Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Marduk: sistema de tecnología de sexto sentido para dispositivos móviles

Tesis que presenta

Alberto Beltrán Herrera

para obtener el Grado de

Maestro en Ciencias

en Computación

Directores de Tesis

Dra. Sonia Guadalupe Mendoza Chapa

Dr. Adriano de Luca Pennacchia

México, D.F. Febrero 2012

Resumen

Las interfaces de usuario fueron creadas con el propósito de facilitar la interacción hombre-máquina. Así, la evolución de la computación conlleva también a interfaces de usuario más poderosas, las cuales buscan ser cada vez más naturales y amigables, adaptándose a las necesidades del usuario mas no inversamente. En la última década un nuevo tipo de interfaces de usuario, denominadas NUI (Natural User Interface), empieza a ganar terreno sobre el paradigma GUI (Graphical User Interface), que ha sido el más popular en la creación de interfaces de usuario. Las interfaces NUI hacen uso de las gesticulaciones del cuerpo humano como datos de entrada, así el usuario puede emplear movimientos que le parecen familiares para crear una interacción más natural, como si estuviera manejando objetos reales. Una sub-categoría de las interfaces NUI es la tecnología de Sexto Sentido, la cual busca expandir el concepto de NUI, añadiendo funciones de movilidad y realidad aumentada. Esta tesis encuentra un campo fértil en este nuevo tipo de tecnología, ya que los prototipos hasta ahora propuestos son meramente experimentales e imprácticos en la vida real. En este trabajo llevamos la tecnología de sexto sentido a los dispositivos móviles, permitiendo que el sistema se vuelva ubiquo y sea facilmente aceptado por los usuarios. Para lograr este objetivo, la presente tesis de maestría aporta una arquitectura desarrollada a partir de las necesidades de la tecnología de Sexto Sentido y de las capacidades del cómputo móvil. Adicionalmente, se presentan las especificaciones de cada módulo y los algoritmos de visión por computadora correspondientes (descomposición de imágenes, reconocimiento de caracteres y tratamiento de imágenes). Los algoritmos desarrollados responden a un estudio sobre las especificaciones de las imágenes tratadas y las características de los dispositivos móviles, acercándonos lo más posible a un punto óptimo en las capacidades de procesamiento, velocidad y memoria. El resultado de nuestro trabajo son: 1) un prototipo de sexto sentido funcional, practico y fácilmente reproducible, 2) una arquitectura que da sustento a las partes física y lógica, 3) un software encargado de los procesos de visión por computadora, los cuales utilizan nuevos algoritmos capaces de realizar su trabajo en los dispositivos móviles de forma eficiente y finalmente 4) un conjunto de aplicaciones que permiten medir y experimentar con las capacidades del sistema, constituyendo un conjunto de pruebas que pueden ser utilizadas como métricas de comparación por cualquier sistema de Sexto Sentido.

Palabras clave: tecnología de sexto sentido, interacción humano-computadora, natural user interface, realidad aumentada, cómputo móvil, cómputo ubicúo.

Abstract

User interfaces were created with the aim of facilitating human-computer interaction. Thus, the evolution of computing also leads to more powerful user interfaces, which try to be more and more natural and friendly, adapting themselves to the user's needs but not inversely. In the last decade a new type of user interfaces, named NUI (Natural User Interface), starts gaining ground on the GUI (Graphical User Interface) paradigm, which has been the most popular in the creation of user interfaces. The NUI interfaces use human body's gesticulations as input data, so the user can employ movements that seem familiar to him in order to create a more natural interaction as if he were handling real objects. A sub-category of the NUI interfaces is the Sixth Sense Technology, which aims to extend the concept of NUI by adding mobility and augmented reality functions. This thesis finds a fertile ground in this new type of technology because the proposed prototypes are merely experimental and hardly practical for real life. In this work, we transpose the Sixth Sense technology to the mobile devices, allowing the system to become ubiquitous and to be well-accepted by the users. In order to achieve this goal, we proposes an architecture developed from the Sixth Sense technology needs and the mobile computing capabilities. In addition, we present each module specifications and the corresponding computer-vision algorithms (image decomposition, character recognition and image processing). The developed algorithms respond to a study about the especifications of the processed images and the characteristics of the mobile devices, approaching as much as possible to an optimum point in the processing, speed and memory capabilities. The results of this work are: 1) a functional, practical, and easily reproducible Sixth Sense prototype, 2) an architecture that supports the physical and logical parts, 3) a software responsible for the computer-vision processes, which use new algorithms able to efficiently carry out their work on mobile devices, and finally 4) a set of applications that allow to measure and experiment with the system capabilities, constituting a test set that can be used as comparison metrics by any Sixth Sense system.

Key words: sixth sense technology; human-compueter interaction; natural user interface; augmented reality; mobile computing; pervasive computing.

Agradecimientos

A CONACYT por el apoyo proporcionado para continuar con mis estudios y que me permitió desarrollarme durante esta etapa de mi vida.

Al CINVESTAV por abrirme sus puertas y procurar mi formación en todos los aspectos.

A mis directores, la Dra. Sonia Guadalupe Mendoza Chapa y el Dr. Adriano De Lucca Penniacci, por su labor dirigiendo este documento. Agradezco especialmente a la Dra. Sonia, a quien considero una excelente directora y una excelente persona.

A todos los profesores del departamento, me enseñaron no sólo lo necesario para ser un gran computólogo, las lecciones que me dieron me han hecho crecer más como persona.

A Sofy, tú debes ser la mejor persona del departamento, tú nos hiciste la vida realmente grata y nos ayudaste más allá de lo que debías. Personas como tú hay realmente pocas.

A mi madre, que siempre me ha dado todo su apoyo de forma incondicional. Gracias por la vida que me diste, he aprovechado cada minuto y espero poder retribuirte todo lo que me has dado.

A mi padre, que ha sido un ejemplo de vida para mí, a quien admiro en demasía. Si he llevado mi vida bajo los principios de justicia y coraje ha sido por ti, espero siempre llenarte de orgullo.

A mis hermanas Griselda y Mariana, las cuales me prestan apoyo siempre que lo necesito, somos muy distintos entre nosotros y al mismo tiempo nos llevamos muy bien, espero poder ser ejemplo para ustedes y poder ayudarlas también siempre que lo necesiten.

A Dios, quien este último año me ha mostrado muchos caminos, no siempre te entiendo, no siempre sé que es lo que debería hacer, sin embargo sé que siempre estás ahí, cuidando. Seguiré cumpliendo mi promesa y jamás me daré por vencido, eres un gran amigo.

A todos mis amigos del CINVESTAV, con quienes he pasado dos años y medio realmente geniales, nos reímos, nos ayudamos, nos apoyamos, hicimos tantas cosas juntos, vivimos tanto que ahora son parte de mi familia. Con algunos pasará un tiempo antes de volvernos a ver, pero siempre serán mis amigos.

Índice general

Índice de figuras

Lista de tablas

Lista de algoritmos

Capítulo 1

Introducción

1.1 Antecedentes

Las primeras formas de interacción hombre-máquina desarrolladas en la computación son conocidas como CLI (*Command Line Interface*), concepto que nace en los años 50 junto con las máquinas de teletipo y los primeros sistemas operativos. Las CLI se caracterizan por ofrecer comunicación basada en comandos de texto que el usuario debe memorizar, y fuerón diseñadas para las primeras computadoras con limitados recursos gráficos. Aunque su uso ya no es frecuente encontramos varias aplicaciones que aun funcionan con este tipo de interfaz, siendo el ejemplo típico la terminal de Linux. Más tarde Xerox creó un nuevo paradigma en interfaces de usuario basadas en elementos gráficos. El primer prototipo experimental fue conocido como Xerox Alto (1973) pero no fue sino hasta 1981 que las interfaces gráficas vieran la luz comercialmente con la computadora Xerox Star 8010. Nacen entonces las GUI (*Graphic User Interface*) extendiendo su popularidad rápidamente entre compañías como IBM, Apple y la entonces naciente Microsoft. Aunque se establecieron diversos manejadores de interfaces graficas la mayoría seguía el paradigma WIMP (*Window, Icon, Menu, Pointer device*). Esta forma de interacción se ha mantenido como estándar desde entonces y hasta la fecha. Si bien se ha aumentado la calidad de los gráficos y animaciones al punto de incluir elementos tridimensionales, se sigue tratando de GUI's.

Un primer intento por mejorar las interfaces de usuario surgió en los noventa y aun tuvo rastros durante la primer década del siglo XXI. Se trató de la realidad virtual, una simulación del mundo real mediante elementos tridimensionales virtuales creados por computadora, donde los sujetos están inmersos en esta reproducción con ayuda de hardware especializado. El complejo cómputo necesario para crear los mundos virtuales en los que se desenvuelve el usuario junto al uso de objetos especiales para la comunicación como gafas y guantes provocó que este paradigma nunca despegara; además los usuarios parecían tardar mucho tiempo en concluir tareas comunes en

comparación con las ya establecidas GUI [Tatu Harviainen and Takala, 2007]. Como bien lo anticiparía Rauterberg y Steiger [Rauterberg and Steiger, 1996] [GWM, 1999] la próxima generación de interfaces de usuario encontraría soporte no sobre la realidad virtual sino sobre la realidad aumentada.

Recientemente, como resultado de la evolución tecnológica, ha surgido el concepto de NUI (*Natural User Interface*) como sucesora de GUI. Según este concepto, desaparecen los periféricos tradicionales de entrada y la comunicación con la máquina se logra con movimientos naturales e intuitivos de la misma forma con la que interactuamos con objetos del medio ambiente. En 2006 Christian Moore establece la primera comunidad de investigadores de NUI, el NUI Group, consolidando el término aunque ya existían aplicaciones comerciales y prototipos que trabajaban bajo esta ideología como el iPod de Apple[1], Microsoft Surface[2] y Perceptive Pixel [3], todos ellos trabajan sobre pantallas o superficies definidas y hardware especializado bajo los principios establecidos por [Chan et al., 2007]. Desde entonces han aparecido numerosos algoritmos, implementaciones y nuevas tecnologías para la creación de NUI, los cuales se han presentado en diversos congresos como el *Consumer Electronic Show*[4] y el TechFest[5], tomando su lugar como una de las tecnologías emergentes más prometedoras.

Desde el nacimiento de las NUI, la intención ha sido correr a la par del movimiento tecnológico y despegarse del entorno estático, ya que hasta el momento los prototipos y proyectos comerciales de las NUI emplean infraestructura montada, inmóvil y especializada. El seguimiento de esta tendencia implica el uso de dispositivos móviles y métodos de visión por computadora [Siltanen and Hyväkkä, 2006]. La denominación para estos sistemas de interacción gestual, portátiles y vestibles es Sixth Sense (Sexto Sentido) [Mistry and Maes, 2009] que se inscriben en el campo de la realidad aumentada pues la interacción se lleva a cabo en proyecciones sobre objetos físicos.

1.2 Planteamiento del problema

Es objeto de esta tesis transponer las NUI y en específico la tecnología de sexto sentido a teléfonos celulares para acercar al usuario común a la última generación en paradigmas de interfaces de usuario sin involucrar costos extraordinarios.

[1] www.apple.com/ipod/
[2] www.microsoft.com/surface/
[3] http://www.perceptivepixel.com/
[4] www.cesweb.org/
[5] www.techfest.org/

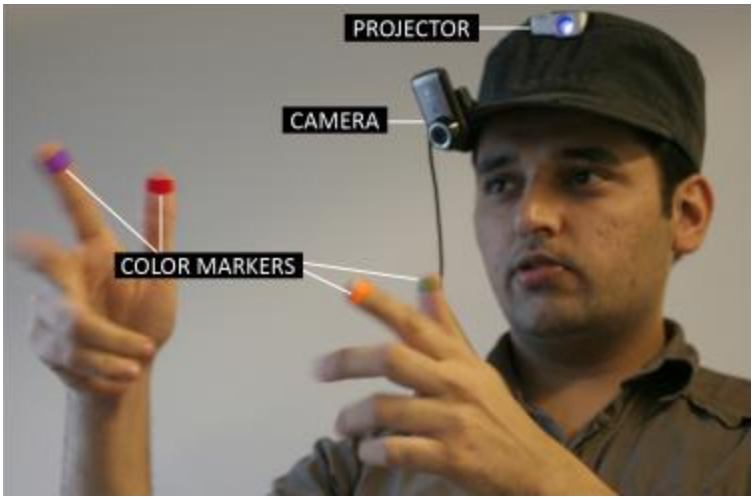

Figura 1.1: Periféricos del sistema de sexto sentido Wear Ur World

Debido a que resultaría incomodo llevar consigo todos los periféricos propuestos por [Pranav Mistry, 2009] (ver figura 1.1) buscamos integrar las NUI en la forma más simple y familiar posible para el usuario. Por lo tanto la propuesta de este trabajo se enfoca en el dispositivo móvil por excelencia, el teléfono celular. La evolución de la tecnología de sexto sentido para trabajar de esta forma podría conducir a la aceptación por parte del público y a la explosión de la tecnología tanto en investigación como en uso comercial.

Se incluyen varias aplicaciones de prueba tratando de relacionar los trabajos existentes, por ejemplo, una de las aplicaciones permite tomar notas al igual que el proyecto *Physic Paper* [Aliakseyeu and bernard Martens, 2001] (ver figura 1.2) pero de forma movil. *Physic Paper* proyecta sobre una mesa una serie de elementos digitales y monitorea las acciones del usuario sobre la proyección mediante camaras montadas al frente, los movimientos de la mano son interpretados y procesados como símbolos o acciones.

Desarrollar cualquier aplicación sobre sexto sentido implica el desarrollo de Lx-Kadabra, el motor de todo el sistema, encargado del reconocimiento de gesticulaciones. Este toma como entrada una serie de fotografías (de la cámara del celular), busca

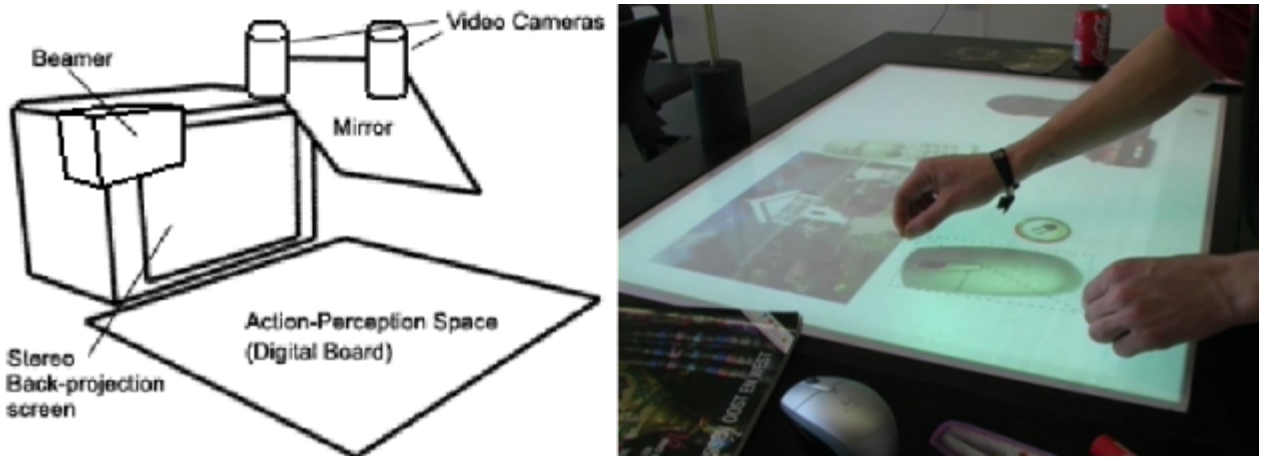

Figura 1.2: Estructura de Physic Paper

el movimiento realizado y lo empareja con un símbolo conocido. La interpretación corresponde a un carácter ASCII o un símbolo de acción, que ejecuta una aplicación o comando predeterminado.

Como soporte para el reconocimiento de caracteres, se utilizan algoritmos *online* como los propuestos en [Calhoun et al., 2002] y [Singh et al., 2010a] que basan el reconocimiento en cambios de posición del puntero, lo que los convierte en algoritmos de bajo peso computacional.

Implícitamente debemos resolver los problemas que trae consigo una implementación de algoritmos de reconocimiento de patrones y visión por computadora en teléfonos celulares como lo son:

- La optimización de algoritmos para que respondan en un tiempo no perceptible por el usuario, de forma que el sistema no se pause o deje de responder por tener que terminar cálculos previos.
- La búsqueda del óptimo entre la calidad de imágen tomada por la cámara y la cantidad de informacion que puede ser procesada por segundo.
- La búsqueda e implementación del algoritmo de calibración de cámaras con mejores resultados que requiera una cantidad menor de recursos.
- La búsqueda e implementación de algoritmos de reconocimiento de formas con base en imágenes y secuencia de imágenes los cuales exijan una menor cantidad de recursos.

- La explotación del uso de todos los sensores del celular a fin de obtener información útil del medio ambiente.

1.3 Objetivos del proyecto

General

Desarrollar un prototipo de un sistema de sexto sentido para celulares, con el fin de hacer más natural la interacción del usuario con su entorno, eliminando el exceso de periféricos e integrando el procesamiento y los dispositivos de entrada/salida en un solo elemento portátil, accesible y popular entre todos los usuarios como lo es el teléfono celular.

Particulares

- Desarrollar un sistema de reconocimiento de caracteres escritos por el usuario a mano alzada, con el fin de transponer los gestos del mundo físico a respuestas del mundo virtual en forma transparente
- Desarrollar un sistema de reconocimiento de señas que facilite la interacción del usuario con su entorno
- Desarrollar una serie de aplicaciones de prueba que permitan validar el sistema como una tecnología de sexto sentido
- Optimizar los algoritmos de reconocimiento de patrones y visión para que se ejecuten eficientemente en un teléfono celular

1.4 Metodología

Para poder proponer una metodología adecuada al proyecto se tuvó en cuenta dos aspectos fundamentales: por un lado tenemos el desarrollo de software para dispositivos móviles y por el otro lado la creación de una interfaz natural de usuario. Debiamosmos tener en cuenta que lo que se esperaba no era sólo un desarrollo teórico, sino también un producto funcional, lo cual nos da la idea de que la metodología usada debía poner en primer lugar el término del producto, moverse rápidamente entre los cambios en la tecnología móvil y poner especial énfasis en cómo el usuario se comunica con el sistema. Teniendo esto en mente el desarrollo de esta tesis se llevó a cabo mediante *metodologías ágiles*, concretamente usamos Movile-D. A continuación se presenta una breve explicación de la metodologia Movile-D y su empleo durante la maquinación de

esta tesis.

Las metodologías ágiles nacen como un nuevo concepto en 2001 en Utah, E.U., como resultado de una reunión de estudiosos de la ingeniería de software con las siguientes motivaciones:

- eliminar el exceso de documentación requerida en cada etapa de los modelos tradicionales;
- la adaptabilidad frente a los cambios que surgen a lo largo del proyecto y
- vincular profundamente las interacciones del usuario con el sistema.

Esta flexibilidad provocó que muchas empresas dejaran atrás los viejos esquemas y pasaran a la programación "ágil". Se trata de un ambiente que evoluciona tan rápido como el computo móvil (en un mismo año nos topamos con dos versiones de un sistema operativo). Las metodologías ágiles han sido muy bien aceptadas, pero hasta hace poco se trabajaba con el mismo tipo de metodologías que se usaban para desarrollar aplicaciones de escritorio, creyendo que no había diferencia. Hoy sabemos que al desarrollar software para este tipo de dispositivos nos encontraremos con problemas específicos como lo son:

- limitado poder de procesamiento y memoria RAM;
- pantallas pequeñas con baja resolución;
- alta fragmentación, i.e. una considerable variedad de estándares, protocolos y técnicas que se aplican de forma distinta en cada dispositivo;
- distribución de dispositivos a nivel mundial, pero sin considerar las necesidades específicas de cada sector y
- alta movilidad y migración de direcciones de red lo que trae consigo latencia

Era claro que se necesitaba una metodología propia para el desarrollo de software sobre dispositivos móviles. En 2004 se da una solución a esta preocupación y nace Movile-D [Abrahamsson et al., 2004] [Ihme and Abrahamsson, 2008] de manos de investigadores del VTT Technical Research Center of Finland en cooperación de varias consultorias especializadas en desarrollo de software sobre dispositivos móviles[6]. Movile-D toma lo mejor de eXtreme Programming (XP), Crystal Methodologies y Rational Unified Process; además se encuentra basado en una serie de estudios de mercado[7].

Movile-D presenta las siguientes ventajas:

[6] http://agile.vtt.fi/case.html
[7] http://agile.vtt.fi/mobiled.html

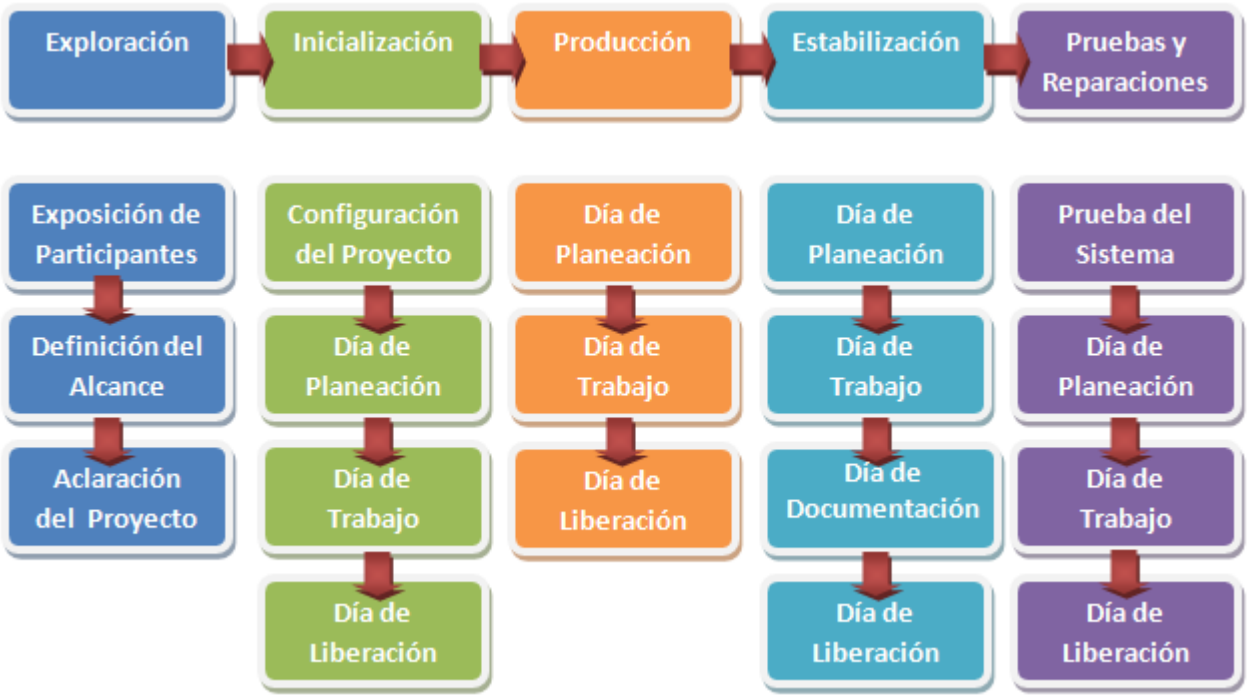

Figura 1.3: Ciclo de desarrollo de Movile-D

- desglosa el proyecto en etapas de tiempo corto,
- flexible a cambiar en cualquiera de las etapas y
- se centra en terminar el producto y en grupos pequeños de trabajo

La figura 1.3 muestra el ciclo que se cumple por cada etapa desglosada. Describiremos cada una y su importancia en esta tesis.

Durante la fase de "Exploración"se establece un plan para el proyecto, aunque esta fase no se considera propiamente dentro del ciclo al ser solo "preparación" para la fase de "Inicialización". La motivación de la fase de "Exploración" es acentuar objetivos, fijar un plan y un cronograma.

En la fase de "Inicialización"así como en cada ciclo, se analiza lo que ya se tiene, se prepara un plan para las siguientes fases y se adaptan nuevas necesidades. Como podemos ver, a partir de esta fase se tiene un día de planificación, uno de trabajo y otro de liberación para terminar lo restante. Esta será nuestra metodologia a seguir. La fase de "Producción"implementa todas las funcionalidades. Durante la fase de

"Estabilización"se termina el prototipo y se integra al sistema; esta fase es vital para esta tesis pues el éxito recae en la integración de partes disjuntas. Finalmente se hacen pruebas controladas y se arreglan los desperfectos. En nuestro caso es necesario hacer siempre una prueba del sistema completo para asegurarnos de no perder la integridad.

1.5 Resultados

Como producto de esta investigación se presenta un software para celulares que valiendose de un pico-proyector aplica tecnología de sexto sentido como interfaces de usuario. El teléfono celular junto al pico-proyector ocupan un volúmen similar al de un smartphone, en concreto 12.5 x 7 x 3 cm, lo que lo convierte en un sistema completamente móvil.

El sistema trabaja con android 2.1 y 2.2 independientemente del teléfono en el que se corra (aunque es forzoso que tenga una salida de video y cámara fotográfica). Puede proyectarse sobre superficies planas en forma óptima y en superficies rugosas con buenos resultados, usando una intensidad de 20 lumens.

Para demostrar la capacidad del sistema se entrega además una serie de aplicaciones que trabajan con interfaces de sexto sentido convirtiendo lo que el usuario escribe en comandos.

1.6 Organización de la tesis

El presente documento se conforma de ocho capítulos estructurados para ser leídos en forma continua. Aquellos con conocimientos previos del tema y familiarizados con la terminología con los objetivos de: entender el funcionamiento del sistema, reproducir el producto final o analizar los resultados, pueden comenzar su lectura desde el capítulo 3. El capítulo 2 comprende el estado del arte, se describen los conceptos necesarios para comprender el documento. Se muestra como la intersección de estos conceptos da lugar a la tecnología de "Sexto Sentido", noción principal sobre la que desarrollaremos. En este mismo apartado se hace un análisis comparativo con los prototipos más importantes creados hasta el momento, sus ventajas, desventajas y características sobresalientes. Durante el capítulo 3 nos introduciremos al análisis y diseño del sistema, su composición arquitectónica y los módulos que lo componen. Se incluye una breve explicación del comportamiento de cada sección y de su comunicación con los otros módulos, a modo de introducción a capítulos posteriores. Explicamos el porqué de la infraestructura de software y hardware seleccionados para esta investigación así como sus características. Los procesos de entrada y salida de datos son presentados durante el capítulo 4, encontraremos en este capítulo una completa explicación de cómo los datos son interpretados a partir de las magnitudes físicas tomadas y como es que el procesamiento vuelve al mundo exterior en forma

de proyecciones. Una vez que se tiene la información de la entrada, es decir, las fotografías en un formato legible para la compuadora, debemos proceder a limpiar la imágen, obteniendo la información verdaderamente útil de ella, esta serie de algoritmos son conocidos como de pre-procesamiento de imágen y son vistos a detalle durante el capítulo 5. Las técnicas, procesos y algoritmos utilizados para crear el motor principal del proyecto conocido como "Lx-Kadabra", son vistas a detalle en el capítulo 6. Explica la extracción de puntos clave de la imágen (con algoritmos de bajos recursos), el reconocimiento de estos como caracteres o símbolos dibujados por el usuario y las acciones que desatan cada uno de ellos. Se ha aplicado un algoritmo de reconocimiento de caracteres en tiempo real especialmente diseñado para dispositivos con procesadores limitados, por lo que además se muestra un estudio de eficiencia de reconocimiento. Como demostración del sistema y de la tenología de sexto sentido se crean varias aplicaciones que hacen uso de este nuevo paradigma. Las aplicaciones juto con su descripción son presentadas a lo largo del capítulo 7. Por último el capítulo 8 presenta las conclusiones de la tesis desde un punto de vista objetivo, analizamos los alcances y posibles implicaciones, así como las mejoras evidentes con los cambios tecnológicos. Se incluye también una sección del trabajo futuro para mejorar la eficiencia en el reconocimiento y miniaturización.

Capítulo 2

Estado del Arte

A lo largo de este capítulo se presenta una literatura amena sobre las interfaces de usuario y su evolución, llegando al punto cúspide en este contexto: la tecnología de Sexto Sentido. Las tendencias en interfaces de usuario han sido marcadas no sólo por la capacidad de nuevos periféricos, la capacidad en el procesamiento de datos ha sido crucial para marcar la línea temporal de cuando las interfaces pueden evolucionar. La aceptación del público juega un factor crucial también, recordemos por ejemplo el fallido intento de la realidad virtual de entrar al mercado, si bien fue gratamente recibida por la industria, los usuarios en general no terminaron de aceptar esta tecnología y finalmente cayo en desuso. La gran tendencia de la sociedad a tecnocratizarse trajo consigo muchos retos, claramente las interfaces existentes fueron una barrera en la adopción de la computación por todo tipo de personas. Investigadores e industria trabajaron para crear un nuevo paradigma que permite a los usuarios una comunicación con las máquinas más natural que nunca, el desarrollo de nuevos modelos de interfaces trae como resultado un cómputo comprensible, amigable y con una curva de aprendizaje baja lo que permite prácticamente a cualquier ser humano usar una computadora sin grandes conocimientos en el área.

2.1 Interfaces de usuario

Una interfaz de usuario define los medios por los cuales un hombre y una máquina se comunican. Si bien imprecisa, la definición es lo suficientemente clara para comprender que una interfaz de usuario actúa como un traductor entre una máquina y quien la controla.

En el contexto de las ciencias computacionales una interfaz de usuario se define como el canal de comunicación que permite a un individuo dar instrucciones a una computadora de una forma humanamente comprensible. Su importancia radica en que permiten una interacción sencilla con las maquinas pues de otra forma sólo un

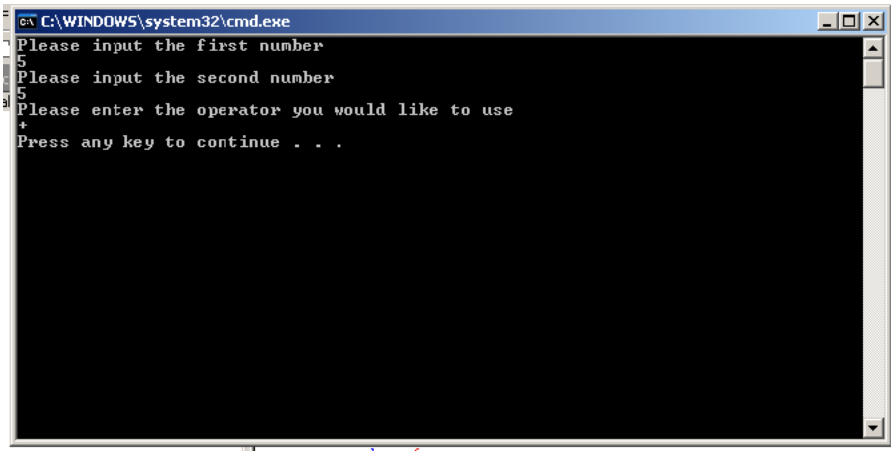

Figura 2.1: En una calculadora por línea de comandos los datos son introducidos uno a uno en un orden secuencial rígido y depende del conocimiento del usuario respecto a las instrucciones y operandos válidos.

grupo de expertos bien formados podría introducir información y obtener resultados de un dispositivo, tal cual se hacía con las viejas computadoras donde había que introducir un código binario al jalar y meter palancas.

Su objetivo es marcado desde su aparición: "crear una forma sencilla, eficiente e intuitiva de transmisión de datos entre usuarios y entidades computacionales". El objetivo ha sido respetado pero ha sido atacado bajo diferentes paradigmas que a su vez han buscado estilizar su presentación y adaptarse al contexto, en la siguiente sección conoceremos cuales han sido las ideologías que marcaron a las interfaces de usuario.

2.2 Historia de las interfases de usuario

Como se vio en el capítulo 1 las *Command Line Interfaces* ó *CLI* son conocidas por ser la primer forma de interface hombre-máquina, en la definición moderna de interface. No es de sorprender que sean poco amigables (basandose sólo en comandos de texto), pues fueron diseñadas únicamente para satisfacer la necesidad de traducción de complicadas instrucciones en lenguaje máquina a formas de expresión comprensibles por el hombre.

Para ejemplificar la evolución de las interfaces de usuario mostraremos un problema particular, se trata de una calculadora simple, observemos en este primer ejemplo la forma particular en que los datos son introducidos en una CLI en la figura 2.1.

El crecimiento en las capacidades de cómputo estará siempre conectado a la evolución de interfaces más complejas y amigables. Así durante la decáda de los ochenta se

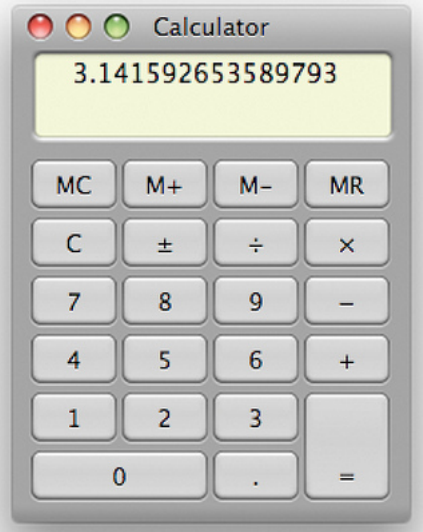

Figura 2.2: La interfaz de la calculadora es similar a una calculadora real lo que facilita el uso de la misma, se ha perdido la rigidez de las CLI pero la interacción aun usa periféricos intermediarios.

crean las *GUI (Graphic User Interface)* de la mano de la compañia Xerox. Se caracterizan por poseer elementos visuales que evitan el tener que escribir cada instrucción, la mayoría de estos elementos visuales responden al paradigma *WIMP (Window, Icon, Menu, Pointer device)*. La facilidad con la que se puede manejar una compuadora haciendo uso de las GUI las popularizó inmediatamente, haciendolas hasta hoy el tipo de interface más usada, teniendo cambios en la calidad de los gráficos, animaciones y efectos tridimensionales, pero conservando el mismo paradigma.

En nuestro ejemplo de la interfaz de la calculadora la GUI simula el aspecto de una calculadora real en una pantalla 2D como lo muestra la figura 2.2.

La primer idea que se concibe para dar el siguiente paso en las interfaces de usuario y que lleva al ambito comercial en una escala importante es la realidad virtual. El concepto era llevar al usuario dentro de la computadora, a un mundo creado de forma artificial con el que el usuario pudiera interactuar de la misma forma que con objetos reales. La idea atrajó la atención de muchas personas, sin embargo el limitado poder de cómputo de aquel entonces convertía los mundos virtuales en mounstruos poligonales, además eran necesarios perifericos extra como: cascos, lentes, guantes,

pantallas especiales y otros tantos propios de cada compañia. Estos factores y algunos rumores sobre los daños oculares que podría provocar el Virtual Boy[1] (principal representante de esta tecnología en la rama del entretenimiento), acabarón por hundir a la realidad virtual en el desuso.

El nuevo milenio traería consigo nuevas concepciones y paradigmas respecto a las interfaces de usuario. Se toma la idea de la realidad virtual pero en sentido opuesto, i.e., la realidad virtual buscaba la inmersión de los usuarios en la computadora, ahora se buscaba que la computadora se sumergiera en la realidad, de esta forma surgen las *NUI (Natural User Interface)*. Las NUI buscan interpretar movimientos que el usuario considera naturales para manipular objetos virtuales como si se tratara de entes reales, e.g., mover un documento de texto por la pantalla como si se tratara de una hoja de papel escrita e igualmente escribir sobre ella como si lo hicieramos con un bolígrafo. Las nuevas generaciones de teléfonos celulares aprovecharían esta tecnología en los smartphone, siendo pionero el iPod[2]. También tomarían parte la nueva generación de consolas con proyectos como el Wii[3], el Kinect[4] y el PlayStation Move[5], sin dejar de lado proyectos fuera de la rama del entretenimiento como el Microsoft Surface[6].

En el paradigma NUI la calculadora podría tomar distintas variantes, podríamos por ejemplo simplemente escribir operaciones en una pantalla o como en la figura 2.3 podriamos interactuar directamente con la información contenida en un dispositivo manipulando números y operaciones con las manos.

En 2009, Pranav Mistry del Massachusetts Institute of Technology toma las NUI y les integra computo móvil y realidad aumentada, lo que bautizaría como *tecnología de Sexto Sentido (Sixth Sense Technology)*[Mistry and Maes, 2009]. En este nuevo paradigma se busca despegar al usuario de los centros de trabajos estáticos y que la computadora vaya siempre con el usuario y este atenta a los movimientos de este, esperando instrucciones, reflejando las respuestas del software en proyecciones alrededor del usuario o sobre objetos reales.

La portabilidad que ofrece la tecnología de Sexto Sentido permitiría llevar nuestra calculadora a cualquier lugar y usarla como instrumento real (ver figura 2.4) pese a que la calculadora no existe físicamente sino como datos y programas.

[1] http://www.nintendo.com/consumer/systems/virtualboy/index.jsp
[2] http://www.apple.com/ipod/
[3] www.nintendo.com/wii
[4] www.xbox.com/kinect
[5] http://us.playstation.com/ps3/playstation-move/
[6] www.microsoft.com/surface/

Figura 2.3: Las entidades informáticas se convierten en objetos con los que se pueden interactuar, literalmente podemos tomar números y funciones.

El contexto histórico nos habla que las interfaces de usuario tienden a hacer la comunicación con las máquinas más sencilla [Petersen and Stricker, 2009], humana e intuitiva, la capacidad de procesamiento de los nuevos equipos nos permite dejar a los dispositivos computacionales cada vez mayor carga de interpretación lo que evita complejidades de aprendizaje y descubrimiento a los seres humanos y hace de la computación un recurso al alcance de la mano de cualquier persona. La figura 2.5. Resume el contenido de esta sección en forma gráfica.

2.3 Natural User Interface

Entendemos a las NUI como la evolución en interfaces de usuario y es que, se escoge a propósito el termino evolución pues hablar de un cambio o revolución crearía confusión con el papel que juega este nuevo paradigma en comunicación hombre-máquina, las NUI no pretenden desplazar a las CLI y GUI e imponerse como dictadores absolutos en el campo de las interfaces de usuario, por el contrario el proceso de adopción de las NUI será gradual y llevará algún tiempo antes que la industria y el mercado terminen de adoptarlas por completo, e incluso cuando ese momento llegue sucederá algo muy

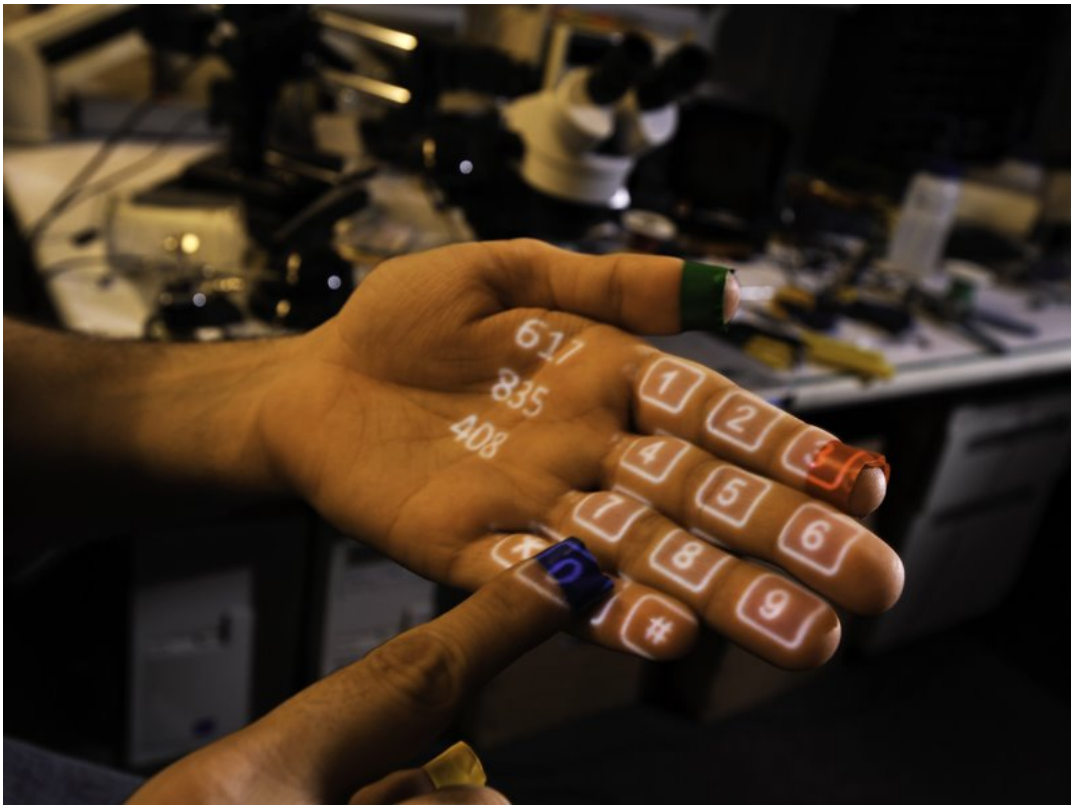

Figura 2.4: Sexto Sentido nos permite tener una calculadora en la mano en cualquier momento y cualquier lugar que se usa tal cual una calculadora real.

similar a lo ocurrido cuando las GUI ganaron campo sobre las CLI. Durante la década de 1980 cuando surgen las GUI el mercado era dominado por interfaces de usuario de texto plano, ya la mayoría de los usuario de computadoras se encontraban familiarizados con esta idea (lo que no quiere decir exactamente que les pareciera agradable), pero el paso a una nueva forma de interacción no sería adoptado fácilmente, para resolver el problema los diseñadores y programadores concluyeron incluir a las CLI dentro de los entornos de trabajo para tareas muy especializadas, de la misma forma las NUI pretenden incluir a las GUI y CLI en su entorno como entradas para procesos muy específicos y siendo las NUI el formato de entrada preferido para casos generales.

El salto de tecnología entre CLI y GUI vino dado por los nuevos dispositivos de entrada gráficos, encabezados por el mouse, lo que permitía la exploración de regiones en la pantalla de una forma simple, al menos mucho más simple que el recorrido matricial y rígido manejado por las CLI. Podemos pensar que el salto entre GUI y NUI se debe también a la intrusión de los nuevos dispositivos de entrada sensoriales como pantallas múlti-táctiles y sensores de movimiento [Izadi et al., 2008], pero se trata de algo más, las NUI representan una nueva forma de pensamiento sobre como interactuamos con los dispositivos informáticos, se centran en la idea de cómo los seres humanos aprenden e interactúan con su ambiente.

Figura 2.5: Evolución histórica de las interfaces de usuario y su factor característico.

2.3.1 Definición

A pesar de que hemos hablado lo suficiente de las NUI como para que el lector creé una definición para sí mismo, es necesario dar formalidad con una definición objetiva.

Una interfaz de usuario natural o NUI (por sus siglas en inglés Natural User Interface) es una interfaz de usuario diseñada para reutilizar las habilidades existentes en los seres humanos de interactuar con el ambiente y enfocarlas a la interacción directa con el contenido informático [K.W. Bollhoefer and Witzsche, 2009].

Para una mayor comprensión de esta definición será preciso analizarla por partes. Decimos que es diseñada no porque implique un trabajo artístico de por medio (aunque es posible que lo haya) sino porque requiere de esfuerzos de planificación y previsión específicos para asegurar que las interacciones con las NUI son apropiadas para el contenido, el contexto y el usuario.

La reutilización de habilidades es el punto de mayor relevancia en la definición, se refiere a que las interfaces deben convivir con los usuarios adaptándose a ellos y a las habilidades de comunicación verbal y no verbal humanas que se interpretan como comandos de entrada, de esta forma los usuarios hacen uso de las habilidades que ya poseen y no es necesario un complejo tutorial o manual de usuario sobre cómo utilizar el sistema.

Por último la interacción directa con el contenido refiere el uso de objetos representando entidades informáticas [K.W. Bollhoefer and Witzsche, 2009], lo que lleva al usuario a convivir directamente con la información. Esto no quiere decir que la interfaz no pueda poseer los controles comunes de una GUI como botones o casillas de verificación cuando sea necesario, sólo que estos deben quedar en segundo término y la manipulación directa de los objetos debe ser la principal forma de interacción.

2.3.2 La naturalidad de las interfaces

Las NUI adquieren su carácter de "naturales"debido a su capacidad de ser intuitivas [Petersen and Stricker, 2009], ciertamente esto todavía es muy ambiguo pues sólo hemos pasado de una palabra a otra, en referencia a Bill Buxton, uno de los grandes expertos en la tecnología de las NUI una interface es natural si explota las habilidades que hemos adquirido a través de nuestra vida [7]. Interpretando estas palabras decimos que una interface es natural si aprovecha las capacidades de bajo nivel del ser humano (hablar, gesticular, reaccionar) más las habilidades aprendidas que hemos desarrollado a través de la interacción con nuestro propio entorno en la vida cotidiana.

Una habilidad se trata de una dinámica que una vez aprendida se convierte en parte de nosotros y es fácil de repetir siempre y cuando mantengamos en uso la habilidad. Un conjunto de habilidades nos llevarán a ejercer una tarea (que es el fin de la interacción con las máquinas), definiéndose en nuestro contexto como una unidad de trabajo que requiere una acción por parte del usuario y entregará resultados específicos. Para entender mejor por qué hacemos hincapié en las habilidades humanas clasificaremos las habilidades en dos grupos.

Las habilidades simples o de bajo nivel son en general las habilidades comunes a todo ser humano y se consideran casi innatas aunque algunas de ellas son parte de los primeros procesos de aprendizaje. Son fáciles de aprender, tienen una baja carga cognitiva y pueden ser reutilizadas y adaptadas para muchas tareas sin gran esfuerzo.

Las habilidades compuestas son aquellas que forzosamente se aprenden para un uso específico, se componen de habilidades simples o de otras habilidades compuestas, en general toma más tiempo y esfuerzo dominarlas, tienen una mayor carga cognitiva y son especializadas.

Ahora analicemos el comportamiento de las GUI, no nacemos aprendiendo cómo usar una computadora, manejar el ratón o el teclado, por el contrario es un proceso de aprendizaje constante y que es considerada una habilidad compleja, lo cual quiere decir que nuestro cerebro ocupa memoria de trabajo por el solo hecho de interactuar con una computadora. Las NUI por otro lado tratan de usar las habilidades simples como entrada a la interfaz, de esta forma manejar una computadora se convierte en un proceso sencillo que no requiere gran tiempo o esfuerzo de aprendizaje, lo que lleva al cerebro humano a desocupar memoria de trabajo centrada sólo en la interacción y poder concretarse en el trabajo desempeñado.

[7] http://channel9.msdn.com/posts/library/Larsen/CES-2010-NUI-whith-Bill-Buxton

2.3.3 Características

Hemos analizado los antecedentes, definición y origen de las NUI, para terminar de hablar de ellas citemos las características que las definen como tales y que nos permiten identificarlas y diseñarlas. La siguiente es una lista de privativas de las NUI.

- Experiencia inmediata. Los usuarios no necesitan aprender algo nuevo sólo valerse de las habilidades simples que ya dominan y aplicarlas en una nueva situación, al usar las habilidades simples comunes a los seres humanos, los usuarios pueden actualizarse rápidamente a cualquier programa.
- Especialidad. Las habilidades pueden explotarse para un grupo específico, por ejemplo si nuestro software está diseñado para arquitectos podemos explotar sus habilidades en la lectura de planos.
- Carga cognitiva. El uso de habilidades simples permite al usuario concentrarse en la labor ejecutada y no en la forma de comunicación de datos, lo que permite a los sistemas ser fáciles de usar y de aprender.
- Aprendizaje progresivo. Incluso en procesos complicados se debe llevar una curva suave de aprendizaje, para tareas complejas la mejor forma de abordarlas es descomponerlas en subtareas que hagan uso de las habilidades simples.
- Interacción directa. El diseño de las interfaces es creado para manejar entidades informáticas como objetos sensitivos. Nuestra interacción con el mundo real tiene estas cualidades por lo que esto dará lugar a interfaces que se sientan más fluidas y naturales.

De la forma de interacción directa se desprenden nuevas características sobre la respuesta del sistema a los impulsos del usuario.

- Proximidad espacial. La proximidad física a un elemento informático que no tiene forma real provoca una sensación de familiaridad con él.
- Proximidad temporal. La interfaz reacciona al mismo tiempo que la acción del usuario.
- Asignación concurrente. Hay una relación entre cada grado de libertad de la acción del usuario y cada grado de libertad de la reacción de la interface.

El conjunto de estas características crea el efecto de sencillez, familiaridad y fluidez, siendo no sólo fácil manipular los elementos de un sistema, el usuario comprende de forma natural el proceso que se lleva a cabo y las limitantes a las que se afronta del mismo modo que comprendemos el ambiente que nos rodea en una forma no del todo profunda pero lo suficiente para darnos cuenta de lo que sucede.

2.4 Sexto Sentido

En estos momentos seguramente el lector ya ha comprendido la importancia de las NUI no como tecnología emergente sino como el proceso bien establecido de evolución de las interfaces de usuario. Las NUI han venido a desarrollarse en un momento en el que el cómputo se encuentra cambiando y ya no podemos usar el término "computadoras" como una generalización de los aparatos con poder de cómputo pues han surgido gran cantidad de máquinas con capacidades similares como teléfonos celulares, tablets, consolas de videojuegos, entre otras [Petersen and Stricker, 2009], por ello sería mejor referirnos a este conjunto como dispositivos informáticos. Si bien muchos de estos dispositivo aún se encuentran atados a una superficie como en el caso de las computadoras y consolas de videojuegos, la tendencia actual es despegar el cómputo de su estado estacionario y convertirlo en cómputo portable como con los teléfonos y tablets.

Un punto en contra de las NUI es el poder de cómputo necesario para que las interfaces sean fluidas. En las PC´s actuales esto no es un problema pero como hemos mencionado la tendencia no es al desarrollo de PC´s si no al de dispositivos móviles, los cuales son conocidos por sus características limitadas, aquí es donde entra en juego un nueva rama en interfaces conocida como sexto sentido.

2.4.1 Definición

En su versión original dictaminada por Pranav Mistry [Mistry and Maes, 2009] la tecnología de sexto sentido es una interface gestual portátil que aumenta el mundo físico que nos rodea mediante proyecciones digitales y nos permite usar gestos naturales para interactuar con la información.

Visto desde el punto de vista de las NUI podemos darnos cuenta que la tecnología de sexto sentido se trata de una rama de las NUI que se centra en interfaces para dispositivos portátiles que hace uso de la realidad aumentada para la convivencia con los entes informáticos.

Las limitantes y características de un dispositivo móvil impiden que una NUI diseñada para una PC u otro dispositivo de mayores capacidades pueda adaptarse con facilidad al cómputo portátil, tampoco podemos usar periféricos convencionales aunque los resultados deben ser similares, de aquí la importancia de esta nueva tecnología pues se enfoca en un campo no explorado que ofrece grandes posibilidades.

2.4.2 Carcterísticas

Como rama de las NUI la tecnología de sexto sentido hereda todas las características vistas en la sección anterior y agrega nuevos elementos como lo son:

- Ubicuidad. Permite al usuario acceder a un sistema computacional sin importar la ubicación del individuo.
- Realidad aumentada. En comparación con las NUI tradicionalmente definidas que muchas veces hacen uso de grandes dispositivos de reconocimiento táctil, sexto sentido aumenta la realidad con proyecciones con las cuales se puede interactuar, eliminando la necesidad de grandes dispositivos periféricos.
- Movilidad. El hardware usado es lo suficientemente portable para que el usuario pueda transportarlo con facilidad, de forma ergonómica y casi invisible al usuario.

Sexto sentido permite a los usuarios convivir en una forma muy natural con los dispositivos computacionales, tal cual lo hacemos con los objetos que rodean nuestro medio ambiente [Pranav Mistry, 2009], con la diferencia que podemos llevar un gran número de objetos e información en nuestro dispositivo portátil preferido y acceder y convivir con ellos cuando lo necesitemos.

2.5 Trabajo Relacionado

Muchos han sido los intentos por crear NUIs. Algunos de ellos han tenido buenos resultados pero usando infraestructura especializada, físicamente espaciosa y que apenas puede moverse una vez instalada. Los propios usuarios se han encargado de hacernos ver la necesidad de miniaturizar estos sistemas para su uso doméstico y de llevar esta tecnología al terreno móvil. Realmente son pocos y recientes los trabajos cuyo objetivo es la implementación de NUIs basadas en gesticulaciones que no usen TouchScreen y que emulen a los sistemas de hardware superior como el Microsoft Surface. El equipo de desarrollo de EyeSight[8] y algunos trabajos como el reportado en [Siltanen and Hyväkkä, 2006] han realizado primeras aproximaciones que no han trascendido, pues apenas reconocen un solo tipo de gesto en un ambiente controlado.

En los siguientes párrafos describiremos algunos de los proyectos más importantes relacionados con NUIs y terminaremos con el proyecto WUW de tecnología de sexto sentido.

[8] www.eyesight-tech.com/

Figura 2.6: Microsoft Surface permite la interacción multi-touch entre objetos y usuarios al mismo tiempo.

2.5.1 Microsoft Surface

Una iniciativa de Microsoft Research que de inmediato paso a comercializarse, aparece contemporáneamente con Pixel Perceptive y con características similares. Se trata de una mesa interactiva multi-touch y multi-usuario que responde a gestos y a objetos del mundo real (ver figura 2.6), los cuales identifica, mostrando información sobre ellos o extrayendo la información que almacenan en caso de tratarse de un dispositivo con memoria [K.W. Bollhoefer and Witzsche, 2009]. Ayuda a la gente a interactuar con contenido digital de forma sencilla e intuitiva. Surface utiliza cámaras y reconocimiento de imágenes en el espectro infrarrojo para reconocer diferentes tipos de objetos como son los dedos humanos. Se toma un conjunto de imágenes que son interpretadas como un gesto que puede manipular el contenido digital al alcance de la mano del usuario. De manera más detallada el algoritmo de reconocimiento de gesticulaciones usado en Suface es expuesto en [Chan et al., 2007]. Debido a su popularidad paso de ser un producto comercial pre-programado a tener un *framework* de desarrollo para que las empresas puedan adaptar Surface a sus necesidades específicas.

A pesar de ser un dispositivo comercial tiene desventajas muy importantes:

- el alto costo que alcanza ($ 12,500.00 USD) se debe al uso de hardware especializado como cámaras de infrarrojo, una mesa de difusión y un ordenador específico;

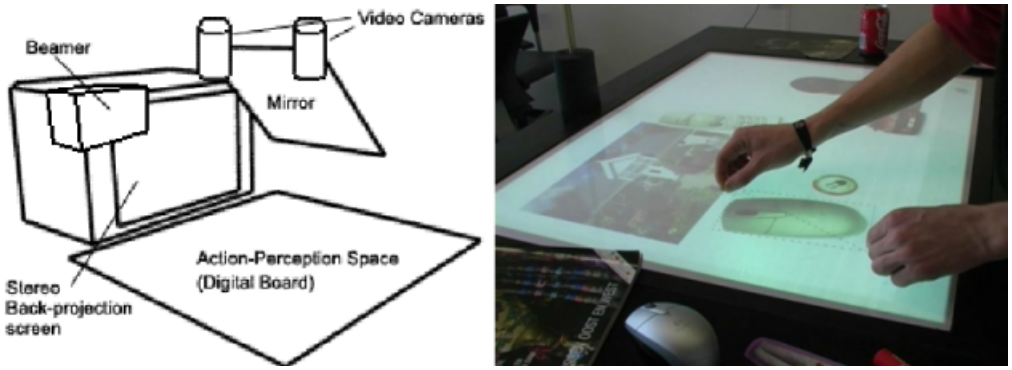

Figura 2.7: Estructura del prototipo "Physic Paper"

- la mesa es fija y no ha sido diseñada para moverse una vez instalada y
- los objetos pueden ser confundidos si sus propiedades son parecidas.

Recientemente Microsoft ha presentado un nuevo prototipo experimental conocido como LightSpace[9], como una evolución de Surface permitiendo movimientos no sólo en la mesa sino también en el aire, siendo aún necesario un ambiente controlado.

2.5.2 Physic Paper

El Physic Paper proyecta sobre una mesa una serie de elementos digitales y monitorea las acciones del usuario sobre la proyección mediante camaras montadas al frente, los movimientos de la mano son interpretados y procesados como símbolos o acciones [Aliakseyeu and bernard Martens, 2001], la estructura del sistema puede ser apreciada en la figura 2.7. El objetivo principal del prototipo es recrear una hoja de papel, la libertad de movimientos que esto implicaría crea importantes problemas por lo que sus creadores limitarón el proyecto a una serie de movimientos predefinidos que deben ser aprendidos por el usuario.

El proyecto tiene alcance limitado como podemos constatar por las siguientes afirmaciones.

- El prototipo es experimental y no se le dio continuidad.
- La infraestructura requerida es dimensionalmente grande y estructuralmente compleja.
- Económica y computacionalmente costoso.
- Alcance limitado en la forma y número de gesticulaciones posibles.

[9] http://research.microsoft.com/en-us/projects/lightspace/

Figura 2.8: El usuario puede escuchar un catalogo de música tal cual si los mismos objetos lo rodearan.

2.5.3 Music Scope Headphones

Demostrando que las interfaces naturales de usuario se deben basar más en las gesticulaciones que en los algoritmos de visión, este proyecto crea auriculares que detectan movimientos naturales de la cabeza y las manos mediante sensores de proximidad y de inclinación [Hamanaka, 2006]. Ofrece a los usuarios un catálogo de música que pueden escuchar como si se tratara de una serie de bocinas que nos rodean (ver figura 2.8); cada canción es reproducida por una sola bocina virtual, de igual forma que sucede en la realidad, escuchamos con mayor intensidad la que se encuentra al frente y con volúmen tenue a las demás reproducciones. Music Scope Headphones permite centrarnos en melodías específicas girando la cabeza, ampliar los menús al inclinarla o cambiar la canción moviendo la cabeza de un lado al otro rápidamente. Al acercar las manos a los audífonos (como cuando nos tapamos los oídos) nos centraremos en una melodía en particular silenciando las canciones restantes. Otro punto interesante es que toda la interfaz se basa meramente en elementos auditivos (i.e., no hay una pantalla o elemento visual alguno).

Los resultados de esta investigación resaltan inconvenientes:

- los usuarios no se sienten cómodos del todo sin una interfaz visual,
- las búsquedas basadas en sonidos no son tan rápidas como las visuales de listas ordenadas y

- el número de gesticulaciones es muy limitado en comparación con otros sistemas.

2.5.4 Kinect

Originalmente desarrollado como complemento para una consola de videojuegos (Xbox 360), Kinect puede ser considerado la NUI más conocida y aceptada comercialmente. Desarrollada por el Dr. Alex Kipman y su equipo, el Kinect permite una interacción completamente natural olvidándose de controles o cables pues toda la interfaz se basa en gestos, movimientos y comandos de voz [10] introducidos por el usuario mediante un hardware especialmente diseñado como una barra horizontal que contiene una cámara RGB, un sensor de profundidad y un micrófono de múltiples matrices, aunado a ello cuenta con su propio procesador con diseño exclusivo para tareas de visión y reconocimiento de voz. Kinect identifica los movimientos mediante algoritmos de visión por computadora los interpreta como un gesto y genera una salida indicando la posición de manos pies y torso del usuario lo que permite a un videojuego o programa decidir qué acción tomar frente a la gesticulación del usuario, además es capaz de reconocer ordenes de voz y los usuario que las producen, estas características hacen de Kinect la NUI por excelencia en el mercado. La figura 2.9 nos permite observar mejor como es la interacción con el dispositivo Kinect y como las acciones de los usuarios tienen una correspondencia en pantalla.

Pese a su gran éxito Kinect tiene aún una lista de errores que no han permitido explotar al máximo sus capacidades:

- Soporta únicamente dos colaboradores al mismo tiempo lo cual reduce su capacidad para ambientes multi-usuario.
- Es afectado por condiciones de luz anómalas como un día en exceso soleado o la falta de luz en los últimos momentos de la tarde.
- La distancia de interacción es poco flexible, los usuarios deben acomodarse a una distancia predeterminada para que sus movimientos se reconozcan con agilidad y precisión.
- El control por voz aún tiene un margen de error muy grande y es necesario calibrarlo a las condiciones ambientales.

[10] http://www.xbox.com/es-ES/Xbox360/Accessories/kinect/Home

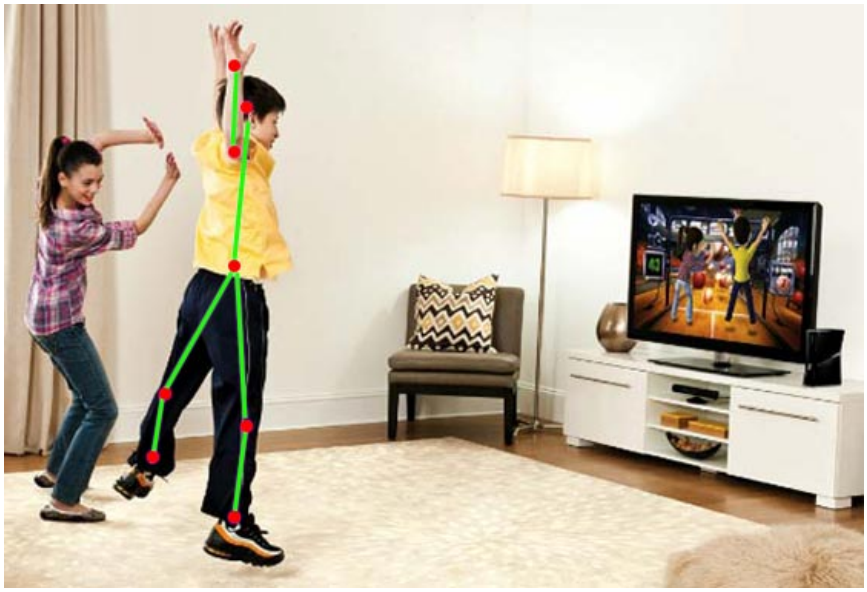

Figura 2.9: La estructura del individuo se muestra como un conjunto de líneas en verde, la pantalla muestra cómo el videojuego interpreta esta estructura y la refleja en un personaje virtual.

Debido al gran impacto que causo en el mercado, rápidamente se decidió crear un SDK para el desarrollo que habrá de tener gran impacto en la próxima generación de sistemas operativos pues Microsoft ha anunciado la compatibilidad de Windows 8 con este dispositivo en el TechForum 2011 [11].

2.5.5 WUW – Wear Ur World

Proyecto del MIT Media Laboratory [Pranav Mistry, 2009] que establece el concepto de sexto sentido, aumenta las regiones físicas que el usuario ve mediante proyecciones, detecta las gesticulaciones de las manos de él o los usuarios, las interpreta como formas de interacción y ejecuta una acción única correspondiente al ademán. El prototipo se compone de una pequeña cámara y un micro-proyector montados en una medalla al cuello, una laptop encargada del procesamiento en una mochila a la espalda a la que se conecta el hardware anterior y marcas de colores en los dedos del usuario (ver figura 2.10). El reconocimiento de los gestos se lleva a cabo por algoritmos de visión que discretizan las marcas de colores. Las imágenes obtenidas por la cámara se procesan en la laptop que envía una señal al proyector, dando la sensación de interacción en tiempo real con la información digital exhibida en objetos del ambiente común.

Las principales limitantes del prototipo son:

[11] http://bitelia.com/2011/02/el-futuro-de-windows

Figura 2.10: WUW se compone de un pico-proyector, una cámara y una unidad de procesamiento, podemos ver la interacción con las proyecciones.

- necesidad de marcas de colores para cada usuario que quiera ejercer influencia gesticular,
- uso de varios periféricos independientes con los que el usuario debe cargar,
- exigencia de una laptop en la espalda para el procesamiento que debe permanecer encendida en todo momento e
- implementación de algoritmos de visión que pueden confundir las marcas de colores con otros elementos del ambiente.

Capítulo 3

Arquitectura

El presente trabajo es un sistema complejo de visión por computadora que engloba diferentes algoritmos y procedimientos para llegar a un único fin: "crear una nueva experiencia en interfaces de usuario". Si bien el programa final es un encapsulado único que contiene todos los procedimientos necesarios para captar los movimientos, interpretarlos como gesticulaciones, así como dar una respuesta física, dentro de este encapsulado se encuentran módulos bien establecidos y enfocados a actividades específicas. Esta forma de repartir el trabajo resulta altamente eficiente, pues se ha estudiado el problema para seccionar el trabajo de manera tal que el flujo de información sea lo más rápido posible, sin permitir la duplicidad de información o la generación de datos inútiles en cualquier momento.

Este capítulo comienza explicando la arquitectura física del sistema, i.e., los elementos de hardware que dan composición real a la tecnología de sexto sentido desarrollada. El flujo de información va de lo real a lo virtual donde se modifica y regresa al campo real. Resulta entonces necesario explicar la conexión existente entre los diferentes elementos del hardware con los módulos del software, para así comprender el sistema como un único elemento y como un conjunto de bloques conectados.

Seguidamente explicaremos las partes que conforman el software y la relación existente entre ellas, sin llegar a ser específicos en un apartado pues la algoritmia y cánones de cada segmento del sistema serán explicadas más adelante en un capítulo propio. Se pretende, por lo tanto, durante el desarrollo de este capítulo, explicar el flujo de información en todo el sistema. De esta forma cuando el lector se enfoque en capítulos posteriores tendrá una idea general, simple y concreta del objetivo del módulo y también entenderá de donde y hacia dónde se dirige la información. La comprensión de las entradas y las salidas le ayudará a entender porque los datos son presentados y tratados de alguna manera en particular y le será más sencillo asimilar

las optimizaciones realizadas a algunos procesos.

3.1 Arquitectura del hardware

Según lo explica [Pranav Mistry, 2009] la parte física de un sistema de tecnología de sexto sentido se compone de una unidad de procesamiento, una cámara que permita la entrada de datos del mundo real y un proyector que permita la salida de la información hacia el ambiente humano. De hecho, no sólo los sistemas de sexto sentido se componen de esta forma, son muchos los prototipos de NUI desarrollados bajo esta estructura física [Leith K. Y. Chan, 2011] [Harrison et al., 2011]. Esto se debe a que sin extender el tamaño del sistema se incluyen los elementos necesarios para manejar algoritmos de visión por computadora, todos ellos embebidos en un sólo aparato. Esta estructura permite también cambiar los dispositivos que componen el aparato completo por otras piezas comerciales de carácteristicas similares. Este último punto permite la reproducción de prototipos sin tener piezas exactas, los cuales se pueden reconstruir a partir de modelos similares disponibles en el mercado. La constitución física del dispositivo desarrollado se ha pensado para seguir con la característica mencionada anteriormente, i.e., pretendemos que sea fácilmente reproducible y que haga uso de tecnología comercialmente disponible, ya que si se desea implementar como una aplicación en la industria o investigación no sería lógico pensar que los mismos elementos del hardware que hemos seleccionado estarán disponibles en otros lugares o en fechas posteriores.

La idea general del sistema es obtener imágenes a través de una cámara fotográfica, luego procesarlas e interpretarlas como un comando y después generar una proyección en una superficie adyacente al usuario. Al tratarse de un sistema que pretende tener la característica de movilidad, resulta obvio pensar que los elementos de hardware deben ser lo suficientemente pequeños como para ser transportados por un individuo cualquiera en situaciones normales, sin que afecte su locomoción o le resulte incómodo. Por tal motivo, hemos escogido un teléfono celular que sirva para el procesamiento y además nos ayude en la entrada de datos pues la mayoría de estos dispositivos en la actualidad cuentan con una cámara fotográfica incorporada, mientras que para la salida de datos se ha escogido un pico-proyector, que no es otra cosa sino un proyector miniaturizado. La figura 3.1 demuestra en forma gráfica la estructura física del sistema.

3.1.1 Entrada física

Como se menciona anteriormente la entrada de la información del mundo real se hará usando la cámara fotográfica incorporada al teléfono celular. En el capítulo 4

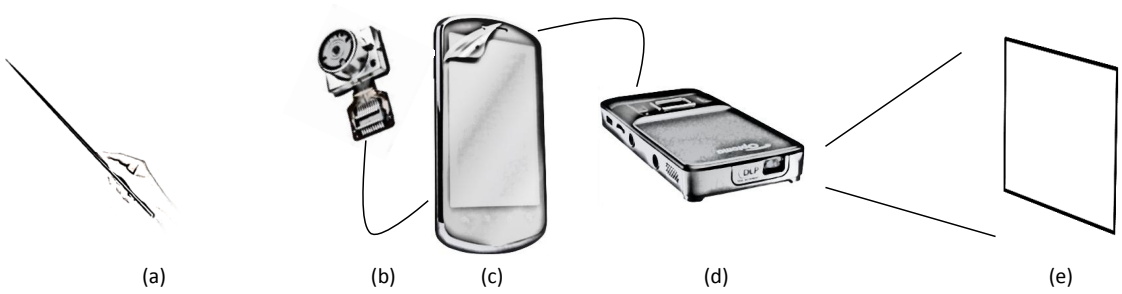

Figura 3.1: Constitución en hardware de la tecnología de sexto sentido desarrollada. (a) Presenta el stylus, (b) simboliza la cámara aunque esta se encuentra empotrada en el móvil, (c) Muestra la unidad de procesamiento, en este caso un Smartphone, (d) representa un pico-proyector, (e) simboliza la proyección

se encontraran las especificaciones tanto de la cámara como del pico-proyector y las justificaciones necesarias para explicar porque se escoge un modelo sobre otro.

La idea principal de este módulo es captar información constantemente del mundo real, casi como si se tratara de una secuencia de video, encontraremos más adelante que de hecho podríamos tomar 24 fotografías por segundo pero dadas las características del sistema, no es necesario pues sólo identificará movimientos básicos, dejando de lado la gama de complejos movimientos que puede crear la mano humana. La percepción se centra en un stylus de punta luminosa, que es nuestro objeto de estudio. Los movimientos y gesticulaciones realizados con este stylus son los que indican al sistema cómo reaccionar.

La cámara entrega la información en un estado puro, i.e., tal cual el chip la percibe, al software el cual debe encargarse de recuperar los datos y transformarlos a un formato que pueda ser comprendido por la computadora. La cámara utilizada es un chip fotográfico Omnivisión OV5633 [Omn, 2008], de 5 MPx del cual más adelante se hablará en detalle. La selección de este modelo en particular ha sido una decisión integrada a la elección del celular que se encargará del procesamiento, ya que uno implica el otro al encontrarse en un sólo dispositivo. En este caso, la cámara es de gran calidad y aunque no aprovecharemos todas las características de nitidez, si nos valdremos del alto número de FPS (*Frames Per Second* o Fotografías Por Segundo) que es capaz de desarrollar. En algunos otros prototipos desarrollados esto ha sido un cuello de botella importante, ya que la mayoría de los dispositivos de la generación anterior eran incapaces de alcanzar más de 10 FPS en un tamaño QVGA (320 x 240 y menores) [Siltanen and Hyväkkä, 2006]. No se podía si quiera pensar en hacer visión por computadora sobre móviles a una mejor calidad. Sin embargo, esta nueva generación de chips nos permite alcanzar un mayor número de FPS en una mejor calidad, principal característica que se ha tenido en cuenta para escoger este modelo.

3.1.2 Procesamiento

Para hacer visión por computadora en tiempo real es necesario un gran procesamiento. Hasta hace poco las características de los teléfonos móviles podían considerarse como escasas, insuficientes y totalmente faltas para este tópico. El rápido crecimiento en el campo de los móviles, consecuencia del auge en el mercado por la alta demanda de teléfonos cada vez más poderosos, propició una generación de teléfonos celulares conocidos como "smartphone" que ya no sólo se presentaban como teléfonos, sino también como unidades de trabajo móviles, destacadas por su gran capacidad de procesamiento y multimedia, por encima de los 650 MHz y su potencial de reproducir y grabar videos de alta calidad.

En el capítulo 5 se explican las capacidades de procesamiento necesarias para esta actividad, que es la más pesada en términos de procesamiento, pues existen actividades como la toma de fotografías que, aunque pueden ser tratadas rápidamente, el hardware tiene un tiempo de respuesta intrínseco entre una y otra. Adelantaremos el precepto de que sería deseable un procesador capaz de manejar por lo menos de 3 a 4 millones de operaciones por segundo como un estado mínimo y de 7 a 8 millones por segundo como un estado satisfactorio. Puede que estas necesidades se vean insignificantes si el procesamiento se llevará a cabo por una laptop o una computadora de escritorio, pero para un teléfono celular resultan muy superiores. Cuando se escogió el teléfono móvil que sería usado como unidad de procesamiento eran pocos los dispositivos de última generación que permitían acercarnos a estas velocidades. De entre ellos se escogió el modelo Samsung Galaxy S o I9000 que es su nombre técnico (ver figura 3.2) por su procesador ARM Cortex A8 Hummingbird de 1GHz [Cor, 2010] y su GPU PowerVR SGX540 [Pow, 2010] que tienen la suficiente capacidad para hacer frente a algunos algoritmos de visión por computadora.

Este smartphone cuenta con un sistema operativo Android 2.1 que durante los experimentos era capaz de desarrollar hasta 8 millones de operaciones por segundo. En una actualización hecha al sistema a la versión 2.2 se lograron hasta 32 millones de operaciones por segundo. Esto se explica debido al hecho de que la actualización consiste en una mejora considerable a la máquina virtual que interpreta los programas que corren bajo esta plataforma, eliminando muchos de los procesos intermedios. Trabajando con esta versión, el programa puede correr con mayor libertad y se evita calentar no sólo el procesador sino todo el dispositivo.

La memoria es otro elemento crítico. En el caso del Galaxy S es de 512 MB, de los cuales alrededor de 160 MB son ocupados por el sistema operativo. Es necesario tratar con grandes volúmenes de información conformados por secuencias fotográficas, por lo que quedarnos sin memoria supondría que el sistema colapsaría. El programa

Figura 3.2: Samsung Galaxy S I9000

requiere de varios arreglos que serán de un tamaño idéntico al de las imágenes tratadas. En particular el programa usa imágenes de 640 x 480 px, lo que daría como resultado arreglos de 307,200 elementos. Si bien la programación ha sido llevada a cabo de forma que el manejo de la memoria sea óptimo, la holgura que nos da una memoria de 512 MB en un dispositivo móvil permite que el sistema sea escalable y haga uso de recursos y bibliotecas de gran capacidad.

Las características presentadas hasta el momento seguramente serían encontradas en una pequeña gama de smartphones, pero una característica en específico, la salida de video, nos lleva a tomar la decisión de escoger el Galaxy S. Este es un concepto recientemente introducido en la telefonía celular comercial, que permite conectar de forma sencilla el teléfono a una salida de video como podría ser un monitor, televisión o proyector. Si pensamos en la etapa de la salida física, el tener una salida de video incluida representa una característica deseable, ya que permite enfocarnos en el desarrollo del software sin preocuparnos por desarrollar un esquema especial para la salida de video.

Las especificaciones hasta ahora mencionadas se consideraban de élite hasta 2010. Al escribir esta tesis resultó grato saber que estas características comienzan a ser de uso cotidiano, lo que permitirá llevar la tecnología de sexto sentido a una mayor

Figura 3.3: Pico-proyector Optoma PK-201

cantidad de dispositivos.

3.1.3 Salida física

Compuesta por el pico-proyector Optoma PK-201 [Opt, 2011], que se puede apreciar en la figura 3.3, la salida física representa la capacidad del prototipo de interactuar con el individuo en forma de realidad aumentada, i.e., las respuestas del software no se limitan a una pantalla, siempre tienen una repercusión física en el ambiente que rodea al usuario al crear proyecciones a su alrededor como respuesta a los gestos de la mano. Este pico-proyector es casi del mismo tamaño que el Galaxy S, lo que ayuda en la movilidad del prototipo. En el capítulo 4 se expresan mejor las justificaciones sobre porque escoger este modelo en particular, por el momento es suficiente con saber que el uso de pico-proyectores en la actualidad es una tendencia común en el desarrollo de NUI [Willis et al., 2011] [Shilkrot et al., 2011].

El modelo PK-201 alcanza hasta 15 lúmenes de potencia luminosa, lo que permite distinguir las imágenes en la mayoría de las condiciones de iluminación, aunque se difumina en condiciones de luz muy brillante. Esta desventaja no es propia de este modelo sino de cualquier pico-proyector y de la mayoría de los proyectores en general. Entre las características que este modelo presenta se distingue una resolución de 848 x 640 px, misma calidad de salida que tiene el Galaxy S, lo que permite percibir los objetos en verdadera magnitud, sin preocuparnos por el hecho de que la proyección deforme los objetos virtuales.

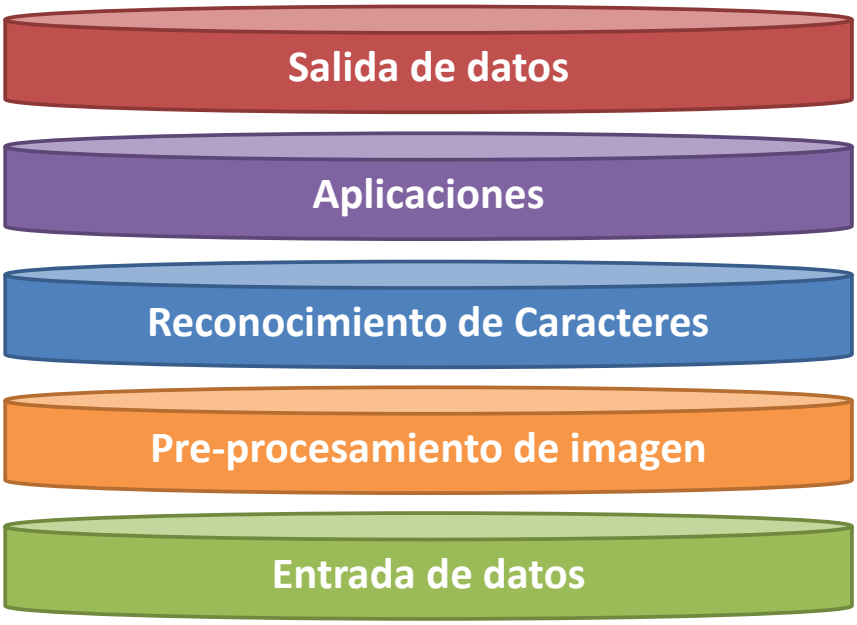

Figura 3.4: Arquitectura por capas del software

3.2 Arquitectura del software

Para administrar el software, este se ha dividido en actividades específicas. Cada bloque que compone el software (mostrados gráficamente en la figura 3.4) se encarga de una actividad en particular, permitiendo que el sistema sea comprensible, modular y coherente. No debemos verlo como una banda de ensamblaje con puntos de control forzosos, de hecho gran parte de las optimizaciones del sistema consisten en evitar módulos que no son necesarios para ese flujo de información en particular. Si bien todos los bloques conforman el sistema, sólo en los flujos de información donde exista una entrada positiva (al existir movimiento del stylus) se ejecutarán todos los bloques, de otra forma omitiremos de algunos de ellos.

3.2.1 Entrada de datos

Después de que la cámara ha capturado las imágenes, estas son entregadas al software para que analice la información y extraiga los datos útiles. No obstante, el chip Omnivision OV5633 entrega un formato de imágen conocido como YUV420SP, que es explicado más a detalle en el capítulo 4, lo que nos imposibilita tratarlo directamente. La tarea de este módulo consiste entonces en transformar la información entregada por la cámara a un formato comprensible para el resto del software. Sin embargo, la conversión de cada imágen puede llegar a ocupar tanto poder de cómputo como el requerido por el resto del programa. Por ello, la etapa de entrada de datos debe ser capaz de hacer la conversión en una forma altamente eficiente, sin perjudicar la

eficiencia del sistema, convirtiendo sólo aquella información que sea relevante.

3.2.2 Pre-procesamiento de imágenes

Una vez que la información de la cámara es legible, entonces necesitamos discretizar los elementos de interés dentro de la imágen si es que estos existen. De esta manera, se define la función del módulo de pre-procesamiento de imágenes, el cual recibe como entrada una imágen y determina un único punto como posición del stylus en esa imágen. El stylus ha sido diseñado pensando en que sea fácilmente identificable, no obstante los algoritmos de reconocimiento de formas requieren un análisis completo de la imágen para operar. Es un hecho que este es el bloque que más recursos computacionales utiliza, ya que no sólo analiza la imágen para encontrar una posición, sino que lo hace varias veces por segundo sin detenerse mientras el programa esté en ejecución, salvo que alguna aplicación lanzada por el programa principal así lo pida.

Posteriormente el capítulo 5 nos explicará en forma clara y detallada cómo transcurre el flujo de información durante esta etapa. De momento nos basta con saber que, aunque el stylus presente colores y formas fáciles de identificar, el algoritmo utilizado es lo suficientemente robusto como para diferenciar entre el stylus y otros objetos parecidos. Se usan filtros sobre el color, la forma y las posiciones relativas que permiten clasificar todos los objetos en la fotografía que cumplan ciertas características. Cada filtro obtiene una serie de candidatos que van siendo descartados hasta tener el objeto con mejores probabilidades de tratarse del stylus. Los procesos deben ser lo suficientemente simples para no absorber el cómputo del procesador y lo suficientemente robustos para no permitirse errores al confundir el stylus con otros elementos de la imágen.

3.2.3 Reconocimiento de carácteres

Los puntos que son entregados por la etapa de pre-procesamiento se convierten en vectores al obtener la diferencia de un punto a otro en el espacio. Un conjunto de vectores bidimensionales representan un trazo o lo que es lo mismo una gesticulación producida con el stylus. Al tratarse de movimientos sobre el espacio bidimensional, lo podemos pensar entonces como dibujos en un lienzo, dibujos simples que carecen de relleno. En consecuencia, tenemos la idea de que el stylus se comporta como un pincel que dibuja una línea en un espacio bidimensional, lo que permite identificar una forma no como un dibujo concreto, sino que la podemos simplificar a la estructura de un carácter. el reconocimiento de movimientos como si se tratara de caracteres ayuda a simplificar el proceso de identificación de gestos. de hecho no tenemos que tratar con formas complejas, sino que expresamos un gesto como el conjunto de sus vectores básicos.

Los vectores son analizados en tiempo real, aproximando el dibujo cada vez más a una letra particular. Los algoritmos que presenta el capítulo 6 reciben uno a uno los vectores, los cuales comienzan a emparejarse con los vectores de letras conocidas por el sistema, que se encuentran almacenados en una base de símbolos la cual organiza el modelo de escritura de cada usuario. El emparejamiento provoca errores o aciertos, la diferencia entre ambos dará un número que de tornarse negativo indicaría un parecido muy pobre con la letra de la base de símbolos, mientras una tendencia positiva indicaría un parecido notorio. La letra con un número mayor por lo tanto será la que más parecido tenga con el trazo dibujado con el stylus.

La capacidad del sistema de reconocer los trazos de un usuario en particular se debe a que, siendo una tecnología diseñada para dispositivos móviles, se espera que sólo un usuario o un reducido grupo de ellos sea quien use el teléfono celular. De esta manera resulta innecesario hacer un extenso reconocimiento por parte del algoritmo como muchos otros que se encargan de reconocer los caracteres independientemente del escriba, pero que requieren una capacidad de procesamiento mucho mayor.

La salida de la etapa de reconocimiento de carácteres es el carácter de la base de símbolos con mayor parecido al trazo. Si bien un trazo idéntico sería difícil de encontrar, el algoritmo está pensado no para tratar de encontrar la letra buscada desde el inicio, lo que podría equivaler a buscar una aguja en un pajar. Una mejor aproximación resulta al descartar rápidamente todas las letras que presentan poco parecido al cabo de cierto número de vectores y así quedarse cada vez con un número menor de candidatos, de los cuales sobresaldrá la letra buscada. Esta forma de descartar letras es como tener un filtro para toda la paja y dejar sólo los objetos más parecidos a la aguja, permitiéndonos diferenciar la aguja más fácilmente.

3.2.4 Aplicaciones

Ahora que conocemos el carácter que representa al trazo dibujado debemos decidir qué hacer con él. Una vez que se ha determinado el carácter adecuado el programa principal se encarga de emitir una respuesta a partir de una serie de aplicaciones de prueba que han sido pensadas para que el usuario sienta la inmersión en el sistema. Algunas de ellas se encuentran basadas en el trabajo de Pranav Mistry [Pranav Mistry, 2009] que es la propuesta original del paradigma de sexto sentido.

Las aplicaciones desarrolladas han sido diseñadas para demostrar la capacidad del sistema en diversos ámbitos multimedia. Por lo tanto, encontraremos desde la sencilla aplicación que nos permite hacer operaciones algebraicas usando el stylus hasta aquellas que nos permiten reproducir videos y navegar por nuestra red social favorita.

Todas las aplicaciones muestran sus resultados en pantalla, pero están diseñadas para que sean visibles mediante el proyector. La visualización en ambos medios se debe a que no podemos escribir directamente en el buffer de video del celular, i.e., la salida de video trasmitirá únicamente lo que sea visible en la pantalla del celular, razón por la cual debemos escribir primero en pantalla todo lo que necesitemos proyectar.

3.2.5 Salida de datos

Finalmente, la información generada en pantalla por las aplicaciones se trasmite al pico-proyector al apagar la cámara por un momento y encender la salida de video del sistema operativo, esto se debe a que por la configuración del hardware no es posible mantener la salida de video al mismo tiempo que la cámara fotográfica, se tiene una especial consideración en la sincronización de estos procesos pues si se llegaran a intersectar el sistema operativo inmediatamente lo detectaría como un error en tiempo de ejecución y terminaría con el programa.

Hasta este momento se han presentado todos los bloques que componen el sistema dando una idea general del flujo de la información, ahora que el lector comprende las entradas y salidas de cada bloque y su relación con los anteriores entonces la lectura de los capítulos posteriores se tornara más amena, pues podemos entrar en los detalles específicos de cada bloque habiendo entendido que información nos precede y que información espera el siguiente bloque que entreguemos.

Capítulo 4

Entrada y salida de datos

Una de las particularidades más importantes del sistema propuesto es poder detectar movimientos hechos naturalmente, en vez de comandos predefinidos en un periférico estático conectado a una unidad de procesamiento. Esto se considera la entrada física. Después de una serie de procesos se entrega una respuesta. La definición de sexto sentido nos indica que la respuesta debe reflejarse en el medio que rodea al usuario, i.e., en una pared, un escritorio o cualquier superficie donde la proyección de la respuesta resulte legible, no limitándose a que la repuesta pueda repetirse en pantalla o en algún otro periférico. Esta descripción corresponde a la salida física del sistema.

La entrada y salida física representan la parte visible del sistema, aquella que permite la interacción con el usuario. Así como en la GUI los botones, menús, ventanas y otros elementos gráficos son los objetos de identidad que permiten reconocer la interface como gráfica, separándola de las CLI, el uso de gesticulaciones en el ambiente nos permiten identificar el sistema como una NUI, mientras el uso de proyecciones y la movilidad del sistema nos permiten clasificarlo dentro de la tecnología de sexto sentido.

La movilidad del sistema resulta un aspecto crucial pues la miniaturización y empotramiento del hardware trae consigo restricciones en su uso, provocadas por la arquitectura electrónica y las propiedades introducidas por los fabricantes. Estos aspectos no han surgido en otros trabajos que hacen uso de electrónica de mayor capacidad y fuentes de procesamiento más potentes, pero al mismo tiempo más difíciles de movilizar y de personalizar. En este capítulo se expondrán los dispositivos usados con el fin de introducir y demostrar datos, el cableado y la configuración necesaria para la reproducción del dispositivo, así mismo, se presentaran los costos y las justificaciones sobre la elección de uno u otro dispositivo.

4.1 Entrada de datos

Para percibir las gesticulaciones del usuario, algunos dispositivos como el Kinect[1] utilizan hardware adicional, en específico proyectores y cámaras de infrarrojos. En nuestra propuesta queremos evitar encarecer el sistema, utilizando la cámara embebida en el Samsung Galaxy I9000 que posee una definición de 5 MPx [Gal, 2009]. Además, con el fin de reducir la complejidad computacional del reconocimiento de formas, usaremos un stylus de punta luminosa que será el objeto a reconocer en las imágenes tomadas por la cámara. Esta etapa se encuentra fuertemente asociada al capítulo 5.

4.1.1 Diseño del stylus

Pensando en cómo identificar objetos en pantalla, los objetos brillantes poseen características más fáciles de analizar que los objetos opacos [Lowe, 1999], por ello buscamos crear un objeto que guie las gesticulaciones con un brillo particular. Haciendo una comparativa con una "varita mágica" creamos un stylus de punta luminosa que consiste en un LED ultra brillante de 1.5 candelas con una resistencia, un push-botón y tres pilas de botón de 1.5 volts.

Como tenemos las indicaciones eléctricas del LED ultra brillante, 3.5 volts y 20 mA [LED, 2011] y el voltaje de la fuente, 4.5 volts, podemos usar la fórmula 4.1 [Boylestad et al., 2003] para determinar el valor de la resistencia:

$$R = \frac{V_F - V_L}{C_L} \tag{4.1}$$

Dónde:
V_F se refiere al voltaje de la fuente
V_L es el voltaje requerido por el LED
C_L es la corriente requerida por el LED

Sustituyendo los valores del párrafo anterior en la fórmula 4.1 tenemos un valor de 50 ohms. El valor de la resistencia comercial más cercana es de 51 Ohms, con estos datos y los mencionados en el párrafo anterior podemos entonces construir el circuito del stylus, el cual es mostrado en la figura 4.1.

[1] http://www.xbox.com/en-US/kinect

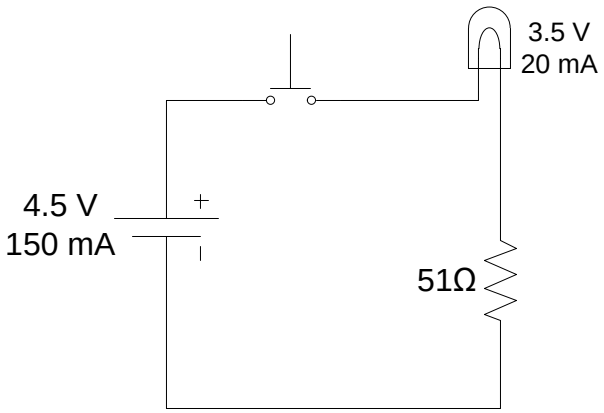

Figura 4.1: Circuito del stylus luminoso

Como vemos se trata de un circuito bastante sencillo, no pretende ser complicado ni la parte primordial de este trabajo. Por el contrario es sólo una parte complementaria que permite ejecutar el trabajo principal en forma más eficiente.

Un último aspecto a tener en consideración es el color del LED. La correcta elección permitirá discernir más fácilmente entre el stylus y otros elementos que puedan ser captados por la fotografía. Por ejemplo, se podría pensar que una luz blanca sería una correcta elección, ya que su umbralización dependería sólo de que sus valores rojo, verde y azul del modelo RGB fueran lo suficientemente altos (cercanos a 255)... nada más alejado de la realidad. En situaciones normales es fácil toparnos con brillos blancos que saturan la percepción de la cámara, provocando objetos completamente blancos, los cuales pueden ser destellos en metales o plásticos, lámparas, focos, agua, o el mismo sol. Siguiendo la lógica de buscar colores fácilmente identificables bajo el modelo RGB, podemos pensar en LEDs rojos, verdes y azules. Sin embargo, el mismo problema que teníamos con los LEDs blancos lo tenemos con los rojos y verdes, ya que todo aparato con botón de encendido/apagado presenta LEDs en color verde y rojo, que provocan saturación en el canal verde y rojo respectivamente. Esto problema sucede también con otras luces brillantes como las de los semáforos y brillos en botellas, así que una saturación en el canal azul es más difícil de encontrar en la naturaleza y más fácil de distinguir por su alto contraste en un sólo canal.

Con todas las propiedades definidas, el stylus físico toma forma. La figura 4.2 muestra una fotografía típica de un movimiento del stylus en el que se aprecia la luz azul que además presenta la propiedad de ser blanca en el centro. En el capítulo 5 se hará uso de esta propiedad para beneficiar el algoritmo de búsqueda de objetos, haciéndolo más eficaz.

Figura 4.2: Al centro se puede apreciar la luz azul con centro blanco emitida por el stylus

4.2 Obtención de imágenes

El reconocimiento de gesticulaciones está diseñado para ser un proceso de visión por computadora auxiliado del hardware incluido en el Samsung I9000, i.e., la cámara fotográfica principal, [2]. El proceso completo de extracción de características y reconocimiento de formas es analizado en los capítulos 5 y 6. Los cuales asumen un flujo de información de entrada existente y limpio. Para que esta suposición sea posible debemos entender antes el proceso fotográfico que se lleva a cabo durante la obtención de imágenes de la cámara, las restricciones que este proceso provoca y así como la forma de transformar los datos puros de la cámara en contenido legible para los demás procesos.

[2]decimos principal pues este modelo en específico cuenta además con una cámara de menor resolución que es usada durante videoconferencias

4.2.1 Arquitectura del proceso fotográfico

El tratamiento digital de imágenes puede volverse complicado entre mayor sea la compresión del formato de imágen, el proceso fotográfico de la cámara del I9000 y de la mayoría de los dispositivos móviles, entregan como resultado una imágen en formato JPG, esta compresión en JPG no es una característica programada, sino más bien una implementación en hardware que poseen como especificación los chips de cámara de la mayoría de los fabricantes. La captura de fotografías en este formato comprimido traería como consecuencia que antes de ser analizada cada imágen debe ser descomprimida en sus canales básicos RGB, llevada a memoria, mantenida ahí en su forma pura y sin compresión, después debe liberarse el espacio de la última imágen cada que captemos una nueva fotografía. Si bien el proceso de descompresión no es complicado, hacerlo 10 veces por segundo en un teléfono que además debe correr otras funciones del sistema operativo y de nuestro sistema genera demasiada carga computacional.

Lo que resulta peor es que el proceso completo para obtener una imágen está conformado por varias etapas dentro del chip que incluyen: 1) pasar la imágen a un buffer interno del chip, 2) comprimir la imágen que en ese momento se encuentra en bruto, 3) hacer una serie de llamadas al sistema operativo y 4) finalmente escribir la imágen en la memoria principal como un flujo de bytes en formato JPG. Los eventos completos pueden encontrarse en la documentación de Android [3] bajo el proceso `takePicture()`. Lamentablemente, al encontrarse programado en hardware, resulta imposible detener el proceso completo antes de que finalice. A pesar de que es posible obtener la imágen sin compresión, el hardware ordenará una interrupción hasta el momento en que termine el proceso completo, haciéndonos imposible el manejo de la cámara mientras aún se encuentra comprimiendo una imágen, esta compresión crea una perdida de tiempo desincronizando los procesos, pues el tiempo de compresión no es constante.

La solución planteada es en realidad creativa. Seguramente el lector ha observado en su teléfono celular que es posible observar en pantalla lo que se quiere fotografiar con movimiento en tiempo real. Esto es posible gracias a que la cámara entrega una secuencia de fotografías llamadas *previews*, que son imágenes en tamaño formato VGA, que no pasan por ningún filtrado o proceso, sino que son entregadas a modo de previsualización de la imágen, i.e., son imágenes tales como la cámara está captándolas sin ningún proceso especial para mejorar calidad o compresión. Es posible intervenir directamente estas imágenes antes de que lleguen a la pantalla principal, gracias al evento `onPreviewCallback` de la cámara de Android, que permite acceder al *buffer* de bytes encargado de almacenar los *previews* [Gu and Duh, 2011].

[3] http://developer.android.com/reference/android/hardware/Camera.html

La configuración de este evento permite además darle a los *previes* las propiedades del tamaño deseado y los FPS necesarios. El único problema ahora es que la imágen en bruto del chip de la cámara no es una imágen RGB sino una imágen en formato YUV24 [McCarthy and El-Sheikh, 2011] [Gu and Duh, 2011], que es un formato especial manejado por las cámaras a nivel hardware.

4.2.2 Descomposición de la imágen

El análisis de la imágen toma los canales RGB de la imágen para umbralizarla con base en la saturación de cada tono de color. Sin embargo como mencionamos en la sección 4.2.1, la cámara maneja en hardware el formato YUV24[4]. Es necesario, por lo tanto, hacer la conversión entre formatos para que nuestros algoritmos trabajen correctamente.

YUV es un formato para la transmisión de video análogo desarrollado para codificar una imágen o video teniendo en cuenta la percepción humana. Este formato permitie no sólo una compresión más adecuada para medios análogos, sino también el encubrimiento de errores de compresión o imperfecciones de transmisión ante la vista humana [Jack, 1995]. Además es usado en la mayoría de los dispositivos electrónicos de transmisión de video como el formato por *default*. En nuestro caso se utiliza para la transmisión de imágenes entre la cámara y la pantalla. Este formato divide el espacio de color en una componente lumínica Y (niveles de blanco y negro) y dos componentes de crominancia U,V (información respecto al color).

Existe una gran variedad de codecs basados en el formato YUV, que fueron desarrollados por diferentes compañías a lo largo de los años para buscar una transmisión más fiable y clara. Entre ellos destaca el formato YUV420 en sus versiones YUV420P y YUV420SP, que han sido tomados por la industria como un estándar pues permiten una mayor compresión de datos (una compresión excesivamente ligera si se compara con JPG) sin comprometer la calidad. El formato que nos interesa, YUV420SP, aplica los mismos principios que el formato YUV, pero este último encontraríamos se guarda y transmite en triadas de elementos <Y,U,V> mientras que YUV420SP es un formato semi-plano, lo que quiere decir que agrupa todos los elementos Y de una línea en un sólo paquete, los elementos U en otro y así mismo con los V. Esta forma de acomodar los elementos resulta mucho más eficiente pues permite comprimir los

[4]Existen chips configurables que pueden reprogramarse para manejar RGB como estilo de captura, pero el chip OV5633 así como la mayoría de los chips de cámaras para teléfonos celulares no incluye esa opción

datos respecto a su posición en pantalla y sin perder calidad.

La transformación de datos entre YUV y RGB normalmente depende sólo de una multiplicación de matrices. Sin embargo, el formato comprimido YUV420SP requiere un tratamiento especial basado en el índice de cada píxel RGB. El algoritmo para pasar entre YUV420SP a RGB (decodeYUV420SP) es descrito en [Masakazu, 2011] y es presentado como el algoritmo 1.

Algorithm 1 decodeYUV420SP

Require: byte YUV[], int Width, int Height
Ensure: Arreglo RGB de la imágen

1: **for** $j = 0, yp = 0$ hasta $j = Height$ **do**
2: $uvp = Width * Height + (j >> 1) * Width, u = 0, v = 0$
3: **for** $i = 0$ hasta $i = Width$ **do**
4: $yp++$
5: $y = Mascara(YUV[yp]) - 16$ //Mascara() regresa los primeros 8 bits de un entero
6: **if** i es par **then**
7: $u = Mascara(YUV[uvp++]) - 128$; $v = Mascara(YUV[uvp++]) - 128$
8: **end if**
9: $y2 = 1192 * y$
10: $R = y2 + 1164 * v$
11: $G = y2 - 833 * v - 400 * u$
12: $B = y2 + 2066 * u$
13: Si R, G o $B < 0$ entonces R, G o $B = 0$, Si R, G o $B > 262143$ entonces R, G o $B = 262143$
14: Anexar R, G y B al arreglo RGB
15: **end for**
16: **end for**
17: **return** RGB

Al ser un formato dependiente del índice el algoritmo, necesita correrse de principio a fin sin interrupciones, i.e., no está diseñado para obtener cada píxel por separado. En el capitulo 5 se hará notorio que esta es una propiedad necesaria para la eficiencia del sistema. Por ello se ha trabajado en una versión especial del algoritmo 1 especialmente desarrollado con el propósito de hacer más eficiente el proceso de conversión. El algoritmo 2 es el resultado de este trabajo.

Algorithm 2 PixelYUVToRGB

Require: byte YUV[], int Width, int Height, int i,j
Ensure: R,G,B del píxel en las coordenadas i,j

1: $y = Mascara(YUV[j * Width + i]) - 16$
2: $uvp = Width * Height + (j >> 1) * Width + i$
3: **if** i es impar **then**
4: $uvp - -$
5: **end if**
6: $u = Mascara(YUV[uvp + 1]) - 128$; $v = Mascara(YUV[uvp]) - 128$
7: $y2 = 1192 * y$
8: $R = y2 + 1164 * v$
9: $G = y2 - 833 * v - 400 * u$
10: $B = y2 + 2066 * u$
11: Si R, G o $B < 0$ entonces R, G o $B = 0$, Si R, G o $B > 262143$ entonces R, G o $B = 262143$
12: **return** R, G y B

4.3 Salida de datos

Según la definición de Sexto Sentido ofrecida por [Mistry and Maes, 2009] es necesario que la respuesta se separe de los periféricos tradicionales de salida, que limitan al usuario a un área de trabajo estática, llevando el resultado de los procesos al medio ambiente que rodea al individuo, provocando que sea la computadora la que se sumerja en el mundo de los humanos y no en viceversa.

La misma definición de sexto sentido citada en el parrafo anterior, se habla de la necesidad de que la respuesta del software deje a un lado los perifericos tradicionales de salida (que limitan al usuario a un área de trabajo estática) y utilice la realidad aumentada como método de salida. Esta idea permite llevar el resultado de los procesos computacionales al medio ambiente que rodea al usuario, i.e., la computadora se sumerje en el mundo humano y no viceversa. Siguiendo este propósito, nuestro sistema presenta las respuestas de la entrada física como una serie de proyecciones alrededor del usuario. De esta manera no es necesario estar atento a la pantalla del teléfono celular ni tener ocupadas las manos, sino que podemos esperar que la respuesta deseada aparezca a conveniencia frente a nosotros.

4.3.1 Proyección

Para crear las proyecciones de salida es necesario agregar un dispositivo adicional al Samsung Galaxy I9000. Se utiliza un pico-proyector, que es un proyector miniaturizado y portable, generalmente del tamaño de un teléfono celular, de baja potencia lumínica (menor a 60 lúmenes) y diseñado para uso personal [Kress and Meyrueis, 2009].

Se ha utilizado un pico-proyector externo ya que, aunque existen en la actualidad celulares que poseen pico-proyectores integrados cómo el LG Expo o el Samsung I8520 BEAM, la capacidad de procesamiento de ambos teléfonos es bastante reducida en comparación con el smartphone seleccionado. Así que prefiriendo las características de CPU sobre un pico-proyector embebido, hemos decidido separar ambos elementos logrando mayor potencia en ambas partes, pues los pico-proyectores embebidos en dispositivos celulares aún son de baja potencia[5] y se usan procesadores de menor capacidad en este tipo de teléfonos para evitar el excesivo consumo de pila.

La tabla 1 muestra algunos de los pico-proyectores disponibles actualmente en el mercado y sus principales características que han sido extraídas de picoprojector-info[6]. El propósito de mostrar este análisis es procurar una correcta comparativa en aspectos esenciales para el sistema, sobre todo aquellos referentes al flujo luminoso o brillantez[7], las dimensiones y el peso.

De los datos de la tabla 4.1 hemos escogido el pico-proyector Optoma PK201, ya que se asemeja en dimensiones al Samsung Galaxy I9000, diferenciándose sólo por el grosor en menos de 5 mm, pero manteniendo la misma altura y anchura, lo que lo integra casi a la perfección. Es de un peso bajo y aunque su brillantez es media, tiene una alta definición, de hecho la misma que el Galaxy I9000, por lo que mantienen una relación de aspecto de 1:1. Además, su costo es aceptable y no aumenta en gran medida el costo del sistema completo.

El sistema operativo Android debe tener encendida la opción "TV-Out" en los ajustes de pantalla, preferentemente en formato NTSC, que es el formato de video americano. Debido a problemas en la arquitectura del sistema operativo no es posible mantener encendida la salida de video y la cámara al mismo tiempo, lo que dificulta nuestra labor. Para nosotros es indispensable trabajar con la cámara y el proyector al mismo tiempo. Como solución a este problema la cámara es activada y desactivada según se encuentren las proyecciones activas o no, de forma que un nuevo objeto cámara se genera cada vez que regresamos a nuestra aplicación principal, desactivando en automático la salida de video.

[5]http://www.mobileworldcongress.com/

[6]http://www.picoprojector-info.com

[7]Se refiere a la potencia lumínica con la que una imágen se proyecta, i.e., a mayor luminiscencia más nítida es una imágen aun en ambientes de mucha luz ambiental

Modelo	Resolución (pixeles)	Brillo (lúmenes)	Tamaño máximo de imágen (pulgadas)	Dimensiones (pulgadas)	Peso (gramos)	Costo (dolares)
Snako Miseal Mini	800 x 600	25	60	2.75 x 2.75 x 2.75	300	430
Samsung MBP200	480 x 320	30	50	4 x 2 x 0.8	140	540
Optoma PK101	640 x 480	11	66	4.1 x 2 x 0.6	100	130
3M MPro-180	1280 x 768	32	80	1.3 x 2.5 x 5.9	340	449
Acer C112	854 x 480	70	100	3.6 x 0.9 x 5.4	245	337
Optoma PK201	854 x 480	20	66	0.7 x 4.6 x 2.4	225	269
Toshiba Lumileo M200	640 x 480	14	60	0.8 x 2.2 x 5.4	145	229
Philips Picopix PPX-1430	800 x 600	30	81	2.5 x 3.9 x 3.9	290	450
Microvision ShowWX plus	848x480	15	100	2 x 9 x 7	122	349

Tabla 4.1: Características de algunos pico-proyectores comeriales

4.3.2 Cableado

La transferencia de las imágenes producidas en el Samsung Galaxy I9000 al pico-proyector PK201 podría resultar particularmente difícil apenas una generación atrás de smartphones. Sin embargo los fabricantes de teléfonos celulares y los sistemas operativos dedicados han planeado la evolución de la tecnología móvil a dispositivos de cómputo ubicuo. De esta manera, en la actual generación, son muchos los dispositivos que permiten una salida directa de video. Entre las marcas que ya ofrecen teléfonos con estas características podemos encontrar a LG, Samsung, HTC, Sony, iPhone, entre otros y en gran variedad de sus productos[8]. Lamentablemente la salida de video sufre el mismo problema que la salida de datos en dispositivos móviles: cada fabricante tiene su propio conector y cableado especial. Dispositivos como los fabricados por HTC utilizan un puerto mini-HDMI bien conocido en el tratamiento de video; algunos otros como iPhone utilizan su propio conector que no se encuentra en ningún otro dispositivo. Siendo esta una barrera física y electrónica, debemos acotar el problema a la conexión de la salida de video del I9000 que hace uso del puerto Jack 3.5mm [Gal, 2009] originalmente destinado en la gran mayoría de los teléfonos celulares a la salida de audio, pero modificado en el I9000 para trabajar con un triple anillo aislante de forma que pueda transmitir audio y video mediante una conexión RCA, común en pantallas y proyectores. Esta solución de puerto no es nueva pues actualmente muchas cámaras y reproductores de video manejan la salida de video mediante el puerto Jack 3.5mm de triple aro o TRRS (*Tip Ring Ring Sleeve*) por la miniaturización que permite.

La verdadera problemática surge en el hecho de que el pico-proyector PK201, al igual que todos los pico-proyectores comerciales, carecen de una entrada de video RCA por el tamaño del puerto necesario [Opt, 2011]. Como alternativa ofrecen un adaptador RCA a Jack 2.5mm, el cual resultaría ideal de no ser porque comercialmente el cable de Jack 3.5 mm a RCA, necesario para la salida de video, es de al menos 1 metro de longitud y el adaptador de RCA a Jack 2.5 mm tiene una longitud similar, lo que provocaría que en realidad tuviéramos una masa de cableado mayor que la que ofrecen el Samsung I9000 y el PK201 en conjunto. Suena ilógico pensar de esta forma, sobre todo porque la conversión a RCA parece una etapa innecesaria... bastaría con pasar de Jack 3.5 mm a 2.5 mm.

Nuevamente nos encontramos con las restricciones comerciales, ya que el cable que conecta los puertos Jack 3.5 mm y 2.5 mm sólo se encuentra en versión de doble anillo, i.e., sólo es factible para la transmisión de audio; el tercer anillo es el que admite un canal de video que después de todo es lo que deseamos. La figura 4.3 nos muestra la

[8] http://www.phonegg.com/List/TV-Out-Cell-Phones.html

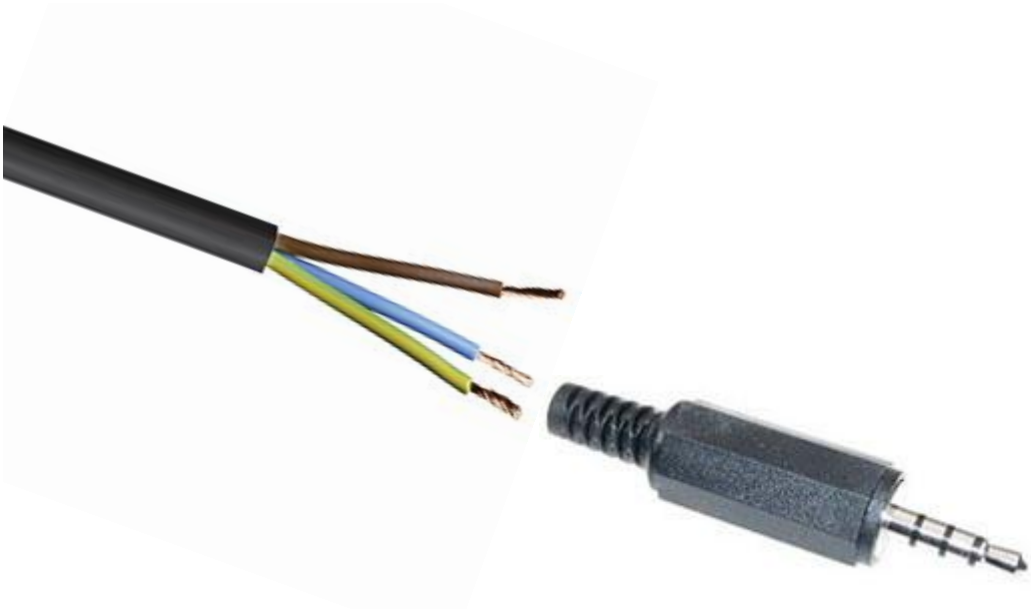

Figura 4.3: Constitución física de un adaptador Jack junto a su cableado

constitución de los adaptadores Jack. El salto repentino en el tema se debe a que la propuesta es no usar un adaptador comercial sino crear nuestro propio cable que se adapte en longitud y capacidad a lo que buscamos.

Como se observa en la figura 4.3 el cable conversor es sólo un coaxial con el mismo número de canales conductores que de anillos en el conector. El cable construido entonces sólo debe soldar los canales conductores a los anillos de cada conector, aislando cada conductor entre sí y terminando con una capa de aislante al medio ambiente para los tres cables en conjunto. El cable resultante se muestra en la figura 4.4, se observan los conectores de 3.5 mm y 2.5 mm a cada extremo. La longitud se ha fijado en 15 cm de forma que el cable queda con la suficiente holgura para conectar el pico-proyector y el teléfono celular, sin tener desperdicios.

En la figura 4.4 se observa que los conectores usados son de sólo dos canales, lo cual es factible ya que al construir nuestro propio cable hemos podido cambiar la configuración interna del mismo. En el esquema de la figura 4.5 se muestra el uso que se le da a cada parte del conector. Como vemos, las dos primeras secciones están reservadas para los canales izquierdo y derecho de audio, la tercera sección es la tierra y la última sección se utiliza para el video [Books, 2010]. Podemos entonces soldar convenientemente los cables de forma que un conector de dos canales quede configurado para manejar un canal de audio, uno de video y una tierra. Dado que no nos interesa transmitir audio, como veremos más adelante en el capítulo 7, resulta

Figura 4.4: Cable construido para la conexión entre el pico-proyector y el telefono celular

más conveniente utilizar la bocina del Samsung I9000 por su potencia y nitidez en comparación con la del PK201. Eliminando el canal derecho de audio, tenemos un cable diseñado especialmente para nuestro problema, se puedan utilizar conectores de triple anillo, pero son más difíciles de conseguir y en general la transmisión audio hacia un proyector resulta inútil si todo el audio será manejado por el teléfono celular en forma más apropiada y con un mejor control de volumen y ecualización.

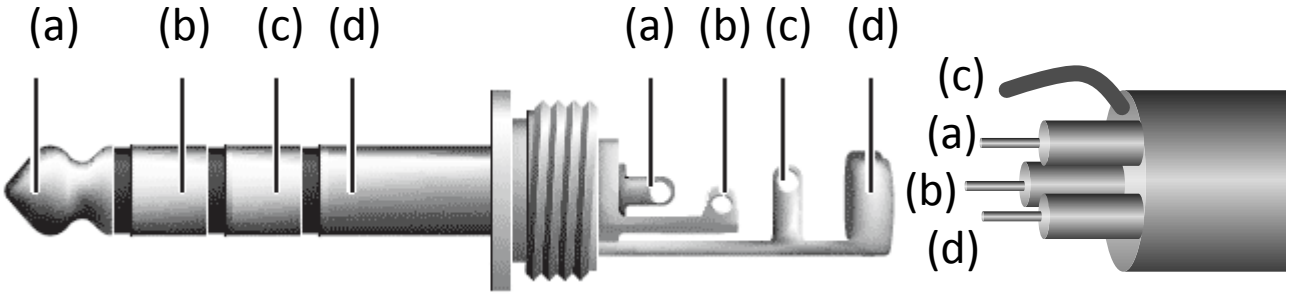

Figura 4.5: (a) Tip, encargado del audio izquierdo (b) Ring 1, encargado del audio derecho (c) Ring 2, tierra común (d) Sleeve, para el micrófono o video

Capítulo 5

Preprocesamiento de imágenes

La forma en la que el teléfono percibe el movimiento del stylus, que más adelante será identificado como una letra, es a través de la cámara posterior del teléfono. Si bien la cámara debe permanecer encendida todo el tiempo, esto no significa que el mundo sea percibido como un video continuo. De hecho, el análisis de una secuencia de video en tiempo real sería altamente complejo, pero los resultados no serían distintos si usamos la cámara en modo ráfaga (muchas fotografías por segundo). Este último paradigma se ha escogido como modo de operación, pues el manejo de la cámara y de las imágenes resulta especialmente sencillo en Android.

El análisis de un video comprendería fases como descompresión, algunos filtrados, análisis de los índices, sustracción del renderizado y algunas otras. Queda claro que el análisis de video en tiempo real es complejo incluso para computadoras de escritorio. El estudio de cada imágen por separado permite una secuencia más ordenada e información menos comprimida. Si pensamos en la información como una matriz de datos, deriva en la creación de algoritmos más intuitivos (aunque más adelante en el capítulo, descompondremos la matriz en un arreglo al acomodar filas y columnas simplemente para optimizar la programación, aunque el concepto abstracto de matriz se mantendrá).

5.1 Formato de imágen

Las imágenes de alta calidad nos permitirían detectar variaciones muy finas entre imágenes consecutivas, sin embargo, entre más píxeles contenga una imágen mayor es el tiempo de respuesta de la cámara. Estas propiedades varían entre los diferentes modelos de chip de la cámara fotográfica, situándonos en el modelo particular que ofrece el Samsung Galaxy S (Omnivision OV5633). La tabla 5.1 nos ofrece una

Formato	Tamaño en píxeles	Máximo de FPS
QSXGA	2592 x 1944	15
1080p	1920 x 1080	30
HD	1280 x 960	60
720p	1280 x 720	60
VGA	640 x 480	60

Tabla 5.1: Capacidad de FPS del Omnivision OV5633 según la resolución

mejor panorámica de las fotografías por segundo (FPS) que pueden ser tomadas en comparación con el tamaño de las imágenes.

Los sistemas de visión por computadora se ven siempre sometidos a la relación existente entre la calidad de las fotografías y la capacidad de procesamiento del equipo [Siltanen and Hyväkkä, 2006], i.e., el análisis de una imágen en formato QSXGA nos llevaría por lo menos 5,038,848 (2592 x 1944) lecturas individuales de cada pixel, mientras que la exploración de una imágen en formato VGA nos tomaría tan solo 307, 200 lecturas, ¡16 veces menos! La decisión de cuál formato de imágen debemos escoger se debe llevar a cabo con base en el conocimiento del algoritmo de reconocimiento de caracteres ¿Cuántas muestras son necesarias por segundo para un correcto reconocimiento? y ¿Cuál es la definición necesaria y suficiente para distinguir un vector de otro?

5.1.1 Análisis del FPS

La primera pregunta encuentra respuesta al descomponer algunas letras en sus vectores, pues como veremos más adelante durante el capítulo 6, los caracteres se descomponen en los vectores que los estructuran, limitandonos al uso de 12 vectores característicos del espacio. Por ejemplo tomemos los caracteres “O”, “@”y “l”, la letra “O”bien redondeada se compone de 11 o 12 vectores; un caso mucho más complejo es el del símbolo “@”que se compone de hasta 20 vectores y un caso mucho más simple es el de la letra “l”compuesta apenas por 1 o 2 vectores. Este ejemplo se esquematiza mejor en la figura 5.1.

El tiempo promedio de cada letra también es importante. Tomando en cuenta un estilo de escritura relajado, que no ha sido apresurado, los promedios en tiempo del trazo de cada letra son de 1.3, 1.9 y 0.3 para las letras “O”, “@”y “l”, respectivamente. Estos datos han sido medidos considerando el trazo de caracteres con el stylus en el aire, el dibujo de letras variará en el tiempo respecto al tamaño.

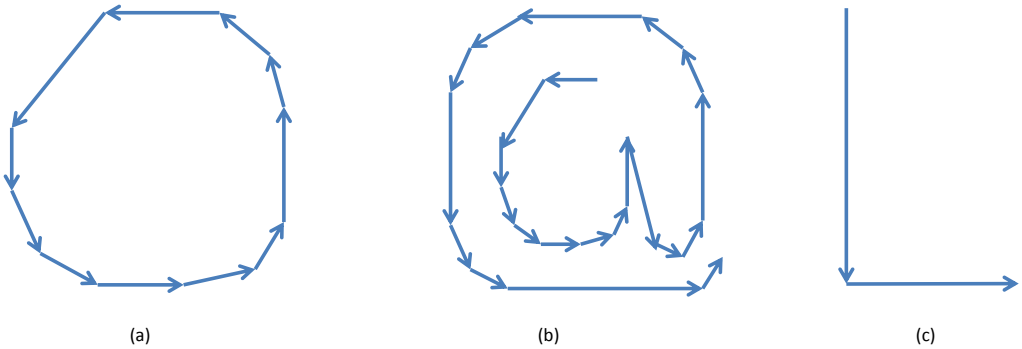

Figura 5.1: (a) muestra una “O”(b) una “@”y (c) una “L”, todas vectorizadas

Fracción	Intervalo en segundos
1/8	0.125
1/9	$0.\overline{111}$
1/10	0.1
1/11	$0.\overline{09}$
1/12	$0.08\overline{33}$
1/13	$0.\overline{0769230}$
1/14	$0.0\overline{714285}$
1/15	$0.0\overline{666}$
1/16	0.0625

Tabla 5.2: Intervalos de tiempo para diferentes FPS. La línea encima de algunos decimales indíca la repetición infinita de los números cubiertos por la línea

Haciendo la comparación de todos los posibles caracteres reconocidos por el sistema y el tiempo que toma dibujarlos hemos establecido que se deben tomar entre 8 y 10 fotografías por segundo para una correcta ejecución del algoritmo de reconocimiento de caracteres. Los anteriores ejemplos ilustran perfectamente estos números: un dibujo de 1.3 segundos nos daría 13 fotogramas a 10 FPS suficiente para detectar una letra “O”; en 1.9 segundos tenemos 19 vectores si capturamos el trazo a una velocidad de 10 FPS que alcanzan para deducir una “@”; en 0.3 podemos obtener 3 vectores a esta misma velocidad, que es más de lo necesario para identificar una “l”. Si bien podemos tomar un número mayor de fotografías por segundo debemos considerar que las fotografías serán tomadas en intervalos discretos de tiempo que la computadora debe medir de forma precisa para evitar inexactitudes en el sistema. Por ejemplo una octava o una décima de segundo parecen ser buenos intervalos pues representan 0.125 y 0.1 segundos respectivamente, pero analicemos que sucede con fracciones de segundo más pequeñas en la tabla 5.2.

La mayoría de las fracciones entregan números con decimales periódicos que

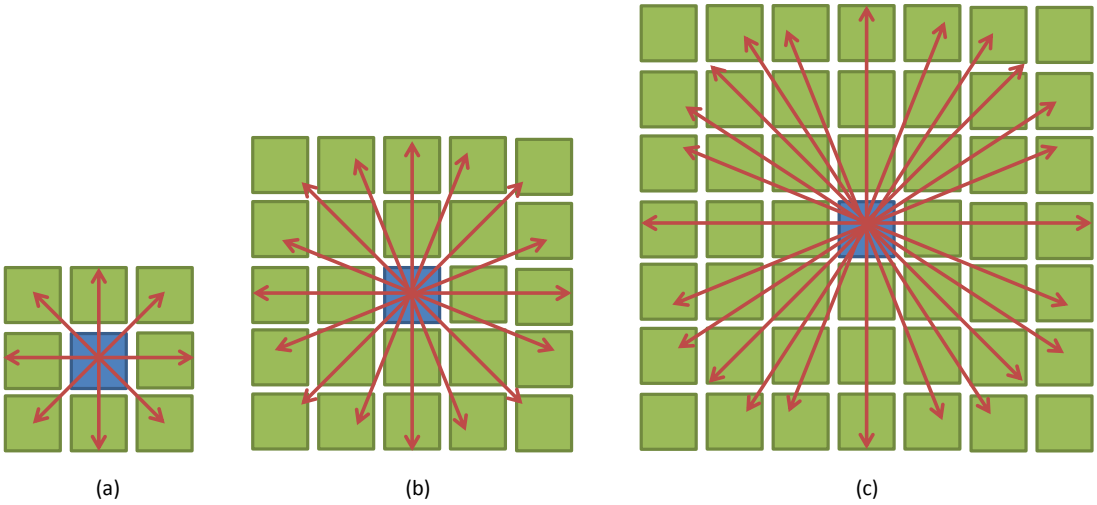

Figura 5.2: (a) presenta una cuadricula de 2 píxeles de distancia (b) una cuadricula de 3 píxeles y (c) una a 4 pixeles. Respectivamente pueden representar 8, 16 y 24 vectores como máximo.

tendríamos que recortar y que a la larga causarían una pequeña variación cada determinado tiempo según el decimal escogido. Los numeradores más exactos que podemos usar son el 8, el 10 y el 16, de entre ellos el que mejor se acomoda es el 10, pues el 8 representa una pérdida pequeña de información, mientras el 16 representa el análisis de un 60 % más de información que no necesariamente será útil.

5.1.2 Análisis del tamaño de imágen

Retomemos la segunda pregunta sobre la decisión del formato de imágen planteada al final de la sección 5.1: ¿Cuál es la definición necesaria y suficiente para distinguir un vector de otro? Afortunadamente, en nuestro caso, no necesitamos una gran resolución de imágen pues para distinguir entre 12 vectores propuestos en el capitulo 6 es necesaria apenas una separación de 3 o 4 píxeles entre el píxel inicial y final en la medición.

La figura 5.2 ejemplifica cómo 3 píxeles sarían suficientes para obtener 16 vectores distintos, pero tendríamos el problema de que sobran 4 vectores que, de alguna forma, deben ser asignados dentro de los primeros 12, mientras que con 4 píxeles obtenemos 24 vectores, suficientes para que a cada vector característico se le asigne dos de los vectores obtenidos.

¿Qué distancia real representa 4 píxeles en los formatos de imágen presentados al inicio de este capítulo? El lector que este familiarizado con el tema probablemente piense que la pregunta ha sido mal formulada, ya que en una fotografía no es posible

Formato	Relación de 4 píxeles a milímetros	Tamaño de una letra de 10 por 10 centímetros en píxeles	Promedio del vector en píxeles	Tamaño de una letra de 3 por 3 centímetros en píxeles	Promedio del vector en pixeles
QSXGA	0.7	460 x 460	250	140 x 140	60
1080p	1	300 x 300	180	90 x 90	52
HD	1.25	280 x 280	120	85 x 85	32
720p	2.5	150 x 150	75	55 x 55	25
VGA	4.5	100 x 100	50	30 x 30	12

Tabla 5.3: Relación entre píxeles y longitudes reales medidas con un escalímetro

encontrar una relación directa entre centímetros y pixeles. De hecho una fotografía se encuentra compuesta de varios elementos en diferentes profundidades, de forma tal que 50 píxeles podrían equivaler a 5 centímetros en un rostro cercano o 5 kilómetros de una lejana montaña. Afortunadamente para nosotros el stylus no puede estar a la distancia en una lejana montaña sino que, dado que el celular se encuentra en el torso del usuario, la posición más alejada se encuentra a la profundidad marcada por lo largo del brazo del usuario, aproximadamente 75 cm [Chaurand et al., 2001] más 10 cm correspondientes a la longitud del stylus, lo que nos da un total de 85 cm de alejamiento máximo, que es el dato más interesante pues conocer la distancia máxima nos permite estudiar el peor caso posible de la resolución de imágen contra la distancia real. Un sencillo experimento con un escalímetro fotografiado a 85 cm de distancia en diversas resoluciones nos arroja los resultados que se observan en la tabla 5.3.

En la figura 5.2 pudimos constatar que 4 píxeles son suficientes para representar los 12 vectores característicos del espacio que se ocuparan durante el capítulo 6, la tabla 5.3 considera ciertas mediciones a una distancia de 85 cm de la cámara, en la primera columna muestra precisamente la relación entre la longitud real y los 4 píxeles que se necesitan como mínimo. También pueden observarse proporciones basadas en 10 y 3 centímetros y el tamaño promedio de los vectores en estas instancias. Una letra de 10 centímetros es una letra cómoda para el usuario que quiere marcar un signo utilizando el stylus. Representa la media de varias letras tomadas al tratar de marcarlas en el aire sin que estas parezcan exageradas pero tampoco incómodamente pequeñas, mientras una letra de 3 cm representa una letra incómodamente pequeña para el usuario al ser trazada en el aire (ya que desde su perspectiva esta letra es aún más pequeña debido a la profundidad). Claramente podemos ver que la definición VGA alcanza la resolución suficiente para distinguir letras pequeñas; si en algún momento se quisiera mejorar el sistema para distinguir letras de poco menos de un

Figura 5.3: Imágen típica tomada por la cámara del Galaxy S, muestra objetos al fondo y una pantalla

centímetro, la calidad 720p sería suficiente.

Los experimentos detallados en esta subsección nos han arrojado los datos de configuración de imágen apropiados para que el sistema trabaje eficientemente: 10 FPS a una resolución VGA de 640 x 480

5.2 Algoritmo de búsqueda de objetos

Ahora que poseemos las configuraciones y herramientas necesarias para analizar la imágen podemos comenzar a discernir los elementos de la imágen que nos interesan, en nuestro caso la punta del stylus luminoso únicamente. La figura 5.3 nos muestra una fotografía típica en la que se encuentran varios elementos tanto lumínicos como opacos además del stylus.

El stylus va moviéndose por el espacio al dibujar una letra pero cada fotografía capta sólo un instante de tiempo, de forma tal que una letra está compuesta por una secuencia de fotografías, como lo muestra la figura 5.4.

En la figura 5.5 el dibujo se vuelve evidente al sobreponer en un sólo espacio las fotografías de la figura 5.4.

Como podemos observar el objeto de interés para nosotros es el espacio lumínico azul con centro blanco provocado por la punta del stylus. Este es el objeto a ser aislado.

Figura 5.4: Serie completa de fotográfias al captar un movimiento del stylus

Podría obviarse el problema, simplemente separando la parte más significativa del canal azul del RGB pero esta es una mala idea, ya que los objetos blancos y la mayoría de las luces y reflejos, pantallas encendidas o plásticos brillantes también entrarán en la gama de la parte alta del canal azul. El algoritmo diseñado debe ser capaz no sólo de identificar todos los objetos en la fotografía con características similares de forma y color, sino también de encontrar al mejor candidato, i.e., que no confunda otros objetos similares con la punta del stylus. Además, debe hacerlo con pocos recursos y un modo eficiente de operación.

Dividiremos en dos etapas el algoritmo para su mayor comprensión: La primera encabezada por un algoritmo de búsqueda de objetos que permite separar de una imágen o segmento de imágen todas las formas de interés, mientras el segundo al que hemos nombrado **algoritmo de la piedra en el estanque** que es un proceso de

Figura 5.5: Sobreposición de las fotografías de la figura 5.4 donde puede apreciarse un movimiento circular

optimización que permite evitar el recorrido completo de la imágen y se centra sólo en las partes con más altas probabilidades de contener objetos de interés.

El fundamento del algoritmo de búsqueda de objetos es formar grandes grupos de píxeles, los cuales representan objetos en la fotografía que cumplen con ciertas características de umbralización. De esta manera, el resultado esperado después de correr el algoritmo sobre una imágen es un arreglo que contiene las características esenciales de cada objeto de interés. No es necesario guardar todo el objeto pues sólo necesitamos conocer su centro para describir como se movió respecto a la fotografía pasada.

Para formar estos grandes grupos actuaremos sobre los píxeles pero no en forma individual como en la mayoría de los algoritmos, sino que tomaremos todos los píxeles seguidos de una línea, evitando de esta forma que las operaciones se realicen sobre

Figura 5.6: Los diferentes colores representan los diferentes grupos o líneas en una lectura horizontal. Se muestran grupos de diferentes tamaños; el algoritmo es más eficiente entre mayor es el tamaño del grupo

cada píxel individual, realizandose sobre la línea completa de píxeles. La figura 5.6 muestra los grupos tal cual serían tomados.

Resulta obvio que los píxeles colindantes forman parte de la misma figura, por ello es conveniente tratarlos como una línea entera. De esta forma decimos que la línea "A"pertenece a la figura "X"en vez de decir que los píxeles $a_1, a_2, \ldots a_n$ pertenecen a la figura "X". El número de operaciones ahorradas es tan grande como el número de píxeles que forman cada nueva línea anexada.

El algoritmo de búsqueda de objetos comienza a crear grupos de formas, cada uno compuesto al principio por una línea horizontal a la cual se irán anexando más líneas horizontales (pues la imágen es leída de izquierda a derecha y de arriba abajo, i.e., por filas), pudiendo ocurrir líneas de un solo píxel.

Consideramos dos tipos de píxeles: Aquellos que contienen información útil, i.e, que parecen pertenecer a uno de los objetos que buscamos, los llamaremos píxeles negros; los que no contienen información de relevancia, osea los que forman parte del fondo u objetos que no son de nuestro intéres, los llamaremos píxeles blancos. Para comenzar a formar líneas leemos la imágen, una fila a la vez, tomando como píxeles negros los que satisfagan la inecuación 5.1 y como blancos todos los demás:

$$\begin{aligned} R &\leq 56 \\ G &\geq 235 \\ B &\geq 235 \end{aligned} \tag{5.1}$$

Siendo R,G y B los valores de los canales rojo, verde y azul respectivamente obtenidos mediante el algoritmo PixelYUV. La inecuación 5.1 permite aislar la luz azúl emitida por el stylus luminoso.

Al tratarse de líneas horizontales sólo necesitamos dos coordenadas para describirlas, pues la coordenada en las abscisas es conocida y permanece, mientras nos

encontramos recorriendo las ordenadas. Un píxel negro sólo puede estar precedido por el elemento nulo o por un píxel blanco; de igual forma un píxel blanco sólo puede estar precedido por nulo o un píxel negro. El primer elemento de una línea se marca al encontrar un píxel negro que ha sido precedido de una serie de píxeles blancos o por el elemento nulo. El último píxel de una línea viene cuando dado un conjunto de píxeles negros, aparezca un píxel blanco.

Conociendo el inicio y fin de una línea, sólo necesitamos operar al encontrar un fin de línea. En esta eventualidad analizamos si se trata de una línea que pertenece a un conjunto ya existente, ya sea a la izquierda o arriba (pues no hemos analizado los píxeles de la derecha ni los de abajo, ya que la búsqueda se realiza de izquierda a derecha y de arriba hacia abajo) o si se trata de una línea que conforma un nuevo objeto pues no es posible asociarla con ningún otro.

Es bien sabido que al umbralizar una imágen suelen quedar huecos blancos entre las formas. Para evitar este fenómeno, definimos un número Z tal que Z represente el número de píxeles blancos que puede haber entre dos líneas para que se les considere parte del mismo objeto.

Para checar si un objeto pertenece a su vecino izquierdo verificamos si la distancia entre ambos es igual o menor a Z; si esto es cierto entonces anexamos las propiedades de la nueva línea a la antigua que ya conocíamos. Al final lo que buscamos sólo es la coordenada del centroide de cada forma, así que para anexar una línea a otra sólo sumamos las coordenadas de los elementos de la línea junto con el número de los mismos. De esta forma, al finalizar un objeto, tendremos una suma de coordenadas ordenadas, una suma de coordenadas abscisas y un número total de elementos, i.e., sólo tres variables en lugar de un gran arreglo de datos.

Un objeto puede también pertenecer a otro que se encuentre sobre él. Al tratarse de un recorrido horizontal y después vertical es claro que la primera vez que encontremos una nueva línea de objeto será al hacer el recorrido horizontal. Cuando finalicemos una línea horizontal registraremos sus coordenadas en un arreglo B. Todas las líneas que hayan sido encontradas en un recorrido horizontal serán registradas en este arreglo, de manera que al hacer un salto y comenzar el siguiente recorrido horizontal este arreglo contendrá las líneas. Para determinar si un objeto pertenece a un elemento superior tenemos que recorrer el arreglo B para determinar si encaja en alguna de las líneas superiores.

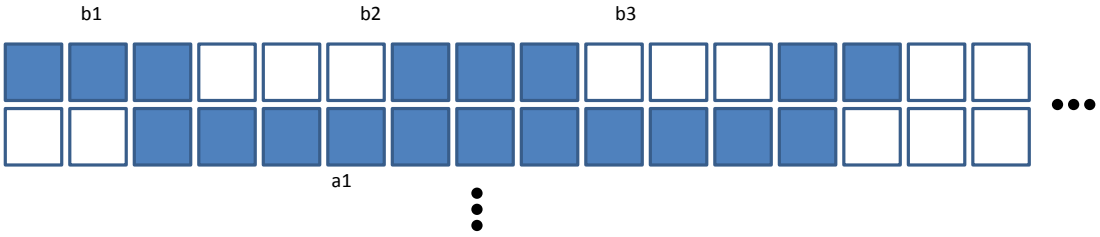

Figura 5.7: En este ejemplo, a1 es la nueva línea descubierta. Al hacer el análisis superior se verifica la pertenencia a b1; como también pertenece a b2 y b3, tanto a1 como b2 y b3 son anexados a b1

Cada vez que hagamos esta verificación no es necesario recorrer todo el arreglo B, más bien avanzamos sobre él cada vez que se descubra una nueva línea, de manera que al final hayamos hecho un sólo recorrido de B. La anterior afirmación se basa en que las líneas son introducidas en forma ordenada de izquierda a derecha. Supongamos el siguiente escenario: El arreglo B se encuentra formado por los objetos $b_1, b_2 \ldots b_m \ldots b_n$; comenzamos a recorrer una nueva fila y encontramos una línea a_1, al verificar la pertenencia hacia algún objeto en B, comenzamos a recorrer el arreglo desde b_1 y hasta que un objeto b_m da positivo en la verificación de pertenencia. Al proseguir el recorrido de la fila nos encontramos una línea a_2, es necesario verificar también si es parte de un objeto ya almacenado en B, para ello no es necesario volver a recorrer desde el objeto b_1, pues el objeto más cercano conocido es b_m, los objetos $b_1, b_2 \ldots b_{m-1}$ se encuentran a uno o más objetos de distancia, entonces a_2 sólo puede pertenecer a alguno (o algunos al unir varios) de los objetos $b_m, b_{m+1} \ldots b_n$. Cada vez que descubrimos una nueva línea recorremos el arreglo B a partir del último elemento que ha empatado con una línea descubierta.

Cuando un objeto a pertenece a uno de los objetos almacenados en B igualmente debemos sumar las propiedades de a al objeto contenido en B, para ir formando un solo objeto, del cual calcularemos su centroide al final. Sin embargo, existen dos casos en los que anexaremos objetos: En el primero, una línea puede pertenecer a uno o varios elementos superiores $(b_r, b_{r+1} \ldots b_q)$. En este caso, tenemos que unir el nuevo objeto a al primer elemento superior b_r al cual se le ha encontrado pertenencia; si corresponde a más de un objeto, i.e., un conjunto $b_r, b_{r+1}, \ldots b_q$, entonces los elementos $b_{r+1} \ldots b_q$ serán adheridos a b_r ya que el objeto a actúa como concatenador. El lector encontrará una explicación gráfica de este caso en la figura 5.7.

El segundo caso sucede cuando la línea a_2 ya antes ha sido anexada a un objeto izquierdo a_1. Al encontrar las concordancias superiores vemos que también pertenece a un objeto b_r o a una secuencia de objetos $b_r, b_{r+1}, \ldots b_q$. Este caso puede interpretarse

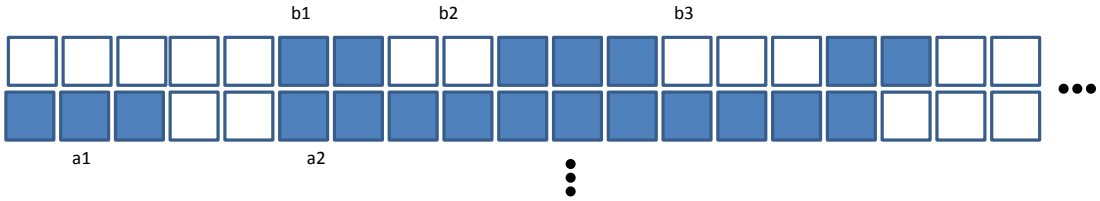

Figura 5.8: Gracias a a_2 todos los objetos mostrados en la figura serán anexadoa a a_1

como lo que sucede al existir las condiciones de pertenencia izquierda y pertenencia superior, mejor ilustrado en la figura 5.8. Como la primera condición en ser evaluada es la pertenencia izquierda, entonces en este momento a_2 se ha anexado a a_1. Al dar afirmativa la pertenencia superior, todos los elementos $b_r, b_{r+1}, \ldots, b_q$ pasan a ser adheridos a a_1.

Cuando no encontramos pertenencia izquierda o superior, la línea a es un nuevo objeto y será registrada como tal.

El algoritmo 3 muestra la forma completa del algoritmo de búsqueda de objetos.

Al observar el algoritmo 3 podemos notar que lo que regresa son los centroides de los objetos contenidos en la imágen y no el objeto en realidad. Esto se debe a que en este trabajo es relevante únicamente el centroide, por lo que hemos desechado

Algorithm 3 Búsqueda de Objetos

Require: byte YUV[], int Width, Height

for $j = 0, yp = 0$ hasta $j = Height$ **do**
  Tope = 0, TopeArriba = 0
  for $i = 0$ hasta $i = Width$ **do**
    $R, G, B = PixelYUVToRGB(YUV, Width, Height, i, j)$
    if $R < 56$ AND $G > 236$ AND $B > 236$ **then**
      $Negro = True$
      if $Blanco$ **then**
        $Primero[Tope] = i$, $Blanco = False$
      end if
    else
      $Blanco = True$
      if $Negro$ **then**
        $Ultimo[Tope] = i - 1$
        $Izquierda = True$, $Arriba = True$
        if $Distancia(Primero[Tope], Ultimo[Tope - 1]) \leq Z$ **then**
          Anexar al centroide izquierdo las propiedades del centroide de la línea delimitada por $Primero[Tope]$ y $Ultimo[Tope]$
          $Izquierda = False$
        end if
        Sean $PrimeroB[]$ y $UltimoB[]$ los contenedores de los puntos que definenen las líneas superiores (Las líneas B)
        while $Ultimo[Tope] \geq PrimeroB[TopeArriba]$ **do**
          if $Primero[Tope] \leq UltimoB[TopeArriba]$ **then**
            $Arriba = False$
            if $Izquierda$ **then**
              Anexar al centroide de la Izquierda el centroide de la línea delimitada por $PrimeroB[TopeArriba]$ y $UltimoArriba[TopeArriba]$
            else
              if Primera iteración del while **then**
                Anexar al centroide limitado por $PrimeroB[TopeArriba]$ y $UltimoArriba[TopeArriba]$ las propiedades del centroide de la línea delimitada por $Primero[Tope]$ y $Ultimo[Tope]$
              else
                Anexar al centroide de la primera iteración el centroide limitado por $PrimeroB[TopeArriba]$ y $UltimoArriba[TopeArriba]$
              end if
            end if
          end if
          $TopeArriba++$
        end while
        if $Izquierda$ AND $Arriba$ **then**
          Registrar un centroide nuevo con las propiedades de la línea limitada por ~~$Primero[Tope]$ y $Ultimo[Tope]$~~
        end if
        $Tope++$
      end if
    end if

las otras propiedades con el fin de ahorrar cálculos y espacio de memoria, aunque no representaría reto alguno el anexo de las propiedades completas del objeto.

5.3 Algoritmo de la piedra en el estanque

Una imágen de 640x480 px propiciaría 307,200 análisis por cada imágen y 2,457,600 análisis por segundo, teniendo el mismo número de accesos a memoria, lo que resulta en el verdadero cuello de botella.

Este problema parecería un punto imposible de eludir, dado que la imágen contiene al objeto a analizar y es obvio pensar que debemos recorrer la imágen completa para encontrar el objeto deseado, pero reflexionemos un momento en la siguiente pregunta: ¿Qué tan lejos está el stylus de su última posición? De hecho si analizamos las figuras 5.4 y 5.5 observamos que durante un trazo las posiciones contiguas de la punta del stylus son relativamente cercanas. La idea sería entonces comenzar el análisis desde el último punto conocido del stylus, expandiendo el área de búsqueda hasta encontrar de nuevo el stylus. Se puede hacer una metáfora al arrojar a un estanque una piedra, la cual creará ondas y en algún momento una de ellas encontrará la orilla. Así mismo, para encontrar la posición actual del stylus, comenzaremos el análisis no desde el primer píxel de la imágen, sino desde la última posición conocida del stylus. No es necesario analizar todos los vecinos circundantes, basta con examinar aquellos que se encuentren a una distancia D entre ellos, formando cuadros. En la figura 5.9 establecemos D=4 y observamos la formación de ondas cuadradas.

Las ondas son soltadas una a la vez. En la figura 5.9 se notan en diferentes colores. Cuando uno de los píxeles marcados en color cumple con la inecuación 5.1 dejamos de analizar los demás y de soltar más ondas, pues el valor verdadero de la inecuación marca que hemos encontrado un objeto parecido al stylus ¿Por qué en la figura 5.9 y en general en el algoritmo usamos ondas cuadradas y no ondas redondas como podría pensarse por la metáfora del párrafo anterior? Esto se debe a la facilidad con que podemos calcular coordenadas cuadradas (con dos simples ciclos for) mientras que si quisiéramos dibujar círculos tendríamos que atenernos a coordenadas polares, la ecuación clásica del círculo o al algoritmo del punto medio.

Al tratar con cuadrados, el área intermedia entre los puntos es igual en todos los casos y así podemos establecer el valor D a un óptimo que no sea tan grande para que la forma buscada pueda caber en los espacios vacíos, ni tan pequeña para que el algoritmo sea idéntico a una búsqueda clásica de 8-vecinos. Bastaría con establecer

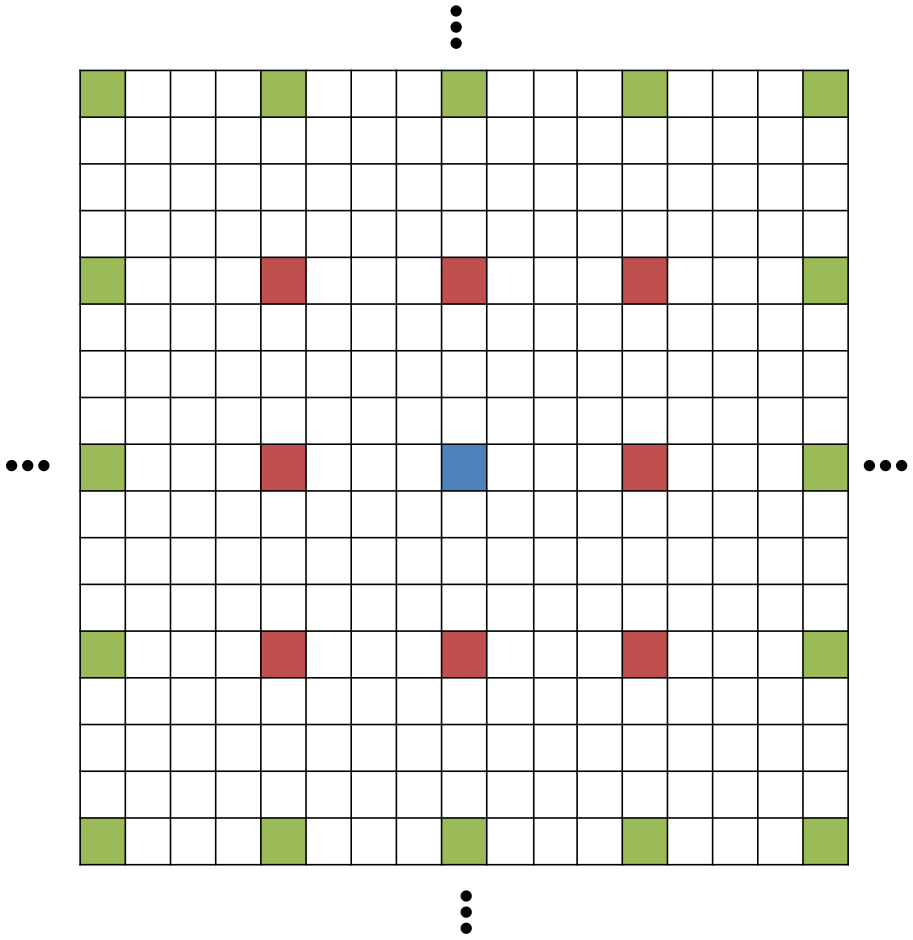

Figura 5.9: Los diferentes colores representan diferentes ondas lanzadas una tras de otra con cierta separación, formando cuadrados cada vez más grandes

$4 \leq D \leq 8$, en los experimentos un valor de 6 ha funcionado perfectamente, sin omitir detalles y permitiendo analizar únicamente 2 de cada 36 píxeles posibles.

Hasta este momento lo que tenemos es únicamente una posición azarosa del objeto a ser candidato de stylus. Ese punto aún no nos dice nada. Volvamos a las metáforas: Si aventáramos la piedra en el estanque no sólo podríamos encontrar un punto de la orilla, si dejáramos transcurrir las ondas podríamos conocer toda la orilla. Con la posición al azar que hemos encontrado podemos aventar de nuevo una piedrita y esta vez pararemos hasta que la inecuación 5.1 se incumpla (anteriormente al buscar un solo punto parábamos cuando esta se cumplía). Se trata entonces de soltar ondas mientras encontremos píxeles que pueden pertenecer al stylus y parar cuando ya no haya zonas de interés.

Como tratamos con ondas cuadradas, sólo podemos obtener estanques a lo más rectangulares. Esta modificación funciona de la siguiente manera: al expandir una onda cuadrada tenemos una línea izquierda, una derecha, una arriba y otra abajo; si todas las líneas cumplen la inecuación 5.1 entonces soltamos una nueva onda más grande, pero si un píxel deja de cumplirla quiere decir que ha tocado un borde donde ya no hay píxeles de la forma de interés. La línea que tenga el píxel dejará entonces de expandirse mientras las demás líneas continúan haciéndolo hasta encontrar un borde. Esta propuesta suena lógica y acorde a la metáfora pues cuando un pedazo de la onda se topa con la orilla del estanque, eso no le impide al resto de la onda seguirse expandiendo.

Como resultado de esta operación tendremos 4 líneas que delimitan el área en la cual se encuentra la forma de interés: una subimágen por decirlo de algún modo, la cual no es mucho mayor que el tamaño de la forma de interés y sobre la cual podemos aplicar el algoritmo de búsqueda de objetos.

Por las mismas razones expuestas al analizar la variable Z, debemos dejar un rango de holgura. No podemos parar al encontrar el primer píxel que incumpla la inecuación 5.1, este píxel podría haber sido producto de una inadecuada umbralización. Establecemos entonces una holgura Zo = 3, para decir que podemos expandirnos hasta 3 ondas más después de haber encontrado un valor de borde, si encontramos de nuevo valores de píxel que cumplen con la inecuación 5.1, entonces se debe reiniciar el índice. Encontrar de nuevo estos píxeles significaría que caimos en un falso positivo debido a los huecos de umbralización y dado que podemos caer más de una vez en estos huecos es necesario reiniciar el índice.

La subimágen obtenida es realmente pequeña en comparación con la imágen original, en promedio 20 veces menor. Los algoritmos de piedra en el estanque son ligeros pues operan sobre el área de la subimágen y al establecer D = 6, sólo se analiza un dieciochoavo de la subimágen (2 de cada 36 pixeles). Esta solución representa una ganancia considerable sobre el paradigma tradicional que analiza por completo todas las imágenes, pues el análisis de una sola imágen en forma tradicional tomaría mas tiempo que el análisis de 10 imágenes con el algoritmo de la piedra en el estanque.

Cabe mencionar que al tomar la primera imágen desconocemos la posición del stylus. Para evitar tener que analizar por completo la imágen podemos aplicar el algoritmo de la piedra en el estanque con punto de origen en el centro de la imágen.

5.4 Discriminación de formas

La unión del algoritmo de piedra en el estanque y el de búsqueda de objetos nos da como resultado todos los objetos que cumplan con las características de interés y que se encuentran en el área compuesta por la subimágen. Sin embargo, sólo uno de estos objetos puede ser el stylus. Para identificar cuál de ellos es el mejor candidato se deben buscar objetos que cumplan con dos condiciones:

- Un tamaño lo suficientemente grande para descartar pequeños brillos
- Un centroide blanco

La primer condición se refiere a descartar todos aquellos objetos de área relativamente pequeña (menor a 25 x 25 px), ya que la luz reflejada por el stylus no es tan pequeña ante la cámara. Por el contrario cuando el stylus se encuentra a la mayor distancia posible (85 cm), el área promedio de la luz emitida es de 125 x 125 px. La segunda condición es observable en la figura 5.4, podemos ver que la periferia del reflejo del stylus se compone por luz azul, mientras que el centro se compone de luz blanca. Como el algoritmo de búsqueda de objetos ya entrega los centroides, entonces sólo verificamos el color del píxel que representa el centroide. Cualquier color distinto del blanco causaría su descarte como candidato.

Por otro lado, puede ocurrir que descartemos todos los objetos de una subimágen por no cumplir las dos propiedades anteriores. En ese caso hemos identificado otra luz azul en la imágen, pero no precisamente la procedente del stylus. Por lo tanto, debemos hacer una nueva búsqueda descartando la zona que ya hemos analizado. Esto se logra al expandir la caja que encierra la subimágen encontrada hasta que un píxel cumpla la inecuación 5.1. El tamaño de esta expansión será guardado por si necesitamos hacer una tercera, cuarta... enésima expansión. El proceso termina cuando encontremos un objeto que cumpla las dos condiciones del párrafo anterior. Este objeto se tomará como el stylus, su centroide como la posición del stylus en el espacio, la cual será heredada a la siguiente imágen para iniciar el algoritmo de piedra en el estanque desde ese punto.

Dos centroides forman un vector el cual se enviará al algoritmo de reconocimiento de caracteres que se encargará de decifrar de que letra o símbolo es el que se está dibujando.

Capítulo 6

Lx-KDabra

Si se analiza desde cierto punto de vista la definición de la tecnología de sexto sentido [Mistry and Maes, 2009], encontramos que en su forma más simple se trata de un sistema de reconocimiento de formas. Más aún, para fines de este trabajo resulta irrelevante reconocer la forma entera pues se cuenta con un stylus de luz puntual, i.e., en todo momento la única forma reconocible debe ser un punto. El hecho de reconocer puntos en el espacio reduce el problema de reconocimiento de formas complejas al reconocimiento del desplazamiento en el tiempo de una luz puntual, pues la diferencia temporal de una sucesión de puntos representa una forma. Este proceso es similar al movimiento de la mano al escribir. A este punto seguramente el lector ya tiene una idea de cómo se pretende tratar el reconocimiento de formas: usaremos reconocimiento de caracteres.

Puede pensarse que el reconocimiento de caracteres y el reconocimiento de formas no varían mucho en la práctica y seguramente para muchos ejemplos así será, pues un carácter terminado (impreso o dibujado a mano) es sólo una forma particular. Sin embargo, en este trabajo, se generan las siguientes cuestiones: ¿Es posible aprovechar la dimensión temporal para optimizar la tarea de reconocimiento? ¿Se necesita realmente conocer todo el carácter? ¿Es necesario y adecuado un sistema complejo de reconocimiento de caracteres para un dispositivo móvil personal? ¿Pueden ejecutarse métodos tradicionales de reconocimiento de caracteres al mismo tiempo que otros procesos de visión por computadora en un dispositivo móvil? Para contestar estas interrogantes se desarrolló Lx-KDabra.

Lx-KDabra es el corazón del presente proyecto. Se trata de un sistema de reconocimiento de caracteres basado en componentes direccionales, i.e. trabaja con cambios de dirección entre puntos en lugar de pixeles. Además Lx-KDabra corre en tiempo real, consume pocos recursos, requiere de pocas operaciones por segundo, necesita un

entrenamiento insignificante en comparación con los algoritmos existentes, es fácilmente paralelizable y fue diseñado para ser personal (reconoce el modelo de escritura de un usuario a la vez).

En este capítulo comenzaremos por describir algunos algoritmos existentes y su evolución, de forma que podamos entender lo complejo de su funcionamiento y la necesidad de simplificarlo. Seguidamente mostraremos las adapataciones pensadas para lo que definiremos como los metodos *online* creando un enlace natural a la siguiente parte del capítulo, nos referimos al funcionamiento básico de Lx-KDabra, donde además se presenta una versión del algoritmo construida con lo hasta ahora planteado. Este algoritmo se enriquecerá con los temás posteriores: resolución de letras conflictivas junto con optimizaciones y mejoras, estos temas son fundamentales pues son los que permiten identificar las letras en una forma eficaz. Finalmente con la adhesión de estos últimos temas daremos a conocer el algoritmo final.

6.1 Antecedentes acerca del reconocimiento de carácteres

Antes de describir el funcionamiento del motor Lx-KDabra, demos una vista rápida a la problemática del reconocimiento de patrones para vincular estos conocimientos a nuestros algoritmos.

El reconocimiento de caracteres es la tarea de transformar un lenguaje representado en forma espacial como marcas graficas en una representación simbólica [Plamondon and Srihari, 2000], convirtiendo una forma escrita de caracteres comprensibles al ser humano en una forma de escritura descifrable por una computadora. El análisis de caracteres impresos suele ser sencillo pues, al no haber gran variación en su forma, mantiene cierta estabilidad en los procesos. Por otro lado, el análisis de caracteres escritos por seres humanos es más complicado pues, aunque un carácter sea dibujado por la misma persona, es probable que varíe la forma, la inclinación y las proporciones. Este tipo de reconocimiento es mejor conocido como "reconocimiento de caracteres a mano alzada", el cual se utiliza ampliamente en sistemas NUI.

Dada la ubicuidad en las transacciones humanas, el reconocimiento de caracteres a mano alzada adquiere un significado práctico en toda actividad que pueda usar la escritura como entrada [J.Ashok, 2010], lo que atrajo rápidamente la atención de los desarrolladores de interfaces de usuario. Al tratarse de una rama separada de las interfaces de usuario, el reconocimiento de carcateres desarrolló características y temáticas propias, pero no todas ellas útiles al desarrollo de NUI. Dentro del reconocimiento de caracteres se encuentran dos grupos claros: *offline* y *online*

[Plamondon and Srihari, 2000].

El reconocimiento de caracteres *offline* tiene como tarea principal la decodificación de caracteres impresos, usando algún método de escanéo y aplicando técnicas de tratamiento digital de imágenes [Plamondon and Srihari, 2000].

El reconocimiento de caracteres *online* consiste en reconocer una letra introducida por medio de cualquier medio que permita transmitir a cada instante la posición de la pluma electrónica o del puntero [Aparna et al., 2004] [Deepu et al., 2004] haciendo posible el proceso de reconocimiento en forma concurrente [Bahlmann, 2006] [Yoshida and Sakoe, 1982] [Plamondon and Srihari, 2000]. La técnica de reconocimiento *online* es ampliamente usada en celulares, tablets, digitalizadoras y dispositivos de pantalla táctil [Srihari et al., 2007].

Esta última subcategoría es sobre la que trabajaremos, pues se puede aprovechar un stylus como método de entrada al igual que una digitalizadora lo hace con su pluma electrónica. En ambos casos el aspecto importante es el seguimiento de la pluma o stylus, por lo que podemos aplicar a nuestro caso los algoritmos desarrollados para la digitalizadara i.e., reconocimiento *online*. En la actualidad, estos algoritmos deberían ser pensados con base en las necesidades de los dispositivos móviles, i.e., rendimiento del sistema en dispositivos lentos y conservación de la batería [Golubitsky and Watt, 2008]. En el resto del capítulo, daremos solución a esta problemática con un algoritmo *online* pensado en las necesidades de los dispositivos móviles.

6.2 Adaptación de los métodos *online*

Dentro del reconocimiento *online*, el método más utilizado se conoce como emparejamiento elástico (*Elastic Matching*). Consiste en establecer correspondencias y sumar distancias entre los puntos de referencia de caracteres conocidos y los puntos de entrada obtenidos por la discretización en tiempo real de la letra dibujada por el escritor [Mitoma et al., 2005]. De esta manera, puede verse como un problema de optimización con la siguiente función objetivo:

$$\frac{1}{I}\sum_{i=1}^{I}\left\|r_i - e_{j(i)}\right\|$$

Variables de control
$j(1), ..., j(i), ..., j(I)$

$$\begin{gathered}\text{Condiciones}\\ j(i) - j(i-1) \in \{0, 1, 2\}\\ j(1) = 1\\ j(I) = J\end{gathered}$$

Aunque la técnica presenta buenos resultados, en la mayoría de los casos el fenómeno conocido como *overfiting* provoca resultados erróneos. Este fenómeno se produce cuando el escritor deforma demasiado su letra, volviéndola muy estrecha o muy amplia: De hecho el símbolo escrito se aleja demasiado de la letra conocida engañando a la función objetivo, ya que las distancias respecto al patrón guardado se vuelven muy grandes [Mitoma et al., 2005] [Ahn et al., 2009] [Uchida and Sakoe, 2003] [Uchida and Sakoe, 2005]. Además, el costo computacional no es despreciable pues para tener un buen funcionamiento el algoritmo necesita por lo menos 60 puntos.

Tomando en cuenta esta cuestión, debemos buscar un nuevo paradigma para resolver el problema de reconocimiento. La idea del emparejamiento elástico es tomar una secuencia de puntos para compararlos, pero si se piensa un poco en cómo son dibujados los caracteres, se puede decir que estos pueden verse no como una secuencia de puntos, sino más bien como un conjunto de líneas, ya que un carácter puede ser representado como una serie de secuencias angulares o vectores [Yoshida and Sakoe, 1982].

Muchos métodos de reconocimiento de caracteres *online* usan la información de las direcciones que componen una letra o símbolo para reconocerlo [Kato et al., 1999] [Xue Gao and Huang, 2001], bajo el principio de que la manera de escribir una letra puede cambiar algunas características de su forma pero no de su estructura primaria [Kamakhya Gupta and Viswanath, 2007]. Esta técnica, conocida como análisis por componentes direccionales, permite comparar carácteres con base en los vectores que los componen, disminuyendo los cálculos necesarios pues cada carácter se compone normalmente de 6 a 15 vectores. Para tener una noción básica de cómo trabaja este tipo de algoritmos, la figura 6.1 permite observar cómo se conforma un prototipo de letra usando un método tradicional como el emparejamiento elástico y la técnica de análisis de componentes direccionales.

Las ventajas de los vectores pueden ser aprovechadas siempre y cuando haya una forma fácil de obtenerlos. En nuestro caso, existe una forma sencilla pues un vector es la diferencia entre dos posiciones consecutivas del stylus.

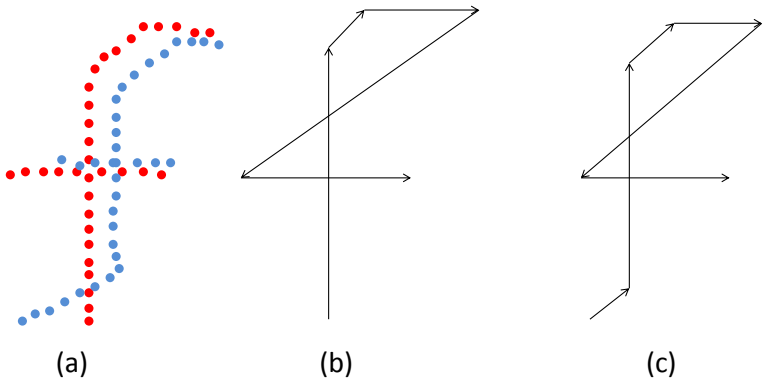

Figura 6.1: En (a) se observa en rojo los puntos que componen la letra "f" conocida y sobrepuesto en azúl el conjunto de puntos de una letra introducida por el usuario. En (b) la misma letra "f" almacenada pero representada por sus vectores. En (c) la letra "f" dibujada en azúl en (a) pero representada por sus vectores. El número de muestras al usar vectores es claramente menor

Como la mayoría de los algoritmos de reconocimiento de patrones, los algoritmos que usan el análisis de componentes direccionales toman una letra ya terminada en una imágen, extraen la información útil, la procesan y comparan el resultado para determinar de qué letra se trata [Golubitsky and Watt, 2008]. Sin importar el enfoque, la extracción y comparación de vectores conlleva un alto costo de procesamiento que, si bien para una computadora personal es tolerable, en un dispositivo móvil podría tornarse en un cuello de botella, aún peor si se requieren hilos concurrentes.

Nuestra propuesta consiste en tomar la idea básica de la técnica de análisis de componentes direccionales y modificarla para trabajar en tiempo real. Esta modificación es posible dado que captamos la luz puntual del stylus en intervalos discretos de tiempo en una fotografía. La diferencia de posición de la luz en dos fotografías subsecuentes constituye un vector. El conjunto de vectores componen una letra, pero no es necesario conocer todos los vectores para iniciar el análisis. Considere el ejemplo de las letras "b" y "d" y los vectores que las conforman (Figura 6.2). Es notable que a pesar de que ambos caracteres se parecen, sus vectores son contrarios. Si cada vez que surge un nuevo vector este es comparado con una base de caracteres vectorizados, se puede diferenciar entre ambas letras sin esperar a poseer todo el juego de direcciones pues, desde los primeros análisis, la diferencia será notoria.

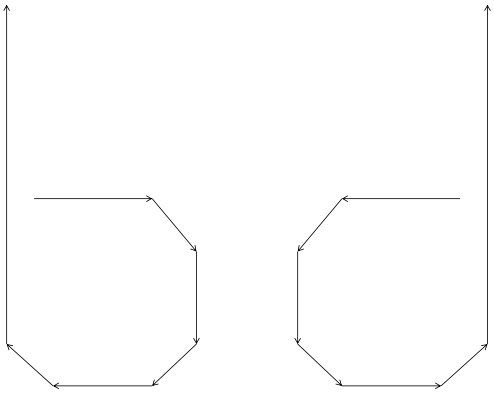

Figura 6.2: “b” y “d” son reflejo una de la otra, lo cual puede engañar a muchos algoritmos invariantes a rotación. Al usar la dirección de los vectores, la diferencia es notoria desde los primeros trazos.

Pese a que las letras “b” y “d” poseen la misma forma, al análizar las direcciones que las componen, las letras pueden verse totalmente distintas dependiendo de quien la dibuje, e.g., alguien podría dibujar una “o” en el sentido de las agujas del reloj y otra persona podría dibujarla en sentido contrario [Chang et al., 2010]. En este momento, hace su aparición la segunda propuesta para el algoritmo. Si se quiere reconocer una letra sería necesaria una gran cantidad de muestras de diferentes estilos de escritura, pero se tiene una ventaja muy importante para limitar el alcance del algoritmo: trabajamos sobre un dispositivo móvil. En otras palabras se piensa que un dispositivo móvil es usado la mayor parte del tiempo por un sólo individuo y en ocasiones esporádicas por un grupo selecto de amigos. Limitar este punto permite reconocer sólo algunos estilos de escritura. Para reconocer el estilo de cada individuo se ocupará un ejemplar simple de su escritura [J.Ashok, 2010] [Connell, 2000], e.g., en nuestro caso se pide al usuario escribir un párrafo de donde extraeremos todos los caracteres necesarios, lo que reduce el número de muestras necesarias por carácter [Kamakhya Gupta and Viswanath, 2007] [Vuori et al., 2001]. Las muestras son almacenadas en archivos que son leídos al arrancar el programa y puestas en memoria para su posterior pareo y análisis.

6.3 Funcionamiento básico de Lx-KDabra

Ya establecidas las características básicas del modelo de reconocimiento a usar, procederemos a desarrollar nuestro algoritmo de análisis de componentes direccionales y clasificaremos el espacio bidimensional en 12 regiones dentro de las cuales caen todos los posibles vectores. [Singh et al., 2010b] muestran que 12 regiones son suficientes para clasificar los cambios de dirección de un carácter. La figura 6.3 muestra esta división y los valores correspondientes a cada región, mismos que son usados en el

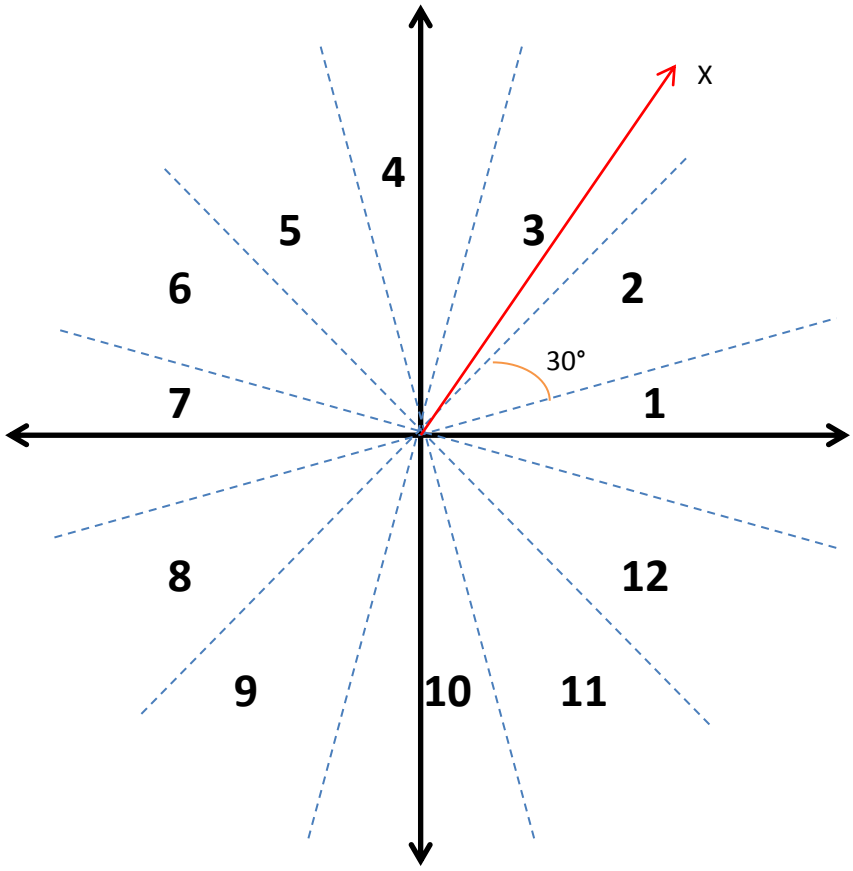

Figura 6.3: El espacio que ocupa cada vector se divide en 12 regiones angulares de 30 grados cada una con centro en el orígen del vector. Los vectores se consideran pertenecientes a una región si su pendiente se encuentra entre las pendientes máxima y mínima de esa región. El vector "X" pertenece a la región 3 pues cae en su espacio.

código del programa y para nombrar a las regiones en el resto del capítulo. La clasificación es sencilla pues sólo depende de la pendiente y del cuadrante en que se localiza el último punto (el primero se toma como origen).

Considerando la división del espacio, un arreglo de enteros que representa los cambios de dirección (región) por cada letra que se reconoce y un índice para cada arreglo, el principio del algoritmo sería el siguiente:

"Dado un nuevo vector, se compara con los vectores contenidos en el arreglo de cada letra en la posición marcada por el índice; los que sean iguales avanzan a la siguiente posición y actualizan el índice; los que sean distintos no actualizan el índice y permanecen en su posición".

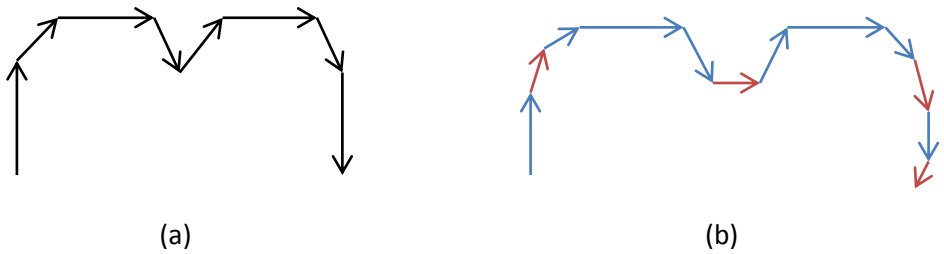

Figura 6.4: (a) Expone una letra "m" vectorizada y almacenada. (b) ejemplifica el trazo vectorizado de una letra "m" dibujada por el usuario; los vectores en azúl marcan coincidencias y los vectores en rojo representan inconsistencias. Es apreciable que se conserva la estructura básica pese a las variaciones.

Gracias a que una región abarca un conjunto de vectores, el principio puede funcionar con tolerancia a ruido y a pequeñas variaciones en la dirección de un vector. La idea del principio es aún inmadura, ya que sólo permite reconocer aquellas letras que tengan gran parecido con las originalmente asignadas como patrones. Las personas mantienen un estilo de escritura (forma de dibujo y cambio de direcciones) pero no por ello todas las letras dibujadas por un mismo escriba son iguales, por el contrario suele haber pequeñas diferencias de una letra a otra. Para que nuestro algoritmo sea robusto a estos cambios, es necesario ampliar la flexibilidad al ruido.

El desarrollo de esta flexibilidad implica la comparación de vectores no solo con la dirección marcada por el índice, sino también con un número de vectores posteriores. Llamaremos a este número de vectores posteriores Ω, que corresponde al hecho de que una letra puede variar en ciertos vectores pero conserva la estructura básica, i.e., en el arreglo que contiene la letra conocida habrá vectores que forzosamente deben aparecer en el mismo orden, pero no necesariamente en la misma posición (ver figura 6.4).

Ahora se puede reescribir el principio del algoritmo de la siguiente forma:

"Dado un nuevo vector, se compara con los vectores contenidos en el arreglo de cada letra en la posición marcada por el índice y hasta Ω pasos; si los vectores son iguales avanzan a la siguiente posición y actualizan el índice sumando n ($n = 1, 2 \ldots \Omega$) siendo n el paso donde se encontró el vector para esa letra; los vectores distintos no actualizan el índice y permanecen en su posición".

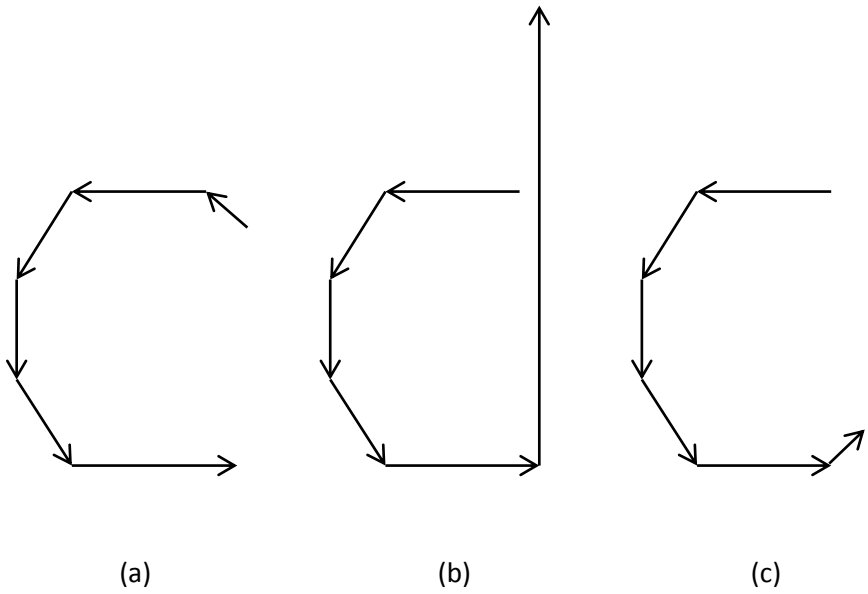

Figura 6.5: (a) y (b) representan respectivamente las letras “c” y “d” guardadas. (c) ejemplifica un patrón de vectores dibujados como entrada. Claramente se trata de una letra “c” pero los arreglos “c” y “d” poseen el mismo número de aciertos.

Los aciertos de un arreglo en particular que se producen al dar positivo en la comparación entre uno de sus vectores almacenados y el vector actual aproximan al arreglo a ser el contenedor de la letra buscada. La distinción de un carácter no es tan sencilla pues, como se observa en la figura 6.5, dos letras pueden tener el mismo número de aciertos pero ser claramente distintas. Este efecto es conocido como “ruido diferencial” al cual representaremos como una magnitud escalar discreta y cerrada en un intervalo.

Para medir el ruido diferencial en un carácter es necesario definir una operación sobre la dirección de los vectores. Llamaremos a esta operación “diferencia radial” la cual corresponde a la distancia radial mínima entre dos regiones, e.g., la diferencia radial entre las regiones 4 y 1 es 3, tomando el sentido contrario a las manecillas del reloj por ser la ruta más corta. Por otro lado la diferencia radial de las regiones 2 y 11 es 3 ya que la ruta más corta es la obtenida al recorrer en sentido de las manecillas del reloj. Esta diferencia entre regiones puede pensarse como geometría modular, la cual está mejor esquematizada en la figura 6.6.

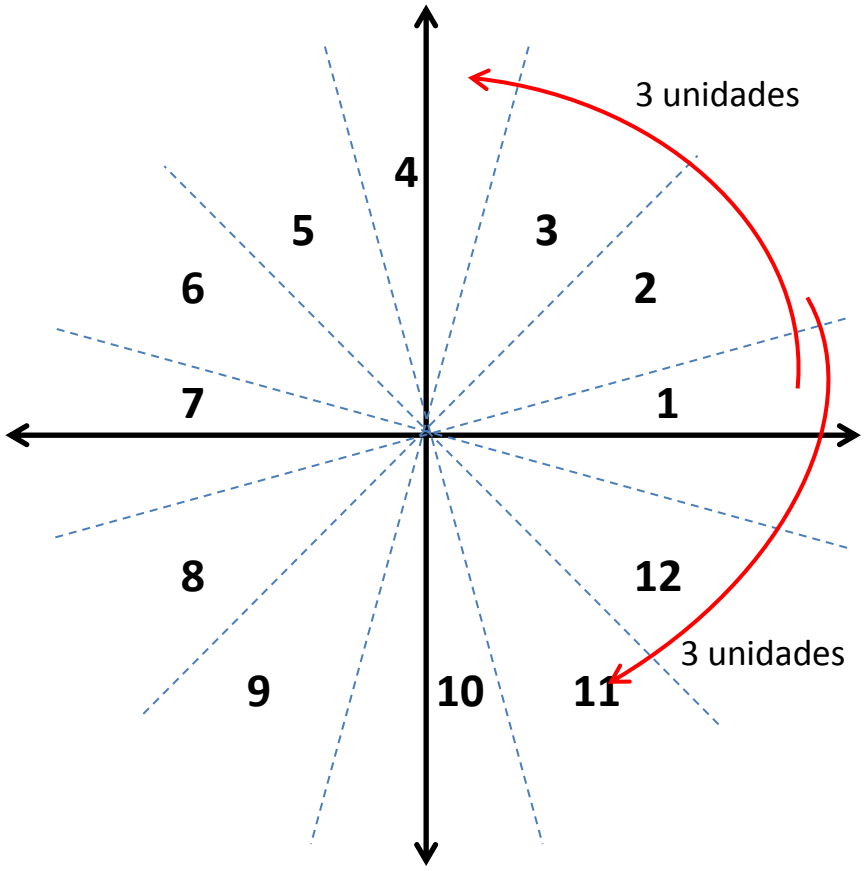

Figura 6.6: La diferencia radial devuelve la menor distancia entre 2 regiones independientemente del sentido.

Existen cuatro situaciones en las que aparece ruido diferencial, en consecuencia la operación de diferencia radial toma distintas regiones como parámetros en cada caso. Los valores arrojados por esta operación son almacenados en un arreglo de errores que sirve para determinar el parecido con la letra a identificar.

Caso 1. Se ha avanzado n pasos de los Ω posibles y en el paso n aparece el vector buscado. En este caso, se aplica la diferencia radial entre el último vector marcado por el índice y los $n-1$ vectores saltados del arreglo hasta descubrir el buscado. Un ejemplo de este caso sería el de los caracteres "b" y "6" pues el carácter "b" podría presentar una ligera curva en la parte izquierda y seguir siendo la letra "b", pero si la curva es muy abierta (hay demasiado ruido radial) se debe considerar el trazo como un carácter "6" y penalizar el arreglo de "b" (ver figura 6.7).

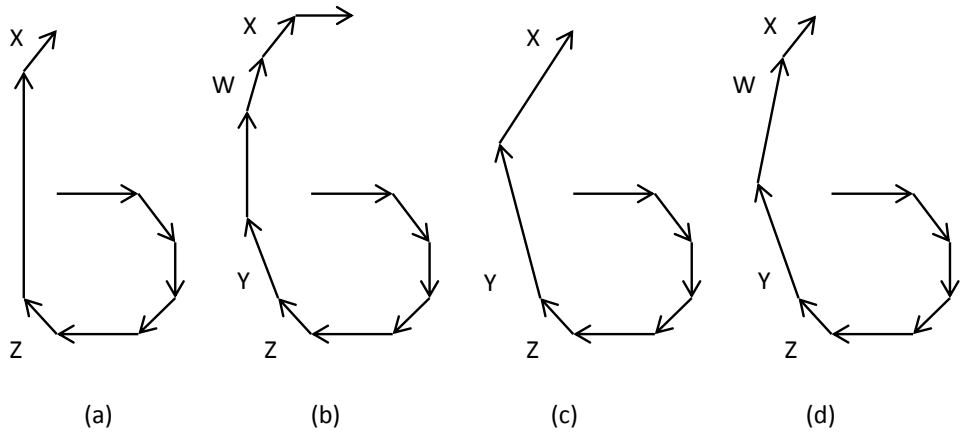

Figura 6.7: (a) y (b) muestran respectivamente los caracteres "b" y "6" almacenados. Los vectores "X" y "Z" prevalecen en los cuatro casos por lo que serán coincidencias siempre que se quiera dibujar los caracteres "b" o "6". En el caso de (c) el vector "Y" no existe en el patrón del carácter "b" mostrado en (a) lo que provoca un error, pero sólo se ha avanzado un paso desde el vector "Z" y hasta encontrar el vector "X", la diferencia radial entre "Y" y "Z" es 1. El arreglo del carácter "b" tiene un error contra varios aciertos por lo que aún presenta parecido con la letra "b". En el caso de (d) hay una curvatura mayor gracias a los vectores "W" y "Y", cada uno con una diferencia radial de 1 con respecto a "Z", lo que aumenta el error respecto al carácter "b" pero lo acerca al carácter "6" pues "Y" y "W" existen en el patrón del carácter "6".

Caso 2. El índice apunta al final del arreglo pero siguen llegando vectores nuevos. Para este caso se aplica la diferencia radial entre la última dirección del arreglo y la dirección del nuevo vector. Para ejemplificar este caso, la figura 6.8 toma las letras "c" y "d" que se dibujan de forma similar, pero son los últimos vectores los que definen de qué letra se trata.

Caso 3. Se analizó el nuevo vector durante Ω pasos, pero no se pudo parear con ningún elemento del arreglo. Como es un nuevo vector que no encajó en la descripción, entonces se aplica la diferencia radial entre el vector al que apunta el índice en el arreglo y el nuevo vector. Este tipo de error se da sobre todo al inicio de una letra o en letras poco parecidas, e.g., las letras "y" y "g" tienen un trazo muy similar que sólo se distingue en la primera etapa, como se muestra en la figura 6.9.

Caso 4. Terminado el análisis en tiempo real notamos que el carácter dibujado tiene menos vectores en comparación con el *i-ésimo* carácter en nuestra base. En este caso, el error es provocado por letras que al inicio son iguales pero después de ciertos

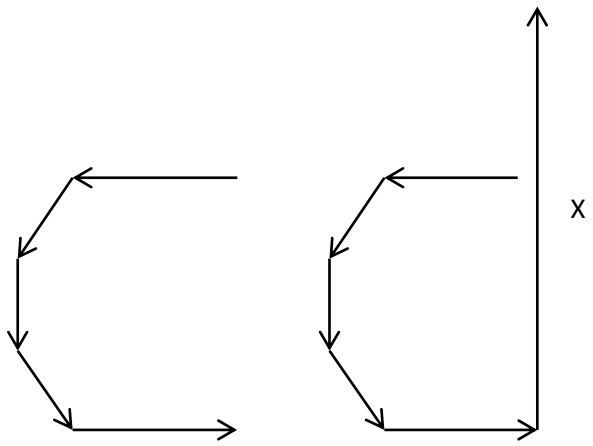

Figura 6.8: La única diferencia entre las letras “c” y “d” es el vector “X” distintivo de la letra “d”.

vectores comienzan a diferenciarse, como el caso de las letras “l” y “L”. Para no confundir ambas letras, la diferencia radial es aplicada entre el último vector del trazo y todos los vectores restantes de la letra almacenada.

Los resultados entregados por la operación de diferencia radial se suman a una variable de error propia de cada arreglo (pues cada uno contiene un carácter distinto) a la cual se restan los aciertos provocados por los vectores que empataron. Tomando en cuenta que la diferencia radial puede arrojar valores entre 0 y 6, es necesario llevar los aciertos al mismo espacio de valores. El valor de un acierto entonces será resuelto como $7 - n$ donde n es el número de pasos y $n = 1, 2 \ldots \Omega$. Esta expresión significa que el valor Ω se encuentra en el rango $[1, 6]$ pues después de 6 los valores se volverían nulos o negativos y no se considerarían aciertos sino errores.

En el momento en que dejen de llegar nuevos vectores, los errores son analizados (considerando que ya le hemos restado los aciertos); la letra buscada, será aquella representada por el arreglo con menor cantidad de errores. Una tendencia negativa de la variable de errores indica gran parecido con la letra, mientras una tendencia positiva indica lo contrario.

Nuestro algoritmo de análisis por componentes direccionales en tiempo real entonces queda escrito de la siguiente forma:

Algorithm 4 Reconocimiento por direcciones

Require: un conjunto de vectores dibujados, la matriz LETRAS contienes los vectores de todas las letras reconocibles

Ensure: el carácter más parecido a la entrada

```
1: while Vector ≠ null do
2:   if Vector ≠ VECTORANTERIOR then
3:     for i = 0 hasta i = LETRAS.lenght do
4:       if INDICES[i] == LETRAS[i].lenght then
5:         ERRORES[i]+ = DiferenciaRadial(Vector, LETRAS[i][INDICES[i]−
           1]) //Caso 2
6:       else
7:         TER = true
8:         for j = 0 hasta j < Ω do
9:           if INDICES[i] + j == LETRAS[i].lenght then
10:            Break
11:          else if LETRAS[i][INDICES[i] + j] == Vector then
12:            ACIERTOS[i]+ = 7 − j
13:            for k = INDICES[i] hasta k < INDICES[i] + j do
14:              ERRORES[i]+ = DiferenciaRadial(Vector, LETRAS[i][k])
                 //Caso 1
15:            end for
16:            INDICES[i]+ = j + 1
17:            TER = false
18:          end if
19:        end for
20:        if TER then
21:          ERRORES[i]+ = DiferenciaRadial(Vector, LETRAS[i][INDICES[i]])
             //Caso 3
22:        end if
23:      end if
24:    end for
25:  end if
26: end while
27: MENOR = ∞
28: for i = 0 hasta i = LETRAS.lenght do
29:   for j = INDICES[i] + 1 hasta j < LETRAS[i].lenght do
30:     ERRORES[i]+ = DiferenciaRadial(Vector, LETRAS[i][j]) //Caso 4
31:   end for
32:   ERRORES[i]− = ACIERTOS[i]
33:   if ERRORES[i] < MENOR then
34:     MENOR = ERRORES[i]
35:     I = i
36:   end if
37: end for
38: return EL CARACTER BUSCADO APUNTADO POR EL INDICE I
```

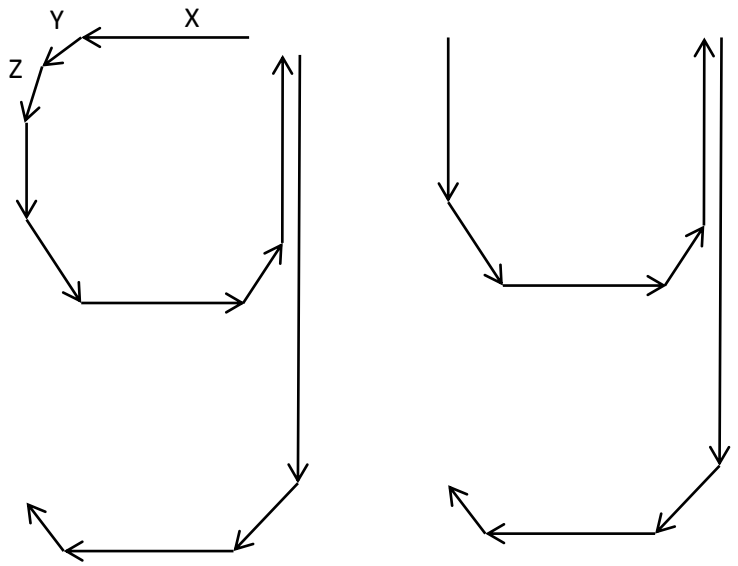

Figura 6.9: Figura kk.8. Sería fácil distinguir una "g" de una "y" pese a su gran similitud en trazado, gracias a los vectores "X", "Y" y "Z" pues si se intenta dibujar una letra "g", estos vectores no encontrarán coincidencia alguna en "y" aún después de Ω pasos ($\Omega < 7$).

Como se puede ver, el algoritmo sólo responde a los cambios de dirección i.e., aunque se tomen muchas muestras en un segundo, estas no son consideradas hasta que ocurra un cambio. Los caracteres más ligeros se conforman de 1 o 2 vectores como las letras "l" o "v", mientras que los carácteres más pesados pueden tener hasta 20 vectores como el número "8". Estas cifras son considerablemente bajas al reflexionar que una red neuronal toma aproximadamente un millardo de píxeles como entrada y que un algoritmo *online* tradicional, como el emparejamiento elástico, necesita entre 60 y 180 muestras. La distinción de cambios en la dirección y no en segmentos continuos de la misma dirección, como lo haría un algoritmo tradicional de análisis de componentes direccionales, garantiza la invariancia a traslación, escalamiento y tiempo de trazado.

6.4 Resolución de letras conflictivas

El algoritmo 1 basta para distinguir la mayor parte de los símbolos de un alfabeto, suponiendo que estos han sido bien trazados. Sin embargo, por razones externas al escriba y al sistema, puede suceder que una letra se aleje de su estructura normal y presente desviación en alguna o varias secciones del esqueleto (aunque queda demostrado que estas desviaciones normalmente sólo ocurren en lateral o en vertical [Kato et al., 1999]). Como se mencionó al principio de la sección 6.3, la capacidad del sistema de adaptarse a dichos cambios se conoce a tolerancia a ruido. Existe una segunda causa importante de ruido que podría volver a nuestro algoritmo vulnerable:

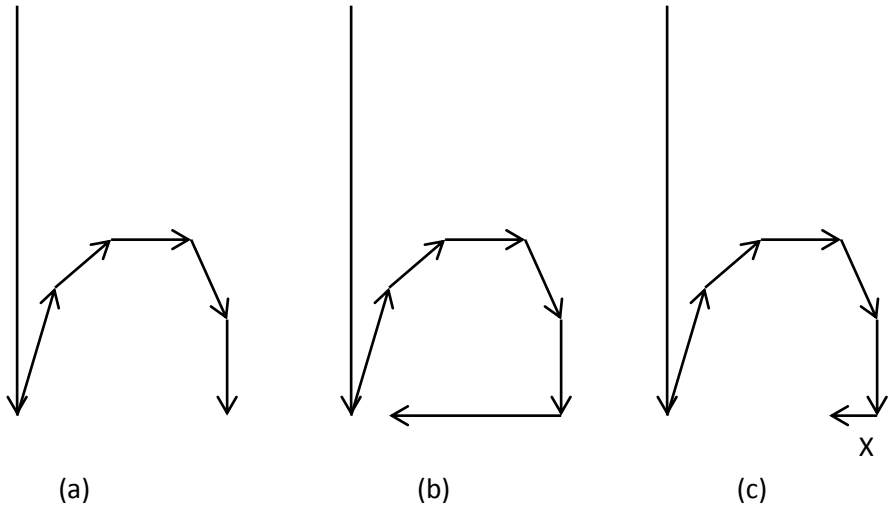

Figura 6.10: (a) y (b) muestran respectivamente las letras "h" y "b" almacenadas. El trazo realizado por el usuario se observa en (c), el vector "X" es un pequeño error final que puede confundir al algoritmo hacia la letra "b".

la similitud de letras y trazos. Sean dos letras X1 y X2 que comparten un trazado en común, el cual representa en ambos casos la mayor parte del carácter. Si el usuario trata de dibujar el carácter X1 pero comete un error, es posible que la semejanza final apunte al carácter X2 de forma errónea. En el ejemplo ilustrado en la figura 6.10, el carácter que el usuario quiere dibujar es claramente la letra "h", pero el trazo final podría provocar que el algoritmo se equivoque, ya que toma sólo los trazos sin importar las dimensiones del mismo.

La resolución del dilema propuesto por la 6.10 es sencilla al tomar las siguientes características básicas y facilmente computables que ya han sido comprobadas como características distintivas de una letra [J.Ashok, 2010] [Liu and Zhou, 2006]:

- Relación de aspecto
- Dirección final
- Centroide

Al momento de guardar los caracteres en la base de símbolos, se hace el cálculo de estas propiedades individuales, las cuales son almacenadas junto con los vectores que las conforman para tenerlas a disposición. Es necesario mencionar que, para conocer los valores válidos de las variables, se toma en cuenta una mayor cantidad de puntos diferentes entre sí, que no necesariamente cumplen la condición de pertenecer a un

vector distinto.

Resulta evidente que el cómputo de estos valores comprometería las invariancias a traslación y escala antes planteadas. Con el fin de mantener todas las propiedades del algoritmo intactas, se vuelve indispensable aplicar un proceso de normalización. La normalización reduce la variación de forma regulando el tamaño, la posición y la forma de cada carácter respecto a su par conocido [Srihari et al., 2007]. Es una técnica usada comúnmente en algoritmos de reconocimiento, pues permite conservar invarianzas y hacer operaciones sin alterar el resultado. Pese a ello, se debe tener cuidado en su manejo, debido a que el rendimiento de un algoritmo de reconocimiento de caracteres a mano alzada suele estar asociado con la correcta normalización de la forma [Srihari et al., 2007] [Liu and Zhou, 2006].

La mayoría de los trabajos relacionados utilizan espacios normalizados bidimensionales cuadrados de tamaño igual a una potencia de dos, siendo especialmente populares los formatos de 64 y 128 unidades [Kato et al., 1999] [Xue Gao and Huang, 2001] [M. and Sridhar, 2007] [Chang et al., 2010] [BAI and HUO, 2005]. La normalización propuesta no desea deformar la letra a un espacio cuadrado, sino conservar la relación de aspecto y usar un origen particular. Dado que no aplican los espacios normalizados de trabajos anteriores, se crea uno propio donde las variables son llevadas a un espacio común de 100 × 100 unidades, tal cual lo marcan las ecuaciones 6.1, donde P es el punto a normalizar. Se usa una ventana con estas proporciones debido a la facilidad didáctica que ofrece, sin embargo, se puede usar una ventana de 64 o 128 de manera que las multiplicaciones y divisiones de las ecuaciones 6.1 se conviertan en corrimientos, obteniendo una ligera optimización. Hay que tomar en cuenta que si el ancho es menor que el alto entonces hay que invertir rx y ry.

$$\begin{aligned} rx &= 100/ancho \\ ry &= ((alto/ancho) * 100)/alto \\ P_x &= (P_x - Origen_x) * rx \\ P_y &= (P_y - Origen_y) * ry \end{aligned} \tag{6.1}$$

Las ecuaciones 6.1 transforman el trazo a un espacio normalizado cuyo origen es el primer punto trazado y cuyo lado más largo es de 100 unidades, en tanto que el más corto está en proporción del primero.

Para hacer la discriminación de caracteres similares, cada una de las variables se calcula e influencia el algoritmo de la siguiente manera.Se calcula el valor absoluto de la relación de aspecto R. Para el 97 % de las letras se tiene que $0.5 < R < 2.0$, lo que convertiría a la relación de aspecto en un parámetro despreciable. Por lo tanto, se multiplica por 10 la diferencia resultante entre la relación de aspecto del trazo r y la del carácter r_i con el que estamos comparando, después sumar a los errores acumulados el valor resultante R que se observa en la ecuación 6.2.

$$R = Abs(r_i - r) * 10 \tag{6.2}$$

La dirección final DF se refiere al resultado de la operación de diferencia radial entre el primer y el último punto del trazo. Al igual que en el caso anterior, el valor obtenido es muy pequeño para considerarlo discretizante ($0 \leq DF \leq 6$) por lo que es necesario multiplicarlo por 10. La operación sobre la dirección final no representa aberración alguna para los casos extremos $DF = 4, 5, 6$, pues esto indica que la letra está excesivamente deformada en comparación con el patrón de vectores; por lo tanto no se trata de la letra buscada. Al error acumulado se suma el valor absoluto de la diferencia entre la dirección final del trazo df y la dirección final df_i de la letra conocida y almacenada con la que comparamos, como se muestra en la ecuación 6.3.

$$DF = DistanciaRadial(df_i, df) \tag{6.3}$$

El centroide es simplemente la suma de todos los puntos entre el número de los mismos, cuyo resultado es un punto en el espacio normalizado que representa la mayor concentración de masa. De la experimentación se tiene que la distancia, entre un centroide calculado para un carácter de la base de símbolos almacenados y los diferentes trazos que pueda presentar posteriormente el mismo carácter dibujado, no suele ser mayor que 20 unidades. De hecho, en la mayoría de los casos, la distancia entre centroides C, suele estar acotada como $0 < C < 12$. Al tratarse de valores grandes, el centroide no necesita ningún tratamiento especial. La ecuación 6.4 se usa para obtener C, tomando como c el centroide del trazo y como c_i el centroide del *i-ésimo* carácter con el cual se compara.

$$C = \parallel c_i - c \parallel \tag{6.4}$$

En la ecuación 6.4 se usa una distancia euclidiana, pero sería igualmente correcto usar una distancia *city block* [Kato et al., 1999] para acortar los cálculos.

6.5 Optimizaciones y mejoras

Sería infactible realizar todos los cálculos presentados para todos los caracteres de la base de carácteres almacenados. Dado que es posible conocer por anticipado los

caracteres que no son candidatos, entonces resulta viable hacer una clasificación ruda de la que se obtendrán sólo los caracteres que tienen la posibilidad de concluir como una respuesta satisfactoria al trazo. Esta idea de primero clasificar a los candidatos y después operar sobre ellos de una forma más compleja, mediante una clasificación fina, ya ha sido tratada y demostrada en muchos otros artículos [Kato et al., 1999]. Haciendo una mejora a los metodos existentes, se dividirá la clasificación ruda en dos: clasificación ruda en tiempo real y clasificación ruda por dirección.

En el caso de la clasificación ruda en tiempo real, tenemos una variable λ, donde $0 \leq \lambda \leq 1$, que representa el índice de tolerancia a fallos en porcentaje. El acumulado de errores siempre debe satisfacer la inecuación 6.5; en caso contrario se descarta al carácter analizado como un candidato a solución. El valor de λ debe ser establecido a criterio, ya que un valor de 1 o muy cercano a él permitiría cualquier deformación derivando la clasificación en operaciones inútiles, mientras que un valor de cero o cercano a él inferiría un clasificador muy estricto que no tolera deformaciones en comparación con el carácter de la base de símbolos. De la experimentación, se concluye que un valor de $\lambda = 0.5$ elimina la mayoría de las letras que obviamente no son solución, pero jamás elimina la solución del rango de candidatos.

$$Error_i < \lambda * K_i \tag{6.5}$$

Donde:
K_i es el número de vectores del *i-ésimo* carácter guardado.

La clasificación ruda en tiempo real se efectúa en cada ciclo que involucre la adición de errores, en tanto que la clasificación ruda por dirección se realiza una vez terminado el análisis de vectores en tiempo real y únicamente para los posibles candidatos. La clasificación ruda por dirección consiste en una refinación de los candidatos del proceso previo, a partir de la dirección final descrita en la sección 6.4, de forma tal que $0 < DF < \delta$, siendo δ una segunda constante de elasticidad a ruido, la cual se describe como la facilidad con la que la parte final de un símbolo puede quedar en una sección distinta a la del patrón almacenado. El trazo de una letra "a" puede ser diferenciado de una letra "d" con esta propiedad, estableciendo adecuadamente δ. Para un buen refinamiento se establece $\delta = 2$, pues un valor mayor provocaría confusiones entre "a" y "d", "6" y "b" y otras por el estilo.

Aquellos caracteres que pasen ambas clasificaciones constituyen un grupo reducido que es evaluado después de agregar los valores de las características propuestas en

la subsección 6.4, de forma que el carácter con un menor número de errores y mayor cantidad de aciertos, es el más parecido a la letra dibujada.

Experimentalmente, el carácter dentro de la base de símbolos que mejor se asemeje al trazo evaluado suele poseer un valor de errores muy pequeño, mientras que todos los demás símbolos se despegan de este valor por diferencias notables.

6.6 Algoritmo final

El algoritmo 5 refleja todas las características expresadas en este capítulo de forma conjunta. Gracias al proceso de clasificación ruda en tiempo real, aquellos carácteres que cuentan con poco parecido al trazo del usuario comienzan a ser descartados, de forma que ahorran una gran cantidad de cálculos, pues tan pronto como son detectados se descarta y el error total de su arreglo es llevado al infinito (en la práctica basta con un valor muy alto). Al terminar el ciclo en tiempo real es necesario recorrer todo el arreglo de errores y determinar el menor. El recorrido no debería ser un problema de procesamiento, pero al agregar las operaciones de la sección 6.4 se debe pensar en ejecutarlas lo menos posible. Con la ayuda de la clasificación ruda en tiempo real, en este punto del algoritmo ya sólo se tienen caracteres candidatos que presentan una similitud aceptable entre ellos. El grupo de candidatos se reduce aun más al usar la clasificación ruda por dirección que evita que carácteres con trazos similares, pero con una deformación significativa, sean borrados del grupo de candidatos. Finalmente la clasificación fina se aplica a un conjunto reducido de caracteres, operando sobre las características básicas de la forma, tales como el centroide, la relación de aspecto y la dirección final.

Algorithm 5 Reconocimiento por direcciones optimizado

Require: un conjunto de vectores dibujados, la matriz LETRAS contienes los vectores de todas las letras reconocibles. CENTROIDES, DIRECCIONFINALES y RELACIONESASPECTO son arreglos que contienes estas propiedades de cada letra guardada

Ensure: el carácter más parecido a la entrada.

1: $NUMVECTORES = 0$
2: $PUNTOORIGEN = PUNTOACTUAL$
3: **while** $Vector \neq null$ **do**
4: $PUNTOFINAL = PUNTOACTUAL$
5: $CENTROIDE+ = PUNTOACTUAL$
6: **if** $PUNTOACTUAL$ Es un punto $EXTREMODERECHO, EXTREMOIZQUIERO, EXTRE$ **then**
7: Remplazar el extremo correspondiente por PUNTOACTUAL
8: **end if**
9: $NUMPUNTOS++$
10: **if** $Vector \neq VECTORANTERIOR$ **then**
11: **for** $i = 0$ hasta $i = LETRAS.lenght$ **do**
12: **if** $ERRORES[i] > LETRAS[i].lenght * \lambda$ **then**
13: $ERRORES[i] = \infty$
14: **else if** $INDICES[i] == LETRAS[i].lenght$ **then**
15: $ERRORES[i]+ = DiferenciaRadial(Vector, LETRAS[i][INDICES[i]-1])$
16: **else**
17: $TER =$ **true**
18: **for** $j = 0$ hasta $j < \Omega$ **do**
19: **if** $INDICES[i] + j == LETRAS[i].lenght$ **then**
20: Break
21: **else if** $LETRAS[i][INDICES[i] + j] == Vector$ **then**
22: $ACIERTOS[i]+ = 7 - j$
23: **for** $k = INDICES[i]$ hasta $k < INDICES[i] + j$ **do**
24: $ERRORES[i]+ = DiferenciaRadial(Vector, LETRAS[i][k])$
25: **end for**
26: $INDICES[i]+ = j + 1$
27: $TER =$ **false**
28: **end if**
29: **end for**
30: **if** TER **then**
31: $ERRORES[i]+ = DiferenciaRadial(Vector, LETRAS[i][INDICES[i]])$
32: **end if**
33: **end if**
34: **end for**
35: **end if**
36: **end while**
37: $MENOR = \infty$
38: $CENTROIDE = NORMALIZA(CENTROIDE, PUNTOFINAL)$
39: **for** $i = 0$ hasta $i = LETRAS.lenght$ **do**
40: **if** $ERRORES[i] < \infty$ **then**
41: $DF = DistanciaRadial(DIRECCIONFINALES[i], VECTOR(PUNTOORIGEN, PUNT$

Capítulo 7

Aplicaciones

Como medio para demostrar la funcionalidad del sistema se han programado una serie de aplicaciones que ayudan al usuario en tareas comunes al usar un Smartphone pero que suelen requerir de la intervención de botones o pantalla táctil. Las aplicaciones han sido diseñadas de modo que el usuario sienta intuitivo su uso aunque al tratarse de programas de prueba no cuentan con una gran interfaz gráfica propia, más bien se vale de otros programas preinstalados para las funciones multimedia (aunque como trabajo a futuro podrían implementarse las propias) pensando en que es más natural para el usuario que nuestro sistema actué sólo como intermediario y permita acceder a sus aplicaciones preferidas para música, videos o internet.

Este capítulo describe las aplicaciones, su uso y el resultado esperado, los porcentajes de asertividad, errores y problemas son descritos en el capítulo de Resultados y Conclusiones

7.1 Marduk

El sistema se encuentra compuesto por una aplicación principal que hemos llamado "Marduk" y una serie de sub-aplicaciones. Marduk tiene el control de la cámara y la salida de video, además es la encargada del reconocimiento de formas, en ella se anidan los métodos de pre-procesamiento de imágenes y reconocimiento de caracteres, es decir que para fines prácticos es la contendora de toda la estructura descrita en el capítulo "Arquitectura del sistema". La figura 7.1 nos muestra a Marduk tal cual se ve en la proyección.

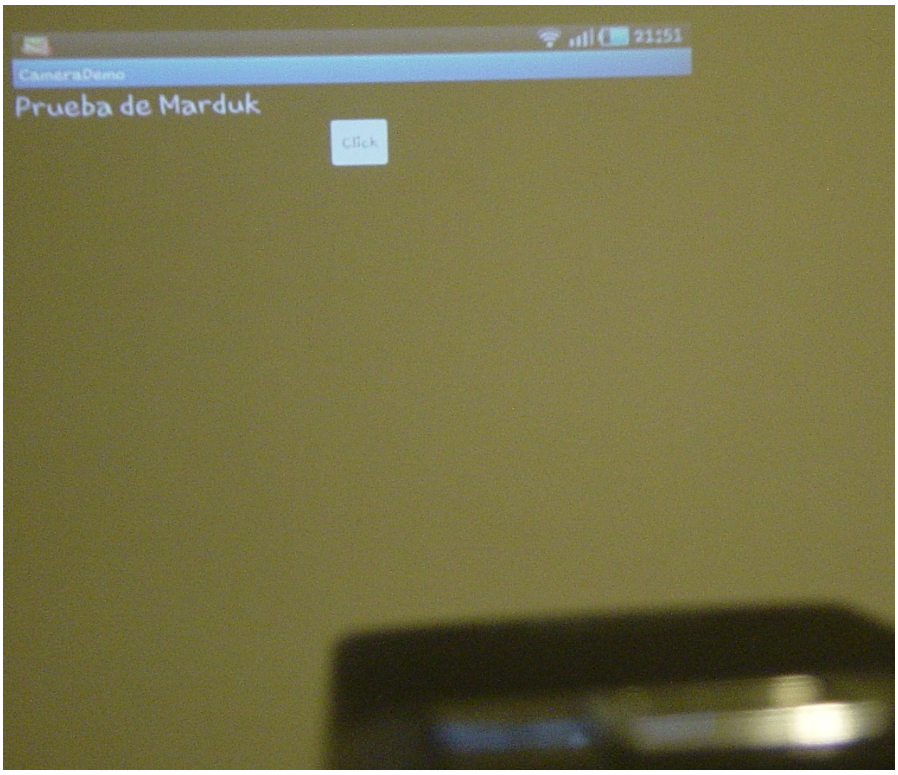

Figura 7.1: Vista de la pantalla principal de Marduk

Como podemos observar la interfaz de usuario es en exceso simple, esto es resultado de la misma idea de sexto sentido en la que buscamos que el usuario interactúe mediante gesticulaciones en el aire y los elementos en las proyecciones, no con lo que aparece en la pantalla. Todo el proceso descrito en el capítulo "Arquitectura del sistema" es transparente al usuario, fuera del entrenamiento del sistema para reconocer el modelo de escritura el usuario no interviene de ninguna otra forma con el sistema. Las variables han sido ajustadas y programadas de acuerdo a los experimentos con diferentes usuarios por lo que no es necesario crear una interfaz que permita ajustar el programa, de hecho probablemente sería perjudicial permitir la libertad de ajustar los parámetros de fineza del reconocimiento o la variable Z descrita en el capítulo de "Pre-procesamiento de imágenes".

Símbolo	Acción
1	Almacena el número 1 en la pila
2	Almacena el número 2 en la pila
3	Almacena el número 3 en la pila
4	Almacena el número 4 en la pila
5	Almacena el número 5 en la pila
6	Almacena el número 6 en la pila
7	Almacena el número 7 en la pila
8	Almacena el número 8 en la pila
9	Almacena el número 9 en la pila
0	Almacena el número 0 en la pila
a	E-mail
f	Facebook
n	Notas
o	Fotografía
+	Prepara para suma
-	Prepara para resta
x	Prepara para multiplicación
÷	Prepara para división
=	Resultado matemático
♪	Música
▷	Video

Tabla 7.1: Símbolos reconocidos por Marduk en la pantalla principal

Dentro del programa principal se pueden reconocer los símbolos de la tabla 7.1, estos desatan ciertas acciones o el surgimiento de sub-aplicaciones. Se ha acortado la lista de todo el alfabeto y signos de puntuación a sólo estos para tener un mayor porcentaje de aciertos.

7.2 Sub-Aplicaciones

El programa principal permite desglosar una serie de eventos o sub-aplicaciones que quedan bajo la jerarquía de esta actividad. Estos eventos tratan de demostrar la funcionalidad del sistema, algunos de ellos pueden relacionarse directamente con las aplicaciones de prueba del trabajo de sexto sentido de [Pranav Mistry, 2009] en el cual con un equipo de mayores capacidades se reconocen diversas gesticulaciones que son tratadas e interpretadas para ejecutar una acción.

No es propósito de esta tesis el desarrollo de codecs de video, música o un explorador de internet, así que para desatar eventos como escuchar una canción, ver un video o abrir una página de internet haremos uso de los recursos que nos brinda Android y de los programas previamente instalados.

Teniendo en cuenta la idea del párrafo anterior describiremos más a detalle los subprogramas que pueden ejecutarse en nuestro sistema.

7.2.1 Red Social

Al trazar una letra "f" con el stylus se concede acceso automático a una red social, por ser sólo una aplicación de prueba se ha optado por dejar como predeterminada la red de Facebook, por ser la red social más importante en nuestro contexto social y temporal, cambiar a otra red social no involucraría grandes cambios. Se abandona Marduk y se dispara la aplicación Facebook que queda al mando de la pantalla tanto en la pantalla del teléfono celular como del pico-proyector, que hasta un momento anterior había permanecido en estado de hibernación y despierta ante el evento, obviaremos este comentario para las demás sub-aplicaciones y debe entenderse que en todas ellas el pico-proyector permanece encendido y a la espera de que el teléfono envíe instrucciones para su encendido y vuelta al modo de hibernación, siempre que se dispara una sub-aplicación se enciende el pico-proyector. En la figura 7.2 se observa mejor la proyección tras el trazo de la "f".

7.2.2 E-Mail

Funciona de forma similar a la sub-aplicación de *Red social* pero haciendo uso de la letra "a", disparando un cliente de e-mail. Debido a que todo celular Android tiene instalado por defecto la aplicación de *GMail* como cliente de correo electrónico se ha optado por hacer uso de este programa de forma que sea intuitivo para el usuario usar el mismo cliente de correo electrónico que utiliza desde su teléfono usualmente. La figura 7.3 muestra la proyección del cliente de correo electrónico *GMail*.

7.2.3 Fotografía

Uno de los ejemplos más relevantes dados en [Pranav Mistry, 2009] para demostrar la potencia de un sistema de sexto sentido es la posibilidad de tomar fotografías a partir de un gesto, en nuestro caso hemos escogido trazar una letra "o", de forma que

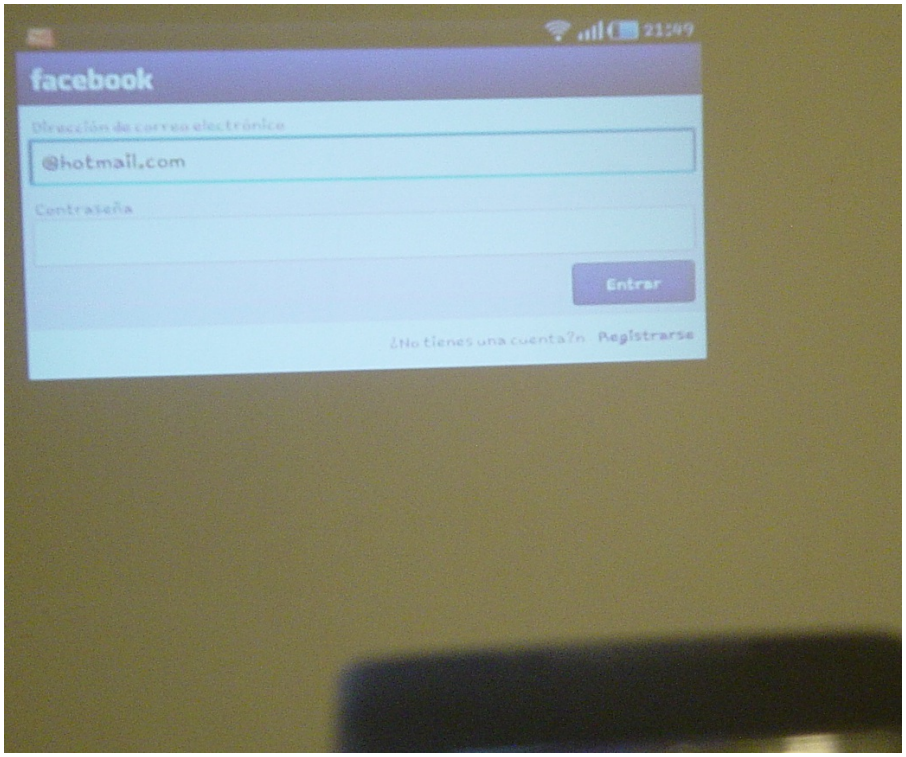

Figura 7.2: Proyección de la aplicación de Facebook tras dibujar una letra "f"

queda similar al círculo usado por [Pranav Mistry, 2009] para tomar una fotografía. El evento fotografía es separado de las fotografías tomadas para el reconocimiento de stylus, se utiliza el formato máximo de imágen soportado por la cámara (en el caso del Samsung Galaxy S es de 2560 x 1920 [Gal, 2009] pero se adapta al celular en el que se corra nuestro sistema) y es guardado en la raíz del sistema de archivos del teléfono celular con el nombre de "*Mi_Foto.jpg*".

Para esta aplicación se usa la API de la cámara que proporciona Android, la misma que ha sido usada durante el pre-procesamiento de imágenes para obtener imágenes de la cámara, sin embargo, en esa etapa hemos interceptado los datos antes de la compresión para poder obtener imágenes en tiempo real sin retrasos. Para esta aplicación no interrumpimos el ciclo sino que aprovechamos la compresión ofrecida por

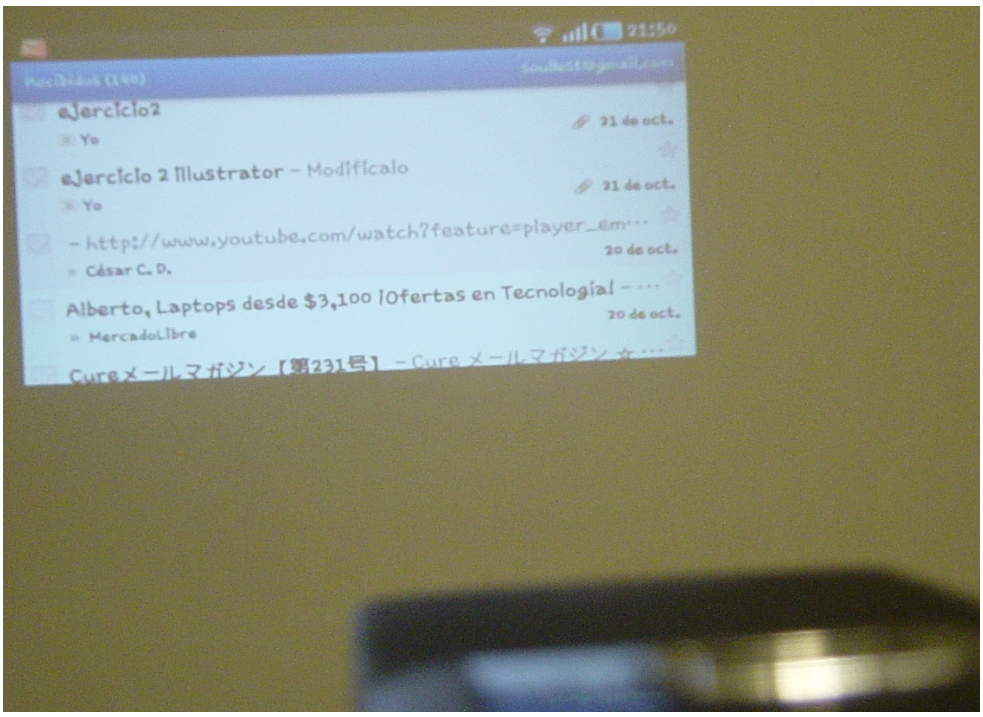

Figura 7.3: Proyección del cliente de correo electrónico

hardware durante el evento takePicture()[1], que nos entrega una imágen comprimida en JPG, la cual nosotros la podemos almacenar en un dispositivo físico. Para este fin se ha escogido la memoria interna del teléfono pues no se puede garantizar la existencia de una memoria SD. Cuando concluye el evento la fotografía almacenada es lanzada a la galería proporcionada por Android, la cual también se encuentra instalada por defecto en todo teléfono que maneje este sistema operativo y es accesible a través de la API proporcionada por Google[2], después se enciende automáticamente el pico-proyector y la fotografía recientemente tomada es proyectada a una superficie cercana. En la figura 7.4 se muestran dos escenarios, en el primero se muestra el stylus mientras dibuja una "o" alrededor de una escena de la cual se quiere obtener una fotografía y en el segundo se muestra la fotografía tomada que es proyectada por el pico-proyector

[1] http://developer.android.com/reference/android/hardware/Camera.html
[2] http://developer.android.com/reference/android/widget/Gallery.html

Figura 7.4: A la izquierda la imágen tomada mientras se realiza el trazo con el stylus, a la derecha la imágen tomada despúes de que la fotografía es capturada y proyectada

7.2.4 Matemática

Como se puede ver en la tabla 7.1, los números del 0 al 9 son reconocidos como símbolos de entrada que son acomodados en la pila hasta ser usados por un operador. Antes de continuar con la explicación de esta sub-aplicación, entenderemos primero la entrada de símbolos numéricos. Cuando un símbolo numérico es reconocido no desata en realidad una acción como todos los símbolos que hasta ahora hemos tratado, más bien estos números comienzan a ser guardados de forma que una sub-aplicación pueda tratarlos posteriormente (hacen uso de los números las aplicaciones matemáticas, de música y de video, en un trabajo futuro los números podrían tener otros usos complementarios). Conforme los números son reconocidos estos pasan a una variable que actúa como buffer a la que llamaremos N1, cada que llega un nuevo número, N1 se multiplica por 10 y se le suma el número recién dibujado, esto con el propósito de que se pueda representar cualquier número a partir de sus dígitos individuales, por ejemplo si quisiéramos trazar un 107 la variable N1 se movería de la siguiente manera:

$$
\begin{gathered}
N1{=}0 \\
N1 * 10 = 0 \\
N1 + 1 = 1 \\
N1{*}10 = 10 \\
N1{+}0{=}10 \\
N1{*}10{=}100 \\
N1{+}7 {=}107
\end{gathered}
$$

Un mecanismo bastante simple pero efectivo, mientras se sigan dibujando números teóricamente podría llegar hasta infinito nuestro número, pero claro nos vemos limitados por la capacidad de almacenamiento del tipo de datos Long.

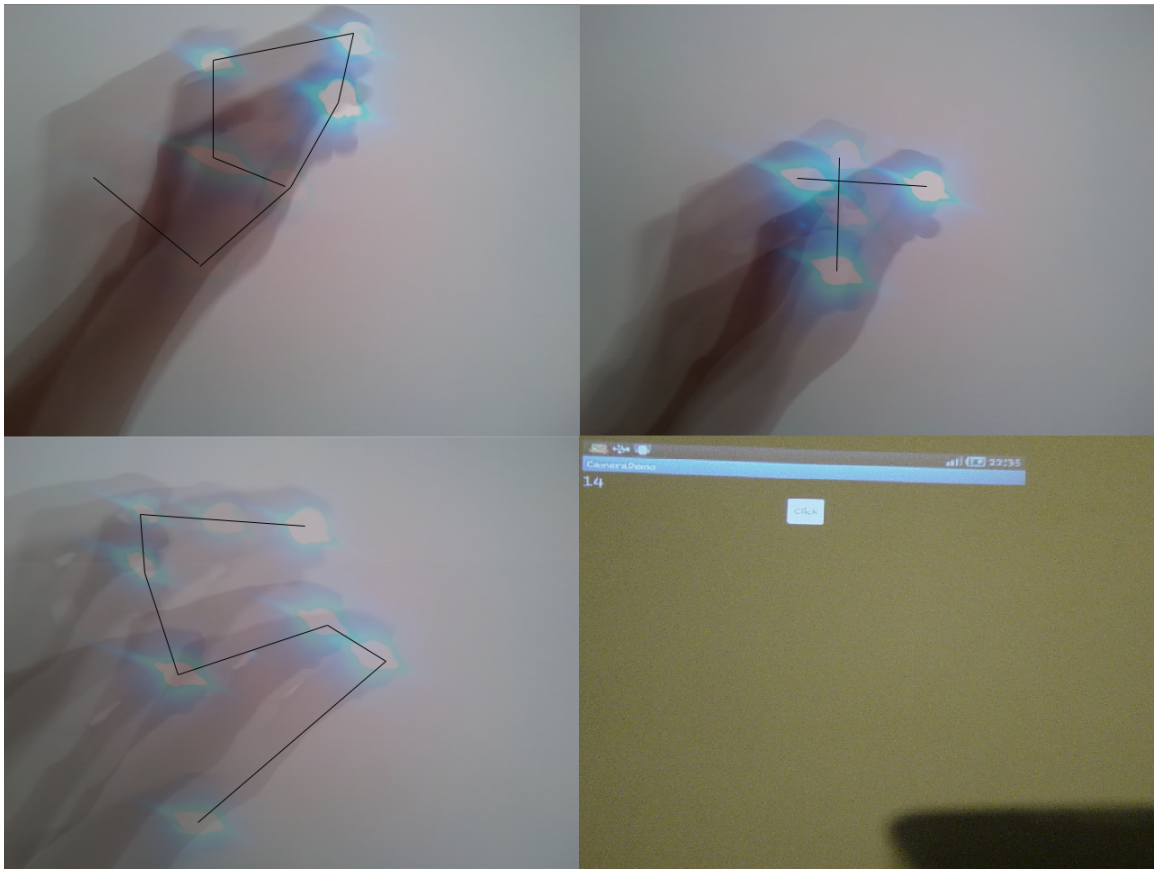

Figura 7.5: Arriba la izquierda el trazo del primer número, un 9. Arriba a la derecha el trazo del operador, en este caso una suma. Abajo a la izquierda el segundo número, un 5. En la inferior derecha la proyección del resultado.

En el caso de la sub-aplicación Matemática al reconocer uno de los símbolos de operadores matemáticos recordaremos el símbolo y comenzaremos a reconocer un nuevo número que almacenaremos en un buffer N2, cuando deseemos conocer el resultado de la operación que hemos trazado con el stylus trazaremos el símbolo "=" lo que desatará en el pico-proyecto la orden de mostrar el resultado de la operación. El ejemplo de la figura 7.5 muestra los diferentes momentos en los que se realiza una operación matemática.

7.2.5 Música

La aplicación de música trabaja también con el buffer N1, después de un número podemos dibujar el símbolo de "Nota musical" que reproducirá una canción con el altavoz del teléfono celular, se usa el reproductor de música de Android (igualmente

instalado en cualquier dispositivo Android) y accesible a través del API[3]. Seria tardado escribir por completo el nombre o ubicación de una canción, pensando en ello se crea una lista de reproducción la cual es almacenada en la raíz de archivos del teléfono celular bajo el nombre "Musica.txt", en ella se guardan las rutas y nombres de las canciones que tenemos guardadas en alguno de los dispositivos de almacenamiento disponibles, de esta manera el número que dibujamos en el aire corresponde al número en la lista de reproducción y por lo tanto a la posición en el archivo de texto de dicha canción.

El porqué del uso del altavoz del celular en lugar del altavoz del pico-proyector es debido por un lado a factores de diseño, generalmente la bocina del pico-proyector queda bloqueada por el teléfono debido a la estructura física del sistema mostrada en el capítulo "Entrada y salida físicas", en este mismo capítulo se habla de algunos problemas que existen al transmitir el audio entre el teléfono celular y el pico-proyector.

Por otro lado el cable que conecta el pico-proyector y el teléfono celular es enchufado en el conector Jack 3.5 destinado a la salida de audio [Gal, 2009] e igualmente es descrito en el capítulo de "Entrada y salida físicas" pero esto trae el terrible problema de que el teléfono cree erróneamente que tiene enchufados los auriculares, situación indeseada pues el sonido del pico-proyector no es lo suficientemente claro ni suficientemente potente, debemos conservar la señal de audio en el teléfono . Una vez más usamos una solución más creativa que técnica, propiamente no es posible hacer un cambio en la salida de audio pues el sistema operativo se ocupa de hacer el cambio entre auriculares y altavoz automáticamente, pero engañaremos al sistema haciéndole creer que nuestra pista musical es una llamada entrante, de esta forma podemos usar los elementos del API del sonido en llamadas[4] la cual si incluye la particularidad de cambiar desde código entre auriculares y altavoces.

Hay que tomar en cuenta que debemos resetear los valores del sonido en llamadas o cuando el usuario quiera usar el teléfono realmente lo iniciara con los altavoces en lugar de la bocina auricular.

La figura 7.6 muestra la proyección de una pista de audio, lamentablemente no podemos exponer sonido en este escrito.

[3] http://developer.android.com/reference/android/media/MediaPlayer.html
[4] http://developer.android.com/reference/android/media/AudioManager.html

Figura 7.6: Proyección de la aplicación de Música

7.2.6 Video

Funciona de la misma forma que la aplicación de Música, pero los números terminan con el símbolo ▷, se toma el número almacenado en N1 que sirve como apuntador a un elemento de la lista de reproducción de videos que se almacena en el archivo "Videos.txt" en la raíz del sistema de archivos del teléfono, conservando la misma estructura que ha sido descrita para el archivo "Musica.txt". Para la reproducción de videos se utiliza el reproductor de videos preinstalado en todo sistema operativo Android, accesible gracias al API[5]. Si bien la reproducción de video se hace mediante el pico-proyector, el sonido se reproduce de igual forma que la aplicación de Música mediante el uso de los altavoces del teléfono. La reproducción de videos es mejor

[5] http://developer.android.com/reference/android/widget/VideoView.html

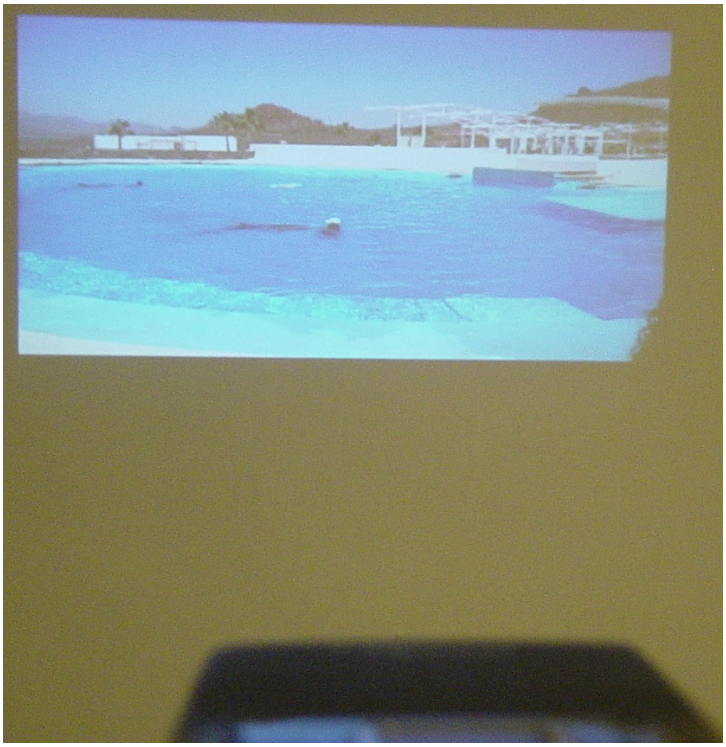

Figura 7.7: Proyección de un video tras el dibujo del símbolo de reproducción de video

esquematizad a en la figura 7.7.

7.2.7 Notas

La aplicación de notas es quizá la más complicada de todas las presentadas, ya que después de trazar el símbolo "n" se carga un nuevo modelo de símbolos a reconocer, cambia de los presentados en la tabla 1 a tener el juego completo del alfabeto español y signos de puntuación para poder crear notas que después aparecerán proyectadas a través del pico proyector o que serán guardadas como un formato de texto. No es necesario escribir con espaciado, se puede escribir todo el mensaje en un solo espacio relativamente pequeño encimando letras, por ejemplo en la palma de la mano se podrían escribir todas las letras encimadas de la palabra "Hola" una encima de la otra, ya que el reconocimiento funciona en tiempo real, al finalizar una letra esta ya

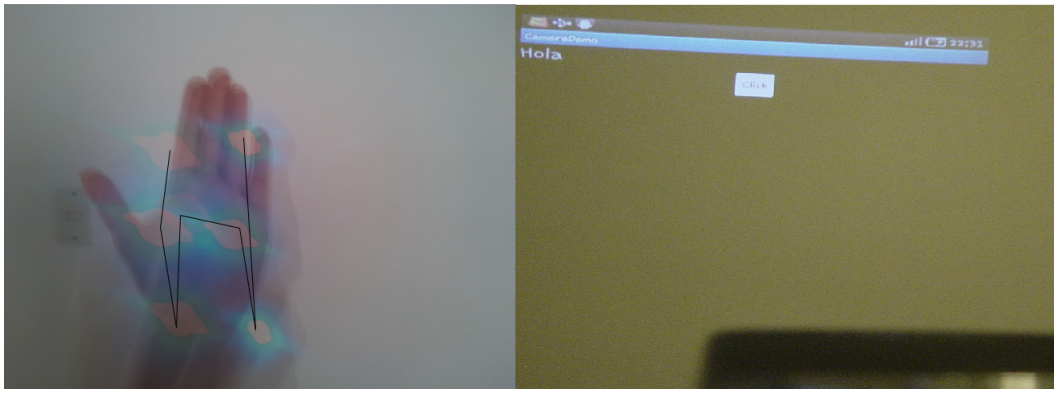

Figura 7.8: A la izquierda el dibujo de una letra "H"sobre la mano. A la derecha la palabra copleta que se dibujo: "Hola"

ha sido reconocida y guardada en la pila para formar palabras. Cuando se dibuja con el stylus el símbolo de tache cerrado se termina la nota y se vuelve a Marduk, esta proyectara la nota mediante el pico-proyector. Siguiendo el ejemplo dado en este párrafo, la figura 7.8 muestra la palabra "Hola" escrita por el stylus dibujada en la palma de la mano.

Capítulo 8

Conclusiones y trabajo futuro

Antes de dar conclusión al trabajo recordaremos la problemática planteada en este trabajo. Nos referimos a la labor de llevar la tecnología de sexto sentido y las NUI a un ambiente totalmente móvil, no como un experimento con fines de recolección de datos, como algunos otros trabajos han intentado, sino como un prototipo funcional que vuelva esta tecnología verdaderamente móvil. Durante este capítulo también se presentan las conclusiones a las que se han llegado a lo largo del desarrollo de este proyecto, presentando los hechos y premisas más relevantes que dan soporte a este trabajo. Finalmente, se aborda el trabajo futuro que queda por desarrollar, así como las predicciones que se hacen sobre la evolución de esta tecnología y como las repercusiones de este trabajo en el desarrollo de las nuevas implementaciones de NUI y sexto sentido.

8.1 Resumen de la problemática

La actual tendencia de la computación a evolucionar en un entorno móvil suele traer consigo muchos problemas, sobre todo aquellos referidos a la capacidad de procesamiento y a la falta de hardware en los dispositivos móviles. En muchas ocasiones al desarrollar para dispositivos móviles, los programadores tienen que seguir protocolos rígidos o utilizar su ingenio para evitar consumir más recursos de los que el teléfono puede ofrecer. Si bien estas son prácticas computacionalmente deseables, en varias ocasiones nos encontramos con problemas difíciles de resolver, utilizando sólo los limitados recursos de un teléfono celular, tablet o cualquier dispositivo móvil.

La definición de sexto sentido ofrecida por Pranav Mistry [Pranav Mistry, 2009] nos permite ubicar esta tecnología en el campo del cómputo móvil, sin embargo, los prototipos creados hasta el momento [Mistry and Maes, 2009] se valen de computadoras portátiles o de cómputo en la nube [Forutanpour and Ren, 2011]. Como prototipos

de investigación resultan perfectamente válidos pero no parecen muy realistas. La tendencia clara es optimizar los algoritmos de visión por computadora y aprovechar las nuevas tecnologías en sistemas operativos y hardware que acompañan a una nueva generación de dispositivos móviles que dejan de ser unidades mono-trabajo para convertirse en completos centros de trabajo móviles.

Este es el campo de acción de esta tesis, construir un prototipo realista de tecnología de sexto sentido que, por primera vez, cumpla con la carácter ística de ser completamente móvil. Se trata de un fuerte trabajo de optimización de algoritmos de visión por computadora que han sido pensados para ejecutarse en terminales móviles y que aprovechan las carácter ísticas de estos dispositivos para evitar cálculos complejos sin sacrificar en ningún momento los resultados.

8.2 Conclusiones

Este proyecto ha logrado exitosamente su objetivo, llevando un smartphone a la tecnología de sexto sentido, la cual ha sido debidamente implementada. Las mejoras realizadas a los algoritmos de visión y de reconocimiento de formas y carácter es han permitido crear una aplicación que, con una cantidad limitada de recursos, crea un nuevo panorama en las interfaces de usuario para dispositivos móviles. Es grato informar que de hecho la aplicación funciona con holgura dentro del smartphone que se ha usado durante el desarrollo, lo que permitiría migrar el programa hacia dispositivos con menores recursos, siempre y cuando mantengan las carácter ísticas de sistema operativo Android y una cámara fotográfica empotrada.

Cabe mencionar que parte del éxito de la tesis se debe al uso de nuevas tecnologías. Las actualizaciones en el sistema operativo que permiten manejar de forma más eficiente los datos en tiempo real, los nuevos chips para cámaras fotográficas de dispositivos móviles con una mayor velocidad en FPS y la capacidad del sistema operativo de hacer uso de elementos de hardware a bajo nivel son algunos de los elementos que ayudaron al buen término de este proyecto. El trabajo ha sido pensado para poder ser reproducido en hardware similar pero no idéntico, en su momento utilizando lo más nuevo en tecnología móvil. De hecho no está por demás mencionar que este es un proyecto que no hubiera podido tener los mismos resultados hace uno o dos años. Al momento de terminar este escrito las capacidades de los smartphone y de otros dispositivos móviles superan a las del hardware utilizado durante el desarrollo de nuestro arquetipo. Con ello la incursión de la tecnología de sexto sentido en los dispositivos móviles permitirá abarcar una amplia gama de dispositivos, no sólo aquellos en la punta de la elite tecnológica.

La arquitectura presentada y el trabajo final pueden ser utilizados como un *framework* para que otros proyectos con intenciones similares puedan llegar más lejos, sin tener que abordar todos los problemas aquí resueltos. Entre las principales aportaciones de este trabajo se tienen las siguientes:

- Creación de un algoritmo eficiente de reconocimiento de carácter es basado en la individualidad de los dispositivos móviles. Este algoritmo es realmente ligero, corre en tiempo real y aprovecha el modelo de escritura del usuario para identificar letras con gran asertividad. Las capacidades de este algoritmo no se limitan a una herramienta para la tecnología de sexto sentido, pues se pensó como un algoritmo de propósito general, lo que permite utilizarlo como parte de diferentes interfaces de usuario que usen texto a mano alzada como modo de entrada.

- Creación de un algoritmo eficiente de reconocimiento de objetos. Tratar con una gran cantidad de fotografías por segundo trae consigo una mejora considerable en el reconocimiento de gesticulaciones, pero es también un cuello de botella, quizás el más importante en este tipo de tecnología. El algoritmo desarrollado permitió centrarnos en la información útil de la imágen sin que el paso por píxeles irrelevantes tenga un costo peculiarmente alto. De hecho, después de localizar por primera vez el stylus, seguirlo a lo largo de la imágen es relativamente sencillo, sin que por ello el algoritmo deje de ser robusto a cambios repentinos en los movimientos, la luz o la posición.

- Desarrollo de una arquitectura base para los sistemas de tecnología de sexto sentido. Habiendo estudiado el problema y conociendo las capacidades de los dispositivos a los que queremos enfocarnos, se desarrolló una arquitectura en software y hardware que da sustento al proyecto. Los bloques fueron diseñados de tal forma que sientan una base para trabajos relacionados o cualquier implementación de NUI que haga uso de visión por computadora como su herramienta principal.

- Generación de un algoritmo propio para descifrar el formato YUV420SP. Con base en un algoritmo conocido para descomponer el formato YUV420SP se creó una versión propia más eficiente para la problemática enfrentada. Este nuevo algoritmo permite descifrar sólo ciertas partes de la imágen, por lo que no requiere un recorrido entero de la imágen como el algoritmo originalmente encontrado en la literatura. Esto se ha podido lograr gracias a una profunda comprensión del formato.

- Aplicaciones basadas en la tecnología de sexto sentido. Se sienta el precedente de aplicaciones que usan la tecnología de sexto sentido como modelo de prueba

para demostrar la capacidad del sistema. Al tratarse de un proyecto pionero en esta tecnología es necesario dejar aplicaciones de prueba que otros trabajos puedan retomar como patrones para medir sus desarrollos y dar a conocer sus resultados.

8.3 Trabajo a futuro

El proyecto ha sido enteramente gratificante al cumplir con los objetivos planteados. No obstante siempre existen mejoras aplicables al trabajo. Esperamos que este trabajo sea un eslabón en una cadena de nuevas tecnologías en interfaces de usuario, que los nuevos proyectos encuentren una base sólida en la que apoyarse con este trabajo, de esta forma haciendo más sencilla la creación de interfaces más complejas que continúen la idea de enfrentar la computación al mundo real y no sea el usuario quien se adapte a la computación.

En este momento, los grandes corporativos de la computación como Microsoft y Apple apuestan por el desarrollo de las NUI como las interfaces del futuro, no sólo con proyectos de entretenimiento como el Kinnect[1] el iPod[2], sino también con el desarrollarlo de una tecnología que permita cerrar la brecha existente entre la computación y los seres humanos. Proyectos como el Microsoft Surface[3] muestran el resultado de intentar disminuir esta brecha, dejando de lado la ficción y convirtiendose en productos comerciales. Con antecedentes como estos se cimientan las bases de lo que serán las nuevas interfaces de usuario. Si bien aún contamos con las tradicionales GUI, el cambio en el paradigma de interfaces de usuario es ya un hecho que encontramos cada vez más presente en nuestra tecnología.

Entre las mejoras que se pueden pensar para el sistema propuesto que lo volverían mucho más amigable y con mayores fortalezas se encuentran las siguientes:

- Integrar a la arquitectura una etapa de calibración de cámaras que permita entender mejor los movimientos realizados en diferentes superficies. Así mismo adaptar las proyecciones a las superficies circundantes sin que esto cause una deformación. Por el contrario, esta nueva etapa en la arquitectura permitiría que las proyecciones se vieran mucho más naturales.
- Incorporación de la tecnología de sexto sentido como un API que no slo permita identificar movimientos en aplicaciones destinadas hacia la tecnología de sexto

[1] http://www.xbox.com/en-US/kinect
[2] http://www.apple.com/mx/ipod/
[3] http://www.microsoft.com/surface/

sentido, sino que también permita remplazar el uso de pantallas táctiles, botones o mouse. De esta forma, toda interacción con el dispositivo móvil se podría realizar mediante gesticulaciones sin entrar en contacto físico con el dispositivo.

- La etapa de pre-procesamiento de imágenes puede mejorarse aún más si trabajamos directamente con el modelo de color YUV420SP sin conversiones. Para ello es necesario hacer un estudio del comportamiento de algunos valores de luminiscencia en este formato en particular. De obtener resultados positivos, podríamos ahorrar una etapa muy importante de cálculo, lo cual conllevaría a una mejora importante en el rendimiento pero también cambios considerables en el algoritmo de reconocimiento de objetos.

- El motor de reconocimiento de carácter es Lx-Kdabra ha demostrado ser altamente eficiente, sin que ello signifique la ausencia de algunos errores aunque estos son pocos. Una buena forma de mejorar el reconocimiento es aplicar un método más robusto sobre el pequeño grupo de candidatos que se genera después del reconocimiento en tiempo real. Si pensamos que el grupo generalmente se compone de no más de cinco candidatos, un método robusto no implicaría grandes costos computacionales, lo que mantiene el bajo perfil del proyecto.

- Lx-Kdabra está diseñado para manejarse con el modelo de escritura de cada usuario. Para evitar que el proceso de aprendizaje sea un proceso tedioso, se puede implementar una base de datos de trazos genérica que actuaría en primera instancia y que sería remplazada al poco tiempo, una vez que se conozcan todos los trazos del usuario con el uso del programa.

- Finalmente un stylus de luz omnidireccional permitiría una interacción más agradable para el usuario. Se puede también pensar en dejar de lado el stylus y comenzar a realizar optimizaciones para adaptar los algoritmos de reconocimiento de formas robustos y así identificar los gestos de las manos directamente.

Bibliografía

[Omn, 2008] (2008). Ov5630/ov5633 5 megapixel product brief.

[Gal, 2009] (2009). Samsung galaxy s gt-i9000 user manual.

[Cor, 2010] (2010). Cortex-a8 technical reference manual.

[Pow, 2010] (2010). Wervr sgx series5 ip core family.

[LED, 2011] (2011). 3/azul ultra.

[Opt, 2011] (2011). Pico pk201 palm-sized versatile performer datasheet.

[Abrahamsson et al., 2004] Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., and Salo, O. (2004). Mobile-d: an agile approach for mobile application development. In *OOPSLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pages 174–175, New York, NY, USA. ACM.

[Ahn et al., 2009] Ahn, J.-H., Lee, J., Jo, J., Choi, Y.-H., and Lee, Y. (2009). Online character recognition using elastic curvature matching. *Advances in Pattern Recognition, International Conference on*, 0:395–397.

[Aliakseyeu and bernard Martens, 2001] Aliakseyeu, D. and bernard Martens, J. (2001). Physical paper as the user interface for an architectural design tool. In *Proceedings of IFIP INTERACT'01: Human-Computer Interaction*, pages 680–681.

[Aparna et al., 2004] Aparna, K. H., Subramanian, V., Kasirajan, M., Prakash, G. V., Chakravarthy, V. S., and Madhvanath, S. (2004). Online handwriting recognition for tamil. In *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, IWFHR '04, pages 438–443, Washington, DC, USA. IEEE Computer Society.

[Bahlmann, 2006] Bahlmann, C. (2006). Directional features in online handwriting recognition. *Pattern Recogn.*, 39:115–125.

[BAI and HUO, 2005] BAI, Z.-L. and HUO, Q. (2005). A study on the use of 8-directional features for online handwritten chinese character recognition. *Document Analysis and Recognition, International Conference on*, 0:262–266.

[Books, 2010] Books, L. (2010). *Audiovisual Connectors: Digital Visual Interface, Scart, Rca Connector, Trs Connector, Serial Digital Interface, D-Subminiature*. General Books LLC.

[Boylestad et al., 2003] Boylestad, R., Nashelsky, L., and Barraza, C. (2003). *Electronica: Teoria de Circuitos Y Dispositivos Electronicos*. Prentice-Hall, Naucalpan, México, 8 edition.

[Calhoun et al., 2002] Calhoun, C., Stahovich, T. F., Kurtoglu, T., and Kara, L. B. (2002). Recognizing multi-stroke symbols. In *IN 2002 AAAI SPRING SYMPOSIUM - SKETCH UNDERSTANDING, (PALO ALTO CA, 2002*, pages 15–23. AAAI Press.

[Chan et al., 2007] Chan, L.-W., Chuang, Y.-F., Chia, Y.-W., Hung, Y.-P., and Hsu, J. (2007). A new method for multi-finger detection using a regular diffuser. In *HCI'07: Proceedings of the 12th international conference on Human-computer interaction*, pages 573–582, Berlin, Heidelberg. Springer-Verlag.

[Chang et al., 2010] Chang, Y. F., Lee, J. C., Rijal, O. M., and Bakar, S. A. R. S. A. (2010). Efficient online handwritten chinese character recognition system using a two-dimensional functional relationship model. *Applied Mathematics and Computer Science*, 20(4):727–738.

[Chaurand et al., 2001] Chaurand, R., León, L., Muñoz, E., and de Guadalajara. Centro de Investigaciones en Ergonomía, U. (2001). *Dimensiones antropométricas de población latinoamericana: México, Cuba, Colombia, Chile*. Colección Modulor. Universidad de Guadalajara, Centro Universitario de Arte, Arquitectura y Diseño, División de Tecnología y Procesos, Departamento de Producción y Desarrollo, Centro de Investigaciones en Ergonomía.

[Connell, 2000] Connell, S. D. (2000). *Online handwriting recognition using multiple pattern class models*. PhD thesis, East Lansing, MI, USA. AAI9971905.

[Deepu et al., 2004] Deepu, V., Madhvanath, S., and Ramakrishnan, A. G. (2004). Principal component analysis for online handwritten character recognition. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, pages 327–330, Washington, DC, USA. IEEE Computer Society.

[Forutanpour and Ren, 2011] Forutanpour, B. and Ren, J. (2011). Projest: Enabling higher levels of collaboration using today's mobile devices. In Jacko, J., editor, *Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments*, volume 6763 of *Lecture Notes in Computer Science*, pages 48–58. Springer Berlin / Heidelberg.

[Golubitsky and Watt, 2008] Golubitsky, O. and Watt, S. M. (2008). Online stroke modeling for handwriting recognition. In *Proceedings of the 2008 conference of the*

center for advanced studies on collaborative research: meeting of minds, CASCON '08, pages 6:72–6:80, New York, NY, USA. ACM.

[Gu and Duh, 2011] Gu, J. and Duh, H. B. L. (2011). Mobile augmented reality game engine. In Furht, B., editor, *Handbook of Augmented Reality*, pages 99–122. Springer New York, KEIO-NUS CUTE Center, National University of Singapore, Singapore, Singapore.

[GWM, 1999] GWM, R. (1999). From gesture to action : natural user interfaces. pages 15–26.

[Hamanaka, 2006] Hamanaka, M. (December 2006). Music scope headphones: Natural user interface for selection of music. In *In Proceedings of the 7th International Conference on Music Information Retrieval*, pages 302–307.

[Harrison et al., 2011] Harrison, C., Benko, H., and Wilson, A. D. (2011). Omnitouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 441–450, New York, NY, USA. ACM.

[Ihme and Abrahamsson, 2008] Ihme, T. and Abrahamsson, P. (2008). The use of architectural patterns in the agile software development of mobile applications. *International Journal of AgileManufacturing*, (8):97–112.

[Izadi et al., 2008] Izadi, S., Hodges, S., Taylor, S., Rosenfeld, D., Villar, N., Butler, A., and Westhues, J. (2008). Going beyond the display: a surface technology with an electronically switchable diffuser. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 269–278, New York, NY, USA. ACM.

[Jack, 1995] Jack, K. (1995). *Video Demystified: A Handbook for the Digital Engineer*. L L H Technology Publishing, 2nd edition.

[J.Ashok, 2010] J.Ashok, E. (2010). Writer identification and recognition using radial basis function. *International Journal of Computer Science and Information Technologies*, 1(2):51–57.

[Kamakhya Gupta and Viswanath, 2007] Kamakhya Gupta, S. V. R. and Viswanath, P. (2007). Speeding up online character recognitio. In *in Proceedings of the Twenty-second International Conference on Image and Vision Computing*, pages 41–45, Hamilton, New Zeland. IEEE Computer Society.

[Kato et al., 1999] Kato, N., Suzuki, M., Omachi, S., Aso, H., and Nemoto, Y. (1999). A handwritten character recognition system using directional element feature and asymmetric mahalanobis distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21:258–262.

[Kress and Meyrueis, 2009] Kress, B. and Meyrueis, P. (2009). *Applied digital optics: from micro-optics to nanophotonics*. Wiley, Great Britain, 1 edition.

[K.W. Bollhoefer and Witzsche, 2009] K.W. Bollhoefer, K. M. and Witzsche, R. (2009). Microsoft surface und das natural user interface (nui). In *Pixelpark white paper*, pages 3–22, Berlin, Deutschland. Pixelpark.

[Leith K. Y. Chan, 2011] Leith K. Y. Chan, H. Y. K. L. (2011). Magicpad: A projection based 3d user interface. In *ACM CHI Conference on Human Factors in Computing Systems (CHI 2011), Workshop on Mobile and Personal Projection (MP2 2011*, Pokfulam Road, Hong Kong.

[Liu and Zhou, 2006] Liu, C.-L. and Zhou, X.-D. (2006). Online japanese character recognition using trajectory-based normalization and direction feature extraction. In Lorette, G., editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule, France. Université de Rennes 1, Suvisoft.

[Lowe, 1999] Lowe, D. (1999). Object recognition from local scale-invariant features. In *Seventh International Conference on Computer Vision ICCV'99*, pages 1150–1157.

[M. and Sridhar, 2007] M., D. and Sridhar, M. K. (2007). A feature based on encoding the relative position of a point in the character for online handwritten character recognition. *Document Analysis and Recognition, International Conference on*, 2:1014–1017.

[Masakazu, 2011] Masakazu, Y. (2011). *gu guru andoroido apuri kaihatsu gaido*. ed. Shuwashisu, Tokyo, Japon, 1st edition.

[McCarthy and El-Sheikh, 2011] McCarthy, J. V. and El-Sheikh, E. M. (2011). Image generation and analysis for the android platform: Exploring computer vision in mobile development environments. In *WORLDCOMP'11 - The 2011 World Congress in Computer Science, Computer Engineering, and Applied Computing*, Department of Computer Science, University of West Florida, Pensacola, FL, USA.

[Mistry and Maes, 2009] Mistry, P. and Maes, P. (2009). Sixthsense: a wearable gestural interface. In *SIGGRAPH ASIA '09: ACM SIGGRAPH ASIA 2009 Sketches*, pages 1–1, New York, NY, USA. ACM.

[Mitoma et al., 2005] Mitoma, H., Uchida, S., and Sakoe, H. (2005). Online character recognition based on elastic matching and quadratic discrimination. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 36–40, Washington, DC, USA. IEEE Computer Society.

[Petersen and Stricker, 2009] Petersen, N. and Stricker, D. (2009). Continuous natural user interface: Reducing the gap between real and digital world. In *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, pages 23–26, Washington, DC, USA. IEEE Computer Society.

[Plamondon and Srihari, 2000] Plamondon, R. and Srihari, S. N. (2000). On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:63–84.

[Pranav Mistry, 2009] Pranav Mistry, P. Maes, L. C. (April 2009). Wuw - wear ur world - a wearable gestural interface. In *ACM conference on Human Factors in Computing CHI2009 Extended Abstracts*, pages 411–416. ACM.

[Rauterberg and Steiger, 1996] Rauterberg, M. and Steiger, P. (Octover 1996). Pattern recognition as a key technology for the next generation of user interfaces. In *In Proc. of IEEE International Conference on Systems, Man and Cybernetics–SMC'96 (Vol. 4, IEEE Catalog Number: 96CH35929*, pages 2805–2810. Piscataway: IEEE.

[Shilkrot et al., 2011] Shilkrot, R., Hunter, S., and Maes, P. (2011). Pocomo: projected collaboration using mobile devices. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 333–336, New York, NY, USA. ACM.

[Siltanen and Hyväkkä, 2006] Siltanen, S. and Hyväkkä, J. (2006). Implementing a natural user interface for camera phones using visual tags. In *AUIC '06: Proceedings of the 7th Australasian User interface conference*, pages 113–116, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Siltanen and Hyväkkä, 2006] Siltanen, S. and Hyväkkä, J. (2006). Implementing a natural user interface for camera phones using visual tags. In *AUIC*, pages 113–116.

[Singh et al., 2010a] Singh, D., Singh, S. K., and Dutta, M. (2010a). Article: Hand written character recognition using twelve directional feature input and neural network. *International Journal of Computer Applications*, 1(3):82–85. Published By Foundation of Computer Science.

[Singh et al., 2010b] Singh, D., Singh, S. K., and Dutta, M. (2010b). Hand written character recognition using twelve directional feature input and neural network. *International Journal of Computer Applications*, 1(3):82–85.

[Srihari et al., 2007] Srihari, S. N., Yang, X., and Ball, G. R. (2007). Offline chinese handwriting recognition: A survey. In *Frontiers of Computer Science in China*, page 2007.

[Tatu Harviainen and Takala, 2007] Tatu Harviainen, L. S. and Takala, T. (October 2007). Usability testing of virtual reality aided design: Framework for prototype development and a test scenario. In *, 4th INTUITION international conference and workshop on virtual reality and virtual environments*. Institute of Communication and Computer Systems of the National Technical University of Athens.

[Uchida and Sakoe, 2003] Uchida, S. and Sakoe, H. (2003). Eigen-deformations for elastic matching based handwritten character recognition. *Pattern Recognition*, 36(9):2031 – 2040. Kernel and Subspace Methods for Computer Vision.

[Uchida and Sakoe, 2005] Uchida, S. and Sakoe, H. (2005). A survey of elastic matching techniques for handwritten character recognition. *IEICE - Trans. Inf. Syst.*, E88-D:1781–1790.

[Vuori et al., 2001] Vuori, V., Laaksonen, J., and Oja, E. (2001). Speeding up online recognition of handwritten characters by pruning the prototype set. In *In Proc. 6rd International Conference on Document Analysis and Recognition*, pages 501–505.

[Willis et al., 2011] Willis, K. D., Poupyrev, I., Hudson, S. E., and Mahler, M. (2011). Sidebyside: ad-hoc multi-user interaction with handheld projectors. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 431–440, New York, NY, USA. ACM.

[Xue Gao and Huang, 2001] Xue Gao, Lian-Wen Jin, J.-X. Y. and Huang, J.-C. (2001). A new stroke-based directional feature extraction approach for handwritten chinese character recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 635–640, Washington, DC, USA. IEEE Computer Society.

[Yoshida and Sakoe, 1982] Yoshida, K. and Sakoe, H. (1982). Online handwritten character recognition for a personal computer system. In *IEEE Transactions on Consumer Electronics*, volume 28, pages 202–209, Kawasaki-City, Kanagawa-Pref, Japan. IEEE Computer Society.