Visión artificial

Eloi Maduell i García

PID_00184756

UOC
Universitat Oberta de Catalunya

Índice

Introducción

La visión artificial o visión por computador es la ciencia y la tecnología que **permite a las "máquinas" ver**, extraer información de las imágenes digitales, resolver alguna tarea o entender la escena que están visionando.

Actualmente, las aplicaciones de la visión artificial están muy extendidas y van desde el campo de la industria (contar botellas, comprobar defectos en una cadena de montaje, interpretar un TAC médico...) y el campo de la medicina (recuento y búsqueda de células), hasta los sistemas más complejos, que permiten a los robots orientarse en un entorno desconocido, pasando por el reconocimiento de patrones de la realidad aumentada, entre otras muchas aplicaciones.

Hoy en día se utilizan cada vez más las técnicas de visión artificial en el campo del diseño interactivo mediante la interacción con superficies multitáctiles (*multitouch*), la interacción con tangibles (objetos) y el reconocimiento de gestos corporales.

Todos estos ejemplos incorporan técnicas de visión artificial.

De alguna manera, en el contexto en el que estamos, este **conjunto de técnicas** nos permite **diseñar interacciones**, de modo que el usuario utiliza su movimiento o su manipulación de objetos para interactuar con la aplicación.

Objetivos

1. Analizar las diferentes herramientas, procesos y proyectos que tienen que ver con la disciplina de la visión por computadora, aplicada al contexto de la interacción.

2. Entender la metodología de un proyecto de visión por computadora para que, mediante la aplicación de estos conceptos, podáis diseñar vuestras propias interacciones, basadas en visión artificial.

1. Conceptos teóricos

1.1. Luz visible, luz invisible

El ojo humano ve una parte del espectro de toda la luz que ilumina el universo. El rango de luz que podemos ver lo denominaremos *luz visible*.

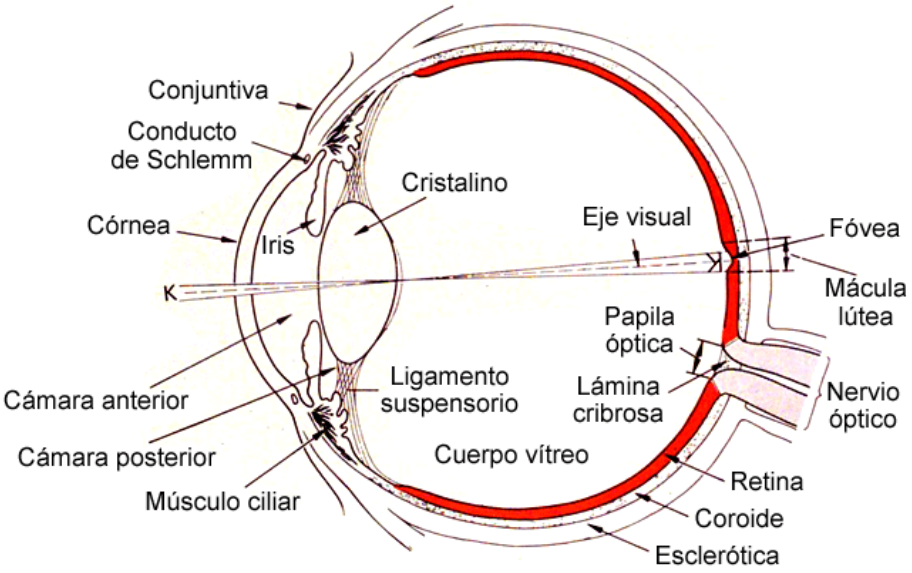
Esto quiere decir que hay frecuencias de luz que no podemos ver pero que existen, como por ejemplo, los infrarrojos y los ultravioletas.

- Los **infrarrojos** son los "colores" no visibles al ojo humano que están por debajo del rojo desde el punto de vista frecuencial.
- Los **ultravioletas** son los "colores" no visibles al ojo humano que están por encima del violeta.

Ved también

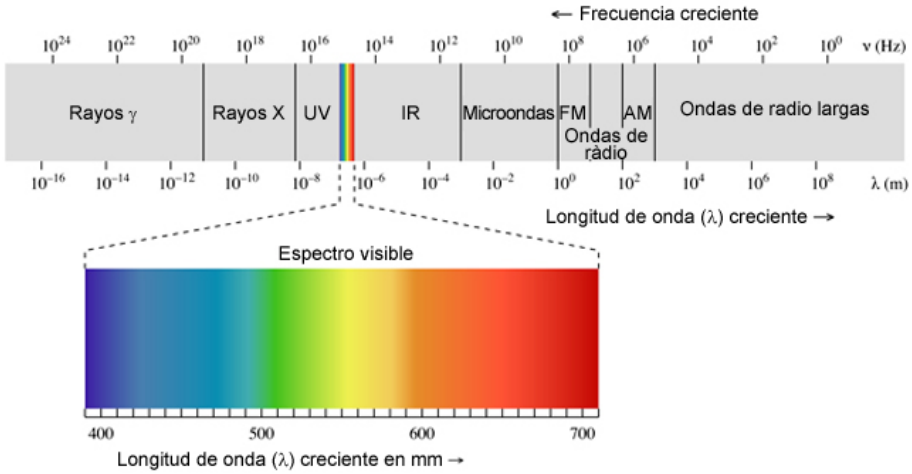
Las imágenes impresas en este documento se pueden consultar en color en el aula virtual de la asignatura en HMTL o PDF.

Esquema de la morfología de un ojo humano



Fuente: Wikipedia

Espectrograma de la luz visible y no visible



Fuente: Wikipedia (http://en.wikipedia.org/wiki/Electromagnetic_radiation).

Color	Longitud de onda
Ultravioleta	< 380 nm
Violeta	380-450 nm
Azul	450-495 nm
Verde	495-570 nm
Amarillo	570-590 nm
Naranja	590-620 nm
Rojo	620-750 nm
Infrarrojo	> 750 nm

Abreviatura: **nm** = nanómetro (unidad de medida del sistema métrico, igual a una billonésima parte de un metro.

El hecho es que la mayoría de sensores de cámaras digitales son sensibles a la luz visible, pero también son sensibles (en diferente medida) a la luz infrarroja y/o a la ultravioleta. Ahora bien, la luz infrarroja no nos es útil para construir una imagen digital, puesto que nos da información de una frecuencia que no podemos ver y que, por lo tanto, no tiene una representación posible en un color.

Por esta razón, la mayoría de cámaras digitales enfocadas a hacer fotografías o fotogramas llevan un filtro anti-IR, o sea antiinfrarojos, para cortar todas las frecuencias por debajo del espectro visible que nos resultarían un ruido innecesario, y solo dejan pasar el rango de luz visible que queremos plasmar con la cámara.

1.2. La imagen en movimiento

Los procesos de visión artificial son posibles gracias a tecnologías basadas en la captura de la imagen (cámaras de vídeo, cámaras web), sumadas a la capacidad de procesamiento de los ordenadores actuales.

En realidad, la tecnología de captación de la imagen se empieza a gestar en el siglo XIX, con la invención de los **daguerrotipos** (o incluso antes, con el descubrimiento del principio de la **cámara oscura**) y la posterior invención de la **fotografía**. Descubrimientos posteriores, como la cronofotografía y otros precursores del cine, acaban desembocando en la invención de las tecnologías de captación de la imagen en movimiento:

> Toda una serie de fotografías disparadas a una gran frecuencia que, reproducidas después a esa misma velocidad, provocan en los espectadores la ilusión del movimiento.

Actualmente, la mayoría de tecnologías de captación de imagen en movimiento (videocámaras) funcionan mediante el uso de sensores electrónicos (CCD y CMOS). Durante la primera década del siglo XXI, se ha estandarizado el uso de la fotografía y el vídeo digitales, lo que permite la grabación de imágenes en alta resolución a un relativo bajo coste y con un elevado número de imágenes por segundo (FPS).

1.3. La imagen digital

1.3.1. Matrices de píxeles

En el mundo digital, las imágenes se representan como una matriz bidimensional de píxeles en la que cada píxel puede adquirir **valores de color codificados** con tres parámetros (**R**: *red*, **G**: *green*, **B**: *blue*).

A pesar de que ello se ha heredado del mundo de la imagen analógica, normalmente trabajaremos con imágenes de proporción 4 × 3 (cuatro unidades de anchura por tres unidades de altura); es el caso de resoluciones estándar como 640 × 480 px, 800 × 600 px y 1.024 × 768 px.

16 × 9

En algunos casos, podemos encontrar imágenes en el formato 16 × 9, puesto que hoy en día se han popularizado las videocámaras de alta definición en formato panorámico.

Frecuentemente, utilizaremos imágenes de baja resolución (entre 160 × 120 y 640 × 480 píxeles) como fuente de análisis de los procesos de visión artificial, ya que en la mayoría de casos no necesitaremos excesivo detalle en las imágenes para efectuar un análisis orientado a la visión por computador.

1.3.2. Bytes, bits y colores

Un píxel normalmente se expresa mediante tres números enteros (R, G, B), que representan los componentes rojo, verde y azul de todo color. Estos valores de R, G y B se suelen expresar en un rango de 8 bits, o sea, de valores entre 0 y 255.

Ejemplo

Por ejemplo, un píxel que tenga unos valores RGB de (255, 0, 0) nos indica un color rojo puro. Un píxel que tenga unos valores RGB (255, 0, 255) nos indica un color producido por una mezcla del rojo puro y el azul puro; en este caso, por lo tanto, obtendremos un color lila intenso.

1.3.3. Frecuencia de imagen (*frame rate*)

La frecuencia de imagen (*frame rate*) hace referencia al número de imágenes por segundo. Es la medida de la frecuencia a la que un reproductor de imágenes muestra diferentes fotogramas (*frames*).

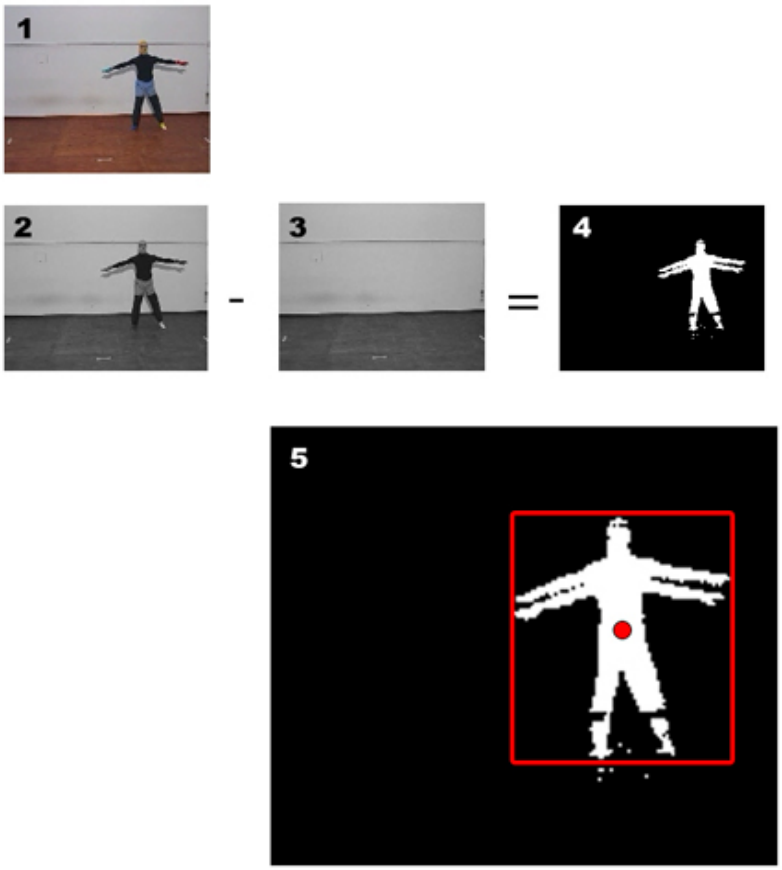
En informática estos fotogramas están constituidos por un número determinado de píxeles que se distribuyen a lo largo de una red de texturas. La frecuencia de los fotogramas es **proporcional** al número de píxeles que se han de generar y que inciden en el rendimiento del ordenador que los reproduce.

La frecuencia de actualización de las imágenes oscila, en el entorno digital, entre los 15 y los 60 FPS (*frames* por segundo). El rango entre 25 y 30 FPS es el más común.

Según nuestras necesidades, deberemos elegir tecnologías de captación de la imagen con una frecuencia de imagen específica. Si queremos analizar y extraer datos de objetos o usuarios que se mueven a grandes velocidades (coches, pájaros, corredores de fútbol...), seguramente necesitaremos un sistema de captación de vídeo con más resolución temporal (una frecuencia de imagen más alta: más fotogramas por segundo).

1.4. Un ejemplo básico

Aunque entraremos en profundidad en la materia en los apartados siguientes, es conveniente hacer un breve análisis de un típico proceso de visión artificial para que, de ahora en adelante, podáis analizar con una cierta perspectiva los conceptos que iremos introduciendo. Fijaos en este ejemplo esquema:



1) Imagen original
2) Imagen procesada en escala de grises
3) Imagen neta del fondo
4) Imagen resultante de la sustracción de fondo y la binarización
5) Extracción de datos (área del *blob* y centro de coordenadas)
Fuente: www.eyesweb.org

En este ejemplo, podéis observar algunos de los procesos típicos de la visión artificial.

A menudo necesitamos conseguir una imagen muy clara y simple del objeto o el usuario del que queremos efectuar un seguimiento. Para no cargar el procesador de los ordenadores, es habitual aplicar los algoritmos de análisis sobre imágenes de color binario (un bit: blanco y negro), de las que se puede extraer el centro de masas del área de píxeles blancos, los contornos...

Para conseguir esta imagen simple y clara, efectuaremos diferentes procesos, como podéis observar en este ejemplo: la sustracción de fondo o la binarización, entre otros.

Una vez conseguida esta imagen binaria final, los algoritmos de visión nos permiten extraer una serie de datos, como veremos a continuación. En este caso particular, se extraen las coordenadas del centro de la masa blanca de píxeles y el área que la rodea (punto y líneas rojas).

2. El espacio y las herramientas

2.1. La cámara

Para empezar a trabajar con la visión artificial, necesitamos un **dispositivo sensible a la luz visible** que nos permita almacenar las imágenes en formato digital. En otras palabras, necesitamos una cámara web, una cámara de vídeo o una capturadora analógica.

Ejemplo de cámara de visión artificial del sector industrial fabricada por Point-Grey
Fuente: http://doc.instantreality.org/

Las cámaras web y la mayoría de cámaras de vídeo se pueden conectar directamente a los ordenadores por medio de los puertos USB, FireWire o Thunderbolt, y podemos capturar sus fotogramas en **tiempo real**. En todo caso, también podemos utilizar una capturadora analógica, que a partir de una señal de vídeo analógico, nos permitirá capturar imágenes y procesarlas.

Ejemplo de cámara web fabricada por Logitech
Fuente: http://doc.instantreality.org/

Una vez que ya tengamos el dispositivo de captura en funcionamiento, hemos de considerar toda una serie de aspectos referentes al entorno en el que se hará la interacción, puesto que la **variación en la iluminación**, la complejidad de la imagen u otros factores pueden dificultar mucho el proceso de visión artificial.

Pensad que el proceso de "enseñar" a un ordenador a tomar decisiones por medio de la "visión" implica muchas dificultades y, por lo tanto, es muy importante trabajar en entornos en los que la luz y el escenario que se quieran analizar sean cuanto más sencillos y estables mejor.

2.2. Iluminación de la escena

La adquisición de imágenes por parte de una cámara varía mucho según la iluminación de la escena. Un cambio lo bastante fuerte en el ambiente lumínico puede hacer que todo el sistema de visión artificial funcione de un modo muy diferente.

Por tanto, siempre que sea posible trabajaremos en entornos en los que el ambiente lumínico sea **cuanto más estable** mejor.

En casos en los que no se pueda conseguir un ambiente estable (como por ejemplo una aplicación al aire libre), deberemos ir adaptando de alguna manera el sistema de adquisición de imágenes a las condiciones cambiantes mediante **algoritmos adaptativos** que permitan analizar periódicamente los cambios de iluminación del entorno y adaptar el sistema de visión a las nuevas condiciones.

Algunas cámaras, como la que lleva el mando de la Wii o muchas del sector industrial, son especialmente sensibles a los infrarrojos, puesto que nos permiten trabajar en un entorno sin luz visible, en la oscuridad, siempre que tengamos alguna fuente de luz infrarroja.

Ejemplo

Por ejemplo, muchos sistemas de videovigilancia incorporan, además de videocámaras sensibles a la luz infrarroja, un anillo de LED que emiten luz infrarroja, invisible al ojo pero visible para la cámara, lo que permite la visibilidad en entornos de oscuridad aparente.

Esta particularidad se aprovecha en muchos casos en el diseño de interfaces interactivas.

Por ejemplo, pongamos el caso de un teatro en el que un actor, a oscuras, describa una trayectoria por el escenario y queramos proyectar en el suelo su recorrido. En principio, con una cámara normal, como estamos a oscuras, no podríamos captar nada. Para poder trabajar en condiciones de oscuridad, lo que haríamos sería iluminar la escena con luz infrarroja, que no es visible pero que sí que es visible para la cámara sensible a los infrarrojos (por ejemplo, una cámara web a la que hubiéramos sacado el filtro anti-IR anteriormente mencionado). Así pues, una cámara de estas características, en una situación como esta, obtiene una imagen muy neta del cuerpo del actor (en blanco) sobre el fondo (negro) del escenario, una imagen perfecta para analizar con visión artificial, puesto que el cuerpo no interfiere con el fondo.

La luz infrarroja se puede generar de varias maneras: hay focos de LED infrarrojos que nos permiten "bombardear" un espacio con luz infrarroja. Otro sistema menos costoso es utilizar una luz de teatro incandescente, a la que pondremos tres filtros: uno rojo, uno verde y uno azul (colores puros). El efecto de poner un filtro rojo hace que la luz salga tintada de rojo, es decir, que eliminemos todos los colores excepto el rojo de la luz filtrada. Si sobre este filtro ponemos un filtro azul, estaremos eliminando todos los colores (incluyendo el rojo que nos había quedado) excepto el azul (que ya no estaba en la luz filtrada porque lo había eliminado el filtro rojo). Y sobre estos dos filtros añadimos el filtro verde, con resultados parecidos. Por lo tanto, hemos eliminado toda la luz visible con los tres filtros, pero no hemos eliminado las frecuencias infrarrojas con los filtros. Lo que hemos hecho ha sido transformar una luz incandescente en una luz infrarroja.

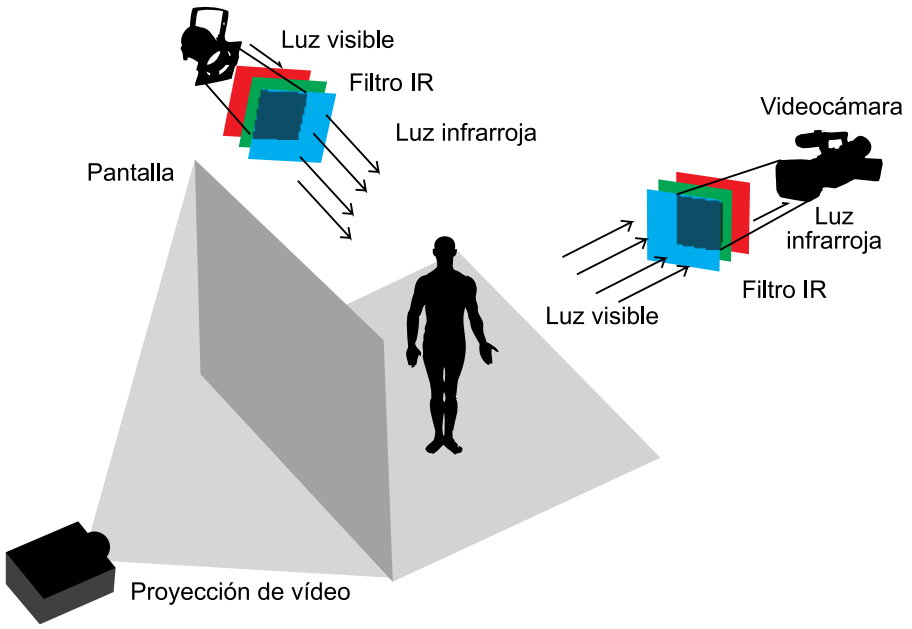
2.3. Proyecciones de vídeo y visibilidad

En un entorno interactivo nos podemos encontrar a menudo con la necesidad de tener en un mismo espacio un sistema de visión artificial y una proyección de vídeo. Como veremos algo más adelante, la complejidad de los algoritmos de visión artificial podría hacer que un sistema tenga que discernir entre una proyección de vídeo interactiva y el usuario que la manipula, algo muy dificultoso si las dos "imágenes" están integradas en un mismo espacio. Como en el apartado anterior, gracias a comprender la respuesta de la luz y sus propiedades, podemos solucionar estas dificultades.

Si no bloqueáramos la entrada de luz visible en la cámara, encontraríamos problemas en el contexto de una interacción con videoproyecciones, puesto que el sistema CV reconocería los elementos proyectados (círculos blancos) como parte integrante de los *blobs* analizados.

La luz proyectada por un proyector, evidentemente, es visible al ojo humano y/o para las cámaras digitales normales. En el caso que planteamos, queremos obtener una imagen del usuario que interactúe con la pantalla de videoproyecciones, pero no de los contenidos de la videoproyección. Para lograrlo, podemos hacer uso de la técnica descrita en el apartado anterior, referente a la luz infrarroja, usando una cámara que "vea" la luz del espectro infrarrojo.

Si a esta cámara le añadimos un filtro que bloquee la luz visible, lo que conseguiremos es que la cámara no sea sensible a la luz visible y solo capte la luz infrarroja. Por lo tanto, esta cámara podrá ver al usuario manipulando el sistema, puesto que su cuerpo hará rebotar la luz infrarroja, pero no podrá ver la proyección (que se encuentra solo dentro del rango de luz visible).



Esquema de la configuración escénica comentada, luz visible filtrada a IR, cámara de vídeo filtrada a IR y retroproyector de vídeo sobre la pantalla de fondo

Reflexión

Como corolario de este ejemplo, podemos concluir que un proyector de vídeo no genera luz infrarroja, puesto que los proyectores llevan un filtro de luz anti-IR, que no permite que "proyecten" luz en el espectro infrarrojo.

2.4. OpenCV

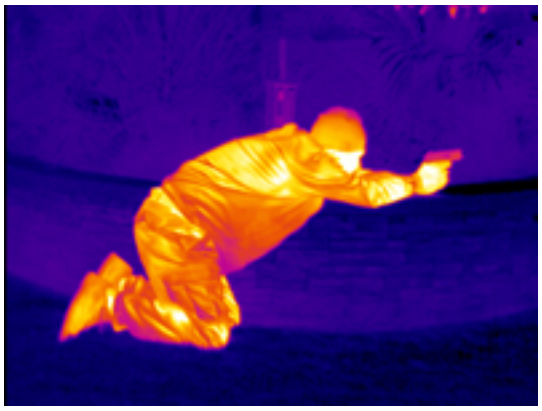
OpenCV u Open Computer Vision es un conjunto de bibliotecas de programación que nos permiten llevar a cabo todo el proceso que hemos visto anteriormente dentro de un ordenador: desde la adquisición de la imagen hasta la extracción de la información.

La gran mayoría de aplicaciones del software libre relacionadas con la visión artificial están basadas en OpenCV. Estas bibliotecas las desarrolló Intel como extensión de las bibliotecas IPL, desde 1999 como proyecto de código abierto, y en el año 2007 se llegó a publicar la versión 1.0. Estas bibliotecas son la base de algoritmos y funciones que finalmente podremos aplicar en nuestras aplicaciones.

2.5. Entornos de programación

Para desarrollar una aplicación que trabaje con visión artificial, hemos de trabajar en un entorno de programación que nos permita integrar las bibliotecas de OpenCV y combinarlas con el resto del sistema interactivo. Aquí dentro podríamos incluir entornos como Processing, openFrameworks, MAX/MSP, Pure Data... Todos estos sistemas de programación nos permiten ejecutar algoritmos y funciones de visión artificial y, a la vez, generar un sistema interactivo que trabaje con gráficos, música, vídeo, etc.

Dentro de los sectores industrial, médico y militar más profesionales, hay una gran cantidad de soluciones y alternativas, tanto de hardware (cámaras térmicas, cámaras lineales, cámaras de alta resolución o alta velocidad) como de software, a OpenCV.



Ejemplo de cámara térmica. La imagen se construye según la temperatura de cada punto
Fuente: www.x20.org

3. Diseñando interacciones: conceptos, algoritmos y funciones. Kinect

Las técnicas de visión artificial siguen normalmente un proceso desde la adquisición de la imagen hasta la toma de decisiones según la información procesada. Lo resumimos a continuación:

3.1. Adquisición de la imagen

Una imagen digital es producida por sensores digitales presentes en cámaras y otros dispositivos digitales que generan una imagen bidimensional (2D), es decir, un conjunto de $N \times M$ **píxeles** o colores o intensidades de un cierto valor que representan el **espacio** que queremos analizar.

En el mundo de la interacción, **podemos utilizar cámaras digitales de diferentes tipos** (cámaras web, DV o USB). En el fondo, cada tipo de aplicación puede necesitar un tipo u otro de cámara según las necesidades.

3.2. Procesamiento de la imagen

Antes de extraer información directamente de la imagen, se acostumbra a hacer un procesamiento previo de la misma para conseguir otra que nos permita hacer el proceso de extracción de datos más sencillo y eficiente.

Ejemplo

Por ejemplo, el procesamiento puede incluir funciones para modificar la luminosidad y el contraste, para reescalar la imagen, los niveles de color, las curvas, la binarización, el desenfoque (*blur*), etc.

Como ejemplo, el hecho de aplicar un desenfoque en una imagen nos permite, de alguna manera, "redondear" los contornos de las formas que aparecen en ella, lo nos puede ser útil en diversas ocasiones. Modificar la luminosidad para tener una imagen más contrastada nos puede ayudar a dar más importancia visual a la parte que queremos analizar. Reescalar y reencuadrar la imagen nos permite centrar el proceso de visión artificial solo sobre la porción de la imagen que nos interesa realmente, ahorrándonos así un procesamiento innecesario.

A continuación, analizaremos algunos algoritmos de procesamiento de la imagen usados frecuentemente en la visión artificial.

Esta será la imagen de ejemplo original que utilizaremos para explicar diferentes funciones del procesamiento de la imagen. Tiene una resolución de 800 píxeles horizontales por 800 píxeles verticales.

Imagen original
Fuente: www.opencv.org

3.2.1. Escala de grises

En muchos casos, el color de la imagen no nos aportará ninguna información relevante para interactuar. Por lo tanto, en estas situaciones se acostumbra a descartar la información de color de la imagen transformándola en una **imagen de tonos de grises**.

De este modo, "ahorraremos" mucha información y los cálculos serán mucho más sencillos de computar. Por ejemplo, si estamos analizando el movimiento dentro de una imagen en color, nos podemos ahorrar aproximadamente 2/3 partes de los píxeles que se han de tratar si descartamos los componentes de color y trabajamos simplemente con la imagen de intensidad luminosa o niveles de grises.

Transformamos la imagen de prueba de color (R, G, B) a tonalidad de grises, es decir, solo con una dimensión de color (L) de luminancia.

Por ejemplo, la imagen original ocupaba en la memoria, en número de bytes:

800 píxeles × 800 píxeles × 3 bytes (R, G, B) = **1.920.000 bytes**.

En cambio, la imagen transformada en blanco y negro ocupará, en número de bytes:

800 píxeles × 800 píxeles × 1 byte (L) = **640.000 bytes**.

3.2.2. Binarización por umbral (*threshold binarization*)

El proceso de binarización por umbral parte de una imagen en tonos de grises y, a partir de un valor definible de umbral de intensidad de luz llamado *threshold,* la imagen se transforma en una imagen binaria, es decir, con píxeles blancos o píxeles negros.

Si el píxel analizado tenía un valor de intensidad inferior al umbral, quedará en negro, y si su intensidad era más elevada que el umbral, quedará en blanco. De alguna manera, esto nos permite descartar los tonos medios grises y facilita mucho la computación de ciertos algoritmos, puesto que otra vez hemos transformado la imagen en algo mucho más ligero y fácil de procesar.

Transformamos la imagen de prueba de color (R, G, B) a blanco y negro, es decir, solo con dos colores: blanco absoluto y negro absoluto.

Tal como hemos visto, la imagen original pesaba 1.920.000 bytes (1 byte = 8 bits); por lo tanto, pesaba 15.360.000 bits.

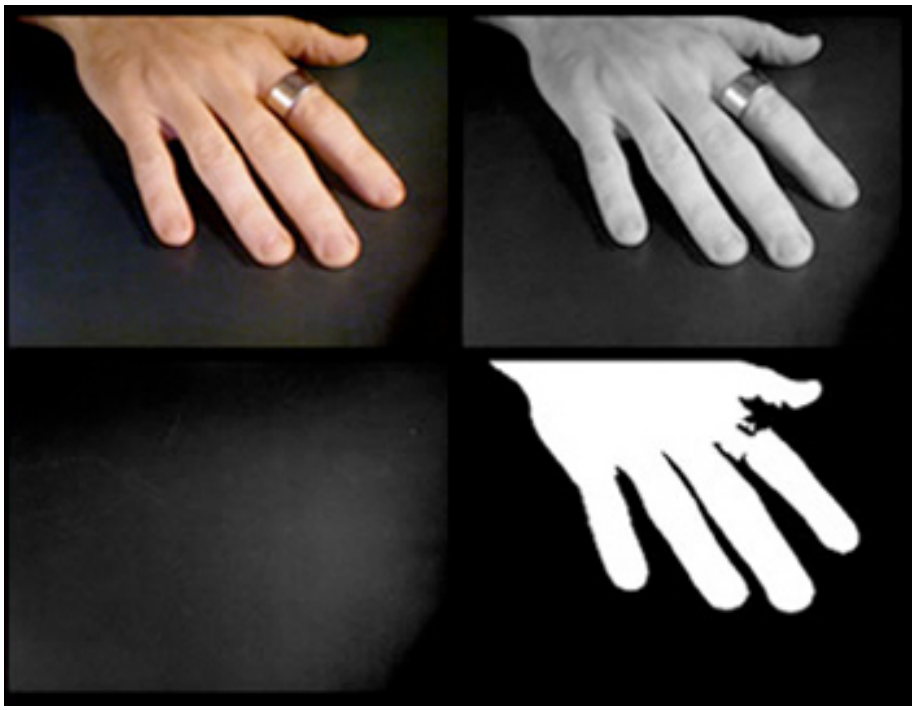
En cambio, la imagen transformada con umbral de blanco y negro ocupará, en número de bits:

800 píxeles × 800 píxeles × 1 bit = 640.000 bits, es decir, 24 veces menos en número de bits que el original.

Velocidad y resolución de la imagen

Muchos de los procesos de visión artificial se basan en trabajar por cada píxel de la imagen internamente; por lo tanto, si con la binarización conseguimos 24 veces menos de

bits que procesar, los algoritmos correrán mucho más rápido. Así pues, un factor determinante en la velocidad de los cálculos que se generan en las aplicaciones de visión artificial es la resolución de la imagen en píxeles. Una imagen de 320 × 240 píxeles será procesada mucho más rápidamente que una imagen de 1.024 × 768 píxeles.



Fuente: www.openframeworks.cc

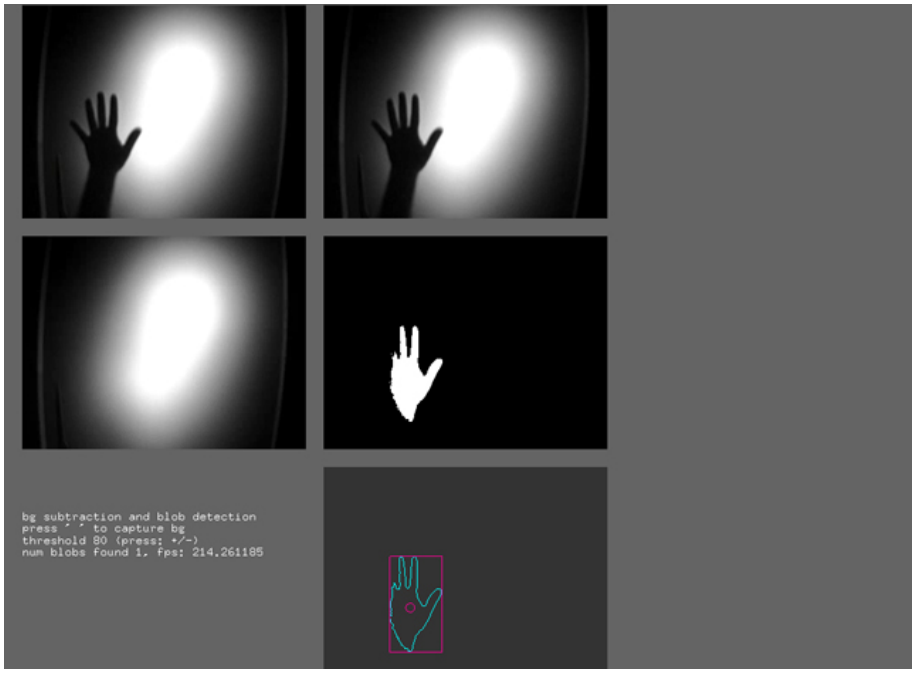
En esta imagen, podemos ver las tres representaciones hasta ahora mencionadas. Arriba a la izquierda, tenemos la representación en color del fotograma capturado por la cámara; arriba a la derecha, la transformación a blanco y negro de la misma imagen, y abajo a la derecha, la representación binaria de la imagen en blanco y negro. El umbral en esta binarización se ha fijado en un valor que permite distinguir fácilmente la mano (blanca) del fondo (negro).

3.2.3. Sustracción de fondo *o background substraction*

La sustracción de fondo o *background substraction* es una técnica de procesamiento de la imagen que permite "restar" una imagen del fondo de la escena con el fotograma actual y obtener, pues, el fotograma actual "menos" el fondo. De este modo, podemos aislar los objetos que han variado respecto de la imagen del fondo de una escena y determinar si ha habido movimiento.

Reflexión

Hay que tener en cuenta que esta técnica requiere poder "capturar" la imagen de "fondo". Su utilidad es en entornos relativamente controlados, en los que podamos conocer cómo será el fondo durante todo el procesamiento o bien en los que nos podamos ir adaptando.



Fuente: www.openframeworks.cc

En esta imagen vemos todo el proceso. La imagen de arriba a la izquierda es lo que "ve" la cámara, y la de abajo a la izquierda es la imagen del fondo o *background* que se ha de restar. Una vez hecha la resta y binarizada la imagen, vemos lo que se ve en la imagen de en medio a la derecha, en la que prácticamente solo aparece la mano en una imagen binaria (con píxeles 100% blancos y 100% negros). Esta última imagen binarizada y restada del fondo ya nos deja casi la silueta perfecta de la mano para aplicar el paso siguiente, que es identificar la "mancha" o *blob* de la mano para obtener cierta información, tal como veremos en el apartado siguiente.

3.2.4. Otras operaciones morfológicas

En muchas situaciones, una vez que hemos obtenido una imagen binaria (en blanco y negro), observamos que aparece **ruido** en la imagen, fruto de los cambios en la iluminación de ambiente y los pequeños reflejos de esta luz en los diferentes objetos y cuerpos en la escena. Este ruido suele consistir simplemente en píxeles que aparecen caóticamente en la imagen y que pueden estorbar seriamente la capacidad de los algoritmos de visión para reconocer *blobs* y patrones gráficos.

Para eliminar estos pequeños y molestos píxeles, hay una serie de algoritmos de "limpieza" de la imagen que nos ayudan a cumplir la tarea de obtención de una imagen binaria lo más neta posible.

1) Mediana (*median*)

El algoritmo de la mediana se aplica sobre las imágenes en escala de grises y provoca una cierta suavización de la imagen útil a la hora de simplificar contornos y áreas irregulares.

El algoritmo de la mediana divide la imagen en un conjunto de divisiones de radio definible y calcula el valor de luminosidad medio de cada una de las subdivisiones. Una vez calculado, sustituye los píxeles que hay dentro de cada división por una única mancha gris con el valor de luminosidad medio de los píxeles originales.

1. Imagen original

2. Aplicación de la media

2) Erosionar/dilatar (*Erode/dilate*)

Los algoritmos de erosión y dilatación se suelen aplicar en serie. Sobre la imagen "ruidosa" se aplica el algoritmo de erosión, que contrae los contornos de todas las áreas blancas un número determinado de píxeles y que elimina completamente las áreas de píxeles blancos más pequeñas e irrelevantes.

Una vez aplicado el algoritmo de erosión, se aplica el algoritmo de dilatación, que ayuda a recuperar la medida original de las áreas importantes y que expande los contornos de las áreas blancas tantos píxeles como sea necesario.

1. Imagen ruidosa

2. Imagen erosionada

3. Imagen dilatada

3.3. Extracción de datos

Una vez que tengamos la imagen preparada, aplicaremos una serie de algoritmos para extraer los datos de la misma. Podemos reconocer en la imagen, por ejemplo, las líneas, los círculos, las manchas, las esquinas, el movimiento, un determinado color..., o bien ciertos patrones de imagen previamente determinados.

Una vez reconocido en la imagen lo que queríamos extraer (sea por cada línea, círculo o mancha reconocidos) de cada elemento, podremos averiguar su posición, velocidad de movimiento, medida o color dominante.

A continuación, analizaremos una serie de algoritmos frecuentemente usados para la extracción de datos en el contexto de proyectos relacionados con visión artificial:

3.3.1. Reconocimiento de regiones o *blob detection*

Llamamos *blobs* a los puntos o las regiones de píxeles más luminosos (o más oscuros) que los de los alrededores y que, por lo tanto, forman una "mancha" aislada del resto de la imagen.

Las herramientas de **reconocimiento de regiones** o ***blob detection*** nos permiten analizar las "manchas" de una imagen binaria (blanca y negra) y extraer información de cada una, como la medida de la mancha, el centro de masas de la mancha, su contorno...

Este tipo de funciones son útiles en muchos casos, por ejemplo, para contar y localizar cuántos objetos tenemos en una escena. A partir de que los hayamos localizado, podremos seguir su trayectoria o analizar su movimiento. Por lo tanto, esta técnica se utiliza en muchos procesos interactivos.

Detección de *blobs*

Tened en cuenta que la detección de *blobs* requiere normalmente, de entrada, una imagen ya binarizada con las funciones de procesamiento de la imagen que acabamos de ver.

3.3.2. *Frame difference*: seguimiento del movimiento o *movement tracking*

Esta técnica nos permite obtener datos del movimiento que hay en la escena que estamos analizando, tanto de la cantidad de movimiento como de su dirección.

La técnica consiste básicamente en restar un fotograma capturado por la cámara con el anterior. La resta nos da una referencia de lo que ha cambiado entre las dos imágenes. Normalmente aplicaremos esta técnica después de haber hecho la sustracción de fondo, es decir, que habremos eliminado mucha información del fondo.

En caso de que no haya cambiado nada entre un fotograma y el anterior, la resta nos dará una imagen prácticamente negra.

Si, por ejemplo, en la cámara estamos viendo un objeto en movimiento, la resta entre dos fotogramas consecutivos nos dará las partes de la imagen que han cambiado. Si la imagen ha cambiado mucho (porque el objeto se desplazaba rápidamente), la resta será una mancha bastante grande; si la imagen ha variado muy poco, la operación de resta nos dará una mancha muy pequeña.

Para acabar de extraer toda la información, lo que haremos es una detección de *blobs*. Como resultado, tendremos, por ejemplo, la medida de la mancha del *blob,* que será una medida de la cantidad de movimiento. O bien, si comparamos los centros de masas de la mancha o *blob* en dos fotogramas consecutivos, podremos extraer una medida de la dirección del movimiento.

3.3.3. Seguimiento de color o *color tracking*

El seguimiento de color o *color tracking* nos permite hacer el seguimiento de un área de píxeles de un determinado color (es decir, de un valor específico RGB).

A pesar de que esta técnica no se incluye en la implementación clásica de las bibliotecas OpenCV, es bastante popular. De hecho, se trata más bien de la suma de un conjunto de técnicas, que implican la extracción selectiva de canales de color, para aislar el color seleccionado y obtener una imagen binaria en la que solo el color del que se ha hecho el seguimiento aparece como un *blob* o una mancha blanca de píxeles.

Generalmente, cuando hacemos un seguimiento de color, seleccionamos el color que queremos recorrer y definimos la medida del área alrededor de este píxel. El algoritmo tendrá que ir reescaneando permanentemente esta área para no perder el seguimiento en caso de que el píxel original cambie de color. Cuanto mayor sea esta área de seguridad, más dificultades tendrá el ordenador para hacer todos los cálculos, puesto que se trata de una operación con gran demanda de capacidad de cálculo por parte del procesador.

Gran popularidad del *color tracking*

El seguimiento de color es muy popular también en entornos de software más allá de las aplicaciones de interactividad en tiempo real. Se usa de manera intensiva en muchos programas de tratamiento profesional de vídeo, para recorrer un punto de una grabación de vídeo, y para añadir efectos de postproducción, estabilizar los movimientos involuntarios de la cámara, etc.

3.3.4. Buscador de caras o *face tracking*

Hay toda una serie de técnicas enfocadas al reconocimiento de patrones o formas concretas en una imagen. Estas técnicas nos permiten definir qué patrones o imágenes buscaremos dentro de la escena. Así es como funcionan la mayoría de técnicas de buscador de caras.

Esta función detecta si hay "caras humanas" dentro de la imagen y nos permite extraer cierta información de la cara (como su posición y su medida) y algunos aspectos morfológicos (que son la base para poder identificar las caras de diferentes usuarios).

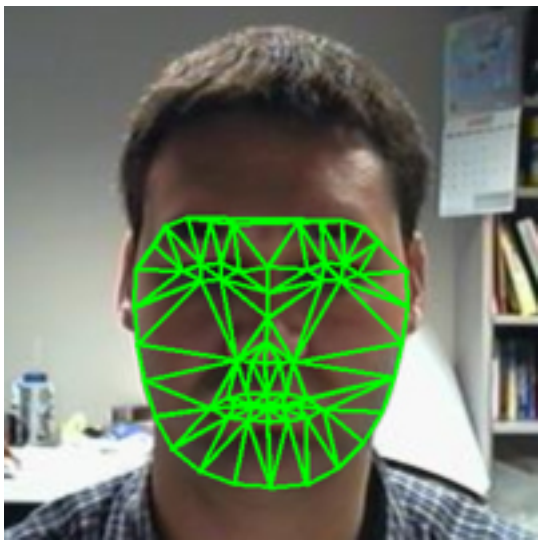
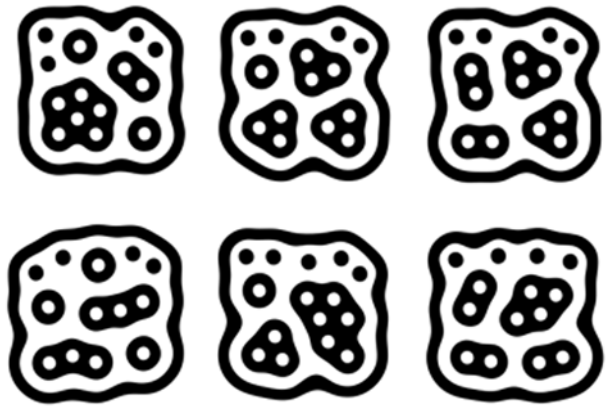


Imagen de ejemplo de buscador de caras en el proyecto AAM Fitting Algorithms
Fuente: Carnegie Mellon University. Robotics Institute

3.3.5. Buscador de características o *feature tracking*

A continuación, denominaremos bajo un mismo nombre toda una serie de técnicas basadas en la extracción de características o *features* de una imagen, que básicamente son puntos de la imagen fácilmente reconocibles por ciertos algoritmos (detección de esquinas, por ejemplo).

Otra aplicación de estas técnicas permite localizar los **fiduciales** de una escena. Los fiduciales serían patrones especialmente diseñados para ser reconocidos, como por ejemplo los fiduciales que utiliza el instrumento interactivo Reactable, desarrollado por el equipo de Sergi Jordà (MTG-UPF).



Marcadores fiduciales del sistema Reactivision, utilizados en Reactable
Fuente: reactivision.sourceforge.net

3.3.6. Realidad aumentada

Como habéis ido viendo, vamos acumulando unas técnicas tras otras y vamos llevando a cabo acciones cada vez más complejas. Del anterior concepto del buscador de características o *feature tracking* se desprende en cierto modo la técnica de la realidad aumentada o *augmented reality*.

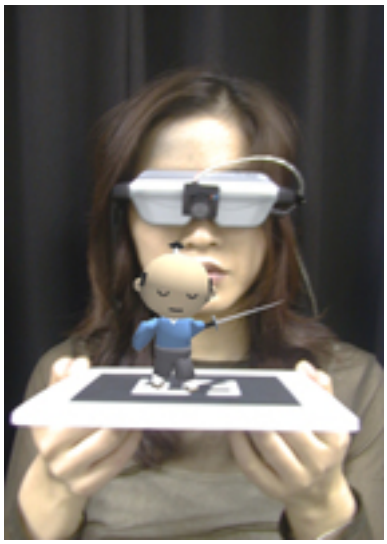
Por *realidad aumentada* entendemos todas las técnicas que de alguna manera integran una imagen virtual dentro de una imagen de un entorno real y, evidentemente, en tiempo real y de manera interactiva.

Una de las bibliotecas más utilizadas se denomina **ARToolKit**, que permite el trabajo desde muchos entornos de programación creativa.

Ejemplo de marcador de un sistema de realidad aumentada

Una de las aplicaciones más conocidas es la capacidad de introducir un elemento en 3D interactivo dentro de la imagen capturada por una cámara. Esta integración es coherente con el punto de vista de la cámara, y nos da la impresión de que los objetos en 3D están presentes en el entorno visto por esta. Evidentemente, esta técnica requiere una cámara para tener sentido, y siempre veremos el mundo en 3D integrado en la visión de la cámara en un monitor.

Esta imagen es un marcador o *marker* para un sistema de realidad aumentada. Este simple patrón es analizado por el sistema de visión artificial y de él podemos extraer la posición y la orientación relativas a la cámara. A partir de aquí, solo nos resta vincular un mundo virtual en 3D en relación con esta marca, copiando su perspectiva y su orientación.



Típica aplicación de la realidad aumentada. Se genera una escena en 3D en relación con la posición del marcador
Fuente: artoolkit.sourceforge.net

Aquí podemos ver una aplicación muy sencilla de realidad aumentada: el usuario tiene en sus manos un papel con un marcador impreso, el sistema de visión reconoce el marcador y su orientación, y genera una figura en 3D sobre este, coherente con su posición, su rotación y su inclinación.

3.4. Kinect

Cuando apareció en el mercado la cámara Kinect de Microsoft, se generalizó la posibilidad de trabajar en el campo de la visión volumétrica o tridimensional.

> Este tipo de cámaras volumétricas nos permiten adquirir una imagen plana en color y una imagen de profundidad, es decir, nos permiten saber a qué distancia (o profundidad) de la cámara se encuentran cada uno de los píxeles de la imagen.

Además, si dentro de la imagen aparece la figura de un cuerpo humano, la propia cámara es capaz de reconocer la postura de dicha figura y de enviarnos a nuestro software la posición y la orientación de tronco, cabeza, brazos y piernas.

Al cabo de pocos días de su aparición en el mercado, surgieron personas que habían conseguido desencriptar el protocolo USB de la Kinect usando métodos de ingeniería inversa y que lo publicaron en Internet. En pocas semanas,

una gran cantidad de entornos de programación interactiva permitían interactuar con la Kinect: Processing, openFrameworks, Quartz Composer, Max/MSP, EyesWeb, Cinder, vvvv, Flash... Así pues, un producto pensado para el mercado de las consolas XBox se empezó a utilizar extensamente en el mundo del diseño interactivo.

Fotografía infrarroja en la que se muestra la nube de puntos infrarrojos que emite la Kinect para obtener la profundidad de la imagen. Fuente: www.mattcutts.com

El hecho de que la cámara Kinect sea un producto comercial masivo permite que su precio sea muy asequible en relación con las prestaciones que nos da en un sistema de visión artificial. Poder disponer, por un lado, de una imagen con profundidad y, por otro, del reconocimiento de figuras humanas nos facilita muchísimo el trabajo de interacción basado en la visión.

Por ejemplo, con una imagen con profundidad de un cuerpo, podemos pedir a la aplicación que solo trabaje sobre los píxeles que hay entre 1,50 y 1,30 m de distancia de la cámara. Si, por ejemplo, el usuario está a un metro de la cámara y mueve las manos hacia delante hasta que entren dentro del rango de 1,50 a 1,30 m, el sistema podrá trabajar con una imagen en la que solo aparecen dibujadas las manos; por lo tanto, las tendremos fácilmente aisladas para poder detectarlas con un *blob-tracking* y extraer su posición.

Es decir, con unas técnicas muy sencillas podemos extraer la posición de las manos con mucha precisión. Si no tuviéramos la imagen de profundidad, nos sería mucho más difícil conseguir el mismo resultado, por no decir imposible.

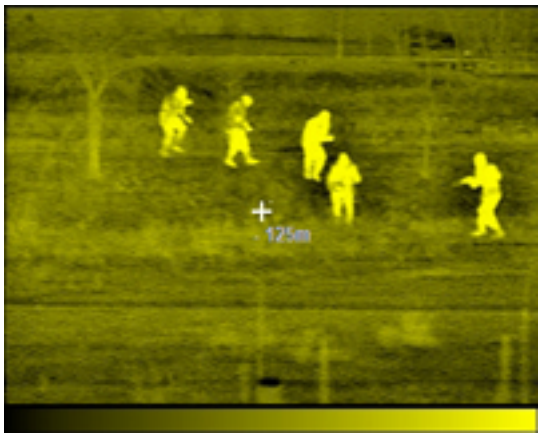
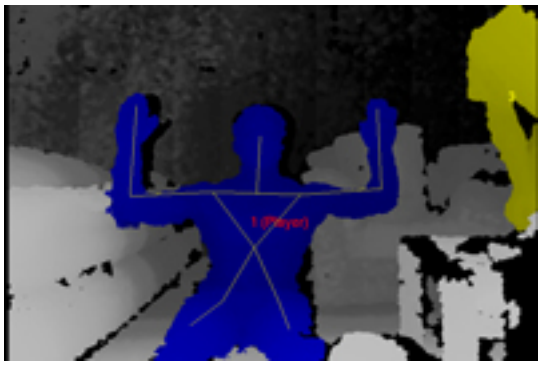


Imagen de profundidad obtenida con una Kinect. Las zonas más claras representan más proximidad a la cámara; las zonas más oscuras representan más alejamiento de la cámara. Fijaos que la parte del cuello está fuera del rango de profundidad y que, de alguna manera, no existe en esta imagen.
Fuente: www.lawriecape.co.uk

Otra característica impresionante de la Kinect es su capacidad para reconocer las posturas y la orientación de una figura humana ante la cámara. El propio hardware de la Kinect nos dice cuántas personas detecta en la imagen y cuál es la postura de cada una de ellas. Los datos que expresan una postura normalmente son los ángulos de las articulaciones. La Kinect divide el cuerpo en tres dimensiones (cabeza, antebrazo, brazo, tronco superior e inferior, cabeza, muslos y piernas). De esta capacidad han surgido multitud de técnicas para interactuar con los usuarios; por ejemplo, generando un modelo en 3D en tiempo real que se mueve tal como se mueve el usuario.



En esta imagen vemos cómo la Kinect pinta una estructura de esqueleto simplificada que copia la posición tridimensional del usuario ante la cámara.
Fuente: weblog.200ok.com.au

3.5. Diseño de interacciones

Una vez que ya hemos extraído los datos de la imagen, hemos de utilizar esta información para tomar decisiones dentro de nuestro sistema.

Esta es la etapa del proceso en la que, a partir de los datos extraídos, podemos tomar decisiones y acciones relacionadas.

Ejemplo

Por ejemplo, si el usuario ha movido los brazos, activaremos una secuencia determinada de operaciones, o si la pieza roja está muy cerca de la azul, haremos sonar una música.

En general, obtendremos unos datos numéricos (coordenadas del centro de masas, cantidades de píxeles de un determinado color, velocidad de movimiento...) que tendremos que transformar y que usar como control de todo tipo de acontecimientos. Obviamente, en esta parte hemos de ser creativos y analizar cuáles son los mapeos más eficientes para diseñar una interacción funcional y comprensible.

4. Más allá. Recursos y bibliografía específica

4.1. Referencias

Computer visions:
http://en.wikipedia.org/wiki/Computer_vision

Processing Tutorials: Getting Started with Video Processing via OpenCV:
http://createdigitalmotion.com/2009/02/
processing-tutorials-getting-started-with-video-processing-via-opencv/

OpenCV Motion Tracking, Face Recognition with Processing: I'm Forever Popping Bubbles:
http://createdigitalmotion.com/2009/02/opencv-
motion-tracking-face-recognition-with-processing-im-
forever-popping-bubbles/

Processing OpenCV Tutorial 1:
http://andybest.net/2009/02/processing-opencv-tutorial-1/

Introduction to programming with OpenCV:http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html

The Five Reactive Books:
http://www.youtube.com/watch?v=nA_UTUvC4h8

Project3 – New Interactions with Kinect and Computer Vision:
http://golancourses.net/2011spring/projects/project-3-interaction/

Microsoft explica cómo funciona Kinect:
http://materiageek.com/2011/03/microsoft-explica-como-funciona-kinect/

4.2. Bibliografía

Programming Interactivity. A Designer's Guide to Processing, Arduino, and openFramework:
http://www.amazon.com/Programming-Interactivity-Designers-Processing-Openframeworks/dp/0596154143