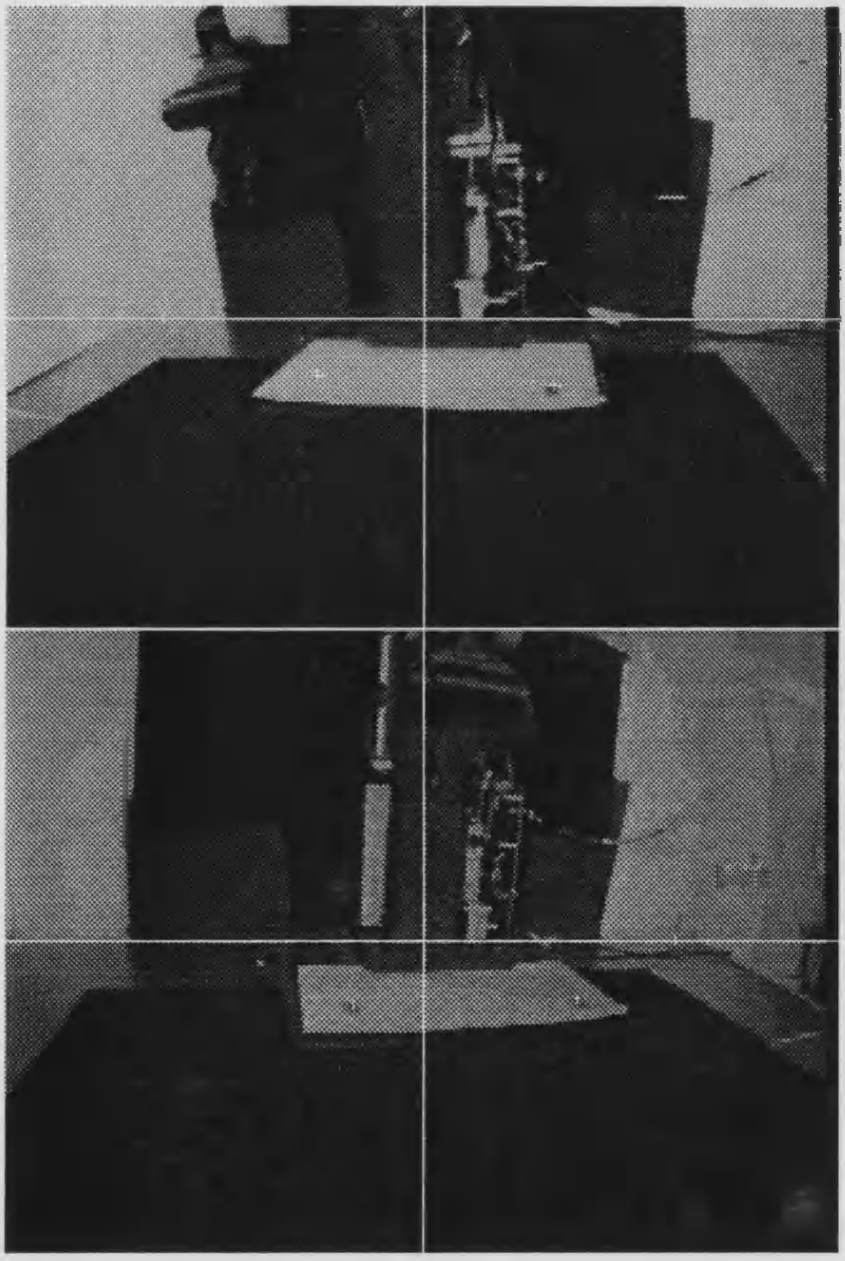


Figura 3.9: Puntos para fijar las cámaras, vistos por cada una.

Capítulo 4

Estado de la investigación en métodos de control basados en visión.

Este capítulo pretende ser una revisión bibliográfica no exhaustiva, sino más bien orientada a la explicación de este trabajo, sobre cómo una parte aún pequeña, pero creciente de la investigación Robótica va abordando de modos nuevos los problemas de visión y su relación con el control. El uso de la información visual para tareas diversas ha estado presente desde los primeros intentos, pero su aplicación está dificultada por problemas como el gran volúmen de calculo necesario, o los precios del hardware de visión, que sólo en los últimos años ha comenzado a ser asequible.

Desde un punto de vista práctico o industrial, son los problemas de ensamblado los que más podrían hipotéticamente beneficiarse de un uso eficiente de la visión, que involucrase reconocimiento de partes llegando en órden y posición aleatorios, localización de puntos de unas y otras que deben coincidir, etc. La importancia industrial de las tareas de ensamblado ya fué notada hace casi 15 años por Nevins y Whitney, por citar sólo un ejemplo, que muestran cómo casi un tercio del trabajo efectuado en la contrucción de vehículos a motor, y casi un cuarto en aparatos electrónicos está dedicado a ensamblar las piezas. ([NW78]). Hoy, habiendo adquirido mucha más experiencia investigadora, multitud de problemas se han resuelto con soluciones "ad hoc", pero una metodología general para el uso de la visión en Robótica está todavía por desarrollar. Durante todo este tiempo la aproximación prevalente ha sido la que llamaremos clásica, no es un sentido despectivo, sino contrariamente, como reconocimiento a su papel de patrón básico de la mayoría de los trabajos.

El ejemplo más conocido y documentado de esta aproximación probablemente sea HANDEY, el sistema de Lozano-Pérez, descrito en varios de sus artículos (p. ej, [LPJMea87]) y más recientemente en un libro ([LPJMO90]). Este es un sistema orientado esencialmente a la planificación, con especial atención a los problemas de encontrar caminos apropiados para el brazo robot, tanto en espacio libre como en las proximidades de los objetos, aunque no está dotado todavía de sensores que controlen la ejecución de la tarea. Todo el razonamiento geométrico necesario para generar las trayectorias se ejecuta sobre modelos detallados y suficientemente precisos (error no superior a 1mm) de todos los objetos que aparecen en la escena, incluído el propio brazo robot. Tales modelos son introducidos de antemano generándolos con

algún tipo de sistema de CAD. Los algoritmos para la detección de colisiones y la planificación de los caminos se basan en el uso del espacio de configuraciones, definido como la representación de una elección de parámetros que caracterizan posición y orientación de cada objeto. Cierto conjunto de valores de estos parámetros llevarán algún punto del objeto al interior de alguno de los obstáculos presentes en su entorno, y consiguientemente serán un punto prohibido; a este conjunto se le llama espacio de configuración de obstáculos. Una trayectoria será un camino en el espacio de configuraciones de un objeto que no intersecte al espacio de configuración de obstáculos de ningún otro. Esta idea tiene la ventaja de ser general, en cuanto que las coordenadas del espacio de configuraciones pueden ser cartesianas, polares, ángulos de orientación de la mano, etc. pero lógicamente no permite cambios dinámicos del entorno. Las trayectorias del brazo, bien libre o con carga, se transforman a secuencias de cuentas usando un modelo cinemático también detallado del manipulador. Lozano-Pérez reporta varios interesantes algoritmos para computar el espacio de configuraciones y los caminos a partir de los modelos; no da, sin embargo, ningún método para abordar los gravísimos problemas provocados por la incertidumbre (errores en el posicionamiento del brazo, en el tamaño, posición u orientación de las piezas) ni para integrar la información de sensores una vez comenzada la tarea. El sensor que utiliza (un scanner de profundidad) localiza y reconoce los objetos, encajándolos en los modelos almacenados, no obstante, los artículos no explican qué nivel de ruido puede llegar a admitir, ni detallan los algoritmos de reconocimiento y encaje. De hecho, uno de los defectos sustanciales, si no del sistema, sí de la documentación presentada sobre él es la ausencia de resultados experimentales sobre el robot real (no sobre su modelo). No obstante, hay que tener en cuenta que la orientación del trabajo es esencialmente hacia la planificación, y que está aún (y probablemente seguirá varios años) en proceso de desarrollo. En cualquier caso, sus resultados serán algún día necesarios, cuando haya métodos eficientes de contener el ruido y la incertidumbre en entornos reales, y de realizar la fusión de datos a bajo nivel; en ese momento los subsistemas eficientes de visión, de navegación (en el caso de robots móviles) y de prensión deberán ser conectados a los modelos y coordinados de algún modo si se busca un comportamiento aceptablemente inteligente. Del mismo modo, los problemas mecánicos y de control asociados a la estabilidad y precisión deben considerarse, y multitud de trabajos de control siguen abordándolos; como ejemplo, véase [Bea82] en el capítulo 3.

Volviendo a la orientación que nos ocupa, el problema de ejecutar tareas con alguna eficiencia sin construir modelos es abordado por Brooks. Véanse a este respecto las referencias citadas en el capítulo anterior, y sobre todo, [Bro91], un artículo en el que aboga por sistemas que no construyan modelos, sino que desarrollen la inteligencia de una manera incremental, construyendo primero las capacidades sensoriales más inferiores mediante aprendizaje de relaciones entre percepción y acción por medio del ensayo en sistemas reales. Por ejemplo, una implantación eficiente del reflejo oculo-motor (aquel por el cual los ojos siguen a un punto móvil) resolvería muchos problemas de posicionado de las cámaras en tareas de ensamblado;

en un caso como este, las cámaras no se orientarían hacia una posición cartesiana en el espacio, conociendo que el brazo está ahí por la información de la parte del sistema que hace el control dinámico, sino que lo seguirían permanentemente, con lo cual podrían alertar si se pierde de vista por ser ocultado, o por otra causa. En resúmen, la información se transmite entre unas partes y otras del sistema a través del mundo, lo cual equivale a usar el mundo como su propio modelo. Esta es la aproximación más extrema, y tiene como inconveniente el que no deja claro cómo estos reflejos se podrán conectar a los sistemas de modelos, cuando éstos lleguen a ser realmente necesarios. Es un problema que la investigación Robótica deberá plantearse en un futuro, pero por el momento, lo que está claro es que es necesario implantar los reflejos esenciales de modo eficiente para que las capas superiores, si debe haberlas, tengan en qué apoyarse.

En esta línea vamos a comentar primero varios trabajos que tienen una relación sólo parcialmente directa con éste, pero que entran dentro de la misma filosofía de ejecutar acciones de posicionado o control sin recurrir a modelos cinemáticos del robot o sistema que están controlando. Muchos de ellos (aunque no todos) usan redes neurales, y por eso se empieza a usar ([Lee93]) el término "neurobótica" para referirseles.

El primero de ellos es el famoso artículo de Barto, Sutton y Anderson sobre el control de un péndulo invertido, "Neuronlike adaptive elements that can solve difficult learning control problems".([BSA83]). En él se usa una red neural para controlar una plataforma con ruedas simulada mediante una fuerza horizontal que puede actuar sobre ella para evitar que el péndulo caiga. Las ecuaciones dinámicas del sistema no se conocen, y la única señal de control disponible es una señal binaria de fallo, que se activa cuando el péndulo ha caído un ángulo mayor que un umbral, o cuando la plataforma llega a uno de los límites. Véase la figura 4.1 Nótese que la señal

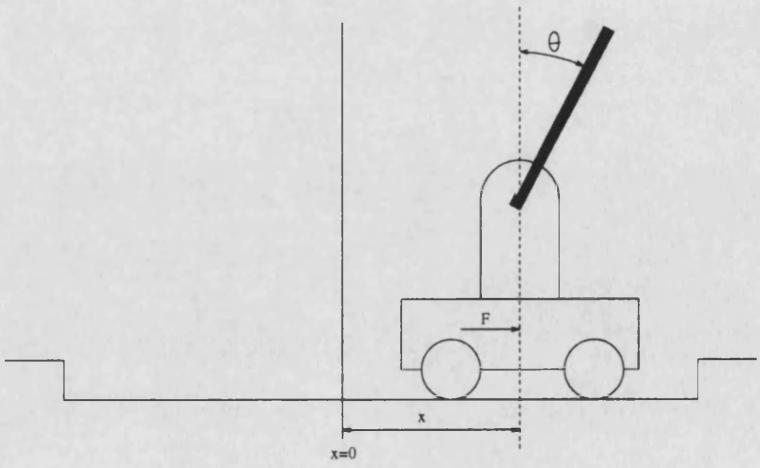


Figura 4.1: El péndulo invertido

de error es mucho más débil que la usada por sistemas usuales de control analógico

o digital, en realidad no evalúa directamente una acción, sino las consecuencias de la misma. Usa dos módulos: un módulo de búsqueda asociativa (ASE) que actúa cuantizando el espacio de entrada y generando una señal que depende de la entrada cuantizada y de la señal de error, más una señal aleatoria. El estado del ASE va cambiendo de tal modo que cada conexión (asociada a una de las cajas de entrada) aumenta su elegibilidad en función de los resultados anteriores. Los detalles pueden encontrarse en el artículo original, [BSA83]. El otro módulo del sistema, llamado el crítico adaptativo (ACE), sirve para mejorar la eficacia del ASE, que no es suficiente, computando una señal de error mejorada que depende de la entrada presente, y de anteriores señales de error; su estado interno también se actualiza según ciertas reglas. La novedad importante de este artículo, y de ahí la frecuencia con que se le cita, es que fué el primero en demostrar la capacidad de métodos no convencionales para controlar sistemas razonablemente complejos con reglas bastante simples. Un aspecto muy importante es que nueve años después, Jervis y Fallside comprobaron que este controlador realmente funciona, construyendo el sistema de Barto. (véase [JF92]). Usaron como controlador un transputer tipo T800, y sus conclusiones fueron que si bien el dispositivo aprendía, y terminaba realizando la tarea con éxito, era necesario partir de un estado del sistema relativamente próximo a la solución, y escoger apropiadamente ciertos parámetros, como el número de intervalos de cuantización en el codificador de las entradas. Esto nos muestra que una simulación no siempre es lo acertada que sus diseñadores suponen, y que es fácil olvidar o subestimar aspectos de los sistemas reales que, cuando se presentan, cambian absolutamente el comportamiento.

En una línea similar de conexión por procedimientos no analíticos entre entradas y salidas debemos citar los trabajos de Kawato y Albus. El primero construye una red neural muy simple con tan sólo tres neuronas (una por cada articulación de un brazo) para simular la dinámica de éste. Las entradas a cada neurona son combinaciones de las variables de entrada (los joints), funciones trigonométricas de éstas, y sus derivadas temporales. Las salidas son los torques para las articulaciones. El tiempo de aprendizaje es relativamente breve, y no se detallan resultados numéricos ni se especifica si se trata de una simulación o de un robot real; véanse los detalles en [KFS87] y en [KvdS93].

El trabajo de Albus, ya antiguo pero constantemente citado, aparece en el libro "Brains, Behaviour and Robotics" ([Alb81]). En él el autor expone sus ideas acerca de una jerarquía de sistemas de control que hipotéticamente podría explicar todos los comportamientos, pero a nivel práctico lo más interesante es el método de aproximación de funciones conocido por CMAC [1]. Este procedimiento se explicará con detalle en el capítulo 7, refiriéndolo en particular al uso que haremos de él. Digamos aquí que está pensado como un modelo del cerebelo, al que se supone ejecutor de una serie de transformaciones $\mathbf{S} \rightarrow \mathbf{M} \rightarrow \mathbf{A} \rightarrow p$ donde $\mathbf{S}$ sería un vector de entradas,$\mathbf{M}$ el conjunto de fibras usadas para codificarlo, $\mathbf{A}$ un conjunto de células contactadas

[1] Cerebellar Model Arithmetic Computer, Modelo de computador cerebelar artimético

por **M** que ejecutan realmente la transformación, y p el valor de salida.

En el caso de que hubiera varios valores de salida haría falta una de estas estructuras por cada uno. En el capítulo 6 de su libro Albus da razones biológicas que hacen plausible este modelo, cuya ventaja principal, como se verá en nuestro estudio del capítulo 7, es la capacidad de generalización. La estructura teselada del espacio de entrada, debida a la cuantización, no es un inconveniente, sino una ventaja, siempre que se multipliquen los canales de entrada, y cada uno desplace la cuantización respecto a los otros; de esta forma se consigue una resolución proporcional al número de canales, y además la estructura de almacenamiento de los pesos hace que cada uno de ellos contribuya en parte al cálculo de la salida para varios puntos próximos. De este modo, cuando el procedimiento de aprendizaje lo varíe, contribuirá también a una salida más correcta en puntos cercanos, donde quizá no haya habido experiencia de aprendizaje. Albus sugiere que el aprendizaje biológico de coordinación motora debe funcionar así, dado que prácticamente nunca se repiten dos movimientos idénticos. La cantidad de memoria y el número de ciclos de aprendizaje son aspectos insuficientemente tratados en el trabajo original, de los cuales haremos posteriormente referencia más extensa.

Otro trabajo que no toca directamente el tema visión-control, pero que es digno de mención por su aplicación de dos métodos anteriores es el de Tham y Prager que usa una CMAC más una técnica de aprendizaje por refuerzo como la de Barto. ([TP92]). Su objetivo es llevar un brazo robot simulado de dos grados de libertad desde un punto a otro en un plano sin chocar con obstáculos puestos en ese plano; la salida de la CMAC son parámetros de una distribución de probabilidad que genera los torques para los joints. La señal de refuerzo indica el choque con algún obstáculo, y entra, al igual que las salidas de la CMAC, en el módulo que genera los torques. Aparte de su relación con los otros trabajos, es mencionado aquí porque su explicación de la implantación y el algoritmo de aprendizaje de la CMAC es incluso más clara que la del propio Albus, y fué la que usamos.

El último trabajo relacionado de modo tangencial con el nuestro es el de Atkeson. ([Atk91]). Su objetivo es predecir sobre un robot igualmente simulado los valores de los torques partiendo de los joints y sus derivadas. El método usado es una regresión lineal sobre las variables de entrada, sus cuadrados y sus productos, pero tomando sólo los puntos vecinos (en distancia euclídea) a aquel para el cuál queremos obtener respuesta, pesándolos apropiadamente de acuerdo a sus distancias. Como se verá en el capítulo siguiente, hemos usado un método similar, pero global, no local. El inconveniente del trabajo de Atkeson radica sobre todo en que es imposible calcular la regresión en un tiempo razonable que la haga útil para controlar el robot, dado el enorme volúmen de cálculo que supone. El autor compara sus resultados con una CMAC, y con una red neural entrenada por retropropagación, pero no da suficientes datos de los parámetros usados en los otros métodos como para poder evaluar debidamente la comparación.

Seguidamente veremos una serie de trabajos que abordan directamente el pro-

blema de la relación visión-control, aunque algunos con una perspectiva bastante diferente. Entre éstos son de destacar los de Geschke, Hashimoto, Westmore, Chaumette y Jang.

El primero ([Ges82]) aborda el problema de la inserción de una pieza en un orificio partiendo de posiciones imprecisas y cámaras calibradas, pero a las que se permite cierto error. Las ideas más importantes que sugiere son el uso de parámetros de la imágen (en su caso, la posición del centro del orificio) como aspectos relevantes computables con poco cálculo que se usan para guiar el brazo, y el empleo de realimentación visual para cancelar los errores en tiempo real, ideas que serán usadas más tarde por los siguientes autores nombrados.

El segundo de ellos ([HKEK91]) usa conceptos de la teoría de control para establecer un bucle de realimentación en el error entre la posición de un aspecto visual de la pieza que se pretende asir y el brazo robot. Este es precisamente la idea central que perseguimos, pero la diferencia es que Hashimoto sigue localizando el aspecto visual en coordenadas cartesianas, usando la diferencia cartesiana como error, y convirtiendo éstas a coordenadas del robot, con la consiguiente pérdida de tiempo y precisión, dado que emplea para la cámara una transformación ideal. Sus resultados son difíciles de evaluar, dado que son sobre simulación.

El tercero ([WW91]) va más allá, al añadir dos ideas nuevas: la cámara en cabeza del brazo, y el uso de un filtro de Kalman ([Oga87]) para predecir el estimador óptimo de la ventana de imágen donde se encontrará el aspecto visual que está siendo detectado, y ahorrar así tiempo de proceso. Esta vez los resultados son sobre robot real, y muestran que el brazo es capaz de seguir un punto, incluso aunque se mueva a baja velocidad. El único problema está en que en sigue usando transformaciones de coordenadas de la cámara a cartesianas, y de éstas a las de los joints, linealizando el jacobiano de la cinemática inversa, lo cual representa más cálculo del que sería deseable.

El cuarto ([CRE91]) es sustancialmente similar y también usa un robot real con buenos resultados. En lugar de un filtro de Kalnam, aquí la reducción del ruido se consigue mediante el uso de varios puntos relevantes que caractericen el objeto, en vez de uno sólo, lo cual aumenta la robustez. Se siguen usando coordenadas cartesianas.

Por fín, el quinto ([JB91]) es también muy similar, pero tiene de interés el que propone una definición matemática de "aspecto visual relevante" basada en la integral sobre la imágen de funciones de transformación; se dan los ejemplos de estas funciones que darían el área de un objeto, los momentos, la proyección sobre una línea, etc, y lo que es más importante, se propone un método para relacionar vectores de entrada conteniendo los valores de los aspectos relevantes con la salida, que son posiciones cartesianas para el punto terminal del brazo, por medio de la matriz jacobiana de la transformación de una a otras. Sin embargo, no indica nada sobre cómo medir o computar tales matrices, que da por conocidas.

A continuación mencionaremos trabajos que nos afectan más de cerca, bien por el

problema concreto, bien por la aproximación. En primer lugar, merece una especial mención el informe técnico sobre el Robot de Rochester, editado por C. Brown, y escrito por él mismo, D. Ballard y muchos otros colaboradores. ([BBB88]). Es un largo trabajo en el que se describe un sistema robótico con propósitos de investigación que está siendo desarrollado en la Universidad de Rochester. Aun ateniéndose en general al paradigma de la Robótica clásica para guiar los experimentos, se usan términos como reflejos, convergencia visual, sistema óculo-motor, etc. tomados de la biología que manifiestan su intención de desarrollar un robot orientado a la imitación de sistemas naturales con propósitos de investigación. En visión su contribución más original es el uso del filtro Cepstral, descrito originalmente por Yeshurun y Schwartz ([YS87]) que puede ser calculado en tiempo real sólo si se dispone de procesadores específicos para hacer la transformada de Fourier; este método permite la determinación de la disparidad entre las imágenes derecha e izquierda en un sistema estereoscópico, permitiendo de esta forma resolver el problema de la correspondencia. En control se plantean también la implantación del reflejo vestíbulo-ocular y del reflejo oculo-motor, aunque usando coordenadas cartesianas y ángulos de orientación para la cabeza del robot (donde las cámaras con papel de ojos van montadas) que deben ser obtenidos por calibración, lo cual es extremadamente difícil con cámaras móviles.

La última serie de artículos que vamos a comentar son los que, además de centrarse en nuestro mismo problema, usan técnicas similares, alguna de las cuales son precisamente objeto de implantación en esta tesis.

El primero es el de Kröse y Van der Smagt. ([KvdSG93]). Se usa en él una red neural de tipo perceptrón muticapa entrenado por retropropagación a la cual entran la diferencia entre la posición (en un plano) y el tamaño deseados para el objeto y los observados; la salida son los incrementos para los ángulos de las articulaciones. La implantación es sobre robot simulado, para el que dan resultados de 0.5 mm de precisión media, una vez entrenada la red. La cámara se fija también en cabeza del brazo. La originalidad aquí radica en el uso de un aspecto visual diferente a las coordenadas usuales, el tamaño, para actuar como entrada al sistema.

El segundo es el informe técnico de Hollinghurst y Cipolla, de la Univ. de Cambridge, sobre su sistema para localizar y coger objetos dirigido por visión ([HC93]). Esta vez el robot es real, y presenta ciertos aspectos especialmente interesantes: en primer lugar, las cámaras no son calibradas por los métodos usuales antes de empezar, sino que se calibran observando cuatro puntos dados de un objeto (en su caso, la propia mano del robot) del que se conocen las correspondencias entre cámara derecha e izquierda. Esto no es suficiente para dar las coordenadas cartesianas absolutas, pero sí referidas a un sistema que es transformación afín del cartesiano, y que por tanto preserva características como la colinealidad y la coplanariedad. Usando un modelo de perspectiva débil para las cámaras, y tomando alrededor de 100 puntos sobre el contorno del objeto, que deberán ser coplanarios (se suponen objetos planos y verticales) se recupera su orientación y posición, con el error introducido por el uso

de una perspectiva ideal. El punto clave es que ese error se cancela por realimentación visual, así como los errores debidos al modelo de la cinemática inversa del robot, y que esto se puede hacer en tiempo real, de modo que la mano puede seguir (y, si se desea, capturar) al objeto.

El tercero es un artículo de S. Lee ([Lee93]) que prueba sobre simulación ciertas características de un sistema visual idealizado de dos dimensiones, pero que considera como una aproximación aceptable a la filosofía de funcionamiento los sistemas reales. Intenta resolver el problema de la correspondencia a nivel de pixels mediante una red neural que toma sus entradas de los estímulos de ambas retinas simultáneamente, en un sistema de coordenadas basado en ángulos muy bien adaptado a la resolución del problema, que pasa su salida a otra red que se supone que es una representación, por supuesto no simbólica, del espacio de configuraciones visuo-motoras correctas del sistema ojos-brazo, el cual tiene también dos grados de libertad. Este artículo no da resultados muy concretos, pero su mérito está en saber escoger sistemas de coordenadas bien adaptados a la tarea que deben realizar, y arquitecturas neurales apropiadas para trabajar con los datos generados por tales sistemas.

El cuarto fue escrito por Clocksin y Moore ([CM89]) y es uno de los primeros en abordar directamente y sobre robot real (incidentalmente, el mismo modelo de robot que hemos usado) el problema de la coordinación visuo-motora en sistemas no calibrados. Por fín los autores prescinden completamente de coordenadas cartesianas, y usan como su sistema las dadas por las cámaras (en pixels) y las enviadas a las articulaciones (en cuentas). Con todas ellas forman un vector, que almacenan en un k-árbol. La ventaja de esta estructura es su breve tiempo de recuperación (proporcional a $n \log n$, siendo n el número de puntos almacenados) y su facilidad para ejecutar sobre ella un algoritmo de búsqueda del vecino más próximo. La idea es introducir primero un número razonable de puntos en la estructura, (periodo de práctica) para intentar después alcanzar un objetivo buscando el punto presente más próximo, y añadiéndole una perturbación aleatoria, que puede ser beneficiosa, o no. Si lo es, irá acercándonos al objetivo, pero aun si no lo es, aumentará el número de puntos conocidos, mejorando el comportamiento la próxima vez. Alternando periodos de práctica con periodos de ejecución se va ganando cada vez más precisión. Este trabajo está planteado sobre todo como un experimento en aprendizaje, y aun cuando no genere un método realmente práctico para mover el robot, es importante porque demuestra que con métodos simples se pueden conseguir resultados muy aceptables.

El quinto artículo que mencionamos es de Conkie y Consgtitvatana, ([CC90b]) y es el que realmente presta inspiración práctica más directa al método de la Teoría de la Red de Tensores, aunque no haga uso de esta formulación ni de los métodos que detallaremos en ella. Usa también un robot real planteando, como siempre, el problema de encontrar la función que relaciona coordenadas visuales y motoras. Al ser ésta desconocida, mide su jacobiano en el punto donde el robot se encuentre, haciendo movimientos que varíen una sóla de las articulaciones, y midiendo la variación

de cada coordenada visual. Este jacobiano es usado para mover el brazo hacia el punto deseado, multiplicándolo por la diferencia en coordenadas visuales entre el actual y el destino; el proceso se aplica iterativamente, hasta tener una diferencia nula. Este artículo fué escrito de modo muy intuitivo, y no demuestra la validez del método, ni prueba la convergencia. No obstante, sus resultados son otro ejemplo más de cómo la sencillez es muchas veces la mejor táctica. Un método muy similar, que usa realimentación visual en la orientación con la que se ve una línea para alinear la pinza de un robot con un bloque se usa en el siguiente artículo de Conkie ([CC90a]).

El siguiente trabajo que ha sido fuente directa para nosotros es el bastante conocido artículo de Ritter, Martinetz y Schulten que aborda exactamente el mismo problema, también con dos cámaras y tres grados de libertad para el brazo, pero sobre robot simulado ([MRS90]). En este caso la aplicación se establece por medio de una red neural autoorganizativa, el modelo de Kohonen, propuesto por éste basándose libremente en observaciones biológicas (véase el artículo original de Kohonen, [Koh82]). Las neuronas se organizan en una topología tridimensional, y cada una de ellas recibe las entradas (no hay capas de entrada o salida). Aquella cuyos pesos convolucionan mejor con la entrada (la ganadora) es actualizada, de modo que su vector de estado se mueve hacia esta entrada, pero también sus vecinas en la ordenación topológica lo son en el mismo sentido, aunque en menor proporción. De este modo la red se va organizando para semejar la topología inducida por los datos de entrada; cada neurona se "especializa" en una pequeña áerea del campo receptivo, y más neuronas se concentran allí donde la densidad de los datos de entrada es mayor. El uso que hacen R,M&S consiste en asociar a cada neurona ocho pesos, cuatro de los cuales representan las coordenadas visuales y cuatro las motoras, además de una matriz que representa el jacobiano de la transformación en ese punto. Los pesos y las matrices son actualizados siguiendo una regla de Widrow-Hoff. Los autores dan resultados sobre robot simulado en el cual después de 30000 ciclos de aprendizaje la función buscada se ha aprendido casi completamente con precisión suficiente sobre el área de trabajo. El uso de la simulación evita ciertos problemas serios que aparecen en el robot real y que serán comentados en el capítulo 8. Como comentario marginal, nótese que muchos de estos trabajos que actúan sobre simulación dan medidas para el error, bien en forma de error cuadrático medio, bien como distancia real, incluso con unidades, y dan también una cota superior para él. Esto nos parece manifiestamente incorrecto, dado que no tiene en cuenta ninguno de los errores experimentales reales, debidos al uso de un sistema de control no perfecto para el robot, o de cámaras con lentes reales, y placas digitalizadoras con error de localización de, al menos, un pixel.

Finalmente, se comentará que los trabajos que directamente sustentan el método de la Teoría de la Red de Tensores son dos artículos de los neurofisiólogos Pellionisz y Llinás ([PL80]) sobre coordinación entre el control ocular y el sentido del equilibrio en los que se da una formulación matemática para explicar cuál es el proceso de la información por el que el cerebelo ejecuta el reflejo vestíbulo-ocular. La explicación detallada se dará en el capítulo 6. De momento, baste decir que consiste

sustancialmente en una linealización, por zonas del espacio sensorial de entrada, de la función de transformación. Una implantación de esto sobre robot real, siguiendo la sugerencia de Pellionisz, puede encontrarse en mi MSc Thesis [Dom91], de la que el capítulo 6 es una reformulación más precisa. La diferencia teórica con el artículo biológico está en el uso de realimentación sobre la señal visual.

Un resumen a nivel general de los métodos que se usan en esta tesis, y de mi postura personal acerca de la Robótica y sus métodos puede encontrarse en [Dom93].

Como conclusión de toda esta revisión podemos decir que, aun sin haber sido exhaustivos, bastantes de los artículos con implicaciones directas en esta tesis están aquí, y como se ve, no son demasiados. El uso de la visión a bajo nivel no era muy común hasta ahora, pero se va extendiendo progresivamente; pensamos que la difusión de estas técnicas será tanto mayor cuanto más se abarate el hardware de visión, y que un campo importante relacionado con la conexión de estos comportamientos en un sistema de ensamblado eficiente, incluyendo descripciones visuales de las tareas, es una investigación que habrá que desarrollar.

Capítulo 5

Un método de regresión.

En este capítulo vamos a exponer la aplicación a nuestro caso de uno de los métodos más usuales para encontrar funciones desconocidas de las cuales conocemos su valor sólo en algunos puntos. Es el modelo de regresión, el más común entre los métodos llamados paramétricos. Este nombre proviene de que es el experimentador quien da un modelo para la dependencia de unas variables repecto a otras, indicando además qué variables son de cada tipo: independientes o dependientes; el modelo deja varios parámetros libres cuyo valor se trata de determinar. Esto es equivalente a un problema de búsqueda en un espacio que tuviese tantas dimensiones como parámetros, en el que cada punto correspondería a una posible solución.

Como inciso, digamos que se suele presentar a estos modelos como contrapuestos a los no paramétricos, de los que a veces se dice que no necesitan un conocimiento previo del problema. Esto no es estrictamente correcto; incluso los modelos de redes neurales, algoritmos genéticos, algoritmos de templado y similares requieren parámetros iniciales: en el caso de las redes, número de neuronas, conexión entre ellas y regla de actualización, por lo menos. En el caso de los algoritmos genéticos, forma de codificación, una expresión para la función de evaluación, elección de operadores de cruzamiento y mutación, etc. Veremos con un ejemplo en el capítulo 8 que, aunque desgraciadamente la metodología de elección de estos parámetros no suele estar muy bien establecida, son absolutamente determinantes de la solución (o al menos, del área del espacio de búsqueda que contendrá la solución) que se termine encontrando, o incluso de si se la encontrará o no (convergencia o no del algoritmo). No obtante, es cierto que en general no se suelen hacer suposiciónes sobre dónde se va a encontrar la solución, o sobre cuál sea la forma de las fronteras de decisión (en el caso de la clasificación) o cuál sea la forma funcional de una transformación (este es nuestro caso). Muchas veces la estructura del algoritmo lleva implícitas limitaciones a la función que se le puede implantar, por ejemplo, un perceptrón aprende funciones lineales exclusivamente. Pero tales limitaciones no suelen aparecer explícitas, así que es difícil saber cuáles son, o si las hay. Un reciente artículo (véase [Fun89]) ha probado que un perceptrón multicapa de al menos tres capas con funciones lineales en sus capas de entrada y salida, funciones sigmoidales en sus capas intermedias y un número suficiente de neuronas puede aproximar una función arbitraria con la precisión que se desee, aunque no está probado que ningún algoritmo concreto converja a tal aproximación en un tiempo finito.

Por contraposición, los modelos paramétricos presentan la ventaja de que no hay un proceso de convergencia, y muchos de ellos garantizan que la solución obtenida es óptima respecto a algún criterio preestablecido. El que esa optimalidad sea o no suficiente para nuestros propósitos es otro problema. Como dijimos, escogimos la regresión porque es de los más comunes, y además existen multitud de herramientas computacionales ampliamente usadas en Estadística que la ejecutan. El objetivo, más que usar el método como tal, es dar una idea de como esta clase de aproximaciones se comporta en el mismo problema con respecto a las otras que veremos después.

El primer paso es, obviamente, elegir el modelo. El segundo es estimarlo, o sea, dar valores para sus parámetros, y el tercero es evaluarlo; evidentemente una forma será probarlo sobre un conjunto de datos de entrada, cosa que haremos en el capítulo 9, pero también podemos considerar a nuestros datos como los valores que sendas variables aleatorias definidas sobre los espacios de entrada y salida pueden tomar, en cuyo caso los parámetros serán obviamente funciones de las entradas y las salidas. A tales funciones se las llama en Estadística estimadores. Volveremos sobre ellos enseguida al hacer la identificación de estos conceptos en nuestro caso, pero lo importante es que la estadística provee también de medios para evaluar la bondad de los estimadores; en lo que sigue se usarán los llamados estimadores de máxima verosimilitud, (ver [DeG88]) que si la muestra es grande, suelen proporcionar una excelente estimación.

Como fín de la introducción, hay que aclarar que las entradas pueden considerarse variables aleatorias, producto de una distribución desconocida, o pueden darse como valores fijos (datos de diseño), pero en cualquier caso esto no altera ninguna de las afirmaciones que se presentarán después.

Particularizando a nuestro caso, el espacio de entrada está constituído por muestras de las lecturas de los valores en pixels del punto terminal del brazo, que organizaremos como un vector de cuatro componentes, (x_L, y_L, x_R, y_R), cada una de las cuales puede tomar un valor en el intervalo $[0, 511]$. La salida está constituída por otro vector (zed, sho, elb, yaw) que contendrá las coordenadas de las cuentas para los motores. En ausencia de cualquier idea acerca de cuál pueda ser la función de transformación

$$\begin{aligned} f &: \Re^4 \longrightarrow \Re^4 \\ (x_l, y_l, x_r, y_r) &\longrightarrow f(x_l, y_l, x_r, y_r) = (zed, sho, elb, yaw) \end{aligned}$$

intentamos, como aproximación más simple, un modelo lineal, o sea,

$$\begin{aligned} zed &= \beta_{11}x_l + \beta_{12}y_l + \beta_{13}x_r + \beta_{14}y_r \\ sho &= \beta_{21}x_l + \beta_{22}y_l + \beta_{23}x_r + \beta_{24}y_r \\ elb &= \beta_{31}x_l + \beta_{32}y_l + \beta_{33}x_r + \beta_{34}y_r \\ yaw &= \beta_{41}x_l + \beta_{42}y_l + \beta_{43}x_r + \beta_{44}y_r \end{aligned}$$

o, escrito en forma matricial,

$$\begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix} = \beta \cdot \begin{pmatrix} x_l \\ y_l \\ x_r \\ y_r \end{pmatrix} \tag{5.1}$$

El procedimiento para determinar la matriz β puede interpretarse tanto desde un punto de vista estadístico como de puro cálculo numérico. Por ser más simple, comenzaremos por el segundo. Supongamos, en principio, cada una de las componentes (zed, sho, elb, yaw) por separado. Si hemos tomado medidas de los valores de estas componentes para un número suficiente de puntos, digamos n, podemos organizarlas en cuatro vectores $\mathbf{Y}$ de tamaño n×1 que supondremos relacionados cada uno con todas las variables de entrada mediante una matriz β_i de tamaño q×1, siendo q el número de variables de entrada, en nuestro caso 4. Los datos de entrada se organizarán en otra matriz, ésta n×q, que llamaremos $\mathbf{X}$. Así pues,

$$\mathbf{Y} = \mathbf{X} \cdot (\beta_i) \tag{5.2}$$

Como se puede ver, este es un sistema de n ecuaciones con q incógnitas. Como el número de muestras (n) es generalmente mucho mayor que q, y ningún vector de muestra es, en general, proporcional a otro, este sistema será incompatible. No obstante, se puede buscar una "seudosolución" que sea óptima en algún sentido. El criterio que hemos escogido es el más usual: la minimización del error cuadrático, es decir, que

$$S^2 = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{q} \beta_{ij} x_j)^2 \tag{5.3}$$

sea un mínimo. Esto es equivalente a decir que

$$S = \|\mathbf{Y} - \beta\mathbf{X}\| \tag{5.4}$$

sea un mínimo.

La matriz β que cumple esta condición es

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{5.5}$$

siendo $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ la llamada seudoinversa izquierda de $\mathbf{X}$. La demostración de esto puede verse en [Oga87]. El método es válido sólo en el caso de que $\mathbf{X}^T\mathbf{X}$ sea invertible, lo cual es equivalente a que $\mathbf{X}$ sea de rango q. En caso contrario, Atkeson aporta una solución llamada "ridge regresion method" (ver [Atk91]) basada en añadir filas a la matriz, pero que no fue necesaria en nuestro caso.

El valor para β se obtuvo con el programa SPSS, y será explicitamente dado en el capítulo 9, junto con los resultados numéricos. El problema estuvo en que la

comprobación sobre un conjunto de test, como se verá allí arrojó resultados muy pobres, con error medio en la posición del punto terminal de más de 60mm, lo cual es obviamente inaceptable. El problema está claramente en una elección incorrecta del modelo. Para comprobar esto, véanse gráficas del comportamiento de cada una de las variables dependientes respecto a cada una de la independientes en las figuras que siguen.

En cada una de estas gráficas se ven varias líneas de puntos debido a que cada valor de la variable independiente presenta, como es lógico, varios valores de la dependiente, en función de las combinaciones de las otras. Lo idóneo hubieran sido gráficas de cada variable dependiente en función de todas las independientes, pero esto es obviamente imposible, dado que necesitaríamos un espacio de cinco dimensiones para dibujarlas. En cualquier caso, se advierte de las mostradas que la dependencia lineal de la variable zed puede ser muy razonablemente asumida, pero que en los demás casos esta suposición es claramente incorrecta. Dado que el resto de las curvas semejan parábolas, una aproximación razonable puede ser suponer una dependencia cuadrática en todas las variables independientes, es decir, cada salida será combinación lineal de las entradas, de sus cuadrados y de sus productos, más por supuesto, un término independiente. Así tendríamos, p. ej,.

$$\begin{aligned} sho \;=\; & a_1 x_l + a_2 y_l + a_3 x_r + a_4 y_r + \\ & a_5 x_l^2 + a_6 y_l^2 + a_7 x_r^2 + a_8 y_r^2 + \\ & a_9 x_l y_l + a_{10} x_l x_r + a_{11} x_l y_r + a_{12} y_l x_r + a_{13} y_l y_r + a_{14} x_r y_r + a_{15} \end{aligned} \tag{5.6}$$

Como se ve, esto es equivalente a un modelo lineal, pero considerando un espacio de características ampliado, donde las entradas serían vectores de la forma $(x_l, y_l, x_r, y_r, x_l^2, y_l^2, x_r^2, y_r^2, x_l y_l, x_l x_r, x_l y_r, y_l x_r, y_l y_r, x_r y_r, 1)$, con lo cual todo lo dicho anteriormente sigue siendo válido, y la ecuación 5.2 subsiste, siendo ahora $\mathbf{X}$ una matriz n×15 y $\beta_\mathbf{i}$ una 15×1, que se determina igualmente según la ecuación 5.5. Los resultados esta vez fueron mejores, con un error medio en la posición de unos 27 mm, lo cual aún es grande, pero no excesivamente teniendo en cuenta lo restringido del modelo.

Seguidamente presentaremos esta aproximación desde un punto de vista estadístico, de lo cual obtendremos interesantes conclusiones respecto a la evaluación de los resultados obtenidos. En lo sucesivo, los términos que aparezcan en cursiva se entienden con el significado que les da la Estadística, y su definición precisa puede encontrarse en [MKB79]. El nuestro es el problema que se conoce como *regresión múltiple* (múltiple en el sentido de que la única variable de salida depende de varias variables de entrada). La ecuación del modelo sería esta vez

$$\mathbf{y} = \mathbf{x} \cdot (\beta_\mathbf{i}) + \mathbf{u} \tag{5.7}$$

siendo $\mathbf{x} = (x_1, \cdots, x_n)$ los valores de las variables independientes (*regresores*) que suponemos dados. Para las n observaciones que hayamos hecho,

$$\mathbf{Y} = \mathbf{X} \cdot (\beta_\mathbf{i}) + \mathbf{U} \tag{5.8}$$

donde $\mathbf{Y}$ es la matriz $n \times 1$ con las diferentes observaciones de la variable independiente, $\mathbf{X}$ es la matriz $n \times q$ con los vectores $\mathbf{x}$ antes definidos puestos uno sobre otro, y $\mathbf{U}$ es una matriz $n \times 1$ que da cuenta de las desviaciones entre el valor dado por el modelo y el realmente obtenido, tanto las debidas a la propias limitaciones del modelo, como al ruido.

Supongamos que tenemos las n observaciones, $(y_i, \mathbf{x_i})$ y que éstas verifican las siguentes hipótesis:

- *Independencia*: las variables $y_1, \cdots, y_n$ deben ser independientes. Esto significa que el valor observado para una de ellas no influye en los observados para las otras.
- *Normalidad*: Cada variable y_i debe tener distribución normal. Como, según dijimos, las $\mathbf{x_i}$ son fijadas, esto equivale a que las u_i sean normales.
- *Homocedasticidad*: Las variables y_i tienen la misma varianza, σ^2

Las dos primeras suposiciones son razonables. La tercera es un poco más fuerte, pero en cualquier caso los modelos que presentamos son suficientemente robustos, en el sentido de que si las hipótesis se verifican sólo parcialmente el modelo sigue dando una estimación razonable de los parámetros. Partiendo de todas estas premisas, se puede demostrar (ver [DeG88], pág. 574-575) que la expresión del *estimador máximamente verosímil* para $\beta_\mathbf{i}$ es precisamente

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{5.9}$$

Nótese que, aunque la notación es la misma que la de la ecuación 5.5, los significados de los símbolos no son exactamente los mismos. No obstante, por la misma razón que se dió allí, este valor minimiza los cuadrados de los errores (aquí, $\sum_{i=1}^{n} u_i^2$), y se conoce por ello como Estimador Mínimo Cuadrático.

Hasta aquí hemos aplicado el método a una sola variable de salida. En el caso de varias (4, para nosotros) puede usarse la ecuación 5.7 pero dando como $\mathbf{Y}$ un vector que tuviese los demás vectores columna organizados como una sóla, uno sobre otro. En ese caso, las demás matrices se ampliarían apropiadamente, y todos los resultados anteriores son válidos.

Otra posible visión de los métodos estadísticos es la *regresión multivariante*, en la que la fórmula principal sería

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{U} \tag{5.10}$$

en la que ahora **Y** es la matriz observada de p variables (como antes, p=4, que son (zed, sho, elb, yaw)) sobre n indivíduos, y que tiene por tanto tamaño n×p. **X** es la matriz conocida, n×q, siendo otra vez q el número de variables de entrada, en nuestro caso, 15; β es la matriz q×p de parámetros de regresión desconocidos, y **U** es una matriz n×p de valores aleatorios no observada cuyas filas son incorreladas (es decir, suponemos que el error cometido en un punto no tiene ninguna relación con el cometido en cualquier otro punto). No obstante, el error cometido en una de las variables puede depender del cometido en otra, y por ello podemos calcular una *matriz de covarianzas* σ, no necesariamente diagonal, definida como

$$\sigma_{ij} = E[(u_i - \mu_{u_i})(u_j - \mu_{u_j})] \tag{5.11}$$

Es claro que $\sigma_{ii} = Var(u_i)$.

Vamos a suponer ahora que cada fila de **U** es el valor de una *variable aleatoria* **U** que sigue una distribución normal multivariante con vector de medias nulas y *matriz de covarianzas* σ, constante. La suposición de media nula viene a decir que los errores se distribuyen alrededor del origen, lo cual es razonable de admitir. Formalmente, las hipótesis son las mismas que ya dimos para el caso univariante (independencia, normalidad y homocedasticidad), pero adaptadas a vectores en lugar de escalares. En ese caso, la *función de verosimilitud* de **Y** en términos de los parámetros β y σ será la *función densidad de probabilidad* para la distribución normal de **U** (es decir, de $\mathbf{Y} - \mathbf{X}\beta$), que es

$$f(Y \mid \beta, \sigma) = \frac{1}{\sqrt{(2\pi \mid \sigma \mid)^n}} e^{-\frac{tr\{(Y-X\beta)\sigma^{-1}(Y-X\beta)^T\}}{2}} \tag{5.12}$$

aunque se suele dar más habitualmente su logaritmo,

$$\ell(Y \mid \beta, \sigma) = -\frac{1}{2} n \log \mid 2\pi\sigma \mid - \frac{1}{2} tr\{(Y - X\beta)\sigma^{-1}(Y - X\beta)^T\} \tag{5.13}$$

Se puede probar (ver [DeG88] en pág. 598) que los valores para β y σ que maximizan esta expresión son

$$\begin{aligned} \hat{\beta} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \\ \hat{\sigma} &= \frac{1}{n}\mathbf{Y}^T(\mathbf{I} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)\mathbf{Y} \end{aligned} \tag{5.14}$$

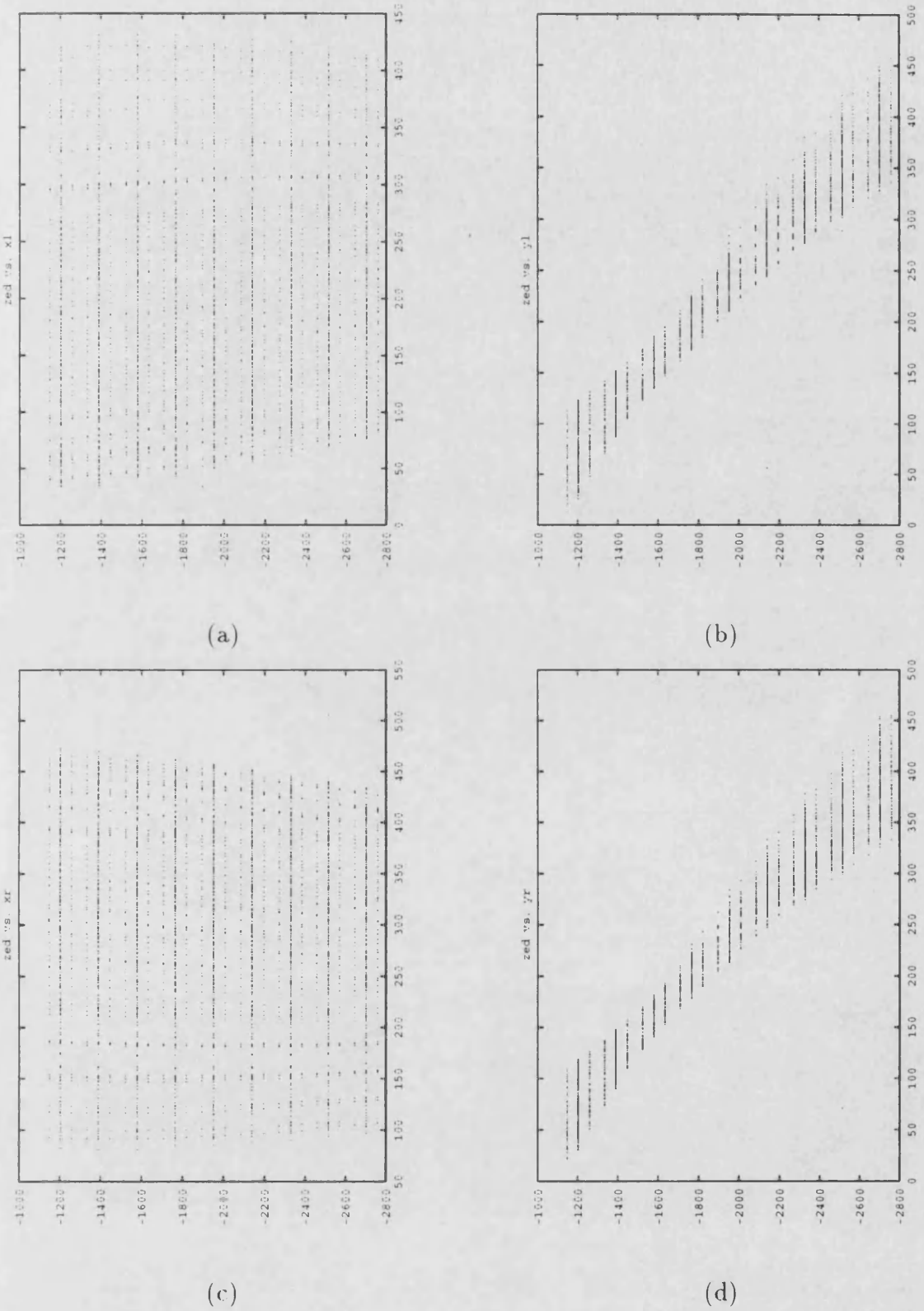
y son, por tanto, los *estimadores máximamente verosímiles* de β y σ. Además, presentan varias propiedades interesantes:

- $\hat{\beta}$ es un *estimador insesgado* de β, es decir, $E(\hat{\beta}) = E(\beta)$
- El vector de medias de los errores, $\hat{\mathbf{U}}$, tiene todas sus componentes nulas.

- $\hat{\beta}$ y $\hat{\mathbf{U}}$ siguen una *distribución normal multivariante.*
- $\hat{\beta}$ es independiente de $\hat{\mathbf{U}}$.

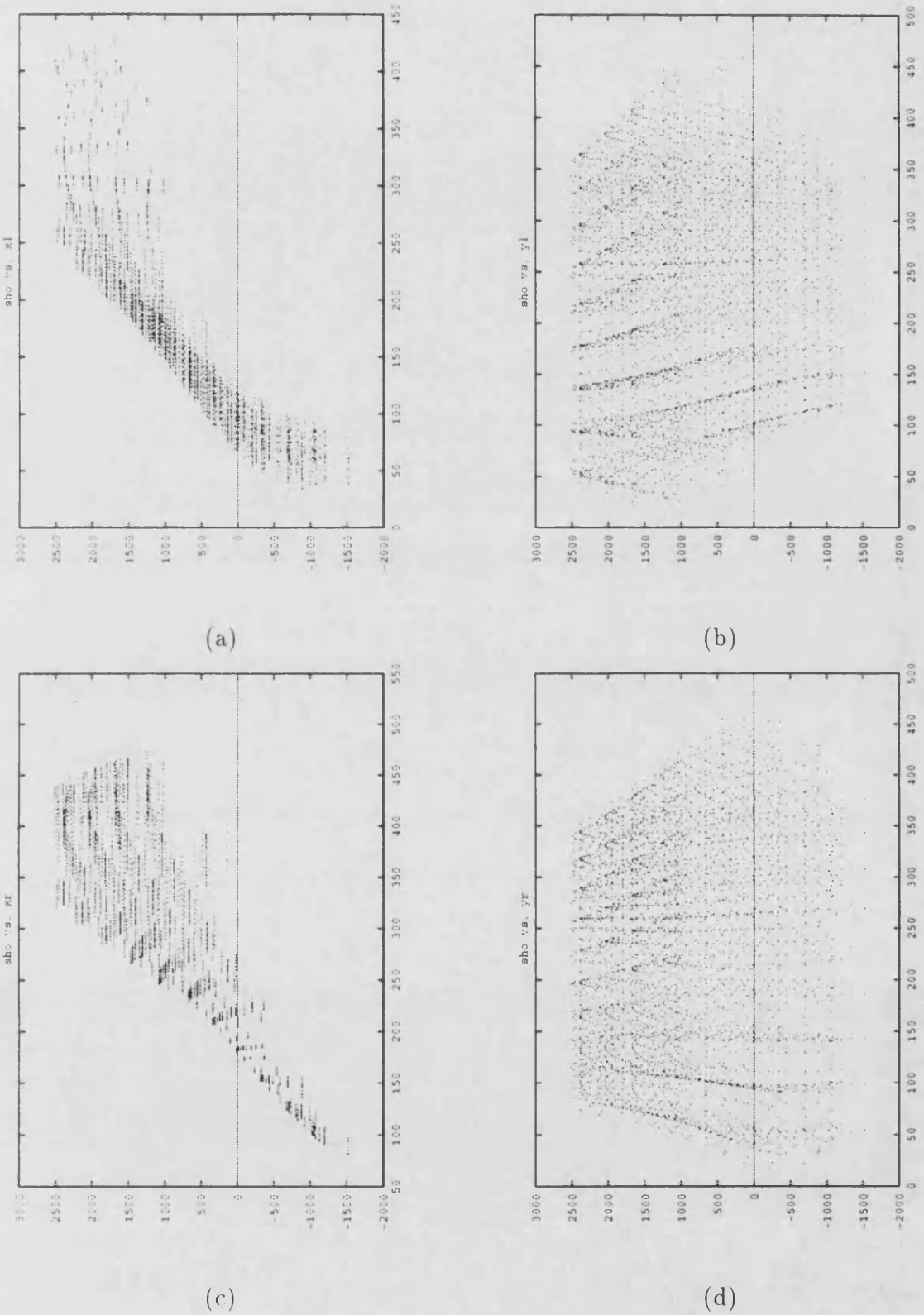
Es interesante hacer notar que en otros trabajos se sugieren métodos más complejos para determinar los parámetros desconocidos de la matriz β. En particular, el que comentamos en la revisión bibliográfica de Atkeson ([Atk91]), que usan adaptaciones de la matriz β al punto particular del espacio en el que se pretende hallar las variables de salida, pesando adecuadamente los puntos de las medidas de acuerdo a su distancia euclídea al punto requerido. Pero esto supone que la métrica del espacio de entrada es localmente euclídea (aunque no adviertan de ello) y además implica recalcular la matriz β para cada petición; esto hace que tales métodos sólo se puedan aplicar sobre simulaciones, y con computadores paralelos, y aun así el error que reportan no compensa su uso, y hace otros métodos que veremos después preferibles. Esta es la razón sustancial que tuvimos para la elección de un método de regresión elemental.

Como conclusión, decir que, independientemente de la mayor o menor precisión en los resultados, tenemos por lo menos la garantía de que los valores dados para los parámetros son los mejores que se podían escoger (o, al menos, aquellos en los que, dados nuestros datos y un criterio de evaluación del error, podemos tener más confianza) y por ello son la elección óptima.



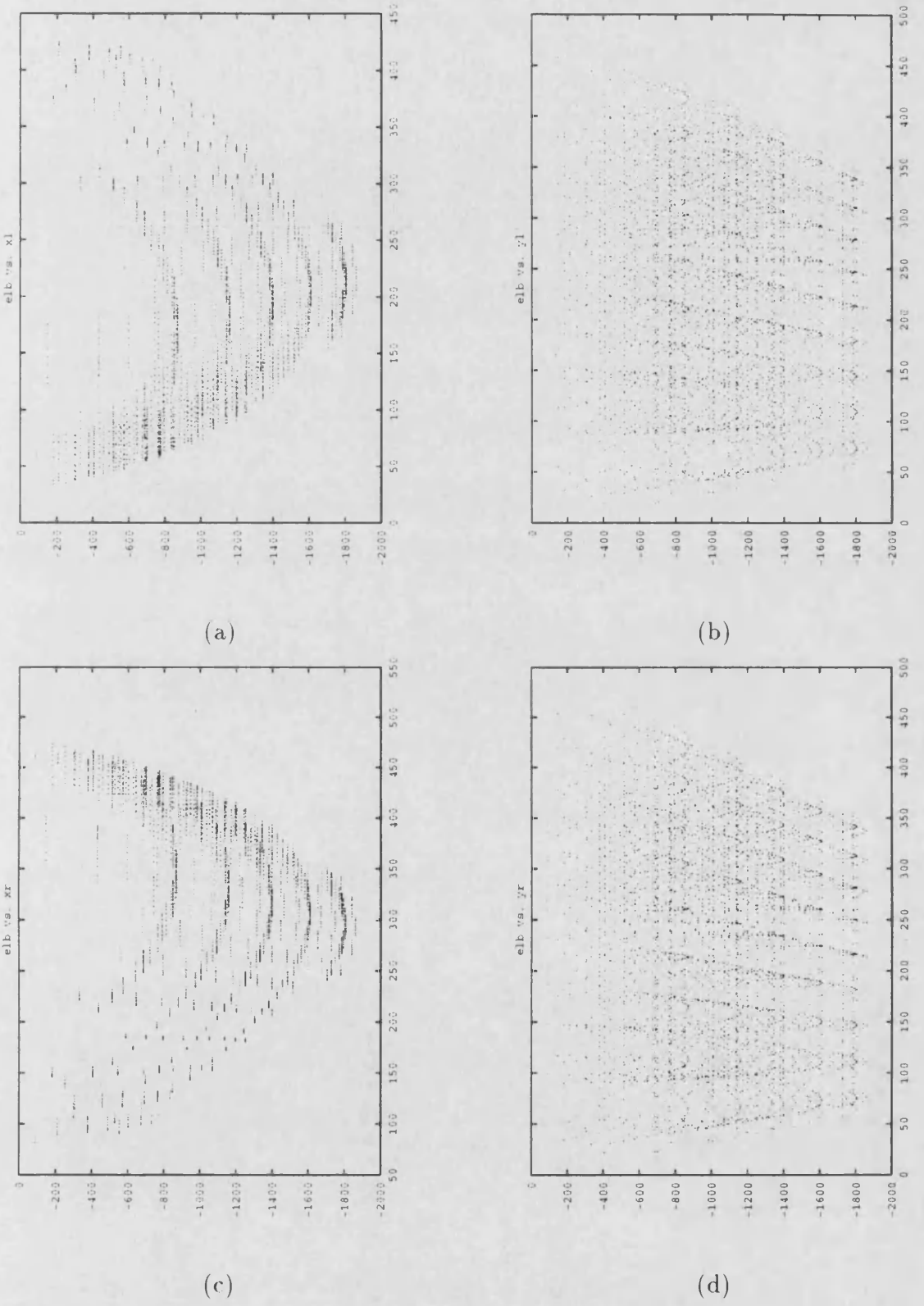
(a) (b) (c) (d)

Figura 5.1: Variable zed respecto a cada coordenada visual



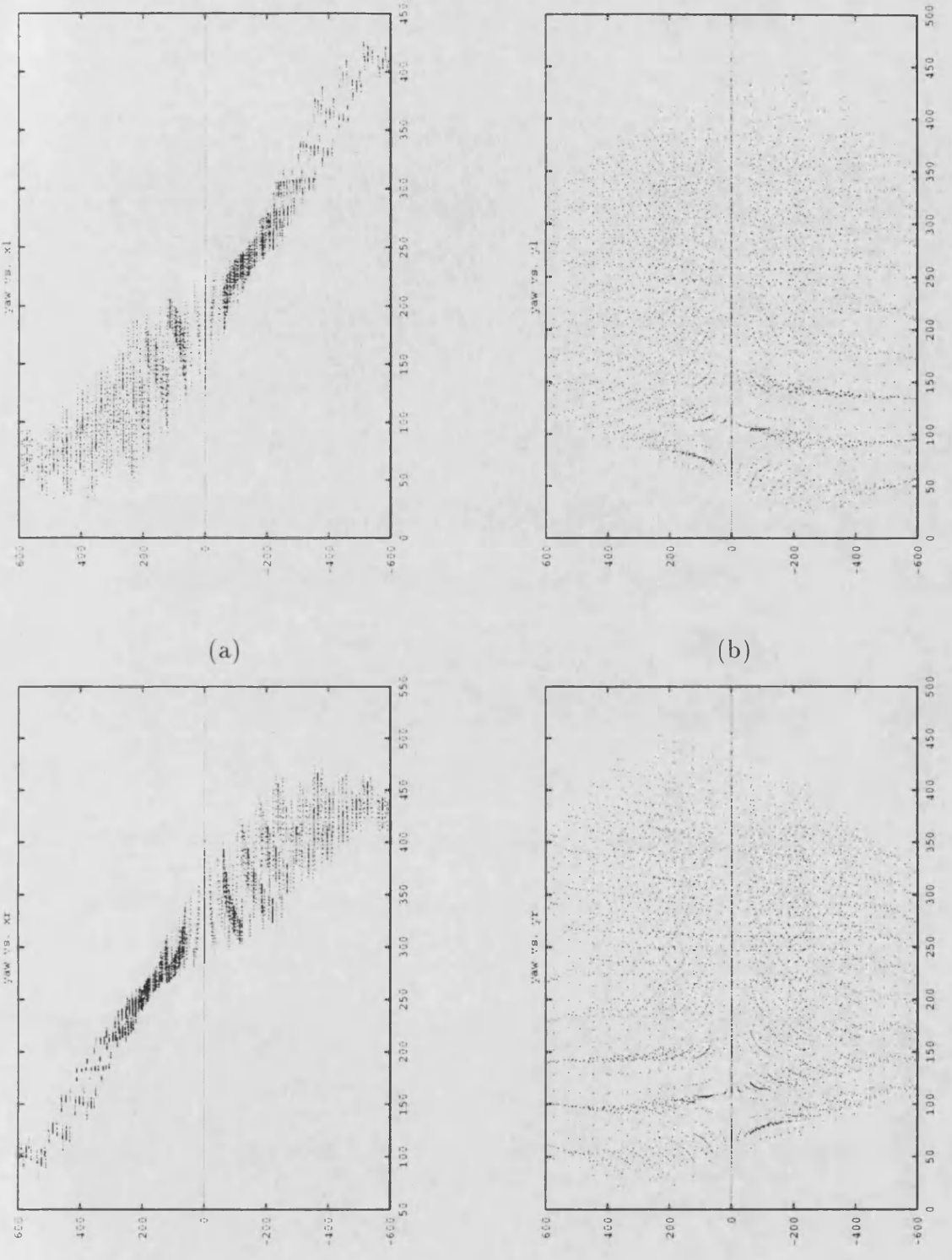
(a) (b) (c) (d)

Figura 5.2: Variable sho respecto a cada coordenada visual



(a) (b) (c) (d)

Figura 5.3: Variable elb respecto a cada coordenada visual



(a) (b) (c) (d)

Figura 5.4: Variable yaw respecto a cada coordenada visual

Capítulo 6

La Teoría de la Red de Tensores.

El segundo método que vamos a exponer, la Teoría de la Red de Tensores, tiene, como ya comentamos en la introducción, un especial sabor biológico. Antes de empezar a detallar su aplicación en el robot explicaremos este punto. La teoría fué, como se dijo en la revisión bibliográfica, desarrollada por los neurofisiólogos Pellionisz y Llinás para explicar el reflejo vestíbulo-ocular. Un experimento simple que puede ayudar a entenderlo consiste en mover la mano rapidamente delante los ojos, manteniendo la cabeza fija y tratando de conservar la vista fija en los dedos. Estos, naturalmente, se ven borrosos porque los ojos no pueden seguirlos con suficiente rapidez. Sin embargo, si mantenemos esta vez la mano fija y movemos la cabeza con la misma rapidez observaremos los dedos nítidos, porque esta vez los ojos sí pueden seguirlos. Aunque en ambos casos el movimiento relativo es el mismo, en el segundo caso la correción de la posición es instantánea (o por lo menos, suficientemente rápida como para seguir las variaciones del movimiento).

La explicación de esto es que existe una relación directa entre la información de orientación y las señales para los músculos de los ojos. La orientación del cráneo es constantemente muestreada por los canales semicirculares, órganos que se encuentran en el oído interno y que están compuestos por tres pequeños tubos de forma toroidal con sus planos longitudinales aproximadamente perpendiculares. Estos tubos contienen un líquido en el que flotan partículas calcáreas (los otolitos) que al moverse excitan alguna de las terminaciones nerviosas que tapizan el interior de los canales. Según cuál de ellas se haya excitado cada canal manda un señal diferente. Estas señales, según Pellionisz y Llinás, van directamente al cerebelo, que las transforma en señales de control para los los tres pares de músculos que controlan cada ojo. Estos músculos son un tanto particulares, dado que pueden estar tanto bajo control consciente (podemos decidir hacia dónde queremos mirar) como inconsciente (el caso de este reflejo).

Lo más interesante del trabajo antedicho es que, una vez descrito el mecanismo anatómico, los autores parten de una serie de medidas de la actividad eléctrica de los nervios que portan la información del sentido del equilibrio junto con la de los nervios que controlan los movimientos del globo ocular, y encuentran que para pequeñas variaciones existe una aplicación lineal que relaciona unas y otras señales, pero que tal aplicación no es la misma para todos los valores de la señal sensorial de entrada.

El modelo que dan para esto es suponer un invariante físico (un tensor) que puede expresarse en dos (o más) sistemas de coordenadas diferentes, y que es la aplicación entre los espacios de entrada y salida. Evidentemente, según en qué sistema de coordenadas se expresen las entradas y salidas, las componentes del tensor variarán, así como su carácter (covariante o contravariante). Pellionisz y Llinás arguyen que las coordenadas sensoriales son covariantes, dado que son absolutamente independientes (la variación del eje sobre el cual se mida una de ellas no implica variación de las demás) mientras que las coordenadas motoras son contravariantes (la variación del eje de una provoca que todas las demás deban variar, para conservar el mismo invariante físico). La figura 6.1 explicará la diferencia entre uno y otro tipo de coordenadas.

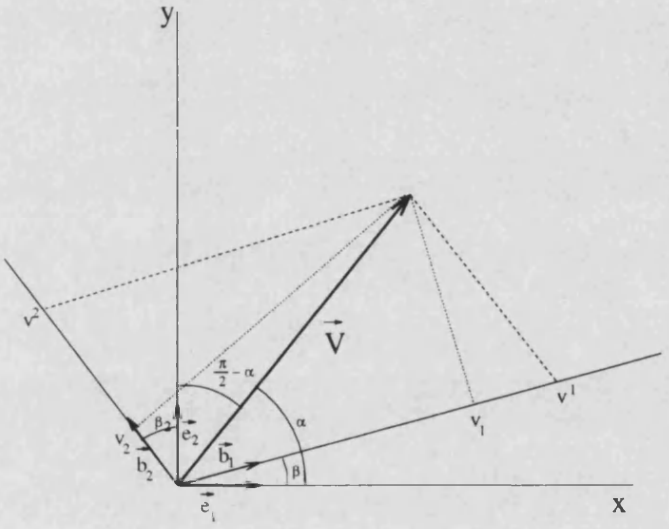


Figura 6.1: Coordenadas covariantes y contravariantes del vector V

En esta figura, X e Y son los ejes cartesianos, con vectores básicos $\vec{e}_1$ y $\vec{e}_2$. Sea un sistema no ortogonal, B, con vectores básicos $\vec{b}_1$ y $\vec{b}_2$, que forman repectivamente ángulos β_1 y β_2 con los ejes X e Y, y sea $\vec{V}$ un vector dado. Para hallar sus componentes respecto al sistema B podemos proyectar el vector perpendicularmente a cada eje, o sobre uno de los ejes, paralelamente al otro. En el primer caso tenemos las coordenadas covariantes, que representaremos con subíndices, v_1 y v_2, y en segundo caso, las contravariantes, denotadas por superíndices, v^1 y v^2. Es obvio de la figura que el girar uno sólo de los ejes, digamos el b_2, (lo cual es equivalente a variar el ángulo β_2) la proyección perpendicular de $\vec{V}$ sobre b_1 (es decir, v_1) no se altera para nada. Por eso se dice que las coordenadas covariantes son independientes. Se puede probar (ver apéndice A) que la matriz de transformación de coordenadas cartesianas a covariantes es

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos\beta_1 & \sin\beta_1 \\ -\sin\beta_2 & \cos\beta_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{6.1}$$

que representaremos como

$$v_i = \mathcal{M} \cdot \mathcal{X} \Rightarrow \mathcal{X} = \mathcal{M}^{-1} \cdot v_i \tag{6.2}$$

y que la matriz análoga para las contravariantes es

$$\begin{pmatrix} v^1 \\ v^2 \end{pmatrix} = \frac{1}{\cos(\beta_1 - \beta_2)} \begin{pmatrix} \cos\beta_2 & \sin\beta_2 \\ -\sin\beta_1 & \cos\beta_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{6.3}$$

es decir,

$$v^i = (\mathcal{M}^{-1})^T \cdot \mathcal{X} \Rightarrow \mathcal{X} = \mathcal{M}^T \cdot v^i \tag{6.4}$$

Igualmente se puede probar que las coordenadas covariantes están relacionadas con las contravariantes por el llamado tensor métrico contravariante, cuya expresión es

$$v^i = g^{ij} v_j \quad siendo \quad g^{ij} = (\mathcal{M}^{-1})^T \cdot \mathcal{M}^{-1} \tag{6.5}$$

o, inversamente, la del tensor métrico covariante,

$$v_i = g_{ij} v^j \quad siendo \quad g_{ij} = \mathcal{M} \cdot \mathcal{M}^T \tag{6.6}$$

Otros resultados interesantes son la expresión de la distancia entre dos puntos, que viene dada por

$$d^2 = v^i g_{ij} v^j \tag{6.7}$$

si conocemos las coord. contravariantes, o

$$d^2 = v_i g^{ij} v_j \tag{6.8}$$

si conocemos las covariantes.

Nótese que las coordenadas cartesianas son simultáneamente covariantes y contravariantes, y que el tensor métrico en este sistema es el tensor identidad.

Para el caso en que tengamos dos sistemas de coordenadas diferentes, $B_1 y B_2$, cuyas transformaciones a las coordenadas cartesianas vengan dadas respectivamente por

$$v^i_{(k)} = \mathcal{M}_{(k)} \cdot \mathcal{X} \Rightarrow \mathcal{X} = \mathcal{M}^{-1}_{(k)} \cdot v^i_{(k)} \quad k = 1, 2. \tag{6.9}$$

y por

$$v_{(k)i} = (\mathcal{M}^{-1}_{(k)})^T \cdot \mathcal{X} \Rightarrow \mathcal{X} = \mathcal{M}^T_{(k)} \cdot v^{(k)i} \quad k = 1, 2. \tag{6.10}$$

podemos escribir todas las posibles transformaciones de un sistema a otro como

$$\begin{aligned} v_{1i} = \quad & \mathcal{M}_1 \cdot {\mathcal{M}_2}^{-1} \cdot v_{2i} && = \mathcal{M}_1 \cdot {\mathcal{M}_2}^T \cdot v_2^i \\ v_1^i = \quad & (\mathcal{M}_1^{-1})^T \cdot {\mathcal{M}_2}^{-1} \cdot v_{2i} && = (\mathcal{M}_1^{-1})^T \cdot {\mathcal{M}_2}^T \cdot v_2^i \\ v_{2i} = \quad & \mathcal{M}_2 \cdot {\mathcal{M}_1}^{-1} \cdot v_{1i} && = \mathcal{M}_2 \cdot {\mathcal{M}_1}^T \cdot v_1^i \\ v_2^i = \quad & (\mathcal{M}_2^{-1})^T \cdot {\mathcal{M}_1}^{-1} \cdot v_{1i} && = (\mathcal{M}_2^{-1})^T \cdot {\mathcal{M}_1}^T \cdot v_1^i \end{aligned} \tag{6.11}$$

La identificación que Pellionisz y Llinás hacen consiste en suponer que las coordenadas sensoriales del sentido del equilibrio son covariantes, puesto que alterar la dirección de uno de los canales semicirculares modificaría exclusivamente su propia lectura, pero no la de los demás, mientras que las coordenadas motoras son contravariantes, dado que cambiar la dirección de tracción de uno de los pares de músculos del ojo obligaría a los otros dos pares a modificar su tensión para mantener la orientación.

Obsérvese también que cualesquiera coordenadas pueden estar sobredeterminadas, es decir, puede haber mayor número de las estrictamente necesarias para determinar el invariante físico.

La "red" de tensores de la que ellos hablan podría haber sido formalizada como un campo tensorial sobre una variedad, del cual se conoce el valor sólo en ciertos puntos, pero de hecho las posibles propiedades de continuidad o diferenciabilidad de este campo no son relevantes para explicar la transformación.

Para la aplicación de todas estas ideas a nuestro caso haremos la identificación de nuestros espacios de entrada y salida, y sus coordenadas. Nuestro invariante físico es la posición en el espacio del punto terminal, que puede ser representada por un vector de n componentes.

El espacio de entrada (expresión de este invariante en coordenadas sensoriales) es obviamente el conjunto de coordenadas visuales de dicho punto terminal visto por cada cámara, medidas en pixels; las denotaremos (X_L, Y_L, X_R, Y_R). Obsérvese que están sobredeterminadas, dado que en principio tres serían suficientes, por estar las cámaras no alineadas. Aunque un conjunto de tres cualesquiera serían válidas, cuando de verdad sea necesario desechar una (como veremos en el método de la CMAC) es conveniente conservar las dos X, debido a la particular posición que hemos escogido para las cámaras, en la cual las Y_L e Y_R resultan tener valores similares para cada punto. Estas coordenadas son covariantes; la razón está en que las de una cámara son absolutamente independientes de las de la otra, y en cuanto a la X y la Y de la misma cámara, se puede argumentar (ver Apéndice A) que en un sistema óptico con simetría rotacional también lo son. Otra característica es que, como puede suponerse, la sobredeterminación lleva a interdependencia en el caso de que los valores provengan de medidas reales, es decir, la dimensión real de este conjunto de cuatro coordenadas es tres, o, lo que es lo mismo, los valores de las mismas que corresponden a puntos físicos forman una variedad de dimensión 3 inmersa en un espacio de 4 dimensiones.

Por otra parte, el espacio de salida está formado por el conjunto de cuentas para los joints, que son números enteros comprendidos en ciertos límites. Hemos escogido trabajar también con cuatro grados de libertad, uno más de los necesarios para la posición, pero dos menos de los necesarios para posición más orientación. Ello es porque la determinación de la orientación no era un objetivo prioritario, pero además porque para determinarla hubiéramos necesitado una cámara más (es decir, al menos seis coordenadas visuales, correspondientes a los seis grados de libertad

de un objeto sin restricciones). De todas formas, las cuatro coordenadas motoras escogidas tienen un propósito especial. Son zed (eje vertical del robot), shoulder (eje vertical del motor que une el primer segmento del brazo con la columna), elbow (eje vertical del motor que une los dos segmentos del brazo) y yaw (eje vertical del motor que gira la mano, al final del segundo segmento). Véanse la figura 3.2 y la figura 3.3. El pitch y el roll, que controlan, junto con el yaw, la orientación de la mano han sido fijados respectivamente a -90^o y 0^o; la idea es que el yaw sea tal que mantenga siempre la mano orientada a roll de 0^o, con lo cual, y para esta posición de las cámaras, los dos diodos son visibles. Podría calcularse analíticamente el yaw a partir de shoulder y elbow para cumplir esta condición, pero hemos decidido que sea también el método el que lo de, para probar que las coordenadas motoras también pueden estar sobredeterminadas. Además de esto, una forma restringida de orientación puede conocerse viendo la orientación visual de la mano, calculada a partir de las coordenadas de cada una de las manchas de los diodos. La línea que las une se puede usar para girar la mano de modo que se la vea paralela a cualquier otra línea localizada en algún objeto de la imagen, con el añadido de que tal orientación estaría relacionada sólo con el roll, que es independiente de las otras coordenadas de cuentas y que no tiene limitación en su valor angular (puede girar media vuelta en cada sentido). Por último, hacer notar que las coordenadas motoras dadas son contravariantes; véase la demostración en el apéndice A.

Seguidamente determinaremos qué medidas tenemos que tomar para conocer la información necesaria para construir el campo tensorial requerido, y cómo debemos tomarlas.

Consideraremos el sistema de coordenadas visuales como el sistema B_1 al que aludimos antes, el sistema de coordenadas motoras como B_2, y las coordenadas cartesianas como $\mathcal{X}$. Entonces, y según la cuarta fórmula de la ecuación 6.11, podemos escribir

$$v_2^i = (\mathcal{M}_2^{-1})^T \cdot \mathcal{M}_1{}^{-1} \cdot v_{1i} \tag{6.12}$$

lo cual es nuestro caso es

$$\begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix} = \begin{pmatrix} \quad \mathcal{M}_2^{-1} \quad \end{pmatrix}^T \cdot \begin{pmatrix} \quad \mathcal{M}_1^{-1} \quad \end{pmatrix} \cdot \begin{pmatrix} X_L \\ Y_L \\ X_R \\ Y_R \end{pmatrix} \tag{6.13}$$

Como dijimos antes, esta fórmula es válida sólo localmente, por tanto, para aplicarla, lo que deberemos hacer es buscar un punto próximo (en coordenadas visuales) al que tenemos, y en el cual conozcamos, por haberlos medido, los tensores de transformación, y el valor de los joints; si denotamos los valores de las magnitudes en este punto, tensores incluídos, con subíndice 0, tendremos que para un punto cualquiera, con coordenadas sensoriales (X_L, Y_L, X_R, Y_R)

y coordenadas motoras (zed, sho, elb, yaw) se cumple que

$$\begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix} = \begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix}_0 + \begin{pmatrix} \quad \mathcal{M}_2^{-1} \quad \end{pmatrix}_0^T \cdot \begin{pmatrix} \quad \mathcal{M}_1^{-1} \quad \end{pmatrix}_0 \cdot \begin{pmatrix} X_L - X_{L0} \\ Y_L - Y_{L0} \\ X_R - X_{R0} \\ Y_R - Y_{R0} \end{pmatrix} \tag{6.14}$$

Falta determinar los valores de los tensores $(\mathcal{M}_2^{-1})^T$ y ${\mathcal{M}_1}^{-1}$ en un número suficiente de puntos del espacio, de modo que cualquier punto arbitrario pueda tener uno de éstos suficientemente próximo. Para ello recurrimos a las fórmulas 6.2 y 6.4, interpretándolas como desarrollos en serie de Taylor alrededor del origen de las funciones que relacionan las coordenadas cartesianas con las visuales y con los joints respectivamente. Efectivamente, si $(zed, sho, elb, yaw) = f(x, y, z)$, siendo f una función

$$f(x, y, z) : \Re^3 \longrightarrow \Re^4$$

que puede ser interpretada como devolviendo un vector cuyas cuatro componentes serían cuatro funciones

$$Zed(x, y, z),\ Sho(x, y, z),\ Elb(x, y, z),\ Yaw(x, y, z) : \Re^3 \longrightarrow \Re$$

entonces

$$\begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix} \simeq \begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix}_0 + \begin{pmatrix} \quad Jf \quad \end{pmatrix}_0 \cdot \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} \tag{6.15}$$

donde Jf es la matriz jacobiana de la transformación f, definida como

$$Jf = \begin{pmatrix} \frac{\partial Zed}{\partial x} & \frac{\partial Zed}{\partial y} & \frac{\partial Zed}{\partial z} \\ \frac{\partial Sho}{\partial x} & \frac{\partial Sho}{\partial y} & \frac{\partial Sho}{\partial z} \\ \frac{\partial Elb}{\partial x} & \frac{\partial Elb}{\partial y} & \frac{\partial Elb}{\partial z} \\ \frac{\partial Yaw}{\partial x} & \frac{\partial Yaw}{\partial y} & \frac{\partial Yaw}{\partial z} \end{pmatrix} \tag{6.16}$$

Identificando las ecuaciones 6.15 y 6.4, si tomamos el origen en $(zed, sho, elb, yaw)_0$, es obvio que el tensor $(\mathcal{M}_2^{-1})^T$ que estabamos buscando es justamente Jf.

Por otra parte, si la relación entre $\mathcal{X}$ y las (X_L, Y_L, X_R, Y_R) la expresamos igualmente como una función $g : \Re^3 \longrightarrow \Re^4$, nuevamente interpretada como un vector de cuatro funciones escalares, $XL(x, y, z)$, $YL(x, y, z)$, $XR(x, y, z)$, $YR(x, y, z)$, entonces

$$\begin{pmatrix} X_L \\ Y_L \\ X_R \\ Y_R \end{pmatrix} \simeq \begin{pmatrix} X_L \\ Y_L \\ X_R \\ Y_R \end{pmatrix}_0 + \begin{pmatrix} \quad Jg \quad \end{pmatrix}_0 \cdot \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} \tag{6.17}$$

y esta vez la matriz jacobiana es

$$Jg = \begin{pmatrix} \frac{\partial XL}{\partial x} & \frac{\partial XL}{\partial y} & \frac{\partial XL}{\partial z} \\ \frac{\partial YL}{\partial x} & \frac{\partial YL}{\partial y} & \frac{\partial YL}{\partial z} \\ \frac{\partial XR}{\partial x} & \frac{\partial XR}{\partial y} & \frac{\partial XR}{\partial z} \\ \frac{\partial YR}{\partial x} & \frac{\partial YR}{\partial y} & \frac{\partial YR}{\partial z} \end{pmatrix} \tag{6.18}$$

Por comparación de las ecuaciones 6.17 y 6.2 concluímos que esta vez $\mathcal{M}_1 = Jg$, por tanto el tensor buscado, ${\mathcal{M}_1}^{-1}$ es $(Jg)^{-1}$. Podría argüirse que es absurdo medir el tensor Jg, para invertirlo después. Podría haberse medido directamente $(Jg)^{-1}$, es decir, ${\mathcal{M}_1}^{-1}$ usando la relación inversa de la ecuación 6.2. Pero es que ésta involucraría las parciales de las coordenadas cartesianas con respecto a las visuales, las cuales, como veremos inmediatamente, no es posible medir.

Describiremos ahora el procedimiento de medición. Para medir las derivadas parciales que necesitamos usaremos una aproximación discreta, suponiendo, por ejemplo, que

$$\frac{\partial X_L}{\partial x} \simeq \frac{\Delta X_L}{\Delta x}$$

y lo mismo para las demás. Los principales problemas que vamos a encontrar al tomar esta aproximación son:

- ¿Para cuántos puntos deberían calcularse las derivadas?
- ¿Cómo deberían estar distribuídos estos puntos?
- ¿Cuáles deberían ser los incrementos para obtener buenos resultados? (es decir, los valores de $\Delta X_L, \Delta X_R, \Delta Y_L, \Delta Y_R, \Delta x, \Delta y, \Delta z$)

La respuesta a la primera pregunta depende de cuán suave sea la variación de las funciones, cuánto error estemos dispuestos a admitir, cuánto tiempo podamos permitirnos gastar en la toma de medidas, y de cuánto espacio podamos disponer para almacenar los resultados. Dada la memoria disponible en nuestros ordenadores (tanto en la estación de trabajo, 32Mb, como en el 486, 8Mb) la última consideración es irrelevante, como veremos después. Para obtener una solución de compromiso entre los otros factores se escogió como criterio que ninguna de las componentes de la matriz de las derivadas podría diferir más del 10% de la correspondiente componente de esa matriz para cualquiera de sus puntos vecinos. Esto asegurará un grado aceptable de linealidad, y si el número de puntos así obtenido es razonable, el problema está resuelto.

Sobre la distribución de los puntos, tomaremos una red ortoédrica, en ausencia de información sobre la forma de las funciones que vamos a medir, ya que llevar el punto terminal del brazo a los puntos de una red regular es fácil de programar y entender.

El tamaño de los incrementos para las derivadas se tomó de 15 mm, lo cual se hizo mirando los resultados de varios incrementos en varios puntos, y tomando el

mayor de ellos para el que la aproximación lineal seguía siendo válida. El hecho de tomar el mayor es consecuencia del intento de que medidas en torno al mismo punto estén lo más separadas que se pueda para minimizar el error relativo (dado que el absoluto es siempre el mismo, $\pm$ 1 pixel).

De acuerdo a todas estas consideraciones, la separación entre cada par de puntos para cumplirlas se escogió de 50 mm. El tamaño del área de trabajo viene determinado por la limitación del robot (lo que puede alcanzar), y como dijimos es un ortoedro de $250 \times 800 \times 400$ mm, de modo que podremos poner en él $5 \times 17 \times 8 = 680$ puntos, aunque no absolutamente todos ellos son alcanzables en cada configuración del robot. Véase la figura 6.2 con un diagrama de la distribución de puntos.

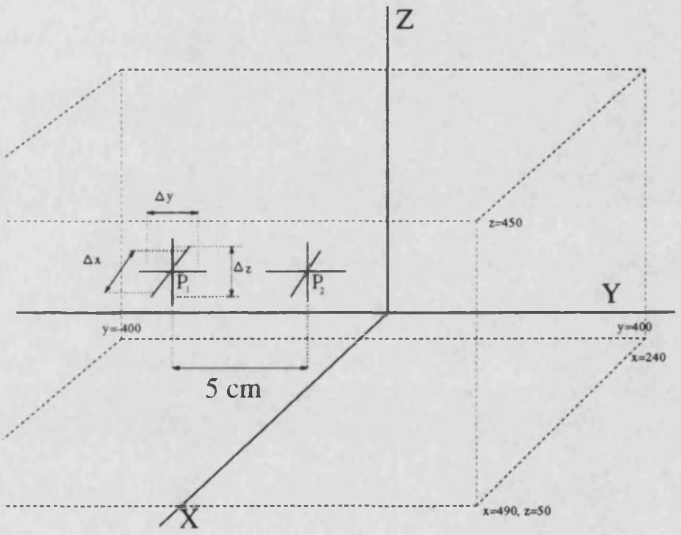


Figura 6.2: La distribución de las medidas en el espacio de trabajo.

Para el cálculo de las derivadas parciales debemos medir los valores de la posición del punto terminal del brazo en el espacio de las cámaras, tanto en el punto escogido como en seis puntos más rodeándole, en las direcciones x,y,z. Ello es porque para el cálculo de la derivada parcial respecto a una de las variables las otras han de permanecer constantes, lo cual sabemos que ocurre al usar x,y,z, pero no estaríamos seguros en otro sistema de coordenadas. Esta es la razón de que no podamos medir otras derivadas, como antes indicamos, y de que tengamos que invertir el tensor obtenido cuando lo que necesitamos es ir del espacio de cámaras al cartesiano. En nuestro caso, y dado que la expresión del tensor es una matriz 4x3 no cuadrada, usaremos en lugar de la inversa, la seudoinversa de Moore-Penrose. Este es un algoritmo incluído en el programa MATLAB, que se usó para tal propósito, y dada una matriz $\mathcal{M}$, de tamaño $m \times n$, devuelve otra matriz, $\mathcal{MI}$ de tamaño $n \times m$ tal que

$$\mathcal{M} \cdot \mathcal{MI} \cdot \mathcal{M} = \mathcal{M}$$

y

$$\mathcal{MI} \cdot \mathcal{M} \cdot \mathcal{MI} = \mathcal{MI}$$

siendo además $\mathcal{MI} \cdot \mathcal{M}$ y $\mathcal{M} \cdot \mathcal{MI}$ hermíticas (en el caso de matrices reales, esto equivale a simétricas). Se puede ver que este algoritmo devuelve $(\mathcal{M} \cdot \mathcal{M}^T)^{-1}\mathcal{M}^T$ en el caso de matrices n×m con n>m, lo cual es habitualmente usado para calcular seudoinversas, siempre que $(\mathcal{M} \cdot \mathcal{M}^T)^{-1}$ sea no singular, dado que minimiza el módulo del error, $\| V - Jg \cdot \mathcal{X} \|$. (véase [Oga87]).

Respecto al tiempo de las mediciones, como dijimos en el capítulo 3, cada una requiere 1.9 segundos, y dado que son 7, tendremos 13.3 segundos, a lo que hay que sumar el tiempo para ir de una posición a otra, que en total resultó ser de unos 7 segundos, da unos 20 segundos por punto. Por ser el número total de puntos no superior a 680, el tiempo total para la toma de todas las medidas es no superior a 13600 segundos, o sea, menos de 4 horas.

Veamos ahora cómo esta formulación tensorial originariamente propuesta puede interpretarse de un modo más sencillo. Si en la ecuación 6.14 la escribimos con los valores que acabamos de obtener para las matrices de transformación tendremos

$$\begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix} = \begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix}_0 + \begin{pmatrix} \frac{\partial Zed}{\partial x} & \frac{\partial Zed}{\partial y} & \frac{\partial Zed}{\partial z} \\ \frac{\partial Sho}{\partial x} & \frac{\partial Sho}{\partial y} & \frac{\partial Sho}{\partial z} \\ \frac{\partial Elb}{\partial x} & \frac{\partial Elb}{\partial y} & \frac{\partial Elb}{\partial z} \\ \frac{\partial Yaw}{\partial x} & \frac{\partial Yaw}{\partial y} & \frac{\partial Yaw}{\partial z} \end{pmatrix}_0 \cdot \begin{pmatrix} \frac{\partial x}{\partial X_L} & \frac{\partial x}{\partial X_L} & \frac{\partial x}{\partial X_L} & \frac{\partial x}{\partial X_L} \\ \frac{\partial y}{\partial X_L} & \frac{\partial y}{\partial X_L} & \frac{\partial y}{\partial X_L} & \frac{\partial y}{\partial X_L} \\ \frac{\partial z}{\partial X_L} & \frac{\partial z}{\partial X_L} & \frac{\partial z}{\partial X_L} & \frac{\partial z}{\partial X_L} \end{pmatrix}_0 \cdot \begin{pmatrix} X_L - X_{L0} \\ Y_L - Y_{L0} \\ X_R - X_{R0} \\ Y_R - Y_{R0} \end{pmatrix} \tag{6.19}$$

lo cual, multiplicando, puede escribirse como

$$\begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix} = \begin{pmatrix} zed \\ sho \\ elb \\ yaw \end{pmatrix}_0 + \begin{pmatrix} \\ H \\ \\ \end{pmatrix}_0 \cdot \begin{pmatrix} X_L - X_{L0} \\ Y_L - Y_{L0} \\ X_R - X_{R0} \\ Y_R - Y_{R0} \end{pmatrix} \tag{6.20}$$

siendo

$$(H)_0 = \begin{pmatrix} \frac{\partial Zed}{\partial x}\frac{\partial x}{\partial X_L} + \frac{\partial Zed}{\partial y}\frac{\partial y}{\partial X_L} + \frac{\partial Zed}{\partial z}\frac{\partial z}{\partial X_L} & \frac{\partial Zed}{\partial x}\frac{\partial x}{\partial Y_L} + \frac{\partial Zed}{\partial y}\frac{\partial y}{\partial Y_L} + \frac{\partial Zed}{\partial z}\frac{\partial z}{\partial Y_L} & \cdots \\ \frac{\partial Sho}{\partial x}\frac{\partial x}{\partial X_L} + \frac{\partial Sho}{\partial y}\frac{\partial y}{\partial X_L} + \frac{\partial Sho}{\partial z}\frac{\partial z}{\partial X_L} & \frac{\partial Sho}{\partial x}\frac{\partial x}{\partial Y_L} + \frac{\partial Sho}{\partial y}\frac{\partial y}{\partial Y_L} + \frac{\partial Sho}{\partial z}\frac{\partial z}{\partial Y_L} & \cdots \\ \frac{\partial Elb}{\partial x}\frac{\partial x}{\partial X_L} + \frac{\partial Elb}{\partial y}\frac{\partial y}{\partial X_L} + \frac{\partial Elb}{\partial z}\frac{\partial z}{\partial X_L} & \frac{\partial Elb}{\partial x}\frac{\partial x}{\partial Y_L} + \frac{\partial Elb}{\partial y}\frac{\partial y}{\partial Y_L} + \frac{\partial Elb}{\partial z}\frac{\partial z}{\partial Y_L} & \cdots \\ \frac{\partial Yaw}{\partial x}\frac{\partial x}{\partial X_L} + \frac{\partial Yaw}{\partial y}\frac{\partial y}{\partial X_L} + \frac{\partial Yaw}{\partial z}\frac{\partial z}{\partial X_L} & \frac{\partial Yaw}{\partial x}\frac{\partial x}{\partial Y_L} + \frac{\partial Yaw}{\partial y}\frac{\partial y}{\partial Y_L} + \frac{\partial Yaw}{\partial z}\frac{\partial z}{\partial Y_L} & \cdots \end{pmatrix}$$

$$\begin{pmatrix} \cdots & \frac{\partial Zed}{\partial x}\frac{\partial x}{\partial X_R} + \frac{\partial Zed}{\partial y}\frac{\partial y}{\partial X_R} + \frac{\partial Zed}{\partial z}\frac{\partial z}{\partial X_R} & \frac{\partial Zed}{\partial x}\frac{\partial x}{\partial Y_R} + \frac{\partial Zed}{\partial y}\frac{\partial y}{\partial Y_R} + \frac{\partial Zed}{\partial z}\frac{\partial z}{\partial Y_R} \\ \cdots & \frac{\partial Sho}{\partial x}\frac{\partial x}{\partial X_R} + \frac{\partial Sho}{\partial y}\frac{\partial y}{\partial X_R} + \frac{\partial Sho}{\partial z}\frac{\partial z}{\partial X_R} & \frac{\partial Sho}{\partial x}\frac{\partial x}{\partial Y_R} + \frac{\partial Sho}{\partial y}\frac{\partial y}{\partial Y_R} + \frac{\partial Sho}{\partial z}\frac{\partial z}{\partial Y_R} \\ \cdots & \frac{\partial Elb}{\partial x}\frac{\partial x}{\partial X_R} + \frac{\partial Elb}{\partial y}\frac{\partial y}{\partial X_R} + \frac{\partial Elb}{\partial z}\frac{\partial z}{\partial X_R} & \frac{\partial Elb}{\partial x}\frac{\partial x}{\partial Y_R} + \frac{\partial Elb}{\partial y}\frac{\partial y}{\partial Y_R} + \frac{\partial Elb}{\partial z}\frac{\partial z}{\partial Y_R} \\ \cdots & \frac{\partial Yaw}{\partial x}\frac{\partial x}{\partial X_R} + \frac{\partial Yaw}{\partial y}\frac{\partial y}{\partial X_R} + \frac{\partial Yaw}{\partial z}\frac{\partial z}{\partial X_R} & \frac{\partial Yaw}{\partial x}\frac{\partial x}{\partial Y_R} + \frac{\partial Yaw}{\partial y}\frac{\partial y}{\partial Y_R} + \frac{\partial Yaw}{\partial z}\frac{\partial z}{\partial Y_R} \end{pmatrix}_0$$

pero, de acuerdo a las reglas del cálculo de derivadas parciales, si una función, digamos Zed depende de otras variables, en nuestro caso, x, y, z, cada una de las cuales es función, entre otras, de X_L, entonces

$$\frac{\partial Zed}{\partial X_L} = \frac{\partial Zed}{\partial x}\frac{\partial x}{\partial X_L} + \frac{\partial Zed}{\partial y}\frac{\partial y}{\partial X_L} + \frac{\partial Zed}{\partial z}\frac{\partial z}{\partial X_L} \tag{6.21}$$

y lo mismo para los demás elementos de la matriz, luego

$$(H)_0 = \begin{pmatrix} \frac{\partial Zed}{\partial X_L} & \frac{\partial Zed}{\partial X_R} & \frac{\partial Zed}{\partial Y_L} & \frac{\partial Zed}{\partial Y_R} \\ \frac{\partial Sho}{\partial X_L} & \frac{\partial Sho}{\partial X_R} & \frac{\partial Sho}{\partial Y_L} & \frac{\partial Sho}{\partial Y_R} \\ \frac{\partial Elb}{\partial X_L} & \frac{\partial Elb}{\partial X_R} & \frac{\partial Elb}{\partial Y_L} & \frac{\partial Elb}{\partial Y_R} \\ \frac{\partial Yaw}{\partial X_L} & \frac{\partial Yaw}{\partial X_R} & \frac{\partial Yaw}{\partial Y_L} & \frac{\partial Yaw}{\partial Y_R} \end{pmatrix}_0 \tag{6.22}$$

De este modo, la ecuación 6.20 es claramente un desarrollo en serie de Taylor de la función final que estamos buscando alrededor del punto $(zed, sho, elb, yaw)_0$, y el método equivale a linealizar la aplicación por zonas alrededor de puntos conocidos. La expresión de la matriz H es, por otra parte, la misma que Conkie y Chongtitvatana dan sin demostración en su artículo. (ver [CC90b]). Esto indica que a la hora de tomar las medidas, deberemos tomar los valores de las coordenadas visuales y de los joints en los puntos que indicamos para calcular las matrices de sus derivadas, y luego seudoinvertir una de ellas y multiplicar el resultado por la otra, obteniendo de este modo la matriz H, que será lo que almacenaremos, juntamente con el punto alrededor del cual es válida, tanto en coordenadas visuales como motoras. No es necesario almacenar información de la posición cartesiana, (salvo que se desee por alguna razón especial), ni tampoco de las matrices de transformación a cartesianas, dado que tales transformaciones han sido subsumidas. No obstante, puede ser útil para ciertas aplicaciones almacenar la matriz de transformación cámara-cartesianas, porque como vimos al principio de este capítulo, ella misma, multiplicada por su traspuesta, nos dará el tensor métrico covariante, que halla distancias reales a partir de diferencias en pixels, para un área determinada no muy grande alrededor del punto donde fué medida, lo cual podría ser interesante en tareas de ensamblado.

Ahora que ya conocemos exactamente qué datos hay que almacenar para ejecutar la tarea podemos hacer una estimación de la memoria necesaria para el almacenamiento. Para cada punto, tendremos las coordenadas visuales, que son 4 enteros, los motoras, otros tantos, y la matriz H, compuesta por 16 números reales. Esto da, con el tamaño de variables de mi compilador, 80 bytes; por tanto los menos de 680 puntos necesitarán no más de 55 Kb. En este método la memoria total empleada para una precisión dada es lineal con el volúmen del espacio de trabajo. Es fácil ver que incluso en un robot industrial, con un espacio de trabajo de hasta ocho veces el usado aquí, e incluso con puntos separados por la mitad de distancia (lo cual multiplica por 8 el número de puntos) la memoria total sería del órden de menos de 4 Mb, perfectamente manejable hoy por cualquier microordenador de uso común.

La última parte de esta capítulo explicará como se usan los datos una vez almacenados para controlar el brazo. La idea es que en una tarea real se localizaría un punto visual relevante, como una esquina, o un punto vacío en el espacio sobre un objeto, del cual se encontrarían sus coordenadas visuales. Se supone que queremos que el punto terminal del brazo se vea en tales coordenadas. Para ello, buscamos entre los puntos que tenemos almacenados cuál es el más próximo a nuestra petición, y recuperamos su tensor H, y el valor de sus coordenadas motoras. Hallamos la

diferencia en pixels para cada coordenada visual entre el punto almacenado y el pedido, y multiplicamos esta diferencia por la matriz H, obteniendo la diferencia en joints, que añadida al valor de joints almacenado nos deberá llevar, suponemos, al punto requerido. Esto es exactamente aplicar la ecuación 6.20. Como se verá en los experimentos, hay siempre un error entre el punto al que llegamos y aquel al que queremos ir, que será evaluado y analizado en el capítulo 9. Pero un dato muy interesante es que este error, como siempre, dado como diferencia en coordenadas visuales, puede ser nuevamente multiplicado por la matriz H, obteniendo nuevas diferencias en joints que sumadas a la última posición nos acercarán cada vez más al objetivo. Esto es claramente un proceso de realimentación, con importantes consecuencias:

- Da cuenta del error intrínseco debido a la linealización de la función de transformación, que, aun localmente, nunca es estrictamente lineal.

- Es capaz de suplir la falta de precisión debida a la toma de un número insuficiente de medidas, siempre que la distancia al punto sea aún razonable

- Es capaz de reducir hasta cierto punto los errores debidos a una desalineación de una o ambas cámaras.

Este proceso convergerá, como se verá en los resultados, en un número pequeño de iteraciones. Para probar esto véase que lo que hacemos es definir una sucesión de puntos que cumplen la siguiente relación:

Siendo:
x_d la entrada del destino
$f(x_d)$ la salida del destino (el punto que queremos obtener)
x_0 un punto próximo a x_d
$f(x_0)$ el valor de la función en x_0 (conocido)
$(\frac{df}{dx})_{x=x_0}$ el valor de su derivada, también conocido

entonces

$$f(x_d) \simeq a_0 = f(x_0) + (\tfrac{df}{dx})_{x=x_0}(x_d - x_0)$$
$$f(x_d) \simeq a_1 = a_0 + (\tfrac{df}{dx})_{x=x_0}(x_d - f^{-1}(a_0))$$
$$\vdots$$
$$f(x_d) \simeq a_n = a_{n-1} + (\tfrac{df}{dx})_{x=x_0}(x_d - f^{-1}(a_{n-1}))$$

Si consideramos $f^{-1}(a_n) = x_n$ como la entrada al sistema y $a_n = y_n$ como su salida, entonces la ecuación de estado es

$y_n = y_{n-1} + K(x_d - x_n) = y_{n-1} - Kx_n + Kx_d$

La diferencia entre la salida en dos instantes consecutivos es, pues,

$y_n - y_{n-1} = Kx_n - Kx_d$

A efectos de comprobación de la estabilidad, la constante Kx_d no tiene influencia, y podemos considerar igualmente la ecuación

$y_n - y_{n-1} = Kx_n$

para la cual, tomando transformada Z, obtenemos

$Y(z) - z^{-1}Y(z) = KX(z)$

lo que da como función de transferencia en lazo abierto de un integrador, $\frac{z}{z-1}$ que como sabemos hace el sistema estable para cualquier entrada finita. Si el sistema es estable, tendrá límite, y por métodos de análisis matemático se puede probar (ver [AA86]) que ese límite es justamente $f(x_d)$.

La robustez respecto a variaciones no muy grandes consiste en que, aun cuando habíamos condicionado los resultados a cámaras fijas, leves golpes a una o ambas de ellas no alteran el resultado final (el brazo sigue llegando al punto pedido), aun cuando lo hace con más ciclos de realimentación. Resultados de este experimento se presentarán también el el capítulo 9.

Hay todavía dos detalles más de implantación que deben ser comentados respecto a la búsqueda del punto más próximo en coordenadas visuales a nuestro punto objetivo de entre todos los que tenemos almacenados. Más próximo significa a menor distancia real; pero, dado que no conocemos la distancia real (por no conocer las coordenadas cartesianas) no podríamos calcular esto. Hay dos soluciones posibles:

- Suponer que la métrica del espacio de cámaras es localmente euclídea (al menos, como aproximación) y calcular la distancia entre dos puntos como
  $d = \sqrt{(X_L^1 - X_L^0)^2 + (Y_L^1 - Y_L^0)^2 + (X_R^1 - X_R^0)^2 + (Y_R^1 - Y_R^0)^2}$
- Almacenar junto a los demás datos para el punto el tensor métrico del espacio de cámaras, y aplicar la ecuación 6.8

En nuestro caso se optó por la primera solución, porque los resultados experimentales fueron prácticamente iguales con ambas, lo cual indica que la suposición sobre aproximación de la métrica a la euclídea a nivel local es aquí correcta; pero no tiene por qué serlo en todos los casos, de modo que se advierte al lector que otras implantaciones del método podrían requerir la solución estricta.

Finalmente, hay un aspecto de eficiencia que debe ser destacado. Para encontrar el punto a la mínima distancia una posibilidad sería ir recorriendo todos los almacenados, calcular su distancia a nuestra entrada, y quedarse con aquel que la tenga menor. En lugar de esto, preferimos usar un algoritmo de búsqueda eficiente en espacios n-dimensionales, que es modificación de un algoritmo de Bentley et al.

(véase [BP80]); la filosofía es la misma que el original: almacenar los puntos conocidos en n listas, una para cada dimensión (tales listas contienen en realidad punteros a los puntos) en las cuáles estén ordenados por su n-ésima componente. La modificación sobre el algoritmo original consiste en que el nuestro trabaja con distancia euclídea en lugar de hacerlo con la L-norma. El lector interesado puede consultar el Apendice B, donde se da el seudocódigo para este algoritmo.

Para terminar este capítulo diremos que implantar este método ha sido una de las tareas más interesantes de esta tesis, en cuanto que responde al reto que Pellionisz y Llinás lanzan como una frase en uno de sus artículos ([PL80]):

> People in robotics should think about implementing this biological formalization as a mean to control real robots

y que nadie, que conozcamos, había recogido hasta ahora. Y es por supuesto especialmente satisfactorio que haya funcionado de modo aceptablemente correcto.

Capítulo 7

El Modelo de Computador Cerebelar Aritmético (CMAC)

El siguiente método que vamos a describir para el control de nuestro brazo robot es el conocido modelo CMAC (Cerebelar Model Arithmetic Computer), propuesto por James Albus como modelo biológico plausible para el funcionamiento del cerebelo. Más aún, según Albus sugiere en otros capítulos de su libro ([Alb81]), su opinión es que puede darse un modelo general de la estructura y funcionamiento del sistema nervioso en su conjunto (incluída su interacción con el mundo, para él, el comportamiento inteligente) en términos de una jerarquía de sistemas de control, organizados en capas sucesivas. Cada capa recogería información de la anterior operando a un nivel cada vez más alto, hasta llegar a la abstracción simbólica. Véase la figura 7.1 que lo explica.

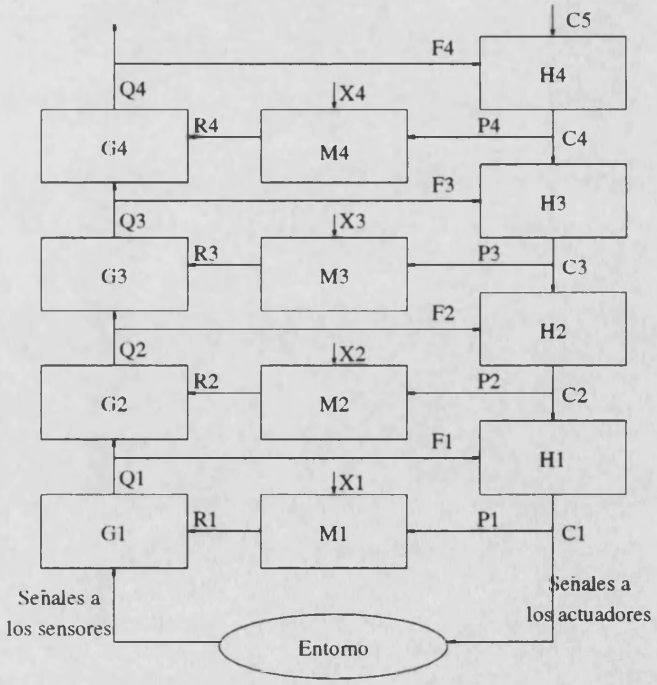


Figura 7.1: La jerarquía de sistemas de control de Albus

En esta figura, los C_i son comandos de control, entendidos a varios niveles: desde órdenes simbólicas en el nivel superior, hasta valores para las señales de los actuadores, en el más inferior. Los módulos H_i toman una señal de control del módulo superior, y una realimentación del módulo de su propio nivel, (F_i) y emiten comandos con subobjetivos (P_i) que van a parar tanto a los módulos de memoria (M_i) como al nivel inferior. Tales módulos de memoria comparan la acción a ejecutar con la que tienen almacenada, más quizá otra información de contexto, X_i, y generan una diferencia sensorial R_i, que los módulos G_i comparan con los datos sensoriales procesados a su nivel correspondiente, para generar así una señal de error, F_i, que realimenta el correspondiente módulo H_i, y también es usada como patrón sensorial de más alto nivel Q_{i-1}. Como se puede ver, los módulos G ejecutan el procesado de la información desde los datos de sensores con ruido, hasta una descripción simbólica; los módulos M son las memorias, tanto de datos sensoriales como de patrones de ellos, y en última instancia, de conceptos, y los módulos H ejecutan el control: los más inferiores podrían ser servomecanismos, los superiores, complejos sistemas de hechos y reglas.

Estas ideas, como el propio Albus reconoce, no pueden ser consideradas por el momento más que como una especulación, dado que es aún imposible medir bien la actividad eléctrica de una neurona, o de un pequeño grupo de ellas, y aun si se pudiese hacer, es dudoso que la información obtenida pudiera ser interpretada de un modo initeligible o relacionable con la jerarquía propuesta. No obstante, otra parte de su trabajo, que se dedica a modelizar los módulos M del nivel más bajo, está mucho más basada en evidencia experimental acerca del funcionamiento del cerebelo, como lo estaba el trabajo de Pellionisz y Llinás, aunque el modelo que propone es un poco diferente. Para explicarlo nos referiremos a la figura 7.2.

Las fibras que entran en la capa de células granulares provienen bien de los sensores fisiológicos (canales semicirculares, sentido del tacto, grado de contracción de los músculos, etc.) como de capas superiores (otras partes del cerebro que controlarían el movimiento voluntario). Las primeras ofrecen una señal sobre la que realimentar, las segundas, no (al menos, directamente). Cuando contactan con la capa superficial del cerebelo unas y otras se entremezclan, volviéndose indistinguibles. Una característica de la transmisión de una señal a través de fibras nerviosas es que éstas son imprecisas y ruidosas, en un nivel más alto del necesario para la resolución y fiabilidad con la que las señales de control de los músculos deben ser manejadas. Esto hace que el sistema nervioso tenga que usar varios canales para transmitir una misma señal, cada uno de ellos máximamente excitado cuando la señal toma un valor en un rango determinado para el que dicho canal es máximamente sensible; la intersección de los rangos de todos los canales máximamente excitados da un pequeño intervalo en el que el valor de la señal debe encontrarse. En el sistema nervioso estos canales corresponden a las fibras de los sensores de las articulaciones (ver la figura). El siguiente paso (la capa de células granulares) se supone que funciona de modo que cada célula recibe entradas de varias fibras aferentes, pero ninguna de ellas recibe

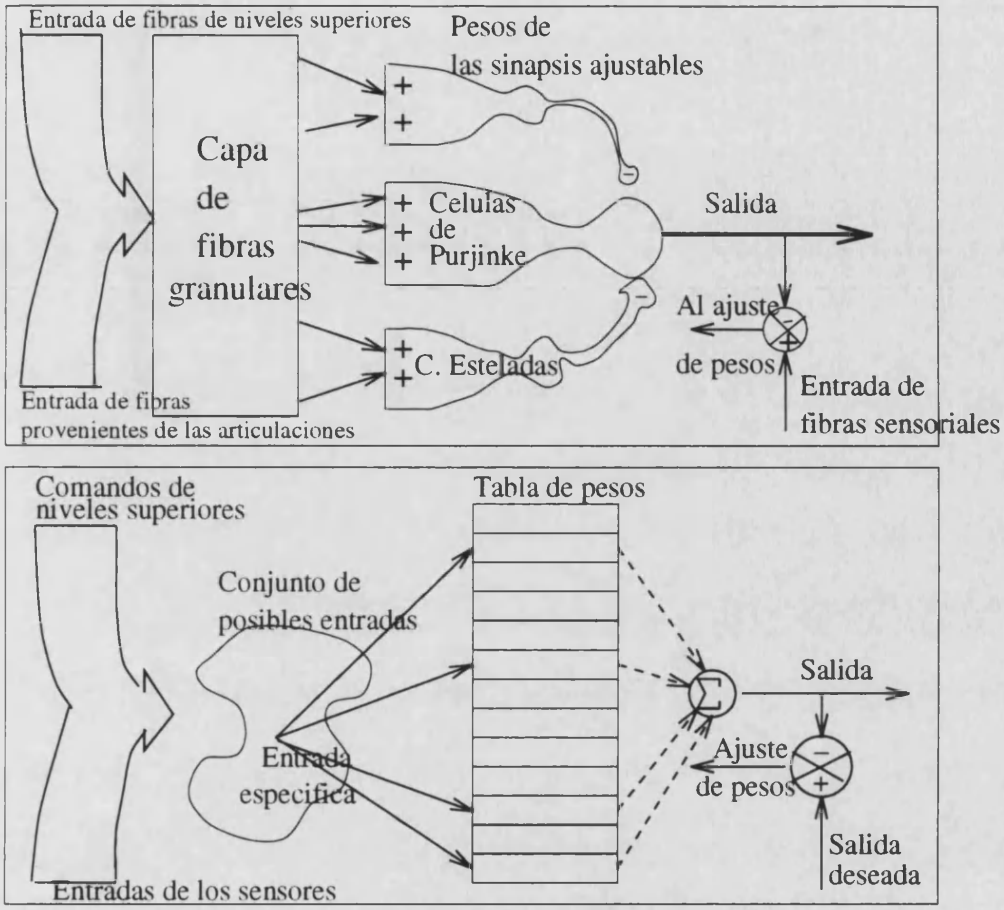


Figura 7.2: Esquema biológico y computacional de la CMAC

señales de la misma combinación de entradas que alguna otra; de esto modo, tal combinación es usada para codificar unívocamente cada salida. El valor de dicha salida se obtiene por suma de los estímulos de las células granulares excitadas. Esto lo efectúan las llamadas células de Purjinke y las células esteladas, cuyas conexiones entre sí y con las células granulares pueden variar su fuerza. El mecanismo de aprendizaje (que es simplemente el llenado de la memoria con datos determinados, o su modificación) se supone que es ejecutado mediante la variación de la fuerza de las conexiones en las células de Purjinke.

Veamos ahora la implantación computacional. Daremos primero una visión general, dado que el método puede ser usado en principio para aprender cualquier aplicación de un espacio de entrada en uno de salida, y después lo particularizaremos a nuestro caso.

Sea un espacio de entrada n-dimensional, $X = (x_1, \cdots, x_n)$ y uno de salida unidimensional, Y. En caso de que la salida tenga varias dimensiones se usará la formulación por separado para cada una de ellas. Sea R_i el rango de valores que cada variable de entrada x_i puede tomar. Vamos a dividir R_i en Q_i intervalos

iguales; entonces la variable x_i tomará un valor que se encontrará en alguno de ellos, digamos el n_i; es claro que $n_i = [x_i/Q_i]$, siendo [ ] la parte entera. Esto es una forma de expresar la imprecisión en la codificación. Pero ahora supongamos que dividimos R_i de modo similar, pero desplazando el inicio de los intervalos una cantidad δ. En ese caso, y bajo la nueva codificación, el valor "impreciso" de la variable x_i sería $n_i' = [(x_i - \delta)/Q_i]$. Supongamos, por fín, que hacemos esto K veces, y que escogemos δ como R_i/KQ_i. En este caso, el aspecto sería, con, p. ej., $K = 4$ y $Q_i = 8$:

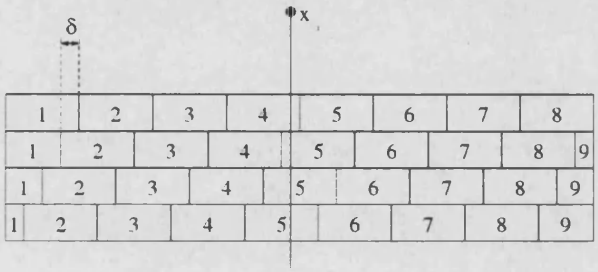


Figura 7.3: Cuantización de una variable en la CMAC

En este caso, el valor x_i vendrá determinado por la cuaterna (4,5,5,5), que codifica un intervalo de anchura $\delta = R_i/KQ_i$. Esta es la máxima precisión para la que la CMAC puede distinguir dos estímulos de entrada, y se supone que debería llegar a ser el máximo error con el que nuestros sensores nos den el valor de x_i. Esto se consigue escogiendo un valor de K lo grande que sea preciso, o recíprocamente, incrementando el número de subintervalos Q. A cada una de las posibilidades de división, K_i, se la llama función de cuantización, y como ya se habrá supuesto, representa cada uno de los canales imprecisos por los que decíamos que se transmitía la información nerviosa. Ahora veamos cómo funciona la CMAC para varias dimensiones y cuál es el modo de almacenamiento. Explicaremos esto sobre la figura 7.4[1]

Supongamos dos variables de entrada, θ_1 y θ_2, y sea p la variable de salida. Sea un punto de entrada, A, cuyas coordenadas hacen que bajo las cuatro funciones de cuantización de la primera dimensión se le identifique por los índices {C,J,O,U}, y bajo las de la segunda, por {c,j,o,u}. Para calcular el valor de p en el punto A se construye el producto cartesiano de los índices de sus dimensiones, que será el conjunto de pares (o de n-tuplas, en el caso de n dimensiones) {(Cc),(Jj),(Oo),(Uu)} Entonces se buscan en cuatro tablas los valores de los pesos w_i indexados por cada par, y el resultado es la suma de tales pesos, en este caso, $p_A^* = w1[(Cc)] + w2[(Jj)] + w3[(Oo)] + w4[(Uu)]$. El número de pesos que hay que almacenar será el número de funciones de cuantización, K, multiplicado por el producto del número de intervalos

[1] Esta figura está reproducida de [TP92] con permiso del autor, a quien se agradece su amabilidad.

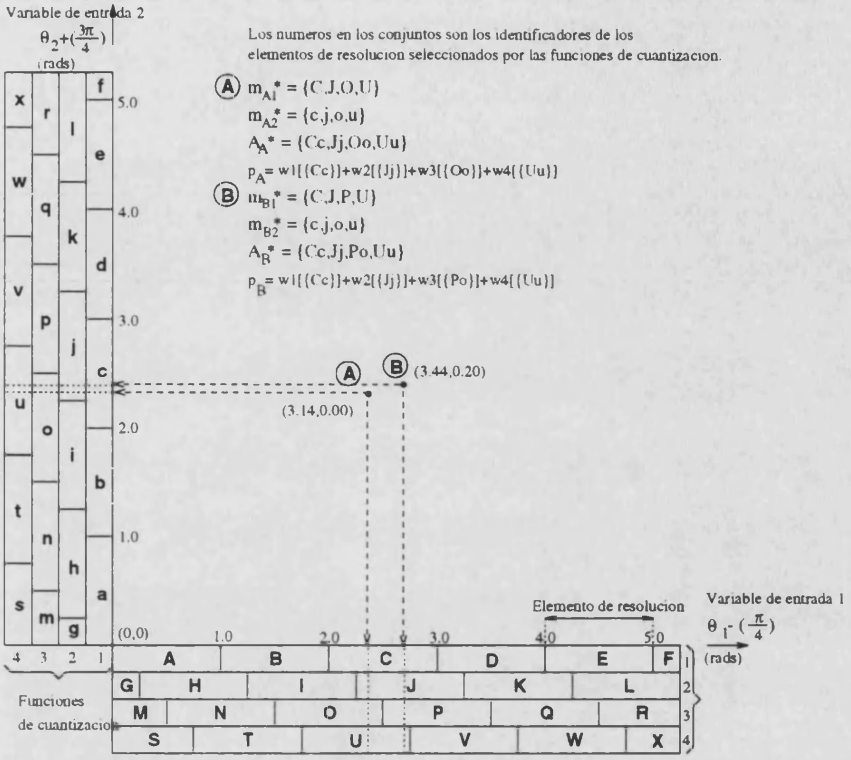


Figura 7.4: Organización de una CMAC

por cada dimensión, o sea,

$$Nw = K \prod_{i=1}^{N} Q_i \tag{7.1}$$

siendo N el número de dimensiones. En muchas ocasiones (será nuestro caso), las variables tienen el mismo rango en todas las dimensiones, el número de intervalos de cuatización se toma entonces el mismo, sea Q, y en ese caso simplemente $Nw = KQ^N$. Los pesos de la tabla son una memoria, y hacen el papel de las células granulares antes mencionadas. Respecto al aprendizaje, que consiste en el llenado de las tablas de pesos, o la variación de su contenido, el algoritmo es muy simple: basta tomar un dato del valor de entrada y salida, sea $(X, p(X))$. Entonces se obtiene el valor dado en este momento por la CMAC para $p(X)$, sea $p^*(X)$. Si este valor coincide hasta la precisión requerida con $p(X)$, no hay que hacer nada. En otro caso, cada uno de los pesos w_i que han intervenido en el cálculo de $p^*(X)$ es incrementado en un valor

$$\Delta w_i = \frac{p(X) - p^*(X)}{K} \tag{7.2}$$

Es trivial ver que comenzando por una tabla llena de ceros, el valor correcto de cada punto es directamente introducido en la tabla. Pero la ventaja del método radica

en su capacidad de generalización. Para verlo, supongamos un punto B, cercano al A, cuyas "coordenadas" serían (ver figura 7.4) {(Cc),(Jj),(Po),(Uu)}. Entonces, $p_B^* = w1[(Cc)] + w2[(Jj)] + w3[(Po)] + w4[(Uu)]$, es decir, tres de los cuatro pesos que intervienen para generar la salida lo hacen también para generar la salida en A, y consiguientemente fueron ya actualizados cuando se introdujo A. Esto, lógicamente, aumenta la velocidad de convergencia, pero también puede provocar un problema de conflicto, especialmente en aquellas zonas del espacio de entrada en que la función que se está encontrando varía muy bruscamente. Específicamente, ninguna variación en un intervalo menor que δ será apropiadamente tenida en cuenta. En realidad, el método está aprendiendo una aproximación constante a tramos a la función real, donde la longitud del tramo es δ. Albus no recoge esta consideración, y supone implícitamente que tal longitud será menor que el error máximo que los sensores cometen. Ignoramos si eso será cierto en el cerebelo, pero en cualquier caso puede ser imposible de implantar en aplicaciones reales, por no disponer de la cantidad de memoria necesaria.

Precisamente, es importante hacer ahora algunas consideraciones acerca de la memoria y el tiempo de aprendizaje y de recuperación de la información. Como dijimos, la memoria (el número de pesos) varía como KQ^N. Dado que el número de dimensiones nos viene impuesto por el problema particular que abordemos, uno puede pensar en jugar con K y con Q para reducir al mínimo la memoria necesaria. Esto presenta un sutil problema: una CMAC puede ser vista como una tabla hash bastante eficiente, que, como es lógico, tiene un límite en su capacidad de almacenamiento antes de que se produzcan colisiones. No es una tabla hash óptima, pero tiene la ventaja de que preserva la continuidad de la función que se modeliza (es decir, pesos próximos en la ordenación que hemos dado para ellos codifican puntos próximos del espacio de entrada) lo cual implica que en el momento en que se produzcan colisiones la exactitud en los resultados dados por la tabla ya no aumentará más, pero por lo menos tampoco se degradará. Deberemos escoger valores para K y Q lo más pequeños que sea posible para ahorrar memoria, pero lo más grandes que sea posible para conseguir suficiente exactitud. Veamos primero cuál sería la memoria mínima para obtener una resolución dada. Si el rango de variación del intervalo para cada variable es R (supongamos, por simplicidad, el mismo para todas las variables, si no, los resultados son análogos) entonces la máxima precisión era $\delta = R/KQ$. Pero como la memoria es proporcional a KQ^N,

$$M = KQ^N = \frac{R}{\delta}Q^{N-1} = \frac{R^N}{\delta^N}\frac{1}{K^{N-1}} \tag{7.3}$$

En el caso mínimo, Q sería 2 (debe haber al menos dos intervalos), por lo que $M = (R/K)2^{N-1}$. El problema es que éste es también el número de pesos, de modo que si nuestro número de datos de entrenamiento distintos es mayor, necesariamente va a haber conflicto. Cuanto mayor sea el número de funciones de cuantización, mayor será el número de puntos que compartan en su composición de la salida el mismo peso, y consiguientemente, mayor la influencia de unos en otros. Hay que

hacer notar que todas estas consideraciones suponen que la CMAC pueda (al menos, en teoría) alcanzar la precisión que hemos fijado para ella. Podemos conformarnos con una precisión menor, y en tal caso, si el número de pesos es de todos modos suficiente dados nuestros datos, la precisión obtenida para el valor de la función puede aún ser razonable. Veremos un ejemplo de que esto sucede con nuestros datos de entrada.

Otras consideraciones importantes son las relativas al tiempo de aprendizaje y a la tasa de aprendizaje. El número de actualizaciones de pesos que hay que hacer por cada nuevo dato introducido es el número de los que contribuyen a él, es decir, K. Cada actualización requiere encontrar el peso en la tabla n-dimensional asociada a la función K-ésima que lo indexa, y sumarle el correspondiente incremento. Recíprocamente, a la hora de la recuperación, también es necesario encontrar y sumar los pesos w_i para dar el resultado. Es fácil ver que el número de operaciones necesarias es del órden de NK. Como, según la fórmula 7.3, la memoria necesaria es inversamente proporcional a K, parecería razonable usar valores grandes para este parámetro, pero nótese que también el coste computacional del aprendizaje es proporcional a K, lo cual ha de tenerse en cuenta para un balance razonable.

Por otra parte, ninguno de los artículos que usan la CMAC (incluído el original de Albus) dan una medida de cuántos datos pueden ser necesarios, o de cuándo se puede parar el aprendizaje. Para dar cuenta de esto hemos definido una tasa de aprendizaje como la variación porcentual media para todos los pesos alterados en el ciclo de aprendizaje presente, y esto promediado sobre los ciclos anteriores.

$$LR = \frac{1}{T} \sum_{i=1}^{T} \sum_{j=1}^{K} \frac{\mid \Delta w_j \mid}{\mid w_j \mid} \tag{7.4}$$

El promediado tiene que ver con la necesidad de suavizar la función, de modo que una variación brusca debida a ruido no altere sustancialmente su comportamiento. Además, la variación de esta función (su derivada) está relacionada con la velocidad de aprendizaje, y su límite, con el grado de exactitud obtenido para el mismo. Veamos cómo. En un supuesto estado estacionario después de un aprendizaje completo, la variación de los pesos sería nula, y consiguientemente la suma de todas las variaciones anteriores permanecería constante, sea C; de este modo nuestra tasa de aprendizaje tendría la forma C/T, y su derivada, $-C/T^2$, no nula. Pero si se observa que esta tasa de aprendizaje a partir de un cierto ciclo apenas varía, o lo hace erráticamente alrededor de un valor (o incluso aumenta) significa que los pesos siguen variando de modo errático. Esto es debido a que se producen colisiones en la tabla, porque cada peso es requrido hacia diferente lugar por diferentes datos de entrada. Deberá pararse el aprendizaje. En este caso, el valor final que haya tomado la tasa de aprendizaje también es informativo: cuanto más pequeño sea el valor por peso de LR (o sea, $LR_{final}/n^o\ pesos$), significa que menos varían, en promedio, éstos cuando se introduce nueva información, y consiguientemente, que más próximos están a su valor correcto. Como conclusión, podemos decir que el comportamiento de la derivada de la tasa de

aprendizaje nos da una pauta para decidir cuando la CMAC está saturada, y es por tanto inútil continuar.

Otra aspecto importante que tampoco cosideran los trabajos anteriores es la distribución de los datos en la memoria. Como quiera que la cuantización del espacio de entrada es regular (es decir, los intervalos Q_i son iguales, y consiguientemente, la memoria asignada a cada intervalo es la misma) puede ocurrir (en CMACS que no alcanzen la máxima precisión posible) que una zona de la memoria se sature mientras otras quedan vacías. Esto depende, por supuesto, de la distribución de los datos de entrada. Lo idóneo sería que se dedicase más memoria a aquellas zonas donde se tienen más datos de entrada, porque generalmente esto se debe a que posteriormente van a ser más usadas. Hay dos posibilidades para hacer esto:

- Usar una resolución variable, que vaya reduciendo la anchura de los intervalos a medida que van entrando más datos.

- Reescalar los datos de entrada de modo que ocupen todo el rango de variación de las variables de entrada.

Esta segunda solución fué la que usamos, teniendo en cuanta la distribución estadística que nuestros datos presentaban. Explicaremos esto con detalle inmediatamente. Tal solución es la más sencilla, pero tiene el inconveniente de que requiere conocer todos los datos antes de entrenar la CMAC, es decir, hay dos fases: una de toma de datos y su almacenamiento en ficheros sin ningún tipo de proceso, y otra de entrenamiento. Esto no suele ser un problema en la mayoría de los sistemas prácticos, pero sería injustificable en un sistema biológico. Este punto no aparece ni se plantea nunca porque en general casi todos los sistemas que usan una CMAC son simulados, y es el experimentador el que genera una distribución aleatoria de datos sintéticos de entrada, la cual siempre suele seguir distribución uniforme. Respecto a la primera solución, el ajuste dinámico de los intervalos con una subdivisión del que sea preciso cuando suficientes datos hayan caído en él daría lugar a una estructura fractal, con un número en principio no limitado de subdivisiones. Esto es una futura mejora que se comentará en el capítulo 10.

Veamos ahora cómo ha funcionado la CMAC en nuestro problema. Nuestras cuatro variables de salida son, como vimos en el capítulo anterior, zed, sho, elb y yaw. Vamos a escoger para las tres primeras una estructura consistente en sendas CMACs cuyas entradas van a ser tres coordenadas visuales. La razón de escoger tres, y no las cuatro, se debe a que si el espacio de entrada tiene una dimensión real de 3, los valores que introduzcamos van a estar confinados a una variedad de dimensión 3; esto quiere decir que en un CMAC de cuatro dimensiones, la inmensa mayoría de la memoria va a quedar desaprovechada, y los datos se concentrarán todos en una pequeña fracción de los pesos disponibles, lo cual degrada sustancialmente la eficacia del método, según comprobamos en una primera implantación fallida. Para verlo

como analogía en una dimensión menos, es como si guardásemos información sobre una función definida en una superficie del espacio dando una tabla de todo el espacio tridimensional. Las coordenadas visuales que vamos a escoger son, de acuerdo a la geometría de nuestra disposición de cámaras, x_l, x_r e y_{lr}, siendo ésta última la media de y_l e y_r. Esto lo hacemos porque los valores de y_l e y_r son en casi todo el espacio similares. Por otra parte, para la cuarta salida (el yaw) podríamos haber seguido la misma estructura, pero como sabemos que es en realidad una coordenada redundante, y que, como dijimos, podía calcularse a partir de sho y elb, lo haremos de esta manera, usando la salida de unas CMACs como entradas a otra. Véase el esquema en la figura 7.5

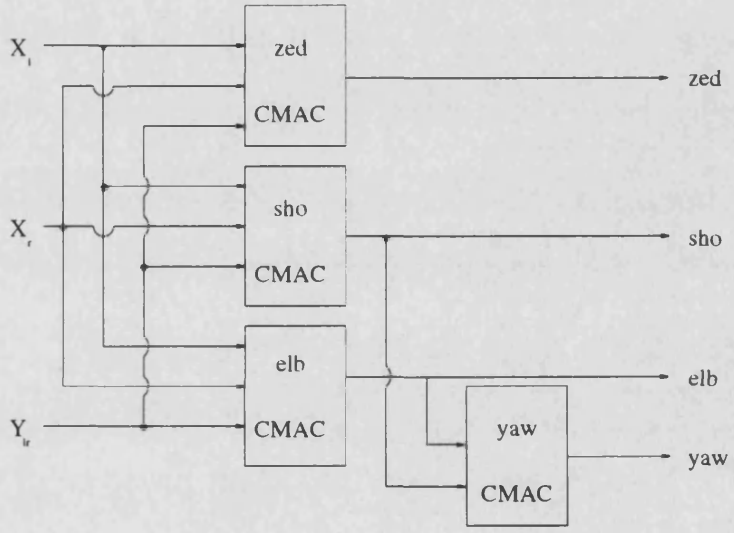


Figura 7.5: Esquema de nuestro uso de las CMACs

La máxima resolución que podemos obtener para las entradas a las tres primeras CMACs es un pixel, y la anchura del intervalo, 512 pixels. Esto significa que $K * Q$ ha de ser igual a 512. Los valores posibles para la memoria empleada en función de K y de Q se muestran en la tabla que sigue, en número de pesos, y en memoria en nuestra máquina, teniendo en cuenta que cada peso ocupa 4 bytes. Los valores de la diagonal son las CMACs con precisión máxima, los valores por debajo de la diagonal rebasarían esta precisión, lo cual no tiene sentido, por tanto no se muestran. La memoria total empleada sería, pues, tres veces ésta (dado que hay tres CMACs de este tipo) más la empleada por el cuarto CMAC, que por ser bidimensional, es inferior a éstas, por lo que cuatro veces la memoria indicada en la tabla es una cota superior suficiente. Las entradas marcadas con una estrella son las que cuadruplicadas no cabrían en la memoria de los sistemas de que disponemos.

Ahora se trata de implantar algunas o todas de estas CMACs y verificar su funcionamiento sobre el robot real. El problema está en que el tiempo de aprendizaje

Q \ K	256		128		64		32		16		8		4		2	
2	2^{11}	8Kb	2^{10}	4Kb	2^{9}	2Kb	2^{8}	1Kb	2^{7}	32b	2^{6}	256b	2^{5}	128b	2^{4}	64b
4			2^{13}	32Kb	2^{12}	16Kb	2^{11}	8Kb	2^{10}	4Kb	2^{9}	2Kb	2^{8}	1Kb	2^{7}	512b
8					2^{15}	128Kb	2^{14}	64Kb	2^{13}	32kb	2^{12}	16Kb	2^{11}	8Kb	2^{10}	4Kb
16							2^{17}	512Kb	2^{16}	256Kb	2^{15}	128Kb	2^{14}	64Kb	2^{13}	32Kb
32									2^{19}	2Mb	2^{18}	1Mb	2^{17}	512Kb	2^{16}	256Kb
64											2^{21}	$8Mb^{(*)}$	2^{20}	$4Mb^{(*)}$	2^{19}	2Mb
128													2^{23}	$32Mb^{(*)}$	2^{22}	$16Mb^{(*)}$
256															2^{25}	$128Mb^{(*)}$

Tabla 7.1: Memoria usada por cada CMAC

sobre el robot real sería excesivo para un test exhaustivo. Hemos decidido usar una simulación, pero basada en datos reales. Para ello, usaremos las medidas tomadas en en método de la Red de Tensores. Como vimos, esta formulación relacionaba coordenadas de las cámaras y de los joints, pero puede también usarse a la inversa: dadas las medidas tomadas, se busca el punto más próximo en coordenadas motoras a aquel al cual mandaríamos al robot; se toma su tensor asociado, y se invierte, lo cual nos da la relación local entre joints y cámaras, es decir, nos indica dónde se verá el punto terminal si llevamos el brazo a cierto valor de las coordenadas motoras. Dado que no se trata de medidas reales, no se puede usar el proceso de realimentación que explicamos en ellas, pero sigue siendo un modelo bastante aproximado, según comprobamos experimentalmente comparando los resultados que daba con los del robot real, y usado para entrenar las CMACs puede darnos una idea de los parámetros, en particular del número de puntos para entrenar, y permite escoger la mejor CMAC, que será probada sobre el robot real. Nótese, por otra parte, que al estar construído el modelo sobre medidas reales, no sobre nuestra concepción del robot, cualesquiera problemas que afecten a su funcionamiento estarán subsumidos en él. Esto lo diferencia de la mayoría de los modelos de simulación usados en otros trabajos.

Veamos ahora algunos resultados con gráficas para la tasa de aprendizaje. Sea, primero, el caso en que la resolución para las CMACs visuales era tan sólo de 2 intervalos de cuantización (Q=2) y el número de funciones de cuantización era 256 (K=256). (véase figura 7.6). Aquí se observa un extraño salto, no visible en otras gráficas, que se debe a que con aproximadamente 5000 ciclos, los pesos han sido ya llenados, y comienza a haber conflictos. Esto es debido a que el número total de pesos es aquí menro que el de puntos (ciclos) de parendizaje, lo cual no se dará en otros casos. Como se observa, a partir de ahí el comportamiento es el esperable, y después de unos 20000 ciclos la variación es aproximadamente nula, lo que indica la saturación estable de la CMAC. Obsérvese que la tasa de aprendizaje final está aproximadamente en 325, o sea, aproximadamente 0.16 por peso, lo cual veremos luego que es altísimo en comparación con otros resultados, y de hecho en términos de precisión real significa que prácticamente todos los puntos son llevados a posiciones absurdas (inaccesibles).

Sea ahora el caso de Q=4 y K=128 (figura 7.7); aquí el comportamiento es

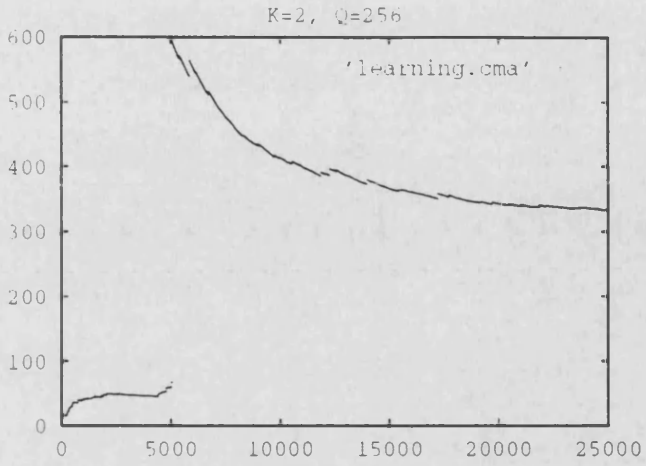


Figura 7.6: Tasa de aprendizaje con Q=2,K=256

completamente diferente: no se observa el salto comentado antes, dado que el número de pesos es suficiente, pero la saturación estable se alcanza inmediatamente, dado que un número aún tan grande de funciones de cuantización (128) da una generalización excesiva. A partir de ahí se observa el comportamiento errático de la tasa de aprendizaje que mencionamos, tendiendo incluso a un ascenso, con un valor final demasiado alto (aprox. 25, como $3 \cdot 10^{-3}$ por peso) lo cual genera sobre la precisión final la misma situación que el caso anterior.

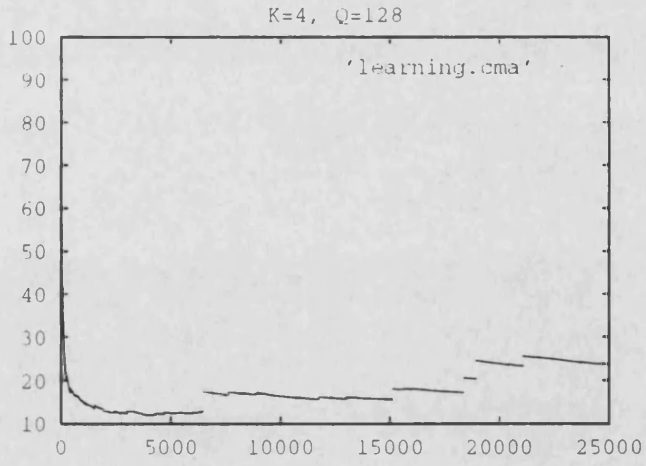


Figura 7.7: Tasa de aprendizaje con Q=4,K=128

La siguiente gráfica es la de Q=8, K=64 (figura 7.8). Es similar a la anterior, pero de comportamiento algo más suave, dado que el mayor número de intervalos de

cuantización recoge mejor la variación de la función objetivo, y consiguientemente, el aporte de nueva información hace variar menos los pesos. La estabilización se consigue sobre los 6000 ciclos, y el valor final es sobre 10, ($3 \cdot 10^{-4}$ por peso), lo cual equivale a que esta vez sí se alcanzan 510 de los 528 puntos de test, pero con error medio de más de 50 mm.

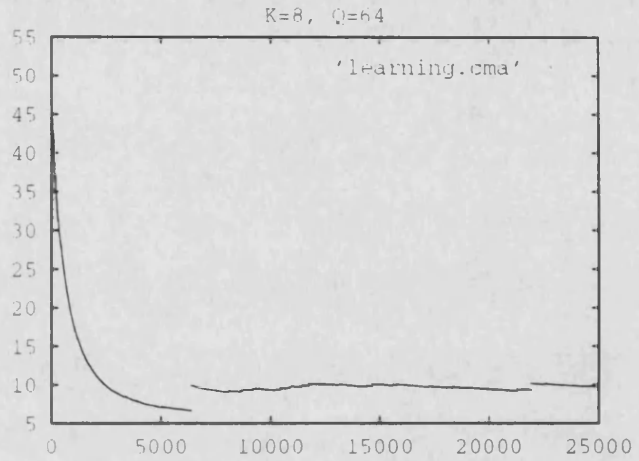


Figura 7.8: Tasa de aprendizaje con Q=8,K=64

Todas éstas eran CMAC's en las que K y Q estaban escogidos para alcanzar la máxima resolución. Veamos ahora que en ocasiones renunciar a éste requisito puede, paradójicamente, mejorar el resultado final con la misma cantidad de memoria. Mostraremos cinco casos.

El primero, con Q=16 y K=4 (figura 7.9). Esta vez la CMAC tarda bastante en aprender, y de hecho sigue aún haciéndolo a los 25000 ciclos, (número máximo que por problemas de tiempo con el robot real parmitiremos en él), pero esta vez el resultado final es mucho mejor, acabando con una tasa de aprendizaje de aproximadamente 0.8 ($1 \cdot 10^{-4}$ por peso), y un gasto de memoria menor, de sólo 2^{14} pesos. Esta gráfica y las siguientes son ejemplos claros del balance entre generalización/velocidad de convergencia y precisión.

Las cuatro últimas gráficas mostrarán el resultado con las cuatro mejores CMACS, que serán las que se usen para las comparaciones finales del capítulo 9. Son las obtenidas con 32 intervalos de cuantización, y diferente número de funciones: 2,4,8 y 16. Al haber en ellas más pesos, y estar además en algunas (la 4_32 y la 8_32) la resolución más próxima a la máxima posible (4 y 2 pixels respectivamente) la variación es más suave, y el resultado final para la tasa de aprendizaje, pequeño (entre $1 \cdot 10^{-5}$ y $5 \cdot 10^{-6}$ por peso). Básicamente, son muy similares, y sus resultados en términos de precisión, que serán analizados en con detalle en el capítulo 9, están entre 8.6 y 9.2 mm. Sus gráficas son las figuras 7.10 hasta 7.13. Un detalle es que la

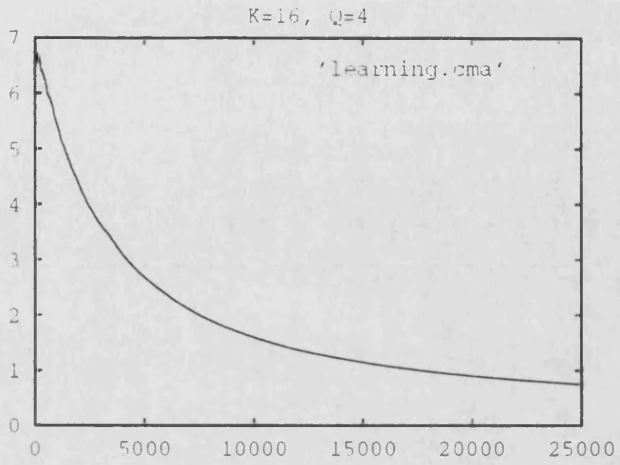


Figura 7.9: Tasa de aprendizaje con Q=16,K=4

16_32, según se verá en el capítulo 9, no es la más precisa, pero ello se debe a que no ha sido entrenada con suficientes puntos por los problemas de tiempo mencionados; en la gráfica se observa que la tasa de aprendizaje seguiría decreciendo sin estabilizarse aún, lo cual siginifica que el aprendizaje podría continuar.

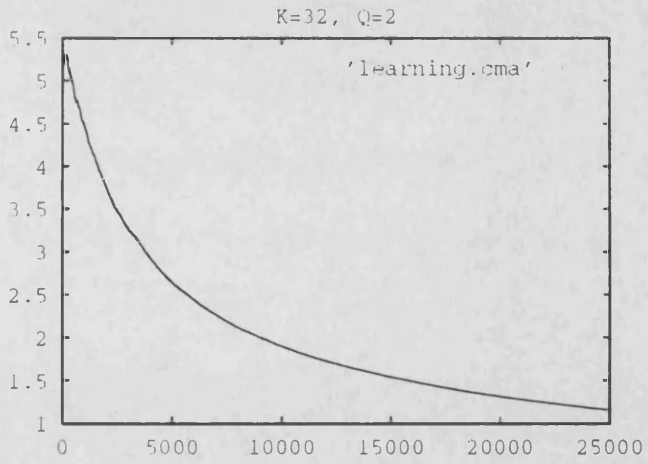


Figura 7.10: Tasa de aprendizaje con Q=32,K=2

Como conclusión de este capítulo, digamos que la CMAC se ha mostrado un método útil, de gran sencillez, y que tiene la ventaja de poderse implantar fácilmente de modo paralelo, con un procesador para cada función de cuantización, y otro para hacer la suma de los pesos devueltos por los anteriores, lo cual, en aplicaciones que lo requieran y en las que el coste computacional principal radique aquí puede ser una

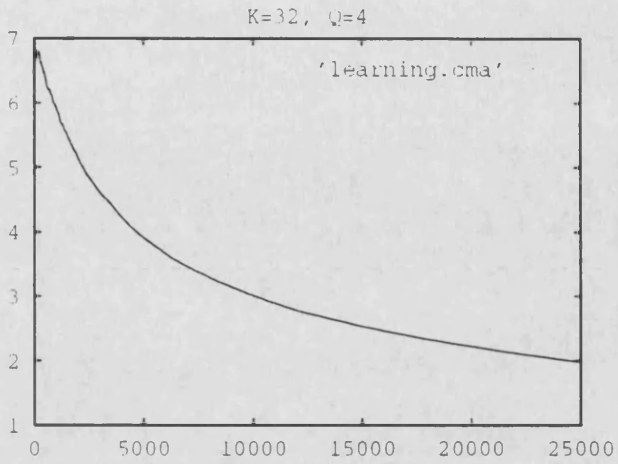


Figura 7.11: Tasa de aprendizaje con Q=32,K=4

ventaja interesante. Respecto a los resultados numéricos, como se verá en el capítulo 9, no son comparables en los mejores casos a los de la TNT. Su mayor deficiencia es la falta de realimentación, lo que implica mayor cantidad de memoria para obtener la misma precisión.

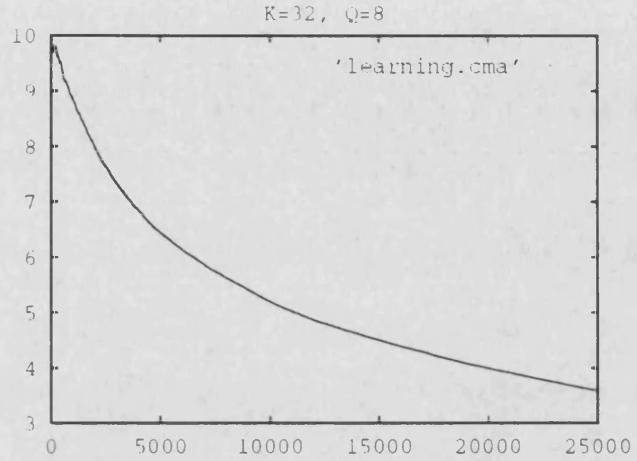


Figura 7.12: Tasa de aprendizaje con Q=32,K=8

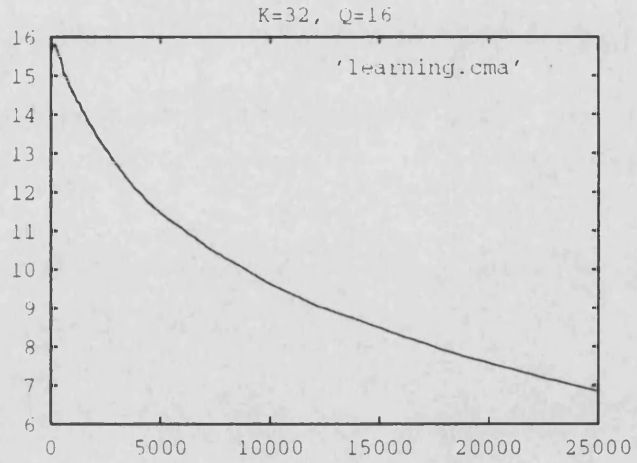


Figura 7.13: Tasa de aprendizaje con Q=32,K=16

Capítulo 8

Algunos modelos de Redes Neurales

8.1 Introducción

En este capítulo abordaremos el problema de la aproximación de la función de transformación cámara-joints por medio del uso de algunos de los modelos más difundidos de red neural, y también alguno de los más modernos, poco usados todavía, y en principio más potentes. Como veremos después, ninguno de estos métodos ha dado resultados satisfactorios en nuestro problema. Daremos cuenta detallada de cuáles han sido los problemas encontrados en cada caso, y cuáles las causas que les atribuímos. Para ello, este capítulo se dividirá en secciones que explicarán cada método en particular. Se supone al lector cierta familiaridad con el vocabulario, métodos y algoritmos fundamentales de las redes neurales; en caso contrario, un buen texto introductorio es el de Krose y van der Smagt, [KvdS93], y una revisión actualizada hasta el año 90 es la de Simpson, [Sim90]. También muchos de los métodos nuevos aparecen explicados en el manual del programa SNNS ([Ins93]).

8.2 La Red Autoorganizativa Tridimensional

Pese a no ser éste el método más sencillo, comenzaremos por él porque es el más específicamente adaptado a la nuestra aplicación, y fué, por tanto, el primero de este tipo que decidimos implantar. Su origen está en varios artículos de Martinetz, Ritter y Schulten (a partir de ahora, R,M&S), el último de los cuales resume prácticamente los anteriores, véase [MRS90]. Parten del trabajo de Kohonen sobre redes autoorganizativas. Este tipo de redes estuvo en principio pensada para ejecutar clasificación no supervisada. A diferencia de los modelos usuales, no tiene capas diferenciadas de entrada o salida. Cada una de las unidades de proceso, cuyo estado viene definido por un vector de pesos $W = (w_1, \ldots, w_n)$ acepta todas las entradas, que se organizan en un vector $U = (u_1, \ldots, u_n)$. Los pesos W convolucionan con las entradas, produciendo la activación A de esa neurona, que numeraremos como i:

$$A_i = \sum_{j=1}^{n} w_{ij} u_j \tag{8.1}$$

La mayor activación marcará la neurona que escogeremos como ganadora, dado que todas ellas se organizan en una sóla capa competitiva. En el caso de que las entradas no estén normalizadas, la menor distancia euclídea a la entrada determinará qué neurona gana. El siguiente paso es la actualización de los pesos según la regla

$$w_{ij}(t+1) = \frac{w_{ij}(t) + \gamma(u_j(t) - w_{ij}(t))}{\|w_{ij}(t) + \gamma(u_j(t) - w_{ij}(t))\|} \tag{8.2}$$

o, en el caso no normalizado,

$$w_{ij}(t+1) = w_{ij}(t) + \gamma(u_j(t) - w_{ij}(t)) \tag{8.3}$$

donde γ es un parámetro de aprendizaje fijado por el usuario.

Nótese que lo que se está haciendo con esto es girar (en el caso no normalizado, desplazar) las neuronas actualizadas hacia el patrón de entrada. La clave del método, y en lo que consiste la propuesta original de Kohonen, es que no todas las neuronas actualizan sus pesos. El usuario debe definir antes de empezar una topología para la red total, que indica qúe neuronas están conectadas con qué otras. En todos los casos que conocemos se usa una disposición en línea o en anillo cerrado para el caso unidimensional, en red cuadriculada para el caso bidimensional (si es cerrado, en toro), o en red tridimensional ortoédrica, como hacen R,M&S. La actualización se hará a cada presentación de un nuevo patrón, sólo sobre la neurona ganadora y sobre un cierto número de sus vecinas en la ordenación topológica. Lo que se consigue de este modo es que cada zona de la red responda a una cierta zona del espacio del entrada, creándose de este modo campos perceptuales distintos. Para conseguir esto de una manera uniforme, el parámetro γ de las ecuaciones 8.2 y 8.3 se sustituye por una función $\gamma(t, d(u,w))$, que tiene un máximo en $(t,0)$ y decrece su valor con la distancia topológica $d(u,w)$ a la neurona ganadora, y que también decrece globalmente su valor con el tiempo t (el número de ciclos de aprendizaje). El problema de la ley para el decrecimiento del parámetro γ es bastante serio, como luego veremos, y está relacionado con la estabilidad. Al final del proceso la topología de la red semeja a la distribución de datos del espacio de entrada, arracimándose más elementos allí donde más densidad de datos hay. Esto trata de reflejar el mecanismo biológico por el que, p. ej., se usan muchas más neuronas para recibir los estímulos de la palma de la mano que de la planta del pie, dado que muchas más terminaciones nerviosas afluyen desde la primera que desde el segundo.

Descendiendo a la implantación sobre robot simulado de R,M&S, la diferencia con el método descrito hasta aquí es que en vez de una clasificación, se desarrollan simultaneamente dos, una del espacio sensorial, y otra del motor, pero residiendo

sobre las mismas unidades de proceso. De este modo, cada neurona tendrá ocho pesos, las cuatro coordenadas de las cámaras para la entrada, y las cuatro coordenadas motoras de la salida, organizadas en dos vectores, cada uno de los cuales se actualiza según las reglas antedichas. El par cámara-joints que es la medida tomada se usa como entrada respectivamente a cada grupo de coordenadas. La consulta se haría tomando una medida de las coordenadas de cámara, y viendo qué neurona tiene sus pesos visuales más cercanos (como vector en distancia euclídea) a tal medida: la salida sería entonces el vector de joints asociado a esa neurona. Esto tiene un inconveniente grave, que es la enorme cantidad de neuronas que harían falta para conseguir una precisión razonable. R,M&S solucionan esto asociando a cada una un tercer elemento: una matriz que represente el jacobiano de la transformación cámara-joints en ese punto. De este modo se puede usar un procedimiento de linealización idéntico al que propusimos en la Teoría de la Red de Tensores, que puede dar buenos resultados con un espacio de almacenamiento pequeño; además tiene dos ventajas sustanciales sobre nuestra aproximación:

- La transformación no es introducida mediante medidas en puntos fijos determinados por el usuario, sino que el sistema la aprende a partir de ejemplos de una distribución aleatoria.
- El propio sistema ajusta su resolución dinámicamente localizando más elementos de proceso allí donde más datos suelen presentarse.

Por otra parte, existe el inconveniente de como entrenar la red para obtener los valores óptimos de tales matrices. El método empleado es hacer aprender cada elemento de la matriz con el mismo algoritmo que hemos descrito para cada peso de la red. Obviamente, necesitamos para esto el valor correcto de la matriz, para mover la que tenemos hacia ella, valor del cual por el momento carecemos. Esto se soluciona buscando un estimador razonable de la matriz. El valor de tal estimador vendrá dado por una regla de Widrow-Hoff aplicada a una fase de movimiento fino, después de que se haya realizado el movimiento marcado por la red en su estado presente. El proceso es como sigue:

- Sea $\mathbf{u}$ el vector de coordenadas visuales del punto al que queremos llegar.
- La red selecciona la neurona, sea la j, con pesos $\mathbf{w}_j$ más próximos a $\mathbf{u}$.
  Sean el vector de joints y la matriz asociados a la neurona j θ_j y A_j respectivamente.
- La fase de movimiento grueso lleva al brazo a θ_j, y se toma medida de las coordenadas visuales en este punto. Sean $\mathbf{v}_i$.
- La fase de movimiento fino llevará ahora el brazo a un punto, θ, que es $\theta = \theta_j + A(\mathbf{u} - \mathbf{w}_j)$
- Se leen las coordenadas visuales en θ, sean $\mathbf{v}_f$
- El estimador para A, A^*, se obtiene como

$$A^* = A_j + \|\mathbf{v}_f - \mathbf{v}_i\|^{-2} A_j(\mathbf{u} - \mathbf{w}_j - \mathbf{v}_f + \mathbf{v}_i)(\mathbf{v}_f - \mathbf{v}_i)^T \qquad (8.4)$$

Esta ecuación puede ser también escrita como

$$A^* = A_j + \|\mathbf{v}_f - \mathbf{v}_i\|^{-2}(\Delta\theta - A_j\Delta\mathbf{v})\Delta\mathbf{v}^T \tag{8.5}$$

En este caso $\Delta\theta$ sería la variación de joints que esperamos obtener al multiplicar la matriz asociada a la neurona, A_j, por la diferencia de coordenadas visuales, y $A_j\Delta\mathbf{v}$ la variación que realmente hemos obtenido. Así, su diferencia es el error en la variación de los joints, o dicho de otro modo, el error en el término de primer órden del desarrollo de la función $\theta(\mathbf{v})$ en un entorno de θ_j. Por tanto, la ecuación 8.5 puede ser entendida como un regla de Widrow-Hoff para la correción de la matriz A. Recordemos que en una red neural que use esta regla la variación de una conexión, y_j, debida a la coordenada x_j del patrón de entrada p sería

$$\Delta y_j = \gamma(d^p - a^p)x_j \tag{8.6}$$

donde d^p es la salida, o respuesta de la red al patrón p, y γ una constante de normalización (en nuestro caso, $\|\mathbf{v}_f - \mathbf{v}_i\|^{-2}$). Se puede probar (ver [Sim90]) que esta regla tiende a minimizar el error cuadrático, el cual para nosotros sería

$$E = \sum_{i=1}^{p} \|\Delta\theta - A_j\Delta\mathbf{v}\|^2 \tag{8.7}$$

El problema grave que surge en este estado es que las matrices A_i están inicialmente llenas con valores aleatorios pequeños. Al multiplicar A_i por la diferencia entre las coordenadas del punto deseado y los pesos de la neurona obtenemos, al menos en los primeros estadios del proceso, incrementos que, sumados a los valores presentes de θ_j, llevarían el brazo a puntos inaccesibles, o a puntos que, aun siendo físicamente accesibles, no serían visibles por las cámaras. Esto hace imposible tomar la medida de $\mathbf{v}_f$, y el punto en cuestión no puede ser usado, con lo que la red no aprende en ese ciclo, y deja la matriz A inalterada, susceptible de provocar otro error cuando esa neurona vuelva a ser seleccionada. R,M&S no se plantean este problema en su simulación. Según se puede ver en gráficas de su artículo, al robot le es permitido llegar a casi cualquier lugar del espacio, y se entiende que las cámaras pueden verlo allí donde esté. De este modo, y con tiempo suficiente, las matrices A se corrigen a valores razonables. Nosotros no podemos admitir esto, dado que al robot real le es físicamente imposible llegar, y a nuestra simulación realista basada en la Teoría de la Red de Tensores, obviamente también. Para remediarlo, decidimos inicializar los vectores de joints asociados a un grupo formado por un 10% del total de neuronas bien repartido por toda la red a valores razonables, aproximadamente los que le corresponderían en la porción de espacio real de la que se supone que esa neurona va a ocuparse. Esto quizá no debería hacerse en un algoritmo que se supone que debe aprender por sí mismo, pero es práctica común entre los implantadores de redes autoorganizativas, y fue la única solución razonable que pudimos encontrar. Con ella se mejoraron sustancialmente los primeros ciclos de aprendizaje, pero a continuación apareció otro problema grave relacionado con la estabilidad. Para

entenderlo, volvamos al algoritmo de actualización de los pesos y de los vectores de joints; dijimos que cuando una neurona era seleccionada como ganadora, sus vecinas se actualizaban también, tanto menos cuanto mayor era su distancia, de acuerdo a una función que llamamos $\gamma(t,d)$, siendo d la distancia topológica en la red entre ambas vecinas. R,M&S escogen para esta función la forma

$$\gamma(t,d) = \epsilon(t)e^{-\frac{d^2}{2\sigma^2(t)}} \tag{8.8}$$

siendo

$$\epsilon(t) = \epsilon_i(\frac{\epsilon_i}{\epsilon_f})^{\frac{t}{t_{max}}} \tag{8.9}$$

$$\sigma(t) = \sigma_i(\frac{\sigma_i}{\sigma_f})^{\frac{t}{t_{max}}} \tag{8.10}$$

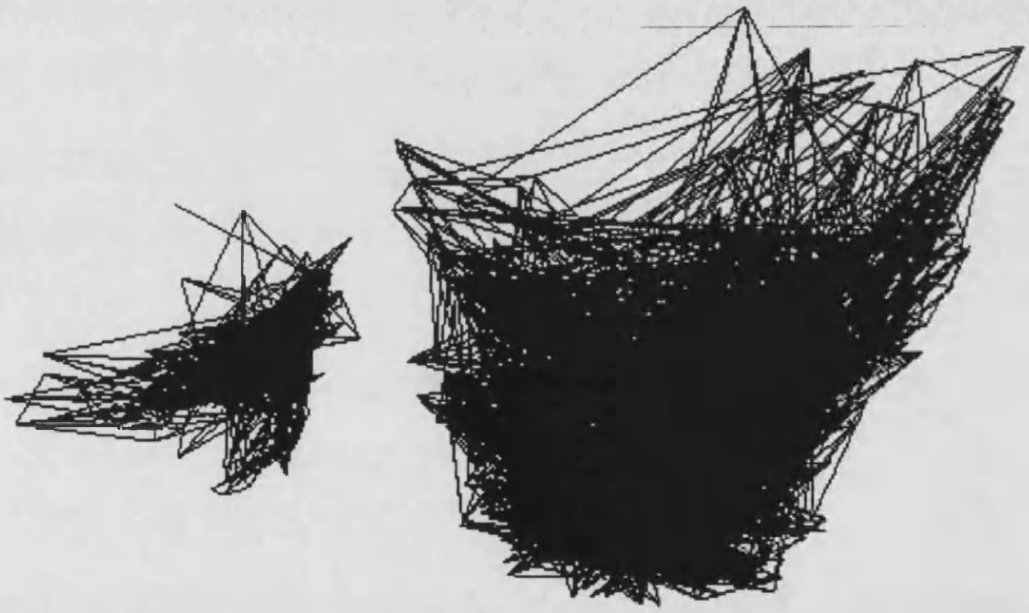
Los valores con subíndices i y f son, respectivamente, valores iniciales y finales para los parámetros en cuestión. Como se puede ver, γ es una gaussiana cuya altura y desviación típica decrecen con el tiempo. t_{max} es el número esperado de ciclos que pensamos que la red tardará en converger. Escoger este gran número de parámetros (nueve, dos para ϵ y dos para σ para las coordenadas visuales, otros tantos para las motoras, más el tiempo máximo) es realmente difícil. En principio, pueden depender del número de neuronas, de la distribución de los datos en el espacio de entrada, del tamaño de la muestra, del nivel de ruido, etc. Considerando el tamaño de nuestro espacio de trabajo comparado con el espacio simulado de R,M&S, y de acuerdo a la experiencia con la Red de Tensores (cada nodo de la cual viene a ser como una de estas neuronas, pero sin proceso de aprendizaje) escogimos una red de 218 unidades, y un número máximo de ciclos de 25000. Reservamos los demás parámetros para hacer pruebas con ellos, hasta encontrar valores razonables. Desgraciadamente, no pudimos encontrar tales valores, como vamos a explicar.

La red resultó extraordinariamente sensible a mínimas variaciones en los parámetros, particularmente en los σ: si eran escogidos inicialmente demasiado grandes, la extensión de cada campo perceptual era excesiva. Cada neurona influía en demasiadas vecinas, obteniéndose al cabo de unos 1000 ciclos un arracimamiento de todas las unidades en torno al centro de la distribución, en un único cluster. Podemos interpretar que cada nueva medida a partir de este estado era considerada como ruido que trata de alterar el estado estable, pero no lo consigue. La red total es estable, pero no ha convergido a la solución requerida.

Recíprocamente, si los σ eran escogidos demasiado grandes, no se daba ningún tipo de autoorganización, y la regla de Widrow-Hoff que debería hacer converger cada matriz, esta vez de modo más o menos autónomo, resultaba inestable, presentando de nuevo el problema original del envío a posiciones inaccesibles. Aquí la situación evolucionaba en dos sentidos: bien la mayoría de las neuronas generaban este tipo de posiciones, dejando de aprender, o bien saltaban alternativamente de una a otra posición a cada patrón de entrada, sin llegar a detenerse en un número alto de

ciclos (más de 6000). Eventualmente, la red terminaba en uno de los dos estados antedichos: colapsado o caótico. Además, resultaba extraordinariamente sensible al ruido (una medida mal tomada), que alteraba la matriz jacobiana de tal modo que todo punto subsiguiente al que esa neurona respondiese era mandado fuera del espacio de trabajo: es decir, la neurona quedaba inhábil para aprender.

Para evitar todo esto tratamos de situar los parámetros σ inicialmente en valores altos, y rápidamente modificarlos hacia valores pequeños para conseguir primero una convergencia gruesa, y a continuación, con cada neurona situada más o menos en su lugar, entrar en una fase de convergencia fina. Esto se consigue, según se puede ver en la ecuación 8.10, eligiendo valores bajos de σ_f. Nuevamente volvimos a intentar gran número de pruebas, y el resultado fue el mismo que en el caso anterior. Véase la figura 8.1 que muestra ambos estados. Son del mismo tipo que las representadas por R.M&S, y en ellas se ve en perspectiva el punto del espacio tridimensional real en que queda cada neurona, y las líneas imaginarias que la unen a sus neuronas vecinas. La situación ideal sería una red ortoédrica más o menos deformada



(a) (b)

Figura 8.1: Los dos estados finales de la red autoorganizativa

Cabe pensar en que este comportamiento pudiera ser debido a un fallo en nuestro programa. Estrictamente, no podemos tener la absoluta certeza de que no sea así, dado que ninguno de los programas de simulación de redes neurales que hemos usado, o de los que hemos solicitado información por la red contiene aún el modelo de Kohonen tridimensional. En nuestro descargo, diremos que ha sido revisado exhaus-

tivamente sin encontrar fallo, y que en conversación privada por correo electrónico con la única persona que contestó a un anuncio puesto en los grupos de news apropiados y que también está tratando de implantar este modelo señaló que obtiene problemas muy similares para la convergencia de los jacobianos, sin haber logrado aún un resultado aceptable.[1]

Como conclusión, digamos que el modelo de red autoorganizativa presenta muchos problemas cuando se implanta sobre un robot real que no aparecen en simulación, o al menos, no en una simulación que permita al brazo ir a cualquier lugar del espacio y que no considere el ruido. Es, de todos modos, deseo del autor seguir intentando en un futuro refinamientos al método que permitan solucionar los problemas encontrados.

8.3 Retropropagación (Backpropagation)

La retropropagación, en la literatura en inglés "backpropagation", es probablemente el método más usado en toda la investigación de redes neurales desde su invención por Rumelhart y Himton ([RHW86]).[2] Su objetivo es entrenar un perceptrón multicapa compuesto por al menos tres capas de neuronas, entrada, capa oculta y salida, cada una de las cuales suele estar completamente conectada con la siguiente (es decir, cada neurona de la capa i con todas las de la i+1). Véase la figura 8.2.

Se basa en la minimización del error cuadrático en la salida para cada entrada n-dimensional p, definido como

$$E^p = \frac{1}{2}\sum_{i=1}^{n}(d_i^p - a_i^p)^2 \tag{8.11}$$

lo cual se consigue por descenso de gradiente, según la fórmula

$$\Delta_p\omega_{ij} = -\gamma\frac{\partial E^p}{\partial \omega_{ij}} \tag{8.12}$$

en la que ω_{ij} es el peso de la conexión de la neurona i con la j, y γ un parámetro de aprendizaje fijado por el usuario. El cálculo de la derivada de E^p respecto a ω_{ij} es trivial para las neuronas de la última capa, y conduce a la regla delta usual, pero para las otras capas debe hacerse iterativamente, usando el error en las entradas de la capa siguiente, que son las salidas de la propia. Las fórmulas y el algoritmo pueden verse en el artículo original, y en cualquiera de los dos libros de redes citados en la introducción, y no incidiremos en él. Pero la parte que interesa de

[1] Esta persona es Connor Doherty, actualmente trabajando en el Institut de Cibernética de la Universitat Politécnica de Catalunya, a quien agradecemos su interés

[2] No obstante, Simpson en [Sim90], pág 113 apunta que varios algoritmos casi idénticos habían sido desarrollados antes en campos diferentes, como estadística o control.

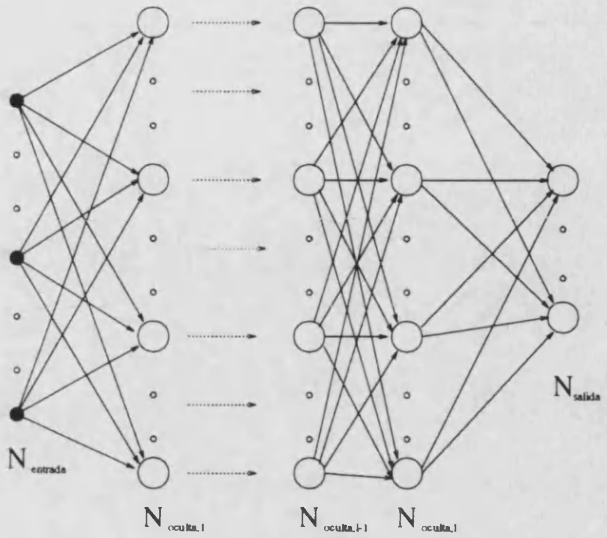


Figura 8.2: Esquema de un perceptrón multicapa

cara a nuestra aplicación es que el método de descenso por gradiente asegura que eventualmente se alcanzará un mínimo del error, pero no garantiza que tal mínimo sea el mínimo global. Visto de otro modo, para un conjunto de entrenamiento fijo, E^p es una función $E^p(\omega_{11}, \ldots, \omega_{1n}, \omega_{h1}, \ldots, \omega_{hl})$. Encontrar el mínimo de esta función de muchas variables significa que las derivadas parciales respecto a cada variable sean simultáneamente 0, pero esto sólo implica mínimo local. Es, por tanto, común que los perceptrones multicapa, tanto los entrenados por retropropagación como por otros métodos, caigan en mínimos locales. No hay una forma bien establecida de evitar esto; más adelante veremos un intento cuando expliquemos la correlación en cascada.

Veamos ahora nuestra implantación en concreto. Para este y los sucesivos experimentos usamos el programa SNNS (Stuttgart Neural Network Simulator), distribuído como software de uso público por el Institute for Parallel and Distributed High Performance Systems de la Universidad de Stuttgart. Es probablemente uno de los simuladores de redes actuales mejor construídos y dispone de un excelente programa de relación con el usuario.

Para el problema en concreto que nos afecta, primero normalizamos las entradas y las salidas a valores entre 0 y 1. Usamos inicialmente un perceptrón usual de tres capas, con 4 neuronas de entrada, dado que el patrón de entrada tiene, en principio, 4 dimensiones; 4 también para la salida, por análoga razón, y en la capa intermedia se probaron, en principio, 81 neuronas. Este número puede parecer grande, pero el número de patrones de entrada es también realmente grande, hasta 25000, y un número pequeño de neuronas se saturaría con rapidez. Los pesos de las

conexiones entre neuronas fueron inicializados a valores aleatorios pequeños. La tasa de aprendizaje, γ, fue escogida de 1.2 tras algunos ensayos, lo cual parecía dar los resultados menos insatisfactorios. Al final de cada ciclo el programa produce como salida el error cuadrático total tras la presentación de todos los patrones. Durante los primeros ciclos la velocidad de aprendizaje era muy grande. Al cabo de un número suficiente de ellos esta velocidad (variación del error entre un ciclo de aprendizaje y el siguiente) iba decreciendo; se recurrió a aumentar entonces la tasa de aprendizaje, lo cual en ese estado podía hacerse sin provocar inestabilidad, y se consiguió seguir aprendiendo durante algunos ciclos más, hasta llegar a la convergencia (los pesos no variaban, lo que quiere decir el error no disminuía), pero desgraciadamente el error final resultó enorme. No fué calculado con exactitud en todos los casos que se intentaron, ni en los que se describirán en los apartados siguientes, dado que una vez calculado en un caso, el error cuadrático total generado por SNNS nos puede dar una idea del órden de magnitud. Para este primer caso que calculamos fue 79.2 en los 20000 patrones y las cuatro coordenadas, lo cual da un error absoluto medio de aproximadamente 0.03 por patron y por coordenada, que después de invertir el reescalado y comparar con la salida esperada, daba un error medio en el espacio real de unos 270 mm., además de muchos puntos que enviaba fuera del espacio de trabajo. Esto es obviamente inaceptable. Para generar un error razonable en la salida el error antes de invertir el reescalado debería ser no mayor de 0.01, lo cual significa un error cuadrático total no superior a 8.

Otras pruebas con la retropropagación simple fueron primero, la reducción del número de dimensiones de la entrada. Como en nuestro problema sabemos de antemano que tenemos sólo tres coordenadas visuales independientes de las cuatro, tomamos como nuevas entradas la x_l, la x_r y la media de y_l e y_r. Esto no mejoró la velocidad de convergencia, pero sí el error final, que descendió a 65.3. Esto quizá pueda considerarse bueno desde el punto de vista de la red, pero en el robot sigue dando resultados reales tan pobres como el caso anterior. A continuación, y dado que se observaban algunas neuronas cuya activación raramente alcanzaba valores apreciables, se pensó que quizá había demasiadas. Se fué reduciendo paulatinamente el número, y tras varias largas pruebas sucesivas, se llegó a una red con 49 neuronas en la capa intermedia, tres en la entrada y cuatro en la salida, que dió un error cuadrático final después de caer en el mínimo local de 57.3, como vemos, muy alejado aún del máximo razonable de 8.

A continuación una última prueba consistió en usar una red con una capa de entrada, una de salida y dos capas ocultas, cada una con la mitad de neuronas. El resultado no mejoró sustancialmente, con error final de 59.0.

Evidentemente, podrían haberse hecho muchas más pruebas, pero nos parece manifiestamente absurdo variar cualquiera de los parámetros (número de neuronas, número de capas, tasa de aprendizaje, etc.) sin tener una idea clara de por qué los estamos variando. Aun cuando alguna combinación encontrada al azar diera resultados aceptables, seríamos incapaces de explicar por qué. Preferimos intentar otros

modelos, algunos de los cuales tienen una base más sólida. Comentaremos más sobre esto en las conclusiones a este capítulo.

Digamos, finalmente, que en teoría un perceptrón de tres capas es siempre una solución posible para cualquier transformación de $\Re^n$ en $\Re^m$. Está probado (véase [Fun89]) que cualquier función $f : \Re^n \longrightarrow \Re$ definida en un compacto K puede aproximarse sobre él por otra función, $\bar{f}$, definida sobre K como

$$\bar{f}(x_1, \ldots, x_n) = \sum_{i=1}^{N} c_i \phi \left(\sum_{j=1}^{n} (\omega_{ij} x_j - \theta_i) \right) \tag{8.13}$$

donde $\{c_1, \ldots, c_N\}$ y $\{\omega_{11}, \ldots, \omega_{1n}, \ldots, \omega_{N1}, \ldots, \omega_{Nn}\}$ son constantes a determinar y ϕ es cualquier función no constante, acotada y monótona creciente en $]-\infty, +\infty[$ de manera que, dado ϵ tan pequeño como se quiera se cumple que

$$\forall \mathbf{x} \in K, \max_{\mathbf{x} \in K} |f(x_1, \ldots, x_n) - \bar{f}(x_1, \ldots, x_n)| < \epsilon \tag{8.14}$$

Como puede verse, esta es exactamente la transformación que ejecuta un perceptrón de tres capas, siendo N el número de neuronas de la capa intermedia, y la función ϕ puede tomarse como la sigmoide, $1/(1 + e^{-x})$, que cumple las condiciones del teorema.

Análogamente se cumple que para cualquier función de $\Re^n \longrightarrow \Re^m$ existe una red del mismo tipo que ejecuta las transformaciones de las m funciones de salida, con la misma capa intermedia para todas ellas, y, en vez de una, m neuronas de salida, cada una con diferentes coeficientes, $\{c_{11}, \ldots, c_{N1}, \ldots, c_{1m}, \ldots, c_{Nm}\}$ que también cumple que

$$\forall \mathbf{x} \in K, \max_{\mathbf{x} \in K} \|f(x_1, \ldots, x_n) - \bar{f}(x_1, \ldots, x_n)\| < \epsilon \tag{8.15}$$

siendo $\| \ \|$ la métrica euclídea usual. Una versión un poco más fuerte de este teorema publicada por Ito ([Ito92]) extiende el resultado a cualquier función contínua sobre todo $\Re^n$, bajo ciertas hipótesis.

Estos resultados, aun siendo extremadamente interesantes a nivel teórico, son demostraciones de existencia, no constructivas, y no dan métodos para calcular los valores de los parámetros, ni siquiera del número de neuronas. Recientemente están comenzando a publicarse trabajos en este sentido, y se esperan inminentes resultados.

8.4 Retropropagación con momento (Momentum Backpropagation)

Este es una algoritmo casi idéntico a la retropropagación usual; la diferencia es que para evitar los problemas de oscilaciones, al variar un peso se introduce un término

de momento que es proporcional al último cambio realizado en él. El efecto de esto es que las zonas planas de la superficie de error se atraviesan relativamente rápido, mientras que la variación decrece cuando la superficie se vuelve rugosa. La nueva fórmula para la actualización de los pesos es

$$\Delta\omega_{ij}(t+1) = \gamma * \delta_j * o_i + \alpha\Delta\omega_{ij}(t) \tag{8.16}$$

donde se define $\delta_j = -\frac{\partial E}{\partial i_i}$, siendo i_i las entradas a la red, y o_i sus salidas. Esta primera parte es la fórmula de la retropropagación usual, y el segundo sumando, en el que α es una constante que se fija a algún valor que de resultados correctos, es el término de momento.

Nuestra implantación usó la estructura de red que mejores resultados había dado en el método anterior, y después de inicializar aleatoriamente los pesos se logró la convergencia en un número menor de ciclos, unos 120 en lugar de 200, pero también con un valor muy similar para el error cuadrático, oscilando alrededor de 58.5. Esto significa que hemos vuelto a caer en un mínimo local, quizá el mismo, pero más deprisa. En realidad, tampoco se esperaba otro resultado con este método, pero dada la facilidad con que SNNS simula las redes, una comprobación era obligada.

8.5 Propagación rápida (Quickpropagation)

Otro método similar, pero más elaborado para aumentar la velocidad de convergencia consiste en usar información local acerca de la curvatura de la superficie de error. Esto requiere el cálculo de las derivadas segundas de E^p. Para ello, las derivadas primeras se calculan iterativamente según el algoritmo de la retropropagación usual, y luego se asume un comportamiento localmente cuadrático de la superficie de error, y se incrementa cada peso como

$$\Delta(t+1)\omega_{ij} = \frac{S(t+1)}{S(t) - S(t+1)}\Delta(t)\omega_{ij} \tag{8.17}$$

siendo S la derivada parcial de la función de error respecto a ω_{ij} en el ciclo que se indique.

En nuestra implantación sobre la misma estructura de red los resultados no fueron, sin embargo, los mismos de casos anteriores. El enorme error de los ciclos iniciales, que los otros métodos reducían rápidamente para caer en el mínimo, provocaba aquí comportamientos erráticos. Desconocemos por completo el comportamiento de la función de error en puntos muy alejados del origen, los cuales generan una mayoría de salidas absurdas con respecto al problema real, y sospechamos que el cociente de la fracción en la ecuación 8.17 adoptaba alternativamente valores muy pequeños o muy grandes porque la suposición de comportamiento localmente cuadrático no puede asumirse para grandes saltos sobre la superficie. El resultado final fue que la

red no llegó a converger, y al cortar el aprendizaje en 500 ciclos seguía presentando comportamiento errático.

8.6 Retropropagación de recuperación (Resilient Backpropagation)

Este algoritmo muy reciente considera la topología local de la función de error para cambiar su comportamiento. Los pesos se actualizan según la siguiente regla:

$$\Delta(t)\omega_{ij} = \begin{cases} -\Delta & , \; si \; \frac{\partial E}{\partial \omega_{ij}}(t) > 0 \\ +\Delta & , \; si \; \frac{\partial E}{\partial \omega_{ij}}(t) < 0 \\ 0 & , \; si \; \frac{\partial E}{\partial \omega_{ij}}(t) = 0 \end{cases} \tag{8.18}$$

donde Δ es una constante, que depende del problema, pero como vamos a ver no es crítica. Esto no suele funcionar muy bien, dado que si las variaciones en la superficie de error se dan en intervalos más pequeños que Δ no se podrá capturarlas. Para remediarlo, cada peso ω_{ij} mantiene un valor de Δ local, que se actualiza como

$$\Delta(t)_{ij} = \begin{cases} \eta^+ * \Delta(t-1)_{ij} & , \; si \; \frac{\partial E}{\partial \omega_{ij}}(t-1) * \frac{\partial E}{\partial \omega_{ij}}(t) > 0 \\ \eta^- * \Delta(t-1)_{ij} & , \; si \; \frac{\partial E}{\partial \omega_{ij}}(t-1) * \frac{\partial E}{\partial \omega_{ij}}(t) < 0 \\ \Delta(t-1)_{ij} & , \; si \; \frac{\partial E}{\partial \omega_{ij}}(t-1) * \frac{\partial E}{\partial \omega_{ij}}(t) = 0 \end{cases} \tag{8.19}$$

donde $0 < \eta^- < 1 < \eta^+$

La magnitud de la actualización no viene dada por la de la derivada, sino sólo por su signo. En zonas planas de la superficie de error el descenso de gradiente va haciéndose cada vez más rápido, pero si un incremento se vuelve excesivo y hace saltar el punto de la red más allá del mínimo, la derivada cambia de signo y el valor de Δ_{ij} vuelve a decrecer. Y además en este caso se le fuerza a que cambie de signo (para retroceder hacia el mínimo que dejó atrás). Esto da información aproximada sobre la topología local de la función de error, pero evita los problemas de inestabilidad que vimos en la propagación rápida.

En nuestro problema, y como siempre con la misma estructura y valores iniciales aleatorios para los pesos, se consiguió una convergencia bastante rápida (70 ciclos antes de la estabilización) y muy regular, presentando la variación del error cuadrático entre cada ciclo y el anterior un comportamiento de decrecimiento sistemático en los primeros ciclos, y de cambio alternativo de signo en los siguientes, como era de esperar. Además el algortimo resulta prácticamente insensible a los valores iniciales de los Δ, que inmediatamente son cambiados a valores razonables, y en cuanto a η^+ y η^-, los provistos por defecto por SNNS funcionaron bien. El problema, sin embargo, vino a ser el mismo que en los casos anteriores: la caída en mínimo local con valor del error cuadrático de 57.5.

Aparte de este resultado, el algoritmo ha mostrado ser el más robusto de los usados para redes de tres capas, y puede revelarse valioso en otras aplicaciones a problemas más sencillos.

8.7 Antipropagación (Counterpropagation)

Este algoritmo es una especie de mezcla entre la retroprogación usual y las redes autoorganizativas de Kohonen. La topología de conexión es la misma que la del perceptrón multicapa, pero aquí la capa intermedia es de tipo competitivo, es decir, igual que explicamos en la red autoorganizativa tridimensional, se calculan todas las activaciones como

$$A_i = \sum_{j=1}^{n} w_{ij} u_j \tag{8.20}$$

y se escoge la neurona con mayor A_i como ganadora, sea la g. Entonces los pesos de las conexiones que la unen a las entradas son actualizadas según la regla de Kohonen,

$$\omega_{ig}(t+1) = \omega_{ig}(t) + \alpha(i_i - w_{ig}(t)) \tag{8.21}$$

es decir, se mueven los pesos hacia las entradas. Una vez actualizada, el patrón se presenta de nuevo, y los pesos de las salidas son cambiados con el mismo criterio:

$$\omega_{ig}(t+1) = \omega_{ig}(t) + \beta(o_i - w_{ig}(t)) \tag{8.22}$$

con lo cual la salida se mueve hacia la salida deseada.

El problema que se presentó en nuesta implantación, esta vez con 81 neuronas en la capa intermedia, es que no eran un número suficiente, dado el gran tamaño del problema. El error resultaba enorme, y variaba de modo errático a cada nueva presentación de los patrones en un orden aleatorio. Esto era esperable, pero el situar el número suficiente de neuronas para nuestro tamaño de problema hubiera sido, según comentamos en la introducción a la implantación de R,M&S, absolutamente desproporcionada. Esto hizo también inaplicable el método.

8.8 Correlación en Cascada (Cascade correlation)

Una vez probadas sistemáticamente las más importantes variaciones hoy en uso a la retropropagación intentamos dos métodos bastante modernos que quizá podrían habernos evitado el problema común a las anteriores de la caída en mínimo local. El primero de ellos es la correlación en cascada, propuesto por Scott Fahlman, y que trata de resolver el problema crucial de cuántas neuronas hacen falta para dar cuenta de una transformación determinada hasta un cierta precisión. Lo soluciona mediante

adición dinámica de las mismas, y se supone que construiría una topología cercana a la mínima posible. El algoritmo comienza inicialmente con una red de sólo dos capas, con las entradas conectadas directamente a las salidas. Se entrenan los pesos que conducen a las unidades de salida con cualquier algoritmo usual (nosotros escogimos la retropropagación simple) hasta que el error (que, obviamente, es aquí muy grande) deje de reducirse. Entonces se genera una colección de varias unidades candidatas, que se conectan mediante pesos de valor aleatorio con todas la unidades de entrada y con todas las ocultas (a medida que vaya habiéndolas). A continuación se maximiza la correlación entre la activación de las unidades candidatas y el error residual de la red entrenando todos los enlaces que conducen a tal unidad; este aprendizaje parcial cesa cuando la correlación no mejora más. En ese momento se escoge la unidad candidata con la máxima correlación, se congelan sus pesos para que ya no varíen más, y se la añade a la capa intermedia como unidad oculta. El proceso comienza de nuevo entrenando otra vez los pesos de las unidades de salida.

La idea que subyace es que cada nueva unidad añadida debería tratar de dar cuenta de lo que las anteriores no han sido capaces de aprender (es decir, del error) y de ahí la búsqueda de una correlación con el error residual. Para el entrenamiento de los pesos de salida se puede usar un descenso de gradiente idéntico al del perceptrón simple (dado que los pesos de capas anteriores están congelados), y para maximizar la correlación se usa un ascenso de gradiente, que es igual, pero con el signo del coeficiente que afecta a la derivada positivo. En la figura 8.3 se muestra un estadio en el que tres neuronas han sido añadidas. Los pesos de las conexiones están representados por los cuadrados, los vacíos, congelados, y los llenos, en proceso de aprendizaje.

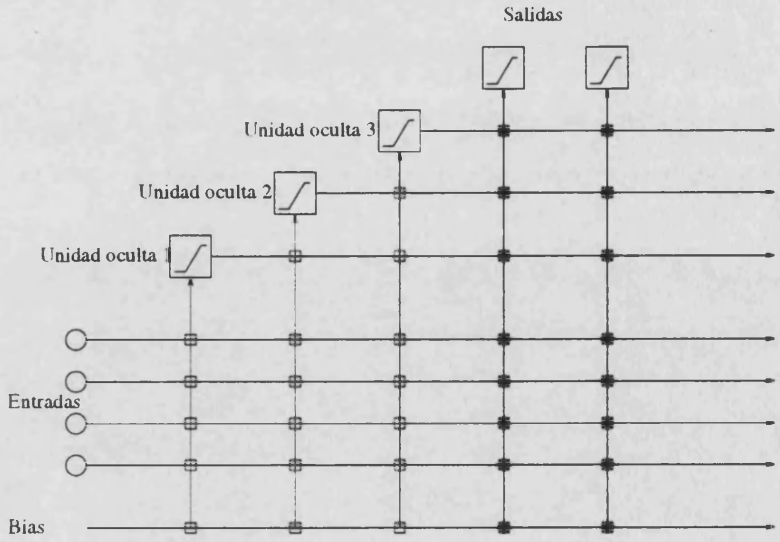


Figura 8.3: Esquema de la red para correlación en cascada

Como se puede entender, el proceso de aprendizaje de este tipo de redes resulta

extraordinariamente lento, porque en cada paso antes de añadir una nueva neurona hay que ejecutar dos procesos completos de retropropagación, cada uno con un número suficiente de ciclos para alcanzar la estabilidad. Considerando el tamaño de nuestro problema y de la red que previsiblemente sería necesario alcanzar, y dada la velocidad que observamos en los primeros ciclos, estimamos el límite inferior al tiempo de aprendizaje en varias semanas de cómputo para una estación de trabajo potente. Por ello, decidimos usar el método combinado con la retropropagación usual de un modo diferente. Se toma la red resultado del método de retroprogación cuando se ha estabilizado en un mínimo local, se congelan sus pesos, y se la usa como red inicial de la correlación en cascada, dejando que ésta añada un cierto número pequeño de neuronas. Era de esperar que la función E^p que tenía mínimo local en cierto punto, al transformarse ahora en una función diferente de más variables (los pesos de las nuevas neuronas) no tendría mínimo local en el mismo lugar, pero sin embargo los valores del nuevo mínimo no estarían excesivamente alejados de los actuales. Ejecutamos este procedimiento añadiendo grupos de tres neuronas entre cada ciclo de retropropagación, para ver si el error se reducía, y los resultados fueron desalentadores. El error cuadrático total apenas bajó unas décimas, quedando igualmente en un valor intolerable para la aplicación real. Ello implica que el mínimo en que la función de error ha caído lo es también en la dirección de las demás dimensiones del espacio que se agreguen. Puede pensarse que el número de neuronas añadido era aún corto, pero la adición de un número significativo de ellas provoca los problemas de tiempo antes comentados.

8.9 Base de Funciones Radiales (Radial Basis Functions)

El último método que decidimos probar es un tipo un tanto diferente de red neural, que tiene un mecanismo de implantación muy similar y también permitiría una paralelización fácil. Además, tiene la cualidad de poseer una fundamentación matemática absolutamente correcta; es el método de aproximación de funciones conocido como base de funciones radiales. La idea es que cualquier función de $\Re^n$ en $\Re^m$ puede escribirse como una combinación lineal de otras funciones desplazadas a un número suficiente de puntos del espacio, según

$$f(\vec{\mathbf{x}}) = \sum_{i=1}^{K} c_i h(|\vec{\mathbf{x}} - \vec{\mathbf{t_i}}|) \tag{8.23}$$

donde K es el número de funciones base (aproximadores) que estamos tomando, y los $\vec{t_i}$ son un conjunto de vectores que actúan como centros y que deben ser encontrados. Igualmente los c_i son parámetros para determinar. El argumento de h es la distancia euclídea entre nuestro patrón de entrada y cada vector $\vec{\mathbf{t_i}}$. Generalmente se suele tomar una gaussiana como función h, pero en cualquier caso deben ser funciones

con simetría radial, con un máximo en 0 y que decrezcan uniformemente al tender la distancia (el radio) a infinito. En el caso usual en que pretendemos emplear el método para aproximar un conjunto de datos discreto el objetivo será minimizar el error, dado por la siguiente función:

$$H[f] = \sum_{j=1}^{p} (y_i - f(\vec{\mathbf{x}_j}))^2 + \lambda \|Pf\|^2 \tag{8.24}$$

donde $\|Pf\|$ es la matriz de autocorrelación de los valores $h(|\vec{t_i} - \vec{t_j}|)$. Este segundo término es un estabilizador para forzar a que f sea lo más suave posible, y λ es un parámetro que determina su influencia. Con λ igual a 0 lo que obtendríamos para la aproximación de la función es un sistema de ecuaciones lineales con p ecuaciones (una por cada patrón de entrada) y K incógnitas, del cual se buscaría la seudosolución que minimizase H, la cual, según vimos al tratar el método estadístico, era la seudoinversa de Moore-Penrose. En el caso en que añadamos el estabilizador, el sistema de ecuaciones puede ser resuelto en los c_i por la fórmula

$$\vec{\mathbf{c}} = (G^T \cdot G + \lambda G_\square)^{-1} \cdot G^T \cdot \vec{\mathbf{y}} \tag{8.25}$$

donde

$$G = \begin{pmatrix} h(|\vec{\mathbf{x_1}} - \vec{\mathbf{t_1}}|) & \dots & h(|\vec{\mathbf{x_1}} - \vec{\mathbf{t_k}}|) \\ \vdots & \ddots & \vdots \\ h(|\vec{\mathbf{x_p}} - \vec{\mathbf{t_1}}|) & \dots & h(|\vec{\mathbf{x_p}} - \vec{\mathbf{t_k}}|) \end{pmatrix} \tag{8.26}$$

y

$$G_\square = \begin{pmatrix} h(|\vec{\mathbf{t_1}} - \vec{\mathbf{t_1}}|) & \dots & h(|\vec{\mathbf{t_1}} - \vec{\mathbf{t_k}}|) \\ \vdots & \ddots & \vdots \\ h(|\vec{\mathbf{t_k}} - \vec{\mathbf{t_1}}|) & \dots & h(|\vec{\mathbf{t_k}} - \vec{\mathbf{t_k}}|) \end{pmatrix} \tag{8.27}$$

Una vez calculadas estas matrices, lo que hemos obtenido es la solución óptima para los c_i, dados unos valores particulares de los vectores $\vec{t}$ y de los parámetros libres de los que dependa la función h. Si escogemos para h una gaussiana, $h(d) = e^{-pd^2}$, el valor de p es el parámetro libre, distinto en general para cada neurona. Otra posibilidad es escoger $h(d) = \sqrt{d^2 + p}$ con p libre, etc.

El procedimiento práctico para ejecutar este algoritmo consiste en escoger una función para h, escoger los $\vec{t_i}$, bien como valores uniformemente repartidos por toda la distribución de entrada, bien como centros de agrupamientos (clusters) escogidos por un algoritmo de autoorganización tipo Kohonen. Entonces se comprueba el comportamiento de la red para diferentes patrones de entrada y se ajusta el parámetro libre (bias) de todas las unidades a un mismo valor, de tal modo que sólo un pequeño grupo de ellas responda con activación elevada a cada patrón. En cierto sentido, esto es como tratar de codificar los patrones de entrada como las posibles combinaciones de grupos de un número pequeño de unidades. A continuación se

ejecuta el procedimento de seudoinversión de las matrices apropiadas, para conseguir los valores de los c_i. Esto normalmente nos conducirá a un error cuadrático bastante grande, porque aun siendo óptimo para los valores dados de los parámetros libres, nadie asegura que sea el óptimo que se podría encontrar dados otros valores de éstos, y ni siquiera que sea bueno. El procedimiento seguirá, pues, tratando de aprender los valores para las componentes de cada $\vec{t_i}$ y para los p mediante descenso por gradiente.

Una vez realizados todos los pasos anteriores habiendo escogido una red de 80 aproximadores con la gaussiana como función base, unos centros de cluster dados por el algoritmo autoorganizativo y un valor del bias (parámetro p del exponente de la gaussiana) apropiado para que no más de 6 u 8 de ellos alcanzasen activaciones altas para cada patrón, se ejecuto el cómputo de las matrices dadas por las ecuaciones 8.26 y 8.27. Es necesario hacer notar que con el parámetro de suavizado la λ igual a 0 la matriz para seudoinvertir, que en este caso es sólo la G, resultó casi singular. Ello es porque al ser la gaussiana de decrecimiento tan rápido, los valores de esta matriz son próximos a 0 o a 1; creemos que esta es la causa de los problemas de estabilidad que encontramos al iniciar el algoritmo de aprendizaje. El coeficiente de la derivada del error respecto al bias resultó ser extraordinariamente crítico. La más mínima variación provocaba el desvío del parámetro p a valores negativos, lo cual es obviamente absurdo porque en ese caso la exponencial resulta con exponente positivo y tiende a infinito, volviéndose inhábil para aproximar una función que sabemos acotada. De modo similar, el coeficiente de variación de los centros era igualmente crítico, aunque no tanto como el anterior, tendiendo en ocasiones a llevar los valores de los supuestos vectores centro del cluster fuera de la distribución.

Visto lo anterior, decidimos usar una función de variación más suave, la raíz, $f(d) = \sqrt{d^2 + p}$, procediendo para fijar los parámetros iniciales como en el caso anterior. Esta función se presta más a provocar interacciones indeseables entre aproximadores vecinos, y consiguientemente, debería converger con más lentitud, pero ser más estable. Efectivamente, así ocurrió, pero el valor final para el error se alcanzó en muy pocos ciclos, y resultó de nuevo inaceptablemente alto, 280.

Con esto concluímos el ensayo de todos los métodos de redes neurales que pudimos razonablemente suponer válidos para abordar nuestro problema. Como se ve, las dificultades en la estabilidad y caída en mínimos locales han dado al traste con todos nuestros intentos. La conclusión es que probablemente sea necesario esperar un poco a que una fundamentación matemática más profunda de las redes neurales, bien en su conjunto, bien separadamente para cada método, de soluciones constructivas al problema de la elección de parámetros y su relación con la estabilidad. Desgraciadamente, mis conocimientos matemáticos me permiten entender los teoremas pertinentes, pero no demostrarlos. Recientemente varios matemáticos americanos y japoneses han mostrado interés por el tema, y el futuro es prometedor. Las redes no son la panacea a todos los problemas de clasificación y aproximación, de hecho, algunos modelos de ellas no hacen sino dar una formulación diferente a problemas ya resueltos en estadística y otras áreas de las matemáticas. Pero aquellos que puedan

ser abordados por estos métodos tendrán la ventaja añadida de su extrema facilidad de paralelización, e incluso sus posibilidades de implantación con dispositivos ópticos, y aunque sólo fuera por esto, son un modelo que debe seguir siendo investigado.

Respecto a la posibilidad de usar otros algoritmos que garantizan, al menos en teoría, la convergencia al mínimo global (algoritmos genéticos, algoritmos de relajación estocástica, etc.), limitaciones de tiempo nos lo han impedido, pero se comentará algo en el capítulo final.

Capítulo 9

Comparación de resultados y análisis estadístico.

En este capítulo vamos a tratar de analizar los resultados proporcionados por todos los métodos para tratar de sacar conclusiones acerca de la bondad o limitaciones de cada uno de ellos. El estudio lo vamos a hacer usando las técnicas estadísticas apropiadas para analizar medidas repetidas, es decir, medidas que varios procedimientos generan sobre el mismo conjunto de datos de test, en las mismas condiciones. La medida que hemos escogido como indicador del error (de la bondad) de cada método es la distancia a la que el punto terminal del brazo queda de la posición a la que debería haber llegado.

Como se ha repetido a lo largo de toda la tesis, nosotros no usamos (y consiguientemente, no podemos ir con nuestros métodos) a puntos expresados en coordenadas cartesianas, sino en coordenadas de la cámara. No obstante, por motivos de claridad y para tener una referencia del tipo habitual a la que el lector esté acostumbrado, tenemos que dar las distancias euclídeas en milímetros. El procedimiento para hacer esto consiste en ir al punto cartesiano donde vamos a medir el error usando la cinemática inversa; entonces tomar medida allí de las coordenadas visuales del punto terminal. A continuación se usa el método correspondiente que estemos ensayando para tratar de llevar el brazo a ese punto, y una vez haya terminado su trabajo, se toma lectura de los joints, y usando la cinemática directa, se determina su posición cartesiana, y se ve cuán alejada está de la original. El usar la cinemática puede provocar una imprecisión añadida, pero no tenemos más remedio si queremos dar los resultados en unidades comprensibles (milímetros, y no cuentas o pixels). En cualquier caso, si la cinemática no es precisa en una determinada zona del espacio, cualquier error sistemático cometido será similar para los dos puntos, el deseado y el obtenido, dado que son próximos.

Respecto a los puntos de test, escogimos una red ortoédrica de 528 puntos distribuídos regularmente por toda el área de trabajo. El único método que, en principio, podría ser por sí mismo sensible al lugar de elección de los puntos es la Teoría de la Red de Tensores, dado que las medidas en ella están tomadas también en una red en concreto. Esto no quiere decir que alguno o algunos métodos no puedan comportarse de modo diferente en función de la zona general del espacio, como luego

veremos, pero en principio el error de los métodos entrenados con nubes de puntos aleatorias no debe variar de modo sistemático para pequeñas variaciones del punto.

Con objeto de evaluar aquellos puntos que más error podrían llegar a dar respecto al método de la Teoría de la Red de Tensores, y tener así una cota superior para él, se escogieron los nodos de la red ortoédrica como los centros de los cubos generados por la red de medidas tomadas en la TNT. Esto significa que son los puntos más alejados que es posible de aquellos en los que originariamente medimos.

Según dijimos al principio, y tal como aconseja la teoría estadística sobre el tratamiento de medidas repetidas, (ver [Seb84]) asociado a cada punto tendremos un vector, cuyas componentes representarán el error dado por cada uno de los métodos, que en nuestro caso son:

Método	Abreviado en adelante
Teoría de la Red de Tensores sin realimentación	DTNT0
Teoría de la Red de Tensores, 1 ciclo de realimentación	DTNT1
Teoría de la Red de Tensores, 2 ciclos de realimentación	DTNT2
Método CMAC, 2 funciones y 32 intervalos de cuantización	D2_32
Método CMAC, 4 funciones y 32 intervalos de cuantización	D4_32
Método CMAC, 8 funciones y 32 intervalos de cuantización	D8_32
Método CMAC, 16 funciones y 32 intervalos de cuantización	D16_32
Método estadístico de la regresión	DESTAD

De entre todas la CMACs posibles, la elección precisamente de éstas se hizo atendiendo sólo al error medio, como manera rápida de ver cuáles tenían alguna posibilidad de competir como métodos válidos.

El procedimiento seguido para la evaluación de los métodos tiene dos fases: primero, un análisis multivariante del vector de medidas considerado conjuntamente, para determinar si hay o no diferencias significativas entre los valores medios de unas y otras de sus componentes. Adelantamos que la conclusión es que efectivamente existe una diferencia apreciable, y esto nos lleva a la segunda fase, que podemos considerar dividida en dos apartados: uno que trata de averiguar si hay diferencias, tanto globales (de todos los métodos simultáneamente considerados) como particulares (de cada método en zonas diferentes del espacio), y otro apartado que establece contrastes univariantes entre cada par de métodos en toda el área de trabajo, y entre sus diferencias zona a zona. Este segundo tratamiento no es estrictamente correcto, dado que, al haberse tomado las medidas en los mismos puntos, están ligadas entre sí; por ello deberá considerarse exploratorio, y sus resultados tomados con cautela.

Las cuestiones anteriores han sido formuladas como diferentes hipótesis que hemos contrastado. El método habitualmente usado en estadística es establecer una hipótesis, y considererla correcta, salvo que los datos la rechazen. En términos cuantitativos esto significa que se le asignará un valor de significación (p-valor) que será tanto

más próximo a 0 cuanto más fuertemente se rechaze la hipótesis. El procedimiento empleado para calcular p-valores depende de la hipótesis y del tipo y número de los datos, y su descripción precisa está fuera del alcance de esta tesis. Baste decir que el programa que usamos (SPSS) está entre el software más potente y probado para este tipo de tratamientos y su elección del método para calcular los p-valores es la comunmente aceptada y contrastada por los estadísticos. Según recomendación de la persona especializada consultada para el tratamiento de nuestros datos, dado el tamaño de nuestra muestra, un valor de significación superior a 0.01 debería ser considerado como suficiente para aceptar la hipótesis que se proponga en cada caso.

Veamos para empezar una tabla de medias, desviaciones típicas, valores máximos y mínimos y número de casos para cada método que permita una primera comparación gruesa. De aquí en adelante la unidades son siempre milímetros. El número de casos varía porque hay algunos puntos sin ningún valor (puntos faltantes) a los que un método determinado no llega (para ser exactos, el error es tal que envía el brazo a una posición inaccesible del espacio de trabajo). En el peor de los casos estos puntos son 15, un 2.8% del total.

Variable	Media	Desviación std.	Mínimo	Máximo	N. casos
DTNT0	3.42	4.10	0.11	29.79	513
DTNT1	2.74	2.91	0.11	24.98	513
DTNT2	2.58	2.47	0.11	23.24	513
D2_32	9.18	10.83	0.12	130.33	527
D4_32	8.61	12.55	0.11	208.83	527
D8_32	8.74	9.09	0.11	118.47	525
D16_32	9.08	9.13	0.12	117.42	526
DESTAD	26.72	20.95	0.50	82.69	528

Tabla 9.1: Estadísticos principales de cada método

En principio, el método con menor media es la Teoría de la Red de Tensores con dos ciclos de realimentación, como era de esperar. Los métodos CMAC dan un resultado bastante tolerable, y si de entre ellos tenemos que escoger uno, el D8_32 parece ser el mejor, porque, aun cuando el error de D4_32 es ligeramente menor, su desviación típica es mayor, lo cual implica menos regularidad. Nótese el altísimo valor del máximo error encontrado para todas las CMAC; puntos como éste, llamados en estadística observaciones aberrantes, provocan resultados anormalmente altos para la media, y generalmente requieren algún tipo de tratamiento especial. En el caso de las CMAC son debidos a que, según vimos, la precisión para la aproximación de la función desconocida en una zona depende del número de datos tomados en ella. Por tanto, los puntos en la frontera del espacio de trabajo que tienen menos vecinos necesariamente deben perder precisión. En el caso de la Red de Tensores, si un punto no tiene un tensor vecino la aproximación lineal con uno demasiado alejado suele provocar una posición absurda, y de ahí que 15 puntos sean inalcanzables. En

las CMACs, sin embargo, tales puntos son alcanzados, pero con gran error. Respecto al método de regresión, es el peor por resultados globales, pero es el único que accede a todos los puntos sin enviar ninguno fuera de los límites, y casi el único (junto con DTNT2) cuya desviación típica es inferior a la media, lo que implica, dentro de sus limitaciones, cierta regularidad.

Veamos ahora una comparación entre los métodos. Se calculó para cada dos métodos, a y b, la correlación entre las medidas de ambos, y el p-valor correspondiente a la hipótesis "El error medio de a es igual al error medio de b, comparando el error punto a punto" (es decir, apareando los valores observados). Valores de p nulos indicarán que la hipótesis debe rechazarse, y cuanto más altos sean, en mayor grado deberemos aceptarla. Recalquemos una vez más que valores numéricamente diferentes de las medias no necesariamente son considerados estadísticamente como diferentes, su similitud o discrepancia depende del número de datos, de la desviación típica de cada método, de cómo y dónde se toman las medidas, etc, y eso es precisamente lo que mide el parámetro p.

El p-valor que hallamos para esta hipótesis era 0 en todos los casos, excepto en los que involucraban a unas CMAC con otras, para las cuales p valía

	D4_32	D8_32	D16_32
D2_32	0.671	0.230	0.322
D4_32	—	0.281	0.401
D8_32	—	—	0.653

Tabla 9.2: p-valor para la comparación entre medias.

De acuerdo a los valores nulos que hemos encontrado para p, las tres variantes de la Teoría de la Red de Tensores pueden considerarse respecto a su media estadísticamente diferentes, según se deducía en una primera mirada a la tabla 9.1. Sin embargo, ocurre lo contrario con las CMACs. De hecho, son tan similares, que escogeremos una sola como representante, la D8_32, que habíamos dado como mejor, para no complicar demasiado el análisis. Respecto a cada uno de los grupos, sus medias resultan claramente distintas entre sí. Un resultado añadido del método, también generado como salida por SPSS, es la correlación entre los métodos y el valor de significación de dicha correlación. La hipótesis que se contrasta aquí es "Los errores del método a son independientes de los del método b". La significación obtenida para esta hipótesis y el valor para el coeficiente de correlación son como sigue:

	DTNT1	DTNT2	D2_32	D4_32	D8_32	D16_32	DESTAD
DTNTO	0.781/0.000	0.690/0.000	0.020/0.646	-0.018/0.687	0.007/0.880	0.036/0.414	0.245/0.000
DTNT1	—	0.900/0.000	0.058/0.190	0.009/0.841	0.044/0.319	0.052/0.241	0.172/0.000
DTNT2	—	—	0.026/0.561	0.003/0.943	0.043/0.326	0.031/0.488	0.159/0.000
D2_32	—	—	—	0.698/0.000	0.646/0.000	0.631/0.000	-0.017/0.700
D4_32	—	—	—	—	0.801/0.000	0.690/0.000	0.002/0.963
D8_32	—	—	—	—	—	0.764/0.000	0.494/-0.030
D16_32	—	—	—	—	—	—	-0.010/0.817

Tabla 9.3: p-valor para la correlación entre métodos

De esta tabla se observa que los errores en las tres realizaciones de la Teoría de la Red de Tensores están fuertemente relacionados, como es lógico, puesto que en cada punto cada uno procede del anterior por realimentación. También hay correlación entre todas las CMAC, lo cual se debe al hecho de haber sido entrenadas con la misma nube de puntos, que ha llenado sus tablas de modo similar. Pero hay todavía un efecto curioso: un valor no muy alto, pero apreciable de la correlación entre la regresión y la Teoría de la Red de Tensores. Interpretamos este hecho recurriendo al fundamento de estos métodos: en realidad, ambos son modelos lineales, pero uno de ellos local, y el otro global. Si la función que tratamos de aproximar presenta algún tramo razonablemente lineal, o si alguna de sus componentes lo es, (y vimos en las gráficas del capítulo 5 que la zed lo era, y la transformación de la zed a la z cartesiana es en nuestro brazo lineal) tal dependencia debe reflejarse de algún modo.

La conclusión de todo esto es que efectivamente hay diferencias significativas entre unos y otros métodos, considerándolos desde un punto de vista conjunto (Recuérdese que este estudio se ha hecho usando las herramientas para el análisis multivariante, en concreto el procedimiento manova[1]). Ahora el problema estriba en hallar dónde radican esas diferencias, tanto espacialmente como entre métodos. Una primera hipótesis que nos parece razonable hacer sobre el comportamiento global de todos ellos es suponer que son más precisos en unas zonas del espacio que en otras, y en concreto que lo son más en el centro del área de trabajo que en la periferia. Esto viene sugerido por el hecho de las cámaras apuntan hacia el centro, y puntos cercanos al eje óptico sufren menos la deformación perspectiva, por lo que el tamaño real del área abarcada por un pixel es menor. Además de ello, los puntos alejados del centro son aquellos en los que comprobamos experimentalmente que la cinemática del robot daba más error, y por último al estar cerca de la frontera de accesibilidad, hay menos vecinos de un punto dado medidos con los que llenar las tablas de las CMACs o la red de la TNT. Así pues, dividiremos el espacio en dos zonas, un ortoedro central, y el resto, de tal modo que aproximadamente 1/2 de los puntos de test queden en la primera zona, y el resto fuera, en lo que llamaremos zona 2. La tabla de medias y desviaciones por métodos y zonas se muestra a continuación:

[1] Multivariate Analisys of Variance

Método	Media zona 1	Desv. std. zona 1	Media zona 2	Desv. std. zona 2
TNT0	3.326	3.917	3.486	4.104
TNT1	2.516	2.612	2.902	3.110
TNT2	2.407	2.141	2.701	2.678
D8_32	7.416	8.791	10.203	12.411
DESTAD	20.415	17.541	30.631	21.779

Tabla 9.4: Estadísticos por zonas para cada método

Estos valores muestran que, en principio, el comportamiento por zonas sí parece ser diferente en cada método, en unos, como DESTAD, de modo más evidente que en otros, como TNT0, pero para comprobarlo haremos igual que antes un análisis que nos de los p-valores para contrastar la hipótesis "El método a se comporta de modo distinto en la zona 1 y en la zona 2". El procedimiento que se ha usado aquí es el llamado Test de la t. Estos valores son:

Método	p-valor
TNT0	0.677
TNT1	0.133
TNT2	0.174
D8_32	0.002
DESTAD	0.004

Tabla 9.5: p-valor de la discrepancia por zonas de cada método

De acuerdo con ellos, la CMAC y el método de regresión sí son claramente diferentes de una zona a otra, lo cual es consistente con los comentarios que hicimos acerca de la precisión de las medidas en zonas extremas. Respecto a la Teoría de la la Red de Tensores, su uso sin realimentación muestra ser sustancialmente igual en ambas zonas, y las diferencias tampoco son grandes en el caso de usar realimentación, lo cual quiere decir que la mejora introducida por ésta opera igualmente bien sobre toda el área de trabajo. Las variaciones nada espectaculares se deben, por otra parte, a que la TNT2 está ya rozando el límite de la precisión experimental de nuestro dispositivo debida al sistema de visión y al control de los joints, y por tanto los errores son más debidos a estas causas que al método en sí. Es interesante hacer notar que en la prueba sobre el robot los errores visuales quedaban en más del 80% de los casos reducidos a 1 o 0 pixels en cada dimensión, lo cual visto como distancias se traduce en valores que van entre 1 y 3 mm., según lo alejado que esté el brazo de la cámara.

Como última comprobación, se construyeron las variables diferencia entre cada par de métodos para verificar si dependen o no de la zona. La tabla de p-valores

para la hipótesis "la diferencia entre los métodos a y b es la misma en la zona 1 que en la 2" es:

Método	p-valor	
TNT0-TNT1	0.342	(+)
TNT0-TNT2	0.623	(+)
TNT0-D8_32	0.014	(*)
TNT0-DESTAD	0.000	
TNT1-TNT2	0.428	(+)
TNT1-D8_32	0.018	(*)
TNT1-DESTAD	0.000	
TNT2-D8_32	0.014	(*)
TNT2-DESTAD	0.000	
D8_32-DESTAD	0.000	

Tabla 9.6: p-valor de la discrepancia por zonas de las diferencias entre cada par de métodos

donde el método usado ha sido de nuevo el Test de la t, pero para la diferencia de errores.

De esta interesante tabla se pueden sacar varias conclusiones: la primera es que las diferencias entre los errores de todas las TNT (es decir, las correciones debidas a la realimentación, marcadas con +) no dependen la zona del espacio, son igualmente válidas en cualquier lugar donde haya un tensor correctamente medido. Otra conclusión es que las diferencias entre el método de regresión y los demás sí dependen fuertemente de la zona, lo cual, volviendo al comentario acerca de la correlación con la TNT, prueba que el área central es mucho más lineal que la periferia. Respecto a las diferencias entre CMAC y TNT, los p-valores (marcados con *) rozan el límite máximo de rechazabilidad que nos habíamos marcado (0.01) y por tanto no se puede decir que aporten mucha información en este caso, aunque tienden más a aceptar la hipótesis que a rechazarla.

Como conclusiones generales del análisis, podemos extraer las siguientes:

- Globalmente considerados, los métodos son diferentes. Las diferencias más importantes se dan entre el método de regresión y la Teoría de la Red de Tensores y entre la regresión y las CMAC. Las diferencias entre la TNT y las CMAC también son muy apreciables, aunque algo menores.

- Las diferencias por zonas señalan a la TNT como el método más robusto, en tanto que es el que mejor y más regularmente se comporta en todo el espacio. El método de regresión y las CMAC presentan un comportamiento bastante deficiente en zonas periféricas.

- Todos los métodos de CMAC son de comportamiento extremadamente similar, por lo que el único criterio razonable para usar uno u otro es atender a su media y desviación típica, pero con la cautela que las conclusiones establecidas suponen.

Otra conclusión importante que podemos sacar es que la superioridad del método de la Teoría de la Red de Tensores se basa no tanto en su mayor precisión como en su capacidad de realimentación. Un experimento interesante que realizamos para comprobar esto fue desalinear intencionadamente ambas cámaras aproximadamente 1 cm. dando a cada una un pequeño golpe. Los tensores medidos siguen siendo válidos, pero estarían ahora asociados a puntos que no son los que les corresponden, sino otros relativamente próximos. Sin embargo, como la variación de las componentes del tensor entre cada par de puntos medidos no es muy grande, cada uno representa todavía una aproximación tolerable, obteniéndose que de diez puntos en los que se hizo el experimento, el brazo llegó con el mínimo error visual de un pixel por dimensión a ocho de ellos, aunque después de un número de ciclos de realimentación más alto de lo usual: entre 4 y 6. Esto es prueba de una cierta robustez del método ante variaciones pequeñas del entorno.

Finalmente, un comentario acerca de la memoria usada: dejando aparte el método de regresión, cuyo resultado es una fórmula y no una tabla, vimos en los capítulos correspondientes que la red con los tensores ocupaba en nuestro caso unos 60 Kb, y la CMAC, no más de 4x64 Kb= 256 Kb. En cualquier caso, vemos que las consideraciones de memoria no serían decisivas en el caso de la TNT, aun para áreas de trabajo incluso 8 veces mayores, como las que podría abarcar un robot industrial real, puesto que para tal método la memoria es lineal con el volumen del espacio de trabajo. Pero en el caso de la CMAC, si queremos mantener la misma resolución deberemos duplicar el número de intervalos de cuantización (si el volumen de octuplica, la arista se duplica), y en términos de memoria, como ésta era proporcional a KQ^N, se multiplicará por 2^N, en el caso de nuestras CMACs de tres dimensiones, por 8. La memoria total sería, pues, del orden de 4x(8x64) Kb= 2 Mb. Esto todavía sigue siendo bastante aceptable para la mayoría de los ordenadores que operan hoy día.

Capítulo 10

Conclusiones y futuras mejoras.

En este capítulo vamos a resumir las principales conclusiones, tanto generales como particulares, a que este trabajo nos ha llevado, y de acuerdo con ellas y con la percepción que tenemos del estado actual de la investigación en Robótica, trataremos de establecer las mejoras que deberán hacerse en la continuación de esta aproximación y las nuevas líneas de trabajo que deberán abrirse.

Como conclusiones particulares, reiterar la comparación entre los métodos hecha en el capítulo anterior, según la cual la Teoría de la Red de Tensores se revela como el más apropiado para este tipo de tarea, fundamentalmente gracias a su capacidad de usar la realimentación sobre la señal visual. El error final roza los límites de precisión del sistema con el que estamos trabajando, que vienen, como dijimos, impuestos por las deficiencias mecánicas y de construcción del robot y por la resolución limitada de las cámaras y placas digitales. Pero estos problemas, en particular el último, son comunes a todos los sistemas robóticos que usan visión, y nos parece que el nuestro es un método elegante de abordarlos, subsumiendo transformaciones intermedias innecesarias. Adolece de varios defectos importantes, como el tiempo de toma de medidas, y la necesidad de trabajar con cámaras fijas. Pero el primero se puede resolver de modo inmediato con hardware específico para la visión, y sobre el segundo comentaremos algo inmediatamente al hablar de mejoras y líneas futuras.

Respecto al modelo de Computador Cerebelar Aritmético, aun sin alcanzar la precisión que sería necesario para hacerlo operativo, se ha mostrado válido como método general para encontrar transformaciones desconocidas entre espacios cualesquiera de entrada y salida, si se dispone de un número suficiente de ejemplos. Este es un punto importante que limita el uso, por el largo tiempo de aprendizaje que puede llegar a requerir. Por otra parte, aun cuando no se hace notar en ninguna de las aplicaciones que lo usan, es necesario conocer de antemano el número real de dimensiones (de coordenadas independientes) del espacio de entrada, lo cual no siempre es posible (aunque sí en la mayoría de aplicaciones robóticas que podemos imaginar). Lo contrario implica un uso ineficientísimo de la memoria que tiene además efectos apreciables en la precisión final. Otro aspecto insuficientemente tratado por la literatura sobre CMACs es la influencia de la distribución de los datos de entrada en el comportamiento local del aproximador una vez entrenado, y en su tiempo de entrenamiento, aspectos que ya discutimos en el capítulo correspondiente.

Como la mayoría de los trabajos publicados usan la simulación, los datos suelen ser generados sintéticamente como distribuciones uniformes, sin pensar en los efectos que esto pueda tener sobre el algoritmo de entrenamiento. Inmediatamente veremos qué mejora podría introducirse para solucionarlo, pero una conclusion importante es que hay diferencias sustanciales entre la realidad y la simulación que no deben nunca ser pasadas por alto.

A este respecto, lo ocurrido con las redes neurales, problemas de convergencia y estabilidad, es un ejemplo claro de tales diferencias, y debe servir como guía a futuros experimentos de implantación de tales métodos. Aun cuando los resultados hayan sido en este caso fallidos, es nuestra intención seguir analizando el problema por esta vía en cuanto aparezcan métodos más robustos que aborden bien los problemas grandes.

De todo lo anterior se deduce que cuanto mayor es el conocimiento sobre el problema que introducimos en el sistema, mayor es la probabilidad de éxito. Las medidas de la Teoría de la Red de Tensores se toman en puntos escogidos del espacio, y son mucho más informativas que las de los demás métodos. Esto debe ser tenido en cuenta a la hora de escoger el método para una aplicación real, y también indica deficiencias en el proceso de aprendizaje de los demás métodos, que tendrán que ser corregidas, puesto que, aparte del interés puramente científico, incluso en aplicaciones reales sería siempre preferible un método que gozara de aprendizaje, dado que podría reprogramarse con mucha más facilidad para adaptarse a cambios en el entorno.

Como conclusiones generales, la más importante de ellas, como ya hicimos notar en la introducción, es el haber probado que las coordenadas visuales son una entrada tan válida como cualquier otra a los sistemas de control, y que además presentan la ventaja añadida de ser coordenadas más fáciles de determinar y más naturales respecto a los sistemas robóticos reales. Otra conclusión importante es haber puesto de manifiesto que en ciertas ocasiones la observación de fenómenos biológicos (el reflejo vestíbulo ocular en la TNT y la estructura de la transmisión de la señal nerviosa en la CMAC) aporta ideas interesantes a la Robótica, y es por tanto algo en lo que merece la pena inspirarse. Aun cuando no se pueda reproducir el fenómeno en toda su complejidad ni al mismo nivel algorítmico con que los sistemas biológicos lo ejecutan, una parte o una idea del mismo pueden ser útiles.

Respecto a las mejoras que en un futuro pensamos introducir en el sistema, consistirían primero en retoques para poder manejar la configuración inversa en el caso de brazos SCARA, que como sabemos, llegan al mismo punto con dos configuraciones distintas, aun cuando a veces sea necesario usar una en concreto, bien por problemas de accesibilidad, bien por colisiones con otros cuerpos en el área de trabajo. Esto puede hacerse de dos maneras: duplicando la información almacenada, o dejando al sistema calcular los valores de los joints de una configuración conocidos los de la otra, bien sea mediante fórmulas, o mediante cualquiera de los métodos empleados en el problema original.

Otra mejora necesaria en un futuro será el control a un nivel más bajo, es decir,

de señales para los motores, proporcionales a los torques, puenteando al sistema de control incluído en el robot. Esto es bastante difícil en el RT100, aunque sabemos que ha sido hecho por personas con fuertes conocimientos de hardware[1], y trataríamos de intentarlo mejor en los futuros robots más precisos de que pudiéramos disponer.

Por último, expondremos brevemente las nuevas líneas de investigación a que este trabajo puede dar pie. Respecto a la parte de control, aun cuando la relación entre coordenadas visuales y motoras ha sido abordada casi por todos los métodos neurales potencialmente útiles sin éxito, existen otras posibilidades que por restricciones de tiempo no han sido usadas, y presentan, al menos en teoría, la característica de evitar quedar atrapados en mínimos locales: son los algoritmos genéticos.

Se basan en codificar las soluciones a un problema en forma (usualmente) de cadenas de bits y cruzar unas con otras para generar nuevas soluciones con una tasa de reproducción relacionada con la eficacia de cada solución para resolver el problema, medida por una función de evaluación provista por el usuario. El problema de estos algoritmos es que es necesario conocer la forma funcional de la solución, cuyos parámetros libres se codifican para encontrar sus valores óptimos. Una aproximación en la que hemos pensado partiría del teorema de Funahashi, encontrando los valores de los coeficientes desconocidos en la ecuación 8.13 o bien empleando la base de funciones radiales, para encontrar los coeficientes de los aproximadores.

Otra línea de trabajo en la que se va a incidir de modo inminente es un estudio más detallado del modelo CMAC y de la influencia de la distribución de los datos de entrada en la precisión y necesidades de memoria. A la hora de optimizar el uso de ésta una posible solución que como se vió en el capítulo correspondiente ya hemos aplicado consiste en transformar las coordenadas de las entradas de modo que éstas siguiesen una distribución uniforme. Pero ello exige conocer su distribución presente, es decir, el conjunto completo de muestras de entrenamiento antes de comenzar el aprendizaje, lo cual no siempre es posible. Una alternativa es hacer que la CMAC dedique más memoria a las zonas del espacio en las que se acumulan más entradas, o en las que la variación de la función de salida es mayor. Ello se conseguirá haciendo que la anchura de las divisiones en cada dimensión (intervalos de cuantización) no sea fija, sino que varíe durante el aprendizaje, y además que se creen más intervalos por subdivisión de los existentes cuando sea preciso: esto dotará a la CMAC de una estructura fractal. La investigación que propondremos tratará de encontrar un algoritmo de subdivisión que sea óptimo respecto al uso de la memoria y razonablemente eficiente.

La siguiente línea de trabajo tiene que ver con la limitación que fuerza a la Teoría de la Red de Tensores a trabajar con cámaras fijas. Como quiera que el centro de las cámaras es el punto con menor error, convendría que apuntase siempre al área de interés, tal como los ojos miran a las manos cuando hacemos algo preciso con ellas. Se usaría un mecanismo que permita el cambio en la orientación de las cámaras,

[1] Peter Bald, Depto. de IA, Univ. de Edimburgo

controlando éste exactamente igual que hicimos con el brazo, con la salvedad de que el invariante físico no sería aquí la posición del punto terminal, sino la orientación de las cámaras. En términos biológicos esto corresponde a la implantación del reflejo oculo-motor, por el cual los ojos siguen a objetos en movimiento. Lógicamente, una implantación en tiempo real, de excepcional interés, requeriría hardware específico.

Intimamente conectado con esta idea está el uso de cámaras de resolución foveal, es decir, mayor en el centro que en el área periférica, tal como la retina humana. Aparte de su aplicación obvia recién citada, el uso de la disposición particular de pixels de estas cámaras facilita la ejecución de la transformada de Gabor, operación que también parece haber evidencia fisiológica de que se realiza en el córtex visual, y que tiene la característica de reducir el volumen de datos al mínimo, sin perder información visual relevante. Esta línea está fuertemente condicionada en el futuro a la disponibilidad de fondos para material apropiado.

Un uso obvio de las técnicas de visión antedichas está en la detección de aspectos visuales relevantes para guiar un ensamblado, aspecto éste importantísimo al que ya se hizo referencia al final del capítulo 4. Otro tipo de algoritmo para abordar este problema está basado en el movimiento (ver [Kit92]), y usaría una microcámara en cabeza del brazo. Algoritmos de esta clase son computacionalmente muy costosos, pero pueden sacar ventaja del hecho de que un movimiento controlado del brazo nos permite fijar una o más componentes del vector velocidad.

Finalmente, quisiera apuntar algunas consideraciones acerca de lo que personalmente esta tesis me ha aportado. A lo largo del desarrollo del trabajo, y en general durante los últimos años, he intentado descubrir qué era Robótica, cuál es el estado actual de la investigación en ella, y cuales son sus limitaciones presentes. Algunas de las conclusiones más útiles que he extraído son:

- Los modelos matemáticos extremadamente precisos del comportamiento de los robots reales son teóricamente muy atractivos, pero tienen sus riesgos. El entorno de un robot real no es perfecto, de modo que incluso un programa aparentemente perfecto puede no representarlo bien.

- Existen aproximaciones alternativas que están emergiendo muy recientemente y que probablemente se extenderán con rapidez. Independientemente de su éxito presente o futuro, tienen el atractivo de estar basadas en una visión diferente de la cuestión que trata de estudiar cómo los seres vivos resuelven problemas reales, no cómo pensamos que deberían hacerlo, lo cual la hace muy interesante como ciencia experimental, que ejecuta sus experimentos no sólo sobre ordenadores, sino sobre ordenadores más dispositivos ópticos y mecánicos. Esto presenta la dificultad de enfrentarse a los errores que uno comete cuando tanto los programas como los dispositivos cuentan, y es una de las habilidades que he tratado de adquirir.

- Los problemas de control avanzan de modo prometedor. La visión, sin embargo, lo hace más lentamente, porque es, posiblemente, el problema más difícil de la

Robótica. Hasta que los aspectos de bajo nivel del sistema de visión humano (o de otros) no sean bien entendidos no podrán hacerse progresos significativos sobre robots que realmente funcionen en entornos moderadamente complejos. Esto constituye un reto para la investigación futura.

Bibliografía

[AA86] Amillo and Arriaga. *Análisis Matemático con aplicaciones a la Computación.* McGraw Hill, 1986.

[Alb81] S. J. Albus. *Brains, Behaviour and Robotics*, chapter 6 (A Neurological Model). BYTE Books, subs. of McGraw Hill, 1981.

[Atk91] C. G. Atkeson. Using locally weighted regression for robot learning. In *Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA*, pages 958–963. IEEE Computer Society Press, April 1991.

[BBB88] D. H. Ballard, T. G. Becker, and C. M. et al. Brown. The rochester robot. Technical Report 257, Computer Science Department, Univ. of Rochester, August 1988.

[BCN88] R. A. Brooks, J. H. Connel, and P.Ñing. Herbert: A second generation mobile robot. AI Memo 1016, Massachussets Institute of Technology, AI Lab, 1988.

[Bea82] M. Brady and et al., editors. *Robot Motion: Planning and Control*, chapter 3. MIT Press Series in AI. MIT Press, 1982.

[BP80] J.L. Bentley and C.H. Papadimitriou. Two papers on computational geometry. Technical Report CMU-CS-80-109, Dept. of Computer Science. Carnegie-Mellon University, April 1980.

[Bro91] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[BSA83] A.G. Barto, R.S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man ans Cybernetics*, SMC-13:834–846, 1983.

[CC90a] P. Chongstitvatana and A. Conkie. Behaviour-based assembly experiments using vision sensing. Technical Report 466, Dept. of Artificial Intelligence, 1990.

[CC90b] A. Conkie and P. Chongstitvatana. An uncalibrated stereo visual servo system. In *Proceedings of the British Machine Vision Conference, Oxford*, pages 277–280, 1990.

[CM89] W. F. Clocksin and A. W. Moore. Experiments in adaptive state-space robotics. In *Proceedings of the 7th AISB Conference*. Morgan Kaufman, 1989.

[CRE91] F. Chaumette, P. Rives, and B. Espiau. Positioning a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA*, pages 2248–2253. IEEE Computer Society Press, April 1991.

[DeG88] M. H. DeGroot. *Probabilidad y Estadistica.* Addison-Wesley Iberoamericana, 1988.

[Dom91] J. Domingo. Stereo part mating. Master's thesis, Department of Artificial Intelligence, Univ. of Edinburgh, September 1991.

[Dom93] J. Domingo. Metodos para el control de robots basados en vision: Una perspectiva comportamental. Conferencia impresa en las Notas al III Curso de Verano de Informática de la Univ. de Castilla-La Mancha, Albacete, Julio 1993.

[Fun89] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[Ges82] C. Geschke. A robot task using visual tracking. *Robotics Today*, Winter 1981-82.

[HC93] N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. Technical Report CUED/F-INFENG/TR 126, Cambridge University Engineering Department, 1993.

[HKEK91] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura. Manipulator control with image-based visual servo. In *Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA*, pages 2267–2272. IEEE Computer Society Press, April 1991.

[Ins93] Institute for Parallel and Distributed High Performance Systems, Univ. of Stuttgart. *SNNS-Stuttgart Neural Network Simulator, User's manual*, 1993.

[Ito92] Y. Ito. Approximation of continuous functions on R^d by linear combinations of shifted rotations of a sigmoid function with and without scaling. *Neural Networks*, 5:105–115, 1992.

[JB91] W. Jang and Z. Bien. Feature-based visual servoing of an eye-in-hand robot with improved tracking performance. In *Proceedings of the IEEE*

International Conference on Robotics and Automation, Sacramento, CA, pages 2254–2260. IEEE Computer Society Press, April 1991.

[JF92] T.T. Jervis and F. Fallside. Pole balancing on a real rig using a reinforcement learning controller. Technical Report CUED/F-INFENG/TR 115, Cambridge University Engineering Department, December 1992.

[Kan81] T. Kanade. A theory of origami world. *Artificial intellignece*, 13(3), 1981.

[KFS87] M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural network model for control and learning of voluntary movement. *Biological Cybernetics*, 57:169–185, 1987.

[Kit92] J. Kittler. General motion estimation and segmentation. Conferencia impresa en los Proceedings del V Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes, SERFAI, Valencia, Septiembre 1992.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[KvdS93] B. J. A. Krose and P. P. van der Smagt. An introduction to neural networks. The University of Amsterdam. Book available by ftp, January 1993.

[KvdSG93] J. A. Krose, P. P. van der Smagt, and F. C. A. Groen. *Neural Networks in Robotics, Bekey et al. eds.*, chapter A One Eyed Self Learning Robot Manipulator, pages 19–27. Kluwer Accademic Publishers, 1993.

[Lee93] S. Lee. *Neural Networks in Robotics, Bekey et al. eds.*, chapter A Neural Net Approach to Robot 3D Perception and Visuo-Motor Coordination, pages 331–347. Kluwer Accademic Publishers, 1993.

[LPJMea87] T. Lozano-Perez, J. L. Jones, E. Mazer, and et al. Handey: A robot system that recognizes, plans, and manipulates. *IEEE Journal on Robotics and Automation*, RA-3:843–849, 1987.

[LPJMO90] T. Lozano-Perez, J. L. Jones, E. Mazer, and P. O'Donnell. *Handey: A Robot Task Planner.* MIT Press Series in AI. MIT Press, 1990.

[LS89] G.F. Luger and W.A. Stubblefield. *Artificial Intelligence and the Design of Expert Systems.* The Benjamin/Cummings Publishing Company, Inc., 1989.

[Mal91] C. Malcolm. Behavioural modules in robotic assembly. Dept. of Artificial Int. Draft Teaching Paper, March 1991.

[Mir93] J. Mira. Computacion neuronal en el camino visual. Conferencia impresa en las Notas al III Curso de Verano de Informática de la Univ. de Castilla-La Mancha, Albacete, Julio 1993.

[MKB79] K. V. Mardia, I. T. Kent, and J. M. Bibby. *Multivariate Analysis.* Neural Networks: Research and Apllications. Academic Press, London, 1979.

[MRS90] T. M. Martinetz, H. J. Ritter, and K. J. Schulten. Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Transactions on Neural Networks*, 1(1):131–136, 1990.

[NS76] A.Ñewell and H. Simon. Computer science as empirical enquiry: Symbols and search. *CACM*, 19(3)(pp. 113-126), 1976.

[NW78] J.L. Nevins and D.E. Whitney. Computer-controlled assembly. *Scientific American*, February 1978.

[Oga87] K. Ogata. *Discrete-time Control Systems.* Perntice-Hall Inc., 1987.

[Oxf90] Oxford Intelligent Machines, Ltd. *UMI-RT100 User's Manual*, 1990.

[PL80] A. Pellionisz and R. Llinas. Tensorial approach to the geometry of the brain function: Cerebellar coordination via a metric tensor. *Neuroscience*, 5:1125–1136, 1980.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, 1:318–362, 1986.

[RK88] A. Rosenfeld and A. Kak. *Digital Picture Processing*, volume 2, pages 220–225. Addison Wesley Co., 1988.

[Seb84] G. A. F. Sebert. *Multivariate Observations.* John Wiley & Sons, 1984.

[Sim90] P. K. Simpson. *Artificial Neural Systems.* Neural Networks: Research and Applications. Pergamon Press, NY, 1990.

[TP92] C.K. Tham and R.W. Prager. Reinforcement learning for multi-linked manipulator control. Technical Report CUED/F-INFENG/TR 104, Cambridge University Engineering Department, June 1992.

[Tsa87] R. Y. Tsai. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Transactions on Robotics and Automation*, 3:323–344, 1987.

[Tur74] K. J. Turner. *Computer Perception of curved objects using a T.V. camera.* PhD thesis, Department of Artificial Intelligence, Univ. of Edinburgh, 1974.

[WW91] D. Westmore and W. J. Wilson. Direct dynamic control of a robot using an end-point mounted camera end kalman filter position estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA*, pages 2376–2384. IEEE Computer Society Press, April 1991.

[YS87] Y. Yeshurun and E.L. Schwartz. Cepstral filter on a columnar image architecture: A fast algorithm for binocular stereo segmentation. Technical Report 286, Courant Inst. of Mathematical Sciences. New York Univ., March 1987. Robotics Research.

Apéndices

Suele ser usual incluir en los apéndices el código del programa usado. Hemos preferido no hacerlo por dos motivos: una, su enorme extensión, y otra, la dudosa utilidad que el código de un programador suele tener para cualquier otro. El número de modificaciones que hay que hacer suele ser tan grande, además de la molestia de redigitarlo, que la inmensa mayoría de la gente prefiere programar sus propias aplicaciones, especialmente en campos tan específicos como este. No obstante, y para facilitar la implantación, hemos incluído el algortimo más intrincado.

Además de esto, se incluye también un apéndice de formulas que hemos considerado que intercaladas en el texto principal hubieran roto su continuidad.

Apéndice A

Principales relaciones tensoriales usadas

Veremos seguidamente una demostración de las principales relaciones tensoriales usadas en la capítulo 6. Nos restringiremos por claridad al caso bidimensional (que es el que luego servirá para probar el carácter contravariante de las coordenads motoras) pero las relaciones serían análogas para cualquier número de dimensiones, considerando las matrices de transformación adecuadas.

Volvamos primero a la figura A.1 de donde extraeremos las relaciones. Sea $\vec{V}$ un vector cualquiera, sean las v_i sus proyecciones perpendiculares a cada eje (componentes covariantes) y v^i las proyecciones sobre cada eje perpendicularmente al otro (componentes contravariantes).

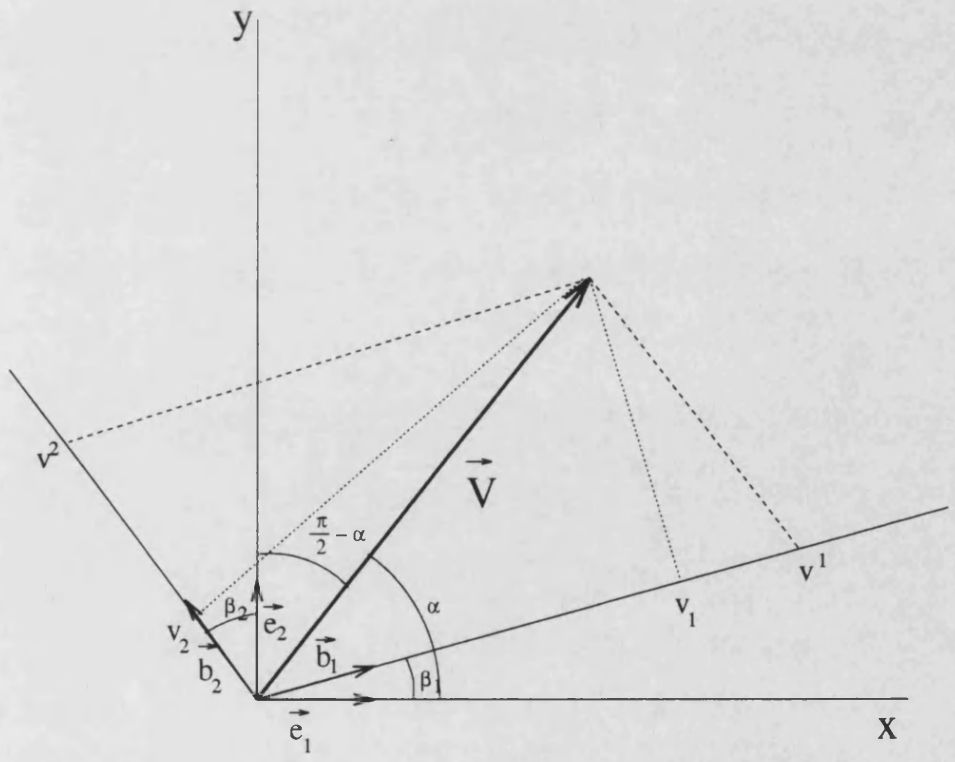


Figura A.1: Coordenadas covariantes y contravariantes del vector V

De esta figura se deduce que

$$v_1 = \quad |v|\cos(\alpha - \beta_1) \tag{A.1}$$
$$v_2 = |v|\cos(\tfrac{\pi}{2} - \alpha + \beta_2) = |v|\sin(\alpha - \beta_2) \tag{A.2}$$

y también

$$v^1 = |v|\frac{\cos(\alpha - \beta_2)}{\cos(\beta_1 - \beta_2)} \tag{A.3}$$
$$v^2 = |v|\frac{\sin(\alpha - \beta_1)}{\cos(\beta_1 - \beta_2)} \tag{A.4}$$

Desarrollando, esto se puede escribir como

$$v_1 = |v|(\cos\alpha\cos\beta_1 + \sin\alpha\sin\beta_1) = y\cos\beta_1 + x\sin\beta_1 \tag{A.5}$$
$$v_2 = |v|(\sin\alpha\cos\beta_2 - \cos\alpha\sin\beta_2) = x\cos\beta_2 - y\sin\beta_2 \tag{A.6}$$

e igualmente

$$v_1 = \frac{|v|}{\cos(\beta_1 - \beta_2)}(\cos\alpha\cos\beta_2 + \sin\alpha\sin\beta_2) = \tfrac{\cos\beta_2}{\cos(\beta_1-\beta_2)}x + \tfrac{\sin\beta_2}{\cos(\beta_1-\beta_2)}y \tag{A.7}$$
$$v_2 = \frac{|v|}{\cos(\beta_1 - \beta_2)}(\sin\alpha\cos\beta_1 - \cos\alpha\sin\beta_1) = \tfrac{\cos\beta_1}{\cos(\beta_1-\beta_2)}y - \tfrac{\sin\beta_1}{\cos(\beta_1-\beta_2)}x \tag{A.8}$$

donde x e y son las coordenadas cartesianas, que denotaremos como vector por X

Esto se puede escribir más claramente en forma matricial como

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos\beta_1 & \sin\beta_1 \\ -\sin\beta_2 & \cos\beta_2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \implies v_i = MX \tag{A.9}$$

$$\begin{pmatrix} v^1 \\ v^2 \end{pmatrix} = \frac{1}{\cos(\beta_1 - \beta_2)}\begin{pmatrix} \cos\beta_2 & \sin\beta_2 \\ -\sin\beta_1 & \cos\beta_1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \implies v^i = (M^{-1})^T X \tag{A.10}$$

de donde, invirtiendo las matrices,

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{\cos(\beta_1 - \beta_2)}\begin{pmatrix} \cos\beta_2 & -\sin\beta_1 \\ \sin\beta_2 & \cos\beta_1 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \implies X = M^{-1}v_i \tag{A.11}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\beta_1 & -\sin\beta_2 \\ \sin\beta_1 & \cos\beta_2 \end{pmatrix}\begin{pmatrix} v^1 \\ v^2 \end{pmatrix} \implies X = M^T v^i \tag{A.12}$$

Estas relaciones son las que se usan en la ecuación 6.11 para convertir diferentes tipos de coordenadas expresadas en sistemas diferentes. Pero además se puede extraer de ellas la relación entre uno y otro tipo de coordenadas en el mismo sistema, que es:

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos\beta_1 & \sin\beta_1 \\ -\sin\beta_2 & \cos\beta_2 \end{pmatrix} \begin{pmatrix} \cos\beta_1 & -\sin\beta_2 \\ \sin\beta_1 & \cos\beta_2 \end{pmatrix} \begin{pmatrix} v^1 \\ v^2 \end{pmatrix}$$

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 & \sin\beta_1\cos\beta_2 - \cos\beta_1\sin\beta_2 \\ \sin\beta_1\cos\beta_2 - \cos\beta_1\sin\beta_2 & 1 \end{pmatrix} \begin{pmatrix} v^1 \\ v^2 \end{pmatrix} \tag{A.13}$$

o, lo que es lo mismo,

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 & \sin(\beta_1 - \beta_2) \\ \sin(\beta_1 - \beta_2) & 1 \end{pmatrix} \begin{pmatrix} v^1 \\ v^2 \end{pmatrix} \tag{A.14}$$

Nótese que si los ejes son perpendiculares (lo cual equivale a $\beta_1 = \beta_2$) esta matriz se transforma en la identidad, y las coordenadas covariantes y contravariantes coinciden.
Esto también se puede escribir como

$$v_i = MX = (MM^T)v^j = g_{ij}v_j \tag{A.15}$$

$$v^i = (M^{-1})^T X = (M^{-1})^T M^{-1} v_j = g^{ij}v_j \tag{A.16}$$

donde g_{ij} es el tensor métrico covariante y g^{ij} el contravariante. Como se puede ver, uno es inverso del otro, y están relacionados con al distancia, dado que el módulo al cuadrado del vector $\vec{V}$ se expresa como

$$|v|^2 = X^T X = (M^{-1}v_i)^T M^{-1} v_j = (v_i)^T (M^{-1})^T M^{-1} v_j = (v_i)^T g^{ij} v_j \tag{A.17}$$

lo cual nos daría la distancia euclídea real conociendo las coordenadas covariantes. Hay fórmula análoga para las contravariantes.

Una vez establecidas todas las fórmulas que se han usado, las emplearemos ahora para probar que las coordenadas que hemos usado para el brazo robot, que son ángulos directamente relacionados con los joints, son contravariantes. Considerando la construcción de nuestro brazo, un SCARA, la coordenada zed (cuentas del motor de deplazamiento vertical) es proporcional a la z cartesiana, y consiguientemente, tendrá el mismo carácter que ésta, es decir, simultáneamente covariante y contravariante, de modo que no representa un problema, y podemos restringir el análisis a las otras dos, que serán los ángulos s (shoulder) y e (elbow) en la figura que sigue:

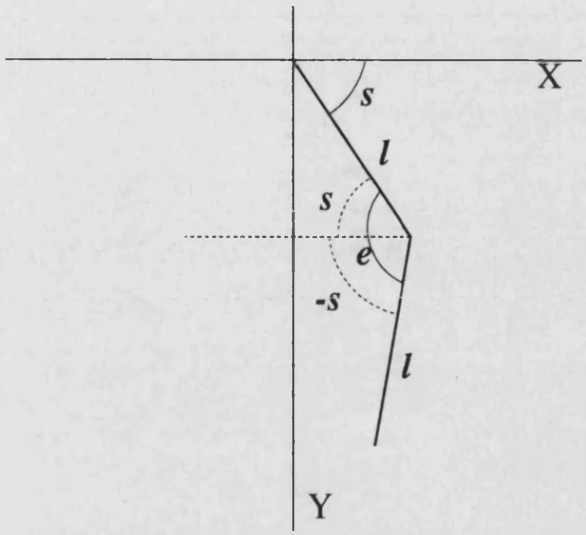


Figura A.2: Coordenadas angulares para el brazo

Las transformaciones de coordenadas son

$$x = l\cos s - l\cos(e-s) \tag{A.18}$$
$$y = l\sin s + l\sin(e-s) \tag{A.19}$$

y diferenciando, obtendremos la matriz que linealiza estas transformaciones en cada punto, para pequeños desplazamientos desde él:

$$dx = (-l\sin s - l\sin(e-s))ds + l\sin(e-s)de \tag{A.20}$$
$$dy = (l\cos s - l\cos(e-s))ds + l\cos(e-s)de \tag{A.21}$$

o, en forma de matriz, y usando las fórmulas trigonométricas pertinentes,

$$\begin{pmatrix} dx \\ dy \end{pmatrix} = l \begin{pmatrix} -2\sin\frac{e}{2}\cos(\frac{e}{2}-s) & \sin(e-s) \\ 2\sin\frac{e}{2}\sin(\frac{e}{2}-s) & \cos(e-s) \end{pmatrix} \begin{pmatrix} ds \\ de \end{pmatrix} = \mathbf{T} \begin{pmatrix} ds \\ de \end{pmatrix} \tag{A.22}$$

Esta matriz que hemos llamado T lleva las coordenadas cartesianas a las del brazo, por tanto, la ecuación anterior deberá identificarse bien con la A.11, si nuestras coordenadas con covariantes, bien con la A.12, si son contravariantes. Hay que decir que la matriz T deberá ser normalizada, para que el vector básico asociado a cada coordenada sea unitario, y así poder identificar con las ecuaciones referidas, que se dedujeron suponiendo esto implícitamente. Las normas de los vectores básicos en nuestro sistema de coordenadas son

$$|\vec{b}_s| = l\sqrt{(-\sin s - \sin(e-s))^2 + (\cos s - \cos(e-s))^2} =$$

$$l\sqrt{2-2(\sin s \sin(e-s)+\cos s\cos(e-s))} = \sqrt{2}l\sqrt{1-\cos(2s-e)} =$$

$$\sqrt{2}l\sqrt{2\sin\left(s-\frac{e}{2}\right)^2} = 2l\sin\left(s-\frac{e}{2}\right) \tag{A.23}$$

y

$$|\vec{b}_e| = l\sqrt{\sin^2(e-s)+\cos^2(e-s)} = l \tag{A.24}$$

Así pues, si la coordenadas son contravariantes obtendríamos el siguiente sistema de ecuaciones:

$$\frac{-2l\sin\frac{e}{2}\cos\left(\frac{e}{2}-s\right)}{2l\sin\frac{e}{2}} = \cos\beta_1$$

$$\frac{2l\sin\frac{e}{2}\sin\left(\frac{e}{2}-s\right)}{2l\sin\frac{e}{2}} = \sin\beta_1$$

$$\frac{l\sin(e-s)}{l} = -\sin\beta_2$$

$$\frac{l\cos(e-s)}{l} = \cos\beta_2 \tag{A.25}$$

el cual, simplificando las fracciones, resulta trivialmente soluble, dando $\beta_1 = \frac{e}{2} - s$ y $\beta_2 = s - e$.

En cambio, si escribimos el sistema de ecuaciones que resulta de identificar con A.11 tenemos

$$\frac{-2l\sin\frac{e}{2}\cos\left(\frac{e}{2}-s\right)}{2l\sin\frac{e}{2}} = \frac{\cos\beta_2}{\cos(\beta_1-\beta_2)}$$

$$\frac{2l\sin\frac{e}{2}\sin\left(\frac{e}{2}-s\right)}{2l\sin\frac{e}{2}} = \frac{\sin\beta_2}{\cos(\beta_1-\beta_2)}$$

$$\frac{l\sin(e-s)}{l} = \frac{-\sin\beta_1}{\cos(\beta_1-\beta_2)}$$

$$\frac{l\cos(e-s)}{l} = \frac{\cos\beta_1}{\cos(\beta_1-\beta_2)} \tag{A.26}$$

Si dividimos la tercera ecuación entre la cuarta, y la segunda entre la primera, obtenemos inmediatamente soluciones para β_1 y β_2 que son $\beta_1 = s - e$ y $\beta_2 = \frac{e}{2} - s$, las cuales, llevadas p. ej. a la tercera ecuación, nos dan $cos\frac{3e}{2} = 1$, con lo que e sería siempre 0, lo cual es obviamente falso. Así pues, este sistema es incompatible, y por tanto las coordenadas (s, e) deben ser contravariantes, c.q.d.

Respecto a las coordenadas de las cámaras, hemos decidido considerarlas covariantes, por semejanza con el sistema biológico. En realidad, son simultáneamente covariantes y contravariantes, dado que sus vectores básicos son perpendiculares en todos los puntos. Esto es porque el sistema óptico de la cámara tiene simetría cilíndrica, lo cual quiere decir que sólo altera la distancia de los puntos al eje óptico, pero no su ángulo, lo que es tanto como decir que varía el módulo del vector básico radial, pero no su orientación. La transformación de cartesianas a polares, y también su inversa, conservan el carácter de las coordenadas, y una vez en polares es fácil ver que la transformación ejecutada por el sistema óptico sigue conservando la perpendicularidad de los vectores básicos. Véase la figura A.3

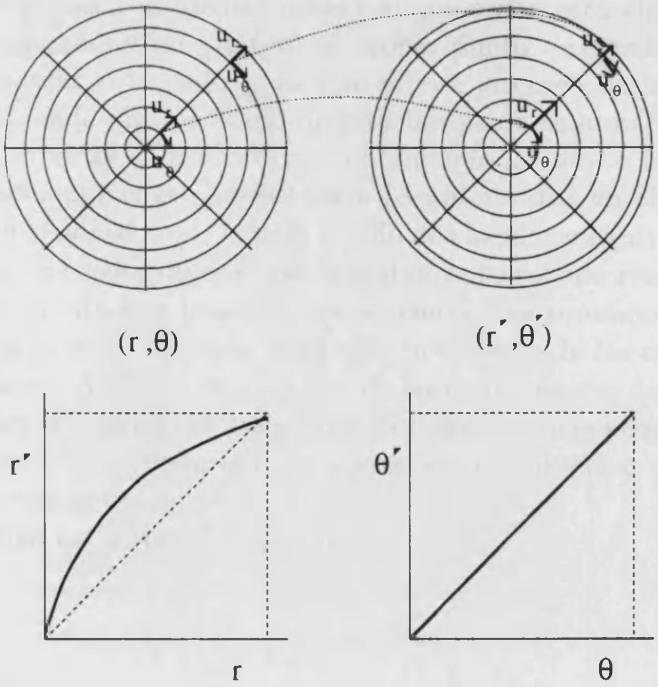


Figura A.3: Transformación ejecutada por una lente con simetría cilíndrica

Apéndice B

Algoritmo de mínima distancia en especios n-dimensionales

Como se recordará, este algoritmo se emplea en la fase de uso de las medidas tomadas por la Teoría de la Red de Tensores, cuando es necesario calcular el vecino más próximo en el espacio de cámaras a nuestra entrada de entre todos los puntos medidos, para recuperar y usar el tensor asociado a él. Está basado en proyectar los puntos n-dimensionales de nuestro conjunto original sobre cada dimensión, y ordenarlos en listas, $L_1, \ldots, L_n$, una por cada coordenada. Recuérdese que el conjunto de puntos en nuestro caso es siempre el mismo, y que por tanto esta ordenación puede hacerse antes de comenzar y almacenarse. Para acelerar aún más la búsqueda y ahorrar espacio cada elemento de cada una de estas listas contendrá no el punto en cuestión, sino referencias a las demás listas (en que órden está almacenado el punto en las otras dimensiones) y un puntero al propio punto. Además el algoritmo usa un campo para marcar si ha sido o no visitado en el curso de la búsqueda de los vecinos a un punto dado. El resultado de esta búsqueda se almacena en la variable vecinos, que será un array de estructuras conteniendo los punto más cercanos y sus distancias, ordenados por ellas. La distancia (denotada dist en el algortimo) puede ser la euclídea, o su cuadrado, que es más rápido de calcular y equivalente a efectos de ordenación. Nótese, de todos modos, que la distancia se calcula realmente muy pocas veces, porque cada punto que pueda aparecer como más próximo según una de las dimensiones se marca como visitado, para que en el bucle de las demás dimensiones no vuelva a calcularse. Además, cuando una de las componentes del vector diferencia (guardadas en el array control) es mayor que la distancia más pequeña, ese punto es eliminado como candidato (dado que su separación del objetivo en una sóla de las dimensiones ya es mayor que el total).
Véase el seudocódigo en la siguiente página.

Sea P el punto cuyos vecinos más próximos queremos encontrar, y sea P(d) la componente d-ésima de tal punto. Entonces:

Para cada dimensión d
  Encontrar en la lista L_d el punto I_d tal que $I_d(d) < P(d)$
   y el valor $P(d) - I_d(d)$ sea el mínimo posible
  Encontrar en la lista L_d el punto S_d tal que $S_d(d) > P(d)$
   y el valor $S_d(d) - P(d)$ sea el mínimo posible
   (Estos dos puntos se encuentran por búsqueda binaria)
Hacer distancia_mínima ← la mayor distancia posible
Desde i=0 hasta el número de vecinos que queremos encontrar hacer
  Hacer
   Para cada dimensión d hacer
    control[d] ← la mayor distancia posible
    Si I_d no ha sido visitado todavía
     calcular la distancia $di_d = dist(I_d, P)$
    Si di_d <distancia_mínima entonces
     distancia_mínima ← di_d
     punto_más_cercano ← I_d
    Marcar I_d como visitado en todas las listas L_d usando
     las referencias cruzadas almacenadas en ellas.
    Si control[d]< $|I_d(d) - P(d)|$ entonces
     hacer control[d] ← $|I_d(d) - P(d)|$
    Hacer I_d ← su predecesor en la lista ordenada L_d
    Si S_d no ha sido visitado todavía
     calcular la distancia $ds_d = dist(I_d, P)$
    Si ds_d <distancia_mínima entonces
     distancia_mínima ← ds_d
     punto_más_cercano ← S_d
    Marcar S_d como visitado en todas las listas L_d usando
     las referencias cruzadas almacenadas en ellas.
    Si control[d]< $|S_d(d) - P(d)|$ entonces
     hacer control[d] ← $|S_d(d) - P(d)|$
    Hacer S_d ← su sucesor en la lista ordenada L_d
   Fin (para cada dimensión)
  Mientras control[d] > mínima_distancia para todas las dimensiones d
  Hacer vecinos[i].P ← punto_más_cercano
  Hacer vecinos[i].dist ← distancia_mínima
Fin (Desde i)

UNIVERSITAT DE VALÈNCIA FACULTAD DE CIENCIAS FÍSICAS

Reunit el Tribunal que subscriu, en el dia de la data, acordà d'atorgar, per unanimitat, a aquesta Tesis Doctoral d'En/ Na/ N' Juan de Mata Domingo Esteve

la qualificació d Apto cum laude

València a 29 d Octubre de 1993

El Secretari,

El President,

UNIVERSITAT DE VALÈNCIA
FACULTAT DE FISICA
DEPARTAMENT D'INFORMATICA
I ELECTRONICA
C/. Doctor Moliner, 50
46100 - BURJASSOT (València)

D. MARCELINO VICENS LORENTE; PROFESOR TITULAR DE LA FACULTAD DE FISICA DE LA UNIVERSITAT DE VALENCIA; Y D: JOAN PELECHANO FABREGAT, TITULAR INTERINO DE ESCUELA UNIVERSITARIA DE LA FACULTAD DE FISICA DE LA UNIVERSIDAD DE VALENCIA

HACEMOS CONSTAR: Que bajo nuestra dirección, D. JUAN DE MATA DOMINGO Esteve ha realizado el trabajo "IMPLANTACION Y COMPARACION DE DIVERSOS METODOS PARA CONTROL DE ROBOTS BASADOS EN VISION" para optar al grado de Doctor en Ciencias Fisicas.

Y para que conste a los efectos legales firmamos la presente en Valencia, Septiembre de 1993.

Fdo.: Marcelino Vicens

Fdo.:Joan Pelechano