Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Prototipo de Silla de Ruedas Controlado por MoCap y SpeechRecognition

Tesis de Licenciatura en Ingeniería en Ciencias de la Computación

Presenta: Celina Osorio Ochoa

Asesor: Dr. Manuel Martín Ortíz

Co-Asesor: Dr. Iván Olmos Pineda

Puebla, Puebla 2013

CONTENIDO

ÍNDICE DE IMÁGENES

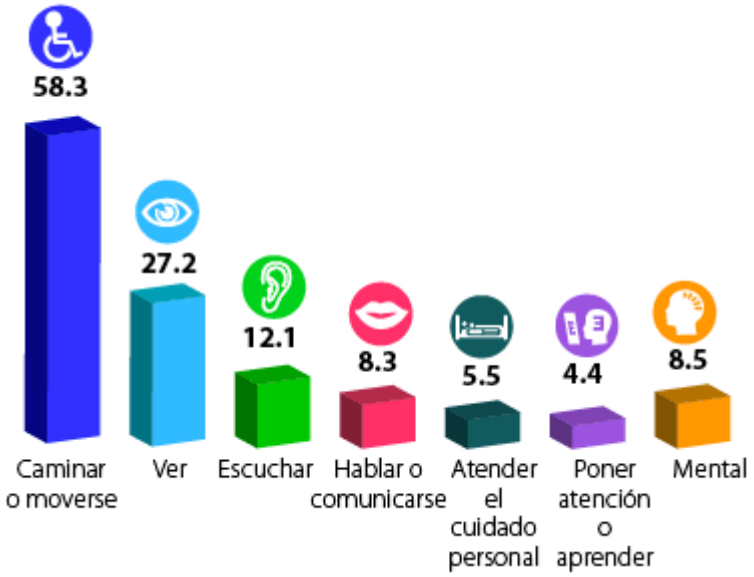
RESUMEN

La atención a los discapacitados ha sido un tema de interés a las diferentes áreas de la ciencia y tecnología. En particular los aportes para problemas de locomoción han sido numerosos. En este trabajo se presenta una interface que permite el control de movimiento de una silla de ruedas a escala mediante métodos de Captura de Movimiento (MoCap) y Speech Recognition. La implementación se basa en la identificación de los movimientos de la cabeza del sujeto mediante MoCap, comandos hablados y una máquina de estados asociada, la cual alimenta a los controles de la silla de ruedas. Se probaron varios esquemas de reconocimiento para su evaluación. La comunicación entre el sistema de reconocimiento y el prototipo de silla se hace vía Bluetooth.

CAPÍTULO I. INTRODUCCIÓN

En la actualidad es posible encontrar diversos trabajos en desarrollo de prototipos de sillas de ruedas para personas con lesiones medulares, sin embargo, las técnicas de captura de movimiento y reconocimiento de voz en conjunto no son muy utilizadas para este tipo de proyectos.

En México, la limitación de la movilidad es la que se presenta con mayor frecuencia, alrededor de la mitad de las limitaciones declaradas por el INEGI en 2010 (Gráfica 1), se refieren a caminar o moverse.



Gráfica 1. Porcentaje de la Población con Discapacidad según Dificultad en la Actividad [1]

No existen estadísticas oficiales acerca de la lesión medular específicamente, pero se sabe que pertenecen a esta clasificación. Éstas son lesiones graves que ocasionan una pérdida de la función motora y sensitiva por lesión en las estructuras nerviosas de la médula espinal, pueden ocasionar fatiga y debilidad en brazos y piernas, por lo que es necesario asistir a los pacientes diagnosticados con lesión medular, en las actividades de la vida diaria básicas (AVDB), lo cual tiene una repercusión personal, social y económica para el paciente y su familia [2]. Por ello, la simplicidad con la cual

pueden capturarse movimientos básicos de la cabeza lleva a la idea de que los pacientes con lesiones medulares podrían controlar su propia silla de ruedas de una manera más sencilla sin la necesidad de recurrir a las soluciones actuales a dicho problema que en su mayoría, han sido diseñadas para tipos particulares de lesión medular y que, en muchos de los casos, no se ajustan a sus necesidades; como las palancas de mando o paneles de direcciones. Aunque cabe mencionar, que existen en México, otros esfuerzos que buscan apoyar a este sector [3] [4] y que aún se encuentran perfeccionando los sistemas antes de llevarlos a una fase de pruebas con aquellos que tienen una limitación de movilidad.

En el caso de este trabajo se plantea el control con la captura de movimiento de la cabeza y reconocimiento de voz. Además, debido a que el prototipo realizado en este trabajo puede ser precedente para una implementación en tamaño real, se pretende que los recursos del sistema de control utilizados no representen un alto costo para los pacientes o sus familiares, aunado a las repercusiones antes mencionadas.

1.1. Antecedentes del Proyecto

Existen trabajos de sillas de ruedas orientados a este tipo de situaciones que son controlados por voz [5] [6], pero dicha técnica sin subsistemas de apoyo no funciona correctamente debido a la cantidad de ruido que llega a haber en un ambiente real y complejo, además con ellos se tiene el problema de identificación del locutor.

Con MoCap ha habido innumerables sistemas que utilizan un control a través del movimiento de ojos [7] [8] [9] [10] [11] [12], en ellos puede ser difícil la ejecución de los comandos pero esta solución está orientada a personas con una muy específica lesión medular que no es directamente abordada por este trabajo, como se verá más adelante.

También existen los sistemas de control automático, los cuales tienen un buen desempeño, funcionan sólo con caminos predeterminados o con características muy específicas [7], por lo que también es necesario el uso de subsistemas que permitan un control semiautomático de las mismas.

Los sistemas clásicos son los controlados por una palanca de mando (Joystick) [13]y se ha visto que funcionan bien para las personas con lesión medular que tienen movilidad en los brazos.

Y finalmente, los sistemas más sofisticados son los controlados con interfaces cerebro-ordenador, las cuales están basados en el procesamiento de las señales electroencefalográficas del sujeto y suelen tener buenos resultados, pero requieren un largo entrenamiento del usuario y ajustes persona-

lizados de algunos factores del sistema, para su correcto funcionamiento [14] [15].

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Desarrollar un software que permita controlar un prototipo de silla de ruedas a través de los movimientos de la cabeza y reconocimiento de voz que sirva como base tecnológica para el desarrollo de un sistema que permita a las personas con lesión medular sin movilidad en brazos y piernas el desplazamiento autónomo en su silla de ruedas con mayor facilidad.

1.2.2. OBJETIVOS ESPECÍFICOS

- Diseñar e implementar un prototipo de silla de ruedas a través del kit de robótica programable LEGO MINDSTORMS NXT.
- Diseñar e implementar un módulo de control global capaz de interactuar y administrar los otros módulos.
- Diseñar e implementar el sub-módulo de captura de movimiento de la cabeza a través de MoCap Óptico.
- Diseñar e implementar el sub-módulo de reconocimiento de voz a través del uso de un micrófono.
- Realizar un ciclo de pruebas al finalizar el diseño e implementación de cada módulo.

1.3. ALCANCES Y LIMITACIONES

Como se mencionó anteriormente, este trabajo es la fase temprana de un proyecto de silla de ruedas para personas con limitaciones de movilidad, por lo cual sólo se desarrolla el sistema de control para un prototipo a escala de una silla de ruedas.

No será posible que una persona sea soportada por el prototipo ya que es muy pequeño (las especificaciones de tamaño se definen más adelante).

CAPÍTULO II. MARCO TEÓRICO

En el sistema de control del prototipo de silla de ruedas a escala, se desarrollaron varios módulos. El módulo que corresponde a la captura de movimiento o MoCap con el cual se obtienen los movimientos de la cabeza, encontrando la posición del rostro. El módulo correspondiente al control por comandos hablados que utiliza Procesamiento Digital del Habla y en específico Speech Recognition. Un módulo en el que se hace un mapeo de los comandos de los módulos anteriores a datos entendibles para otro módulo que es el encargado de controlar el prototipo de silla de ruedas a escala construido con LEGO MINDSTORMS NXT.

Además de lo anterior se hizo un análisis de la lesión medular, para entender el enfoque de este sistema de control.

2.1. MOCAP

Captura de Movimiento o *MoCap* (Por sus siglas en inglés *Motion Capture*) se refiere al muestreo y registro de movimientos de humanos, animales y objetos inanimados como datos 2D o 3D [16]. Los datos pueden ser usados para estudiar el movimiento capturado o para dar la ilusión de vida a modelos computacionales en 3D. A pesar de que la mayoría de las aplicaciones *MoCap* requieren equipamiento especial, existen un gran número de compañías, escuelas y organizaciones que usan tecnología de este tipo [17] [18] [19]. Para hacer *MoCap*, es importante hacer pre-procesamiento de datos y una planeación de cómo se hará la captura, de qué tipo y qué algoritmos se implementarán.

2.1.1. ANTECEDENTES DE MOCAP

Podría pensarse que el *MoCap* sólo existe a causa de la existencia de computadoras y de la tecnología actual, pero los primeros intentos de capturar el movimiento datan de 1872, cuando Eadweard Muybridge a través de la invención del zoopraxiscopio –dispositivo que proyecta rápidamente imágenes secuenciales- capturó animales y humanos en movimiento (Imagen 1) [20] [21]. Después de Muybridge hubo múltiples aportaciones al *MoCap*, entre ellas, el estroboscopio –dispositivo que captura ob-

jetos que se mueven rápidamente en una película- de Edgerton en 1931 y el Rotoscopio – dispositivo que producía animación *frame* por *frame*- de Max Fleischer. Este último fue utilizado posteriormente para producir animaciones a Disney, comenzando por Blanca Nieves.

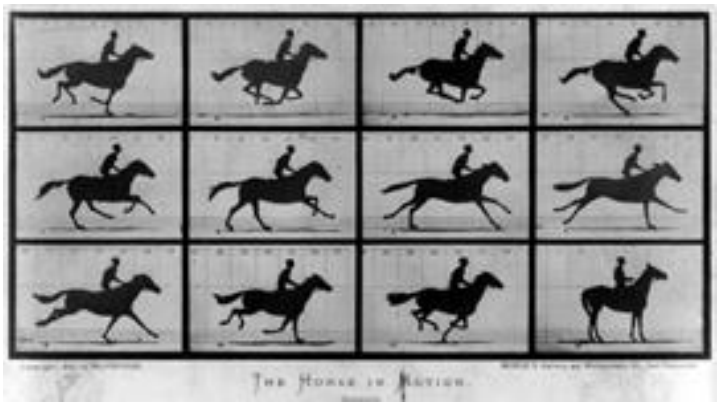


Imagen 1. "The Horse in Motion" de Eadweard Muybridge

Alrededor de 1980, la industria de las imágenes generadas por computadora (CGI) descubrió el potencial que existía en la investigación y el desarrollo de la tecnología de *MoCap* digital y en 1985 la empresa Wavefront Technologies desarrolló y comercializó el primer software para animación en computadora. Entonces, la mayor parte de las animaciones producidas eran sólo anuncios cortos de 15 a 30 segundos de duración.

Desde los esfuerzos de los pioneros en *MoCap* se han observado notables desarrollos y aportaciones en el *MoCap* digital. En años recientes, además de las aplicaciones para medicina y entretenimiento, se han encontrado aplicaciones de *MoCap* en diversos campos, como el diseño, los deportes y la ingeniería.

2.1.2. *TIPOS DE MOCAP*

Los sistemas de *MoCap* comercialmente disponibles hoy día, pueden ser categorizados en tres grandes grupos: Sistemas Ópticos, Sistemas Magnéticos y Sistemas Mecánicos. Cada tipo con sus fortalezas y debilidades. Aunque también existen los Sistemas Ultrasónicos e Inerciales, los más importantes son los tres listados arriba.

Sistemas Ópticos de MoCap

Estos sistemas utilizan una o más cámaras para obtener información, la mayoría mediante marcadores que pueden tener propiedades reflectivas –pasivos- o emisores de luz –activos-(Imagen 2). Los marcadores pasivos son hechos de materiales reflectivos y por lo general, sus formas son esféricas o circulares; aunque la forma y el tamaño, pueden depender de la

resolución de la cámara y del sujeto/objeto capturado. Los marcadores activos son diodos emisores de luz (LEDs), que pueden iluminarse uno a uno, o todos al mismo tiempo modulando la amplitud y la frecuencia de cada LED, permitiendo así que los sistemas puedan identificar a los marcadores.

Imagen 2. Marcadores Pasivos para MoCap Óptico [22]

Las cámaras en un sistema de MoCap óptico deben capturar entre 30 y 2000 muestras por segundo. Además, dependiendo de la aplicación, se sugiere el uso de dos cámaras, de tal forma que pueda evitarse que los marcadores queden invisibles ante algún movimiento (oclusión); pues esto podría generar pérdida de información y por lo tanto inconsistencia de los datos.

Aunque también existen tecnologías de MoCap que no necesitan utilizar marcadores, como Mova's Contour Reality Capture System (Imagen 3), en el que se captura la piel usando un maquillaje fosforescente, de tal forma que el conjunto de cámaras capturará la textura y la geometría del sujeto en movimiento [23].

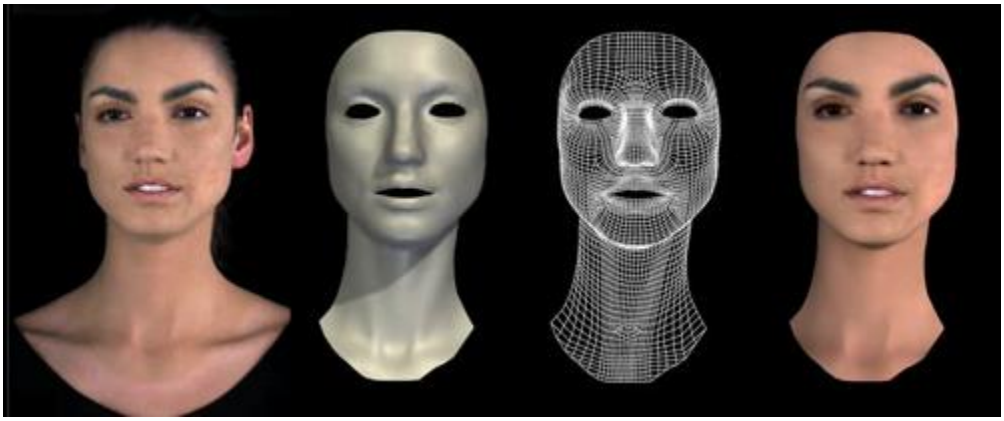


Imagen 3. Proceso de Captura de MOVA [23]

Ventajas y Desventajas de los Sistemas Ópticos de MoCap

En los sistemas ópticos, los datos son precisos y la velocidad de captura es alta, además se puede capturar a varios sujetos simultáneamente con un gran número de marcadores en ellos, que además pueden ser cambiados de lugar fácilmente. Todo ello, permite que se puedan generar esqueletos con movimientos libres del sujeto de captura.

Sin embargo, se requiere una gran cantidad de post-procesamiento y una buena iluminación, puede ocurrir oclusión de marcadores cuando el sujeto se mueve -lo cual ocasiona pérdida de datos-, es difícil observar la reconstrucción del escenario capturado en tiempo real y el hardware suele ser más caro que en los otros tipos de MoCap.

Sistemas Magnéticos de MoCap

Son llamados Magnéticos o Electromagnéticos, en éstos se utilizan sensores de seguimiento que se colocan en el sujeto de captura para medir la relación espacial a un transmisor magnético (Imagen 4). La salida de los sensores contiene el desplazamiento y la orientación, por lo que puede ser utilizado en tiempo real sin necesidad de un post-procesamiento.

Con estos sistemas no ocurre oclusión, pero puede existir interferencia causada por objetos metálicos o electrónicos, lo que podría resultar en datos de salida distorsionados. Además será necesario pensar en el uso de baterías y su recarga. Su velocidad de muestreo es de 144 a 240 muestras por segundo y no es tan simple cambiar la configuración de los marcadores.

Imagen 4. Sensor de Seguimiento de Sistema Electromagnético [24]

Ventajas y Desventajas de los Sistemas Magnéticos de MoCap

En estos sistemas, la posición y la orientación están disponibles sin hacer post-procesamiento, esto significa que también es posible observar el escenario que se captura en tiempo real. En ellos no ocurre oclusión, se puede capturar a varios sujetos con diferentes configuraciones de marcadores y son más económicos que los sistemas ópticos.

Sin embargo, los sensores que fungen como marcadores están propensos a interferencia magnética y electromagnética -por lo que los datos pueden ser ruidosos-, la velocidad con que se obtienen los datos es más lenta que con los sistemas ópticos, es difícil cambiar la configuración de los sensores y requieren baterías.

Sistemas Mecánicos de MoCap

Estos sistemas miden directamente los ángulos de las articulaciones de un sujeto que usa un dispositivo articulado que consiste de barras rectas y potenciómetros. Las barras son unidas a los potenciómetros en las articulaciones del cuerpo ya que fueron diseñadas para medir los ángulos de las articulaciones cuando el sujeto se mueve.

Los sistemas mecánicos pueden ser usados en tiempo real, no son propensos a oclusión ni a interferencias eléctricas o magnéticas. Pero no miden muy bien las traslaciones globales debido a que utilizan acelerómetros.

Ventajas y Desventajas de los Sistemas Mecánicos de MoCap

Estos sistemas son relativamente económicos, puede verse la captura en tiempo real, no ocurre oclusión, no hay interferencia eléctrica o magnética y son altamente portables. No obstante, hay restricciones en la captura del movimiento del sujeto ya que el sistema es frágil, la configuración de los marcadores es fija y tienen una velocidad de muestreo baja.

2.1.3. ALGORITMOS DE MOCAP ÓPTICO

Los algoritmos de procesamiento de imágenes que pueden ser utilizados para desarrollar aplicaciones con captura de movimiento óptica son variados, por lo que sólo se definirá el que se usó para controlar el prototipo de silla de ruedas, que consiste en la detección de la cara y luego su posición en el espacio.

2.1.4. VIOLA-JONES [25]

Este enfoque de detección de objetos combina cuatro conceptos principales: Simples características rectangulares -llamadas Haar Features (Características Haar)-, una imagen integral para una rápida detección de características, el método de aprendizaje automático AdaBoost y un clasificador en cascada que combina muchas características eficientemente.

Características o Features

Las características que Viola y Jones usaron están basadas en Haar Wavelets. Éstos son longitudes de onda cuadradas (una en intervalo alto y otra en intervalo bajo). En dos dimensiones, una onda cuadrada es un par de rectángulos adyacentes –uno claro y el otro oscuro-.

Las combinaciones de rectángulos usadas para la detección visual de objetos no son Haar Wavelets específicamente, sino que contienen combinaciones de rectángulos mejor adecuadas para las tareas de reconocimiento visual. Debido a esa diferencia, estas características son llamadas Haar Features o Haar Like Features, en lugar de Haar Wavelets. Un ejemplo de ellas se sitúa en la Imagen 5, A y B son características de dos rectángulos, C de tres y D de cuatro.

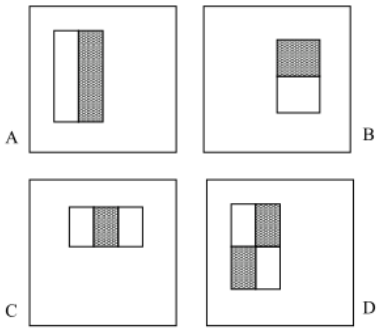


Imagen 5. Ejemplo de Características Rectangulares

La presencia de un Haar Feature es determinada restando la suma de los pixeles de la región oscura de la suma de los pixeles de la región clara. Si la diferencia es arriba de un umbral (que se determina durante el entrenamiento) se dice que la característica (o el Haar Feature) está presente.

Para determinar la presencia o ausencia de Haar Features en cada parte de la imagen a diferentes escalas, de forma eficiente, Viola y Jones usaron una representación de la imagen llamada Imagen Integral, en la que el valor integral para cada pixel es la suma de todos los pixeles arriba y a la izquierda del mismo:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y')$$

Donde $ii(x,y)$es de la imagen integral e $i(x,y)$es de la imagen original. Usando el siguiente par de recurrencias:

$$s(x,y) = s(x,y-1) + i\,(x,y)\ (1)$$

$$ii(x,y) = ii(x-1,y) + s(x,y)\ (2)$$

(donde $s(x,y)$es la suma acumulativa del renglón, $s(x,-1) = 0$ e $ii(-1,\ y) = 0$)

La imagen puede ser calculada en sólo una pasada sobre la imagen con unas pocas operaciones por pixel.

En la Imagen 6, el pixel localizado en (x,y) contiene la suma de los valores de los pixeles dentro de la región rectangular sombreada y puede ser calculado usando cuatro referencias de la matriz. El valor de la imagen integral en la posición 1 es la suma de los pixeles en el rectángulo A. El valor en la posición 2 es $A+B$, en la posición 3 es $A+C$, y en la posición 4 es $A+B+C+D$. La suma de los pixeles en D puede ser calculada como $4+1-(2+3)$ o como $(x_4,y_4)+(x_1,y_1)-(x_2,y_2)-(x_3,y_3)$.

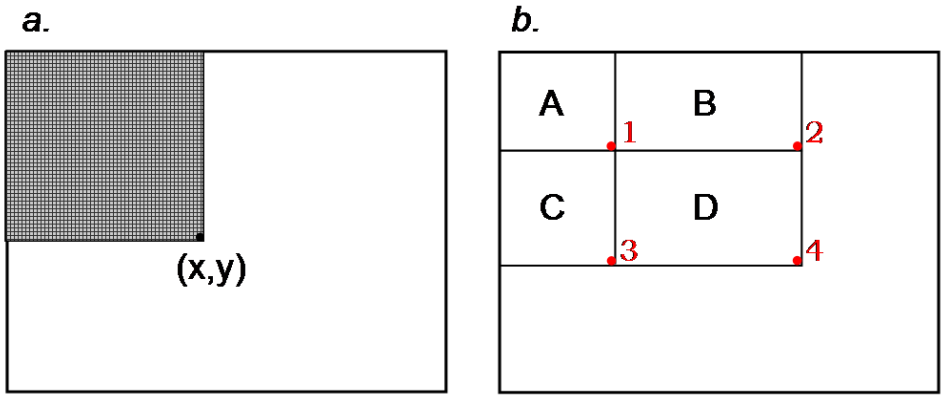


Imagen 6. Imagen Integral

Sin embargo, aunque este cómputo es eficiente y la resolución base del detector es de 24x24, el número de características resultantes está por arriba de 180,000.

Funciones de Aprendizaje para la Clasificación

Dados un conjunto de características y un conjunto de entrenamiento de imágenes positivas y negativas, se utiliza una variante del algoritmo AdaBoost tanto para seleccionar un conjunto más pequeño de características, como para entrenar al clasificador. AdaBoost en su forma original es usado

para aumentar el rendimiento de la clasificación de un algoritmo de aprendizaje simple (o débil).

Como se mencionó, existen más de 180,000 características rectangulares asociadas con cada sub-imagen y procesar el conjunto completo de características podría ser bastante costoso. Por ello, Viola-Jones propusieron tomar algunas características y con ellas proporcionar una clasificación efectiva. Para esto se utiliza el algoritmo débil que está diseñado para seleccionar sólo la característica rectangular que mejor separa los ejemplos positivos y negativos.

AdaBoost selecciona un conjunto de clasificadores débiles y asigna un peso a cada uno (Tabla 1). La combinación ponderada se convierte en un clasificador fuerte.

- Dadas las imágenes $(x_1, y_1), \dots, (x_n, y_n)$ donde $y_i = 0{,}1$ para ejemplos positivos y negativos respectivamente.
- Inicializar pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$ respectivamente, donde m y l son números negativos y positivos respectivamente.
- Para $t = 1, \dots, T$:

1. Normalizar los pesos,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

   Tal que w_t es una distribución de probabilidad.
2. Para cada característica j entrenar un clasificador h_j el cual está restringido a usar una sola característica. El error es evaluado con respecto a $w_t, \epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
3. Elegir el clasificador h_t con el menor error ϵ_t.
4. Actualizar los pesos:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

   donde $e_i = 0$ si el ejemplo x_i es clasificado correctamente, $e_i = 1$ de otra manera y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- El clasificador final es:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & de\ otra\ forma \end{cases}$$

  donde $\alpha_t = log \frac{1}{\beta_t}$

Tabla 1. Algoritmo AdaBoost para Clasificar

Clasificadores en Cascada o Cascade Classifiers

Viola y Jones combinaron series de clasificadores AdaBoost como una cadena de filtros (Imagen 7) que es especialmente eficiente clasificando regiones de la imagen.

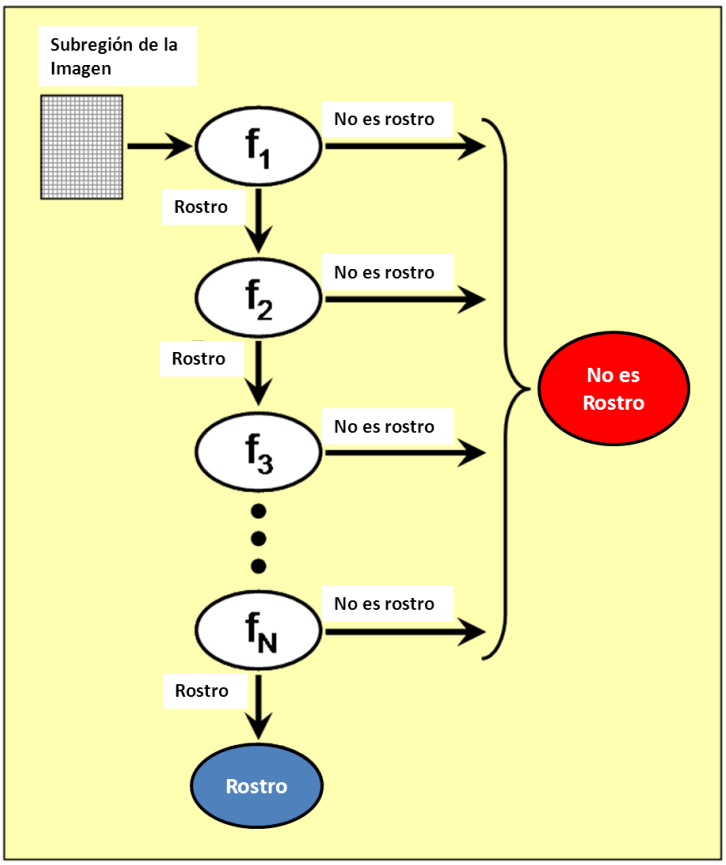


Imagen 7. Cadena de Clasificadores y su Funcionamiento con Subregiones de la Imagen

Cada filtro es un clasificador AdaBoost separado con un muy pequeño número de clasificadores débiles.

El umbral de aceptación en cada nivel es puesto lo suficientemente bajo para dejar pasar todos, o casi todos los ejemplos de rostro en la etapa de entrenamiento. Los filtros en cada nivel son entrenados para clasificar la imágenes de entrenamiento que pasaron por todas las etapas anteriores (El conjunto de entrenamiento es una gran base de datos de caras). Durante su uso, si por alguno de estos filtros falla el paso de la subregión de la imagen, esa región es clasificada inmediatamente como "No es rostro". Cuando la región de la imagen pasa el algún filtro, ésta se dirige al siguiente filtro en la cadena. Las regiones de la imagen que pasan a través de todos los filtros en la cadena son clasificadas como "Rostro". Viola y Jones denominaron a la cadena de filtrado como cascada.

El orden de los filtros en la cascada está basado en los pesos que AdaBoost asigna y el más grande valor del filtro ponderado va primero, para eliminar las regiones de la imagen que no son caras tan pronto como sea posible.

2.1.5. HAAR-LIKE FEATURES EN EMGUCV

El algoritmo Haar-Like Features de Viola-Jones ha sido implementado en la librería OpenCV [26]. Con OpenCV es posible hacer el proceso completo de entrenamiento (descrito en el Apéndice A) o solamente utilizar el método de detección de rostros.

Para utilizar la librería en el desarrollo del proyecto en Microsoft Visual Studio 2010 en C# se utilizó el wrapper EmguCV [27].

2.2. PROCESAMIENTO DIGITAL DEL HABLA

Aunque el término speech recognition, se ha traducido al español como reconocimiento del habla, es más común encontrar en la literatura el término reconocimiento de voz haciendo alusión al primero. A pesar de ello, se utilizará el término reconocimiento del habla al proceso de traducción de las palabras habladas a texto.

La comunicación hablada es una habilidad esencial, poseída por el ser humano. Y aunque es posible comunicarse de forma visual es demasiado ineficiente comparado con lo que es posible comunicar a través del habla.

Ha sido confirmado que las estructuras acústicas y lingüísticas del habla han sido intrínsecamente relacionadas con nuestra habilidad intelectual, y además están íntimamente relacionadas con nuestro desarrollo social y cultural [28].

La historia de la investigación en el habla empezó con el desarrollo de sintetizadores mecánicos de habla hacia el final del siglo XVIII y con vibración vocal y mecanismos de escucha a mediados del siglo IX. Antes de la invención del PCM (modulación por impulsos codificados) en 1938, ya se había tratado la onda del habla con técnicas de procesamiento analógico. La invención del PCM y el desarrollo de circuitos digitales y computadoras electrónicas han hecho posible el procesamiento digital del habla y ello ha traído un gran progreso en procesamiento de la información hablada, especialmente después de 1960.

Los dos trabajos de investigación más importantes en lo 60's fueron el Sistema de Análisis-Síntesis del Habla basado en el Método de Máxima Verosimilitud presentado por NTT's Electrical Comunication Laboratories y también el trabajo en Codificación Predictiva presentado por Bell Laboratories. Estos trabajos produjeron esencialmente el más grande progreso en tecno-

logía de procesamiento de la información hablada, pues ambas utilizan la técnica de compresión de la información usando predicción lineal de ondas de habla que están basadas en técnicas matemáticas para procesos estocásticos.

Otras técnicas de procesamiento del habla han sido desarrolladas y en conjunto han contribuido a la realización de un amplio rango de sistemas que operan bajo los principios de codificación del habla, análisis-síntesis del habla, síntesis del habla, reconocimiento del habla y reconocimiento de voz o reconocimiento del hablante.

2.2.1. SPEECH RECOGNITION O RECONOCIMIENTO DEL HABLA

Reconocimiento del Habla o *Speech Recognition* es el proceso de extracción y determinación de la información lingüística, también conocida como información fonética. En un sentido más amplio, reconocimiento del habla incluye reconocimiento de voz o reconocimiento del hablante, lo que implicaría extraer información individual para indicar quién está hablando. Pero para efectos de este trabajo, se referirá al término solamente como el reconocimiento de la información lingüística [29].

Ventajas y Desventajas del Speech Recognition

El uso del reconocimiento del habla en los sistemas de información tiene múltiples ventajas, es fácil de usar debido a que no requiere habilidades especiales tales como escribir en un teclado o presionar botones para realizar operaciones, además la palabra hablada puede ser usada como entrada de información más rápido que la palabra escrita y mientras el usuario realiza alguna otra actividad que involucre las manos, piernas, ojos y oídos. Los dispositivos para capturar el sonido (en este caso el habla), suelen ser económicos y para la transmisión de esta información en tiempo real, pueden utilizarse tecnologías existentes como el teléfono o el internet.

Pero independientemente de las ventajas que puede tener el reconocimiento del habla, existen diversas desventajas relacionadas con el tratamiento del ruido o la separación de palabras dependiendo del hablante o el idioma. Todo ello, puede resumirse como sigue.

- Coarticulación. Cuando se dice una oración, el espectro de una palabra es diferente cuando está acompañada o precedida por diferentes palabras.
- Segmentación. El espectro de una secuencia de palabras dichas, tiene una estructura tal, que es difícil encontrar la separación.
- Individualidad. Las características acústicas, varían entre personas, incluso cuando se pronuncian las mismas palabras.

- Conocimiento lingüístico insuficiente. Las características físicas de los fonemas no proporcionan tanta información como el conocimiento del lenguaje. Las personas pueden saber qué es lo que alguien va a decir debido a que conocen la estructura del lenguaje y qué palabras pueden preceder en una oración, sin embargo, es difícil intentar modelar estas reglas.

2.2.2. TIPOS DE SPEECH RECOGNITION

Existen dos tipos de clasificadores del habla, los que reconocen palabras aisladas y los que reconocen flujos continuos del habla. Éstos se subdividen en Transcripción y Entendimiento.

Para reconocer flujos continuos, es necesario conocer el lenguaje, para aplicar reglas de gramática que gobiernan esas secuencias. También se pueden clasificar en dependientes del hablante o independientes del hablante.

En lo particular, no se implementó un algoritmo de Speech Recognition, para ello, se utilizó la librería Microsoft Speech Platform Version 11, de la cual no se detalla el algoritmo pero si se da una breve descripción de su funcionamiento.

2.2.3. MICROSOFT SPEECH PLATFORM SDK 11 PARA SPEECH RECOGNITION

Microsoft Speech Platform SDK 11 (MSP SDK 11) proporciona las herramientas con las cuales pueden adquirirse y monitorearse entradas habladas, crearse gramáticas para Speech Recognition, capturar información de los eventos generados por el motor de Speech Recognition y configurar y administrar los motores de Speech Recognition [30].

Un motor de *Speech Recognition* o *Speech Recognizer* de MSP SDK 11 toma un flujo de audio como entrada y lo convierte en una transcripción textual. El proceso de *Speech Recognition* puede ser pensado en dos partes, la primera procesa el flujo de audio, aísla los segmentos de sonido que probablemente pueden ser palabras y los convierte en series de valores numéricos que caracterizan el sonido en la señal. La segunda parte, es un motor de búsqueda especializado que toma la salida producida por el primer proceso y busca a través de tres bases de datos: un modelo acústico, un léxico, un modelo del lenguaje.

El modelo acústico representa los sonidos acústicos de un lenguaje, y puede ser entrenado para reconocer las características de los patrones del habla de un usuario en particular. El léxico lista un largo número de palabras de un lenguaje y provee información sobre cómo se pronuncia cada

palabra. Y el modelo del lenguaje representa las formas en las cuales las palabras son combinadas en un lenguaje.

Para cualquier segmento de sonido, hay muchas cosas que el hablante podría estar diciendo. La calidad de un reconocedor es determinada por qué tan refinada es su búsqueda, eliminando las coincidencias pobres y seleccionando las coincidencias más parecidas. Esto depende en gran medida de la efectividad de los algoritmos para procesar el sonido, de los modelos y de las búsquedas en ellos.

2.3. LEGO MINDSTORMS NXT 2.0

LEGO MINDSTORMS NXT 2.0 es el más nuevo set de las series LEGO MINDSTORMS, contiene 619 piezas y existen muchas formas de programarlo y/o controlarlo. Se puede utilizar el software oficial de LEGO MINDSTORM que es libre y fácil de usar para niños, pues cuenta con un modo de programación de arrastrar y soltar íconos en un área de desarrollo visual, pero el acuerdo de licencia para el uso de este software es demasiado restrictivo y no permite el uso comercial de proyectos desarrollados con él.

Es posible utilizar Protocolo de Comunicación LEGO NXT (LCP) que es completamente libre y utiliza programación de bajo nivel.

También puede utilizarse software de terceros, existen muchos wrappers libres y no libres para el protocolo de comunicación LEGO y su firmware.

2.3.1. PROTOCOLO DE COMUNICACIÓN LEGO NXT

Este protocolo es usado para controlar el NXT desde un dispositivo remoto como un teléfono móvil, una computadora u otros robots enviando un volumen de bytes. Estos comandos se dividen en dos grupos:

- Comandos de sistema. Comandos que controlan la administración del bloque o del protocolo como el acceso al sistema de archivos en el bloque inteligente o la configuración de la comunicación.
- Comandos directos. Comandos que controlan el movimiento del robot a través de los motores o la lectura de los sensores.

Algunos comandos son sólo peticiones de retroalimentación por parte del robot, como obtener datos de los sensores y se envían a través de telegramas de dos bytes.

El Lego NXT puede operar mediante dos formas de comunicación: Bluetooth y USB 2.0 y a dos niveles de velocidad de transferencia: 1Mbit/s (Velocidad alta) y 9600 bits/s (velocidad baja), cual sea el caso. El protocolo de transferencia utilizado es el I^2C.

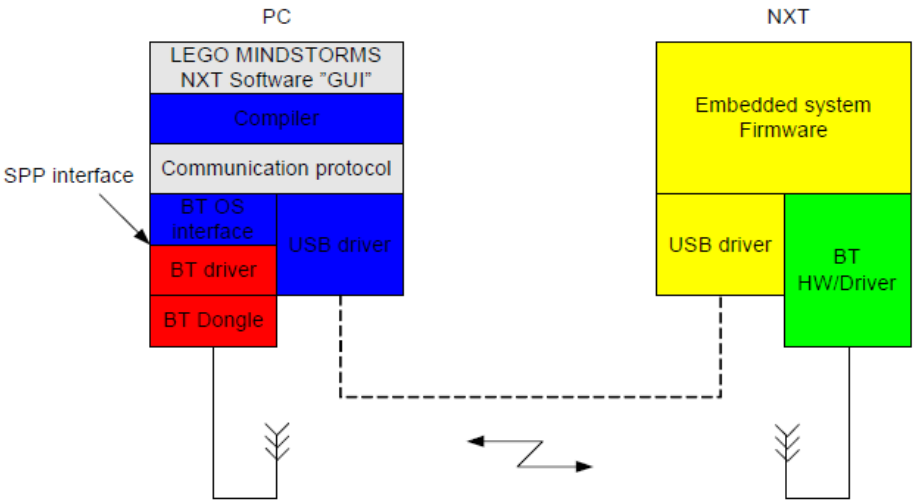


Imagen 8. Esquema de las capas de transferencia de datos LEGO NXT

En la Imagen 8 se muestran las capas de transferencia de datos entre la PC y el NXT más importantes con la posibilidad de utilizar los dos diferentes medios de comunicación mencionados.

El protocolo acepta las siguientes operaciones:

- Descarga de archivos
- Cargar archivos
- Borrar datos en el NXT
- Comunicación directa con el NXT (ejecución de comandos, mensajes de línea de comandos)

La estructura de los paquetes utilizados en la transferencia de datos con el lego NXT es la mostrada en la Imagen 9.

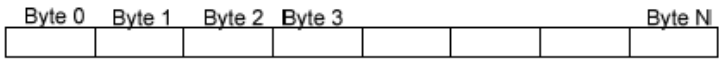


Imagen 9. Estructura de transferencia de datos

Byte 0: Sirve para especificar el tipo de mensaje que será enviado y también para especificar si se requiere de una respuesta.

Byte 1: Especificar al modulo de carga de datos qué hacer con los datos (leer, escribir o borrar).

Byte 2-N: información adicional.

El tamaño del buffer de entrada es de 64 bytes -cuando éste se llena-, que se transfieren a la memoria flash. Este proceso toma un tiempo de 6 milise-

gundos, periodo de tiempo en el cual el procesador no puede recibir más información.

Comunicación Bluetooth

Para la comunicación Bluetooth se utiliza la estructura de un puerto serial a través del puerto UART del procesador ARM7 en el Bloque Inteligente. Existe una desventaja considerable del uso del Bluetooth y es que hay un tiempo de penalización (alrededor de 30 milisegundos) con el chip Bluecore [REF] en el momento en el que este cambia su funcionalidad de enviar a recibir y viceversa.

La implementación del modulo Bluetooth en el NXT utiliza paquetes de datos que especifican el tamaño total de los mensajes, de forma que se reduzca el numero de veces que se realiza el cambio de modo de operación del chip Bluecore.

Comunicación USB

La comunicación vía USB es realizada en volúmenes con un tamaño de 64 bytes que es la capacidad del buffer del procesador ARM7. En la PC se utiliza el driver Phantom que habilita la comunicación con el NXT.

2.3.2. *LIBRERÍA MINDSQUALLS*

Como se mencionó anteriormente, existe diferente software de terceros con el cual es posible controlar el LEGO MINDSTORMS NXT 2.0 como MindSqualls [31].

La librería MindSqualls es una de las librerías mejor documentadas, de fácil uso y lo suficientemente robusta para controlar remotamente el LEGO MINDSTORM NXT 2.0 en C# con las funciones y características deseadas para el sistema de control desarrollado.

Aunque está escrita en C# puede ser usada por cualquiera de los lenguajes de programación .Net, ofrece soporte para todos los comandos directos y es software libre gratuito.

2.4. LESIÓN MEDULAR

La médula espinal es el camino que los mensajes usan para viajar entre el cerebro y las diferentes partes del cuerpo y está rodeada por anillos de hueso llamados vértebras. Estos constituyen la columna vertebral. En general, entre más alta sea la lesión en la columna vertebral, más problemas de funciones experimentará la persona. Cada vértebra toma su nombre de acuerdo con su localización (Imagen 10).

Imagen 10. Descripción de la Columna Vertebral

Las lesiones de la médula espinal pueden hacer difícil para los nervios entregar las señales entre el cerebro y partes del cuerpo [32]. Generalmente, se clasifican según su severidad -completa o incompleta-, nivel vertebral o de acuerdo al estándar de la American Spinal Injury Association (ASIA) como ASIA A, B, C, D o E (Tabla 2) que toma en cuenta el nivel neurológico, es decir, el último nivel sano tanto sensitivo como motor [33].

Lesión Medular Completa A	**No hay preservación sensitiva ni motora por debajo del nivel de lesión y se abarcan segmentos sacros, es decir, no existe tampoco sensibilidad ni control para miccionar ni defecar.**
Lesión Medular Incompleta B	Hay preservación de la sensibilidad pero no motor por debajo del nivel neurológico abarcando segmentos sacros, es decir existe sensibilidad para defecar y miccionar, pero no control voluntario
Lesión Medular Incompleta C	Hay preservación de la sensibilidad y la fuerza por debajo del nivel de lesión pero los músculos, se encuentran débiles y se consideran no funcionales.
Lesión Medular Incompleta D	Los músculos por debajo del nivel neurológico son funcionales un 75% de ellos
Lesión Medular Incompleta E	La fuerza y la sensibilidad prácticamente esta normal.

Tabla 2. Clasificación de Lesiones Medulares según ASIA [34]

Dado que las primeras siete vértebras en el cuello se llaman cervicales, se le llama a la primera C1, a la siguiente C2, hasta llegar a la séptima. Generalmente las lesiones cervicales (Imagen 11), causan pérdida de las funciones en los brazos y piernas; algunas personas con lesiones por encima de C4 requieren asistencia de algún dispositivo para respirar; lesiones a nivel C5 conservan el control de hombros y bíceps pero no la mano o el puño; en lesiones a nivel C6 existe control del puño sin funcionalidad de la mano y en lesiones a nivel C7 hay quienes incluso pueden estirar los brazos pero tener problemas de destreza con manos y dedos. Las lesiones en las vértebras siguientes normalmente no afectan la movilidad en brazos y manos.

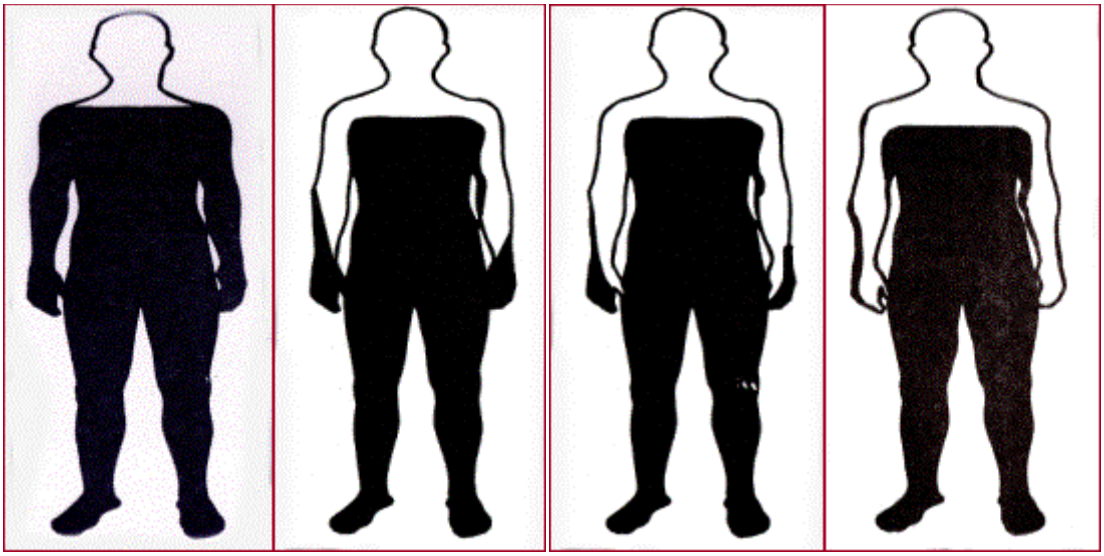


Imagen 11. Lesión Cervical en C4, C5, C6 y C7 (de Izquierda a Derecha) [35]

CAPÍTULO III. PLANTEAMIENTO DEL PROBLEMA Y ANÁLISIS

3.1. PLANTEAMIENTO DEL PROBLEMA Y ANÁLISIS

La cantidad de gente que necesita desplazarse con ayuda de algún dispositivo incrementa continuamente [36]. Esto significa que tomando ventaja de las tecnologías se pueden crear sistemas de control que permitan que la calidad de vida de estas personas mejore a través de su autonomía para desplazarse y además sea más fácil su integración en la sociedad.

Las contribuciones actuales desarrolladas para alcanzar estos objetivos, dependen de la lesión medular y de las limitaciones de movimiento de las personas y proponen diferentes medios para el control de las sillas de ruedas (Antecedentes del Proyecto, Pág. 6).

Para éste trabajo, se tomará en cuenta la clasificación según el nivel vertebral. En específico la solución propuesta al problema de desplazamiento en silla de ruedas para personas con lesiones medulares, está orientada a aquellas con lesión medular en la cervical, cuando hay poco o nulo movimiento en brazos y manos y no se puede utilizar un control tradicional de silla de ruedas basado en palancas de mando.

Para ello se desarrolló un prototipo de silla de ruedas a escala que a través de movimiento de la cabeza y comandos de voz ejecuta movimientos sencillos en diferentes direcciones.

3.2. REQUERIMIENTOS

3.2.1. REQUERIMIENTOS DE SOFTWARE

Debido a que el sistema de control para el prototipo de silla de ruedas está orientado a personas con lesión medular en la cervical, los requerimientos básicos son los siguientes:

- El usuario del sistema de control es capaz de controlar el prototipo mediante el uso conjunto de comandos hablados gestionados a través de la librería Microsoft Speech Platform Version 11 y comandos de movimiento de la cabeza gestionados a través de la librería EmguCV con la implementación de Haar Like Features.
- Se ejecutan los comandos a modo de confirmación, es decir a cada comando hablado le corresponde un comando de movimiento de la cabeza. Si no se corresponden, el prototipo no se mueve.
- Los comandos del movimiento de la cabeza son: Arriba, Abajo, Izquierda, Derecha y Centro, es necesario que la cara esté posicionada en el centro del campo de visión de la cámara o cerca de él.
- Los comandos del movimiento de la cabeza se obtienen a través de la posición del rostro con respecto al campo de visión de la cámara.
- Se requiere una fase de calibración para ajustar los límites de movimiento para la cara del sujeto.
- Los comandos de habla son los descritos en la gramática (Apéndice B) y en la Tabla 3 (Columna: Comando Hablado).
- A cada movimiento del prototipo, corresponde un par de comandos (Tabla 3) y el movimiento de éste se gestiona a través de la librería MindSqualls.

3.2.2. REQUERIMIENTOS DE HARDWARE

Para la construcción del prototipo se utilizan los siguientes componentes de un Kit de Robótica Programable 8547 LEGO® MINDSTORMS® NXT 2.0 [37] (Imagen 12):

- Bloque Inteligente NXT 9841: Consiste en el cerebro del robot, a través de él, los comandos generados por el sistema de control en la computadora, se transmiten a los motores. Contiene un microprocesador 32-bit ARM7, soporte para comunicación por bluetooth, un puerto USB 2.0, 4 puertos de entrada, 3 puertos de salida y es alimentado por una Batería Recargable de litio 9693.
- Batería Recargable 9693: Se trata de la batería recargable de litio que utiliza el bloque inteligente NXT 9841.
- 2 Servo Motores Interactivos 9842: Tienen integrado un sensor de rotación que mide la velocidad y la distancia, datos que reporta al Bloque Inteligente NXT 9841.
- 1 Sensor Ultrasónico 9846: Es capaz de detectar un objeto y medir su proximidad en pulgadas o en centímetros, luego la reporta al Bloque Inteligente NXT 9841.

Imagen 12. Modelo de Robot con el Kit de Robótica Programable 8547 LEGO® MINDSTORMS® NXT 2.0

Comando de Movimiento de la Cabeza	Comando Hablado	Acción del Prototipo
Centro	Muévete hacia adelante Muévete a adelante Muévete adelante	Avanza en línea recta
Derecha	Muévete hacia la derecha Muévete a la derecha Muévete la derecha	Avanza hacia la derecha
Derecha	Gira hacia la derecha Gira a la derecha Gira la derecha	Gira sobre su eje hacia la derecha
Derecha	Retrocede hacia la derecha Retrocede a la derecha Retrocede la derecha	Retrocede moviéndose hacia la derecha
Izquierda	Muévete hacia la izquierda Muévete a la izquierda Muévete la izquierda	Avanza hacia la izquierda
Izquierda	Gira hacia la izquierda Gira a la izquierda Gira la izquierda	Gira sobre su eje hacia la izquierda
Izquierda	Retrocede hacia la izquierda Retrocede a la izquierda Retrocede la izquierda	Retrocede moviéndose hacia la izquierda
Arriba	Retrocede ahora Muévete hacia atrás Muévete a atrás Muévete atrás	Retrocede en línea recta
Abajo	(Cualquiera)	Se detiene
(Cualquiera)	Detente ahora	Se detiene

Tabla 3. Relación entre Comandos Hablados y de Movimiento de la Cabeza para Movimiento del Prototipo

Para hospedar el software se utiliza una computadora portátil con las siguientes características y software instalado:

Características	Software
▪ Procesador 1.7 GHz ▪ Sistema Operativo Windows 7 Home Basic ▪ Memoria 2GB ▪ Cámara de 640×480 pixels ▪ Pantalla ancha HD de 14" ▪ Puerto USB 2.0	▪ Microsoft .NET Framework 4.0 ▪ Microsoft Speech Platform SDK 11 ▪ Microsoft Speech Platform Runtime 11 ▪ Runtime Language Español México ▪ Microsoft Visual Studio 2010 ▪ Emgu.CV-2.4.2 ▪ MindSqualls Version 2.2

3.3. Diseño Conceptual

3.3.1. Configuración de Dispositivos

La forma en la que se acomodan los dispositivos es fundamental para un buen funcionamiento del sistema, no sería lo mismo colocar la cámara frente al sujeto que controla al prototipo que colocarla a un lado de él. Los comandos de movimiento de la cabeza no funcionarían pues es necesario capturar el rostro y por lo tanto nunca habría confirmación de los comandos de voz para realizar algún movimiento.

La cámara se coloca de frente a la persona que controla el prototipo a una distancia aproximada entre 40 y 50 cm y en un ángulo aproximado entre 90° y 110°(Imagen 13).

El micrófono puede colocarse cerca de la boca pero para un mejor funcionamiento es importante hacer pruebas de sonido antes de utilizarlo en conjunto con el sistema.

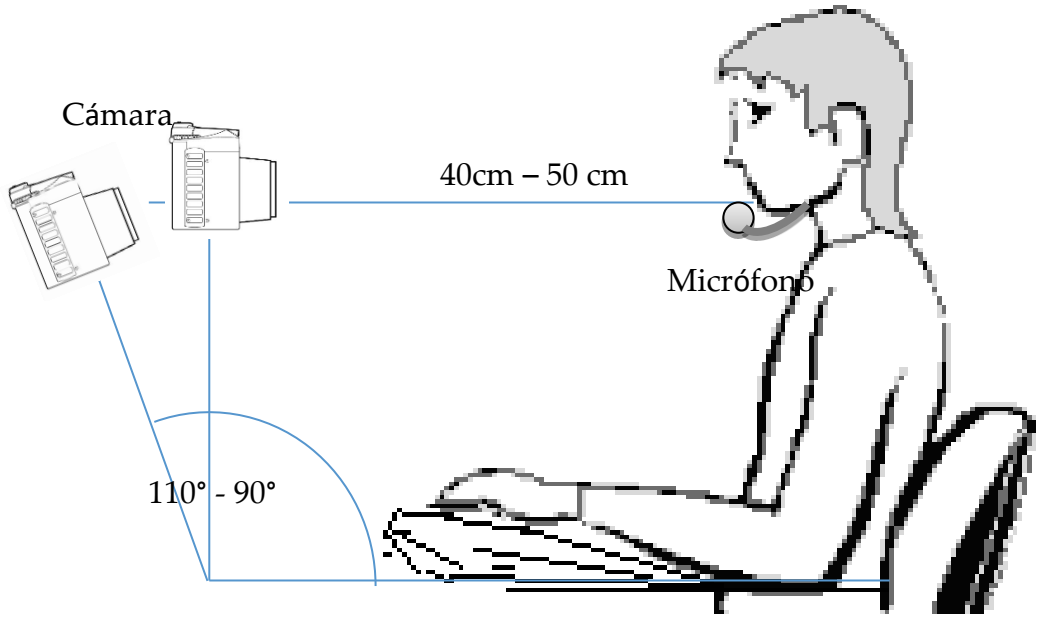


Imagen 13. Configuración de la Cámara y Micrófono con Respecto a la Persona

3.3.2. *Diagrama de Funcionamiento General*

El sistema está compuesto por tres partes principales, el controlador del prototipo, el módulo de captura del habla y el módulo de captura del rostro, su funcionamiento general se muestra en el Diagrama 1.

Al iniciar activan los siguientes módulos:

1. Módulo que captura el habla
2. Módulo que muestra las capturas de la cámara, ajusta los límites de movimiento del rostro y -cuando es necesario- obtiene datos sobre la posición del mismo.
3. Módulo que se encargará de traducir los datos obtenidos de ambos módulos a valores numéricos entendibles por el controlador del prototipo de acuerdo a la Tabla 3 para que éste pueda moverse.

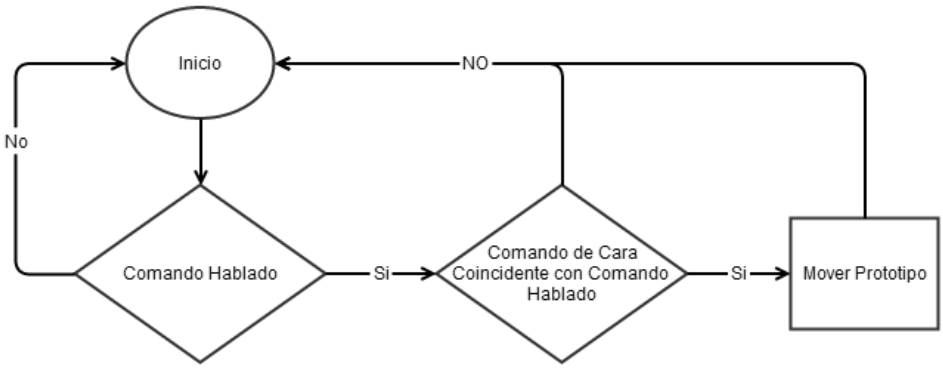


Diagrama 1. Funcionamiento General del Sistema de Control del Prototipo de Silla de Ruedas

CAPÍTULO IV. DISEÑO E IMPLEMENTACIÓN

Para el desarrollo del sistema se utilizó una metodología en espiral, en cada giro se diseñó, implementó y probó cada módulo. Finalmente se hizo una fase de integración en la que también se siguió todo el ciclo de desarrollo. A continuación se muestran los diagramas correspondientes a esta fase para una mejor comprensión del sistema completo.

4.1. DISEÑO

Las clases que se diseñaron para ser implementadas son ControladorLEGO,

CapturadorCara, CapturadorVoz y ControladorPrincipal, coloreadas en el Diagrama 3 con azul y las clases coloreadas de verde son las que se utilizaron de las librerías. La clase SpeechRecognitionEngine de la librería Microsoft Speech Platform SDK 11, NXTBrick de MindSqualls y HaarCascade de EmguCV, éstas se encuentran resumidas en el diagrama, pues sólo se muestra de ellas, métodos y variables que se usarán en la fase de implementación. Para entender el funcionamiento del sistema se puede revisar el Diagrama 2.

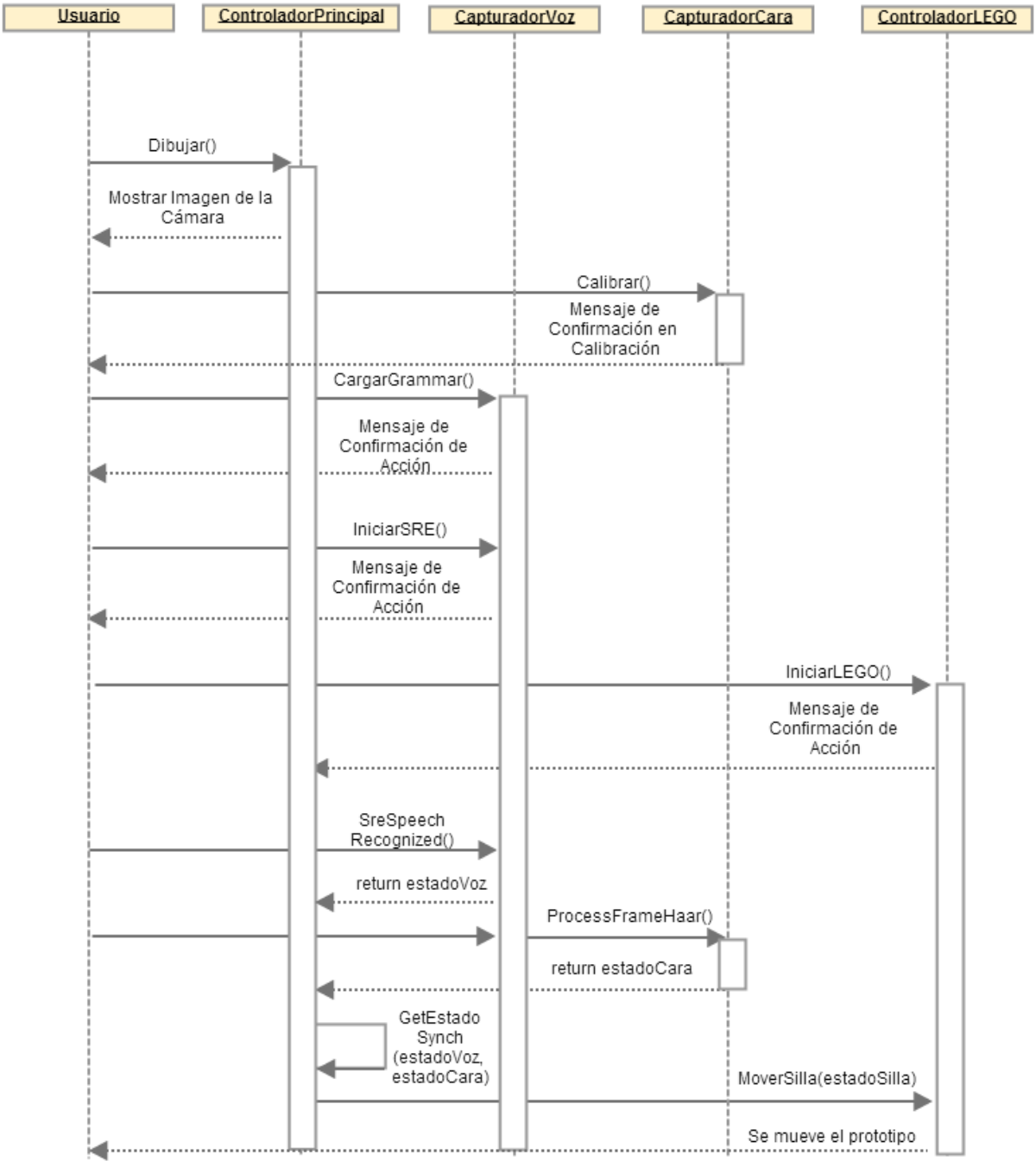


Diagrama 2. Diagrama de Secuencia del Sistema de Control del Prototipo de Silla de Ruedas

4.2. IMPLEMENTACIÓN

La implementación del sistema, debido al origen de librerías se desarrollo en lenguaje C# en Microsoft Visual Studio 2010. A continuación se describe brevemente la implementación de las clases que muestran en el Diagrama 3. Diagrama de Clases del Sistema de Control del Prototipo de Silla de RuedasDiagrama 3.

4.2.1. CLASE CONTROLADORPRINCIPAL

La clase ControladorPrincipal, contiene métodos para controlar cuándo comienzan o terminan, los métodos implementados en ControladorLEGO, CapturadorCara y CapturadorVoz. Contiene solamente un constructor en el cual se inicializan los componentes gráficos, los objetos *capturaCara* y *capturaVoz*, las variables *_iniciaHaar*, *estadoAnteriorVoz* y *estadoAnterior-Cara* y además se inicia un timer que maneja un método encargado de dibujar en el entorno gráfico lo que la cámara está capturando.

El método IniciarLEGO es manejado por otro timer, por esa razón el método recibe como parámetros un object *sender* y un EventArgs *e*, aunque en realidad no es necesario utilizar estas variables dentro del método. Dentro de él, se obtienen los estados en que los objetos *capturaCara* y *capturaVoz* se encuentran y se hace una llamada a otro método que revisa la correspondencia entre los estados con respecto a la Tabla 3 para finalmente indicarle al objeto *controlaLego* en qué dirección se moverá el prototipo de silla de ruedas.

El método GetEstadoSynch recibe como parámetros los estados actuales en que se encuentran los objetos *capturaCara* y *capturaVoz* adentro revisa si ambos estados corresponden a un movimiento coherente para *controlaLego* y finalmente regresa un valor entero mayor a cero si los estados indican un movimiento o cero sino.

El método CargarGrammarFromFile contiene lo necesario para lanzar una ventana de apertura de archivos, que permite abrir la gramática guardada en un archivo en formato grxml y luego asignarla al objeto *capturaVoz*.

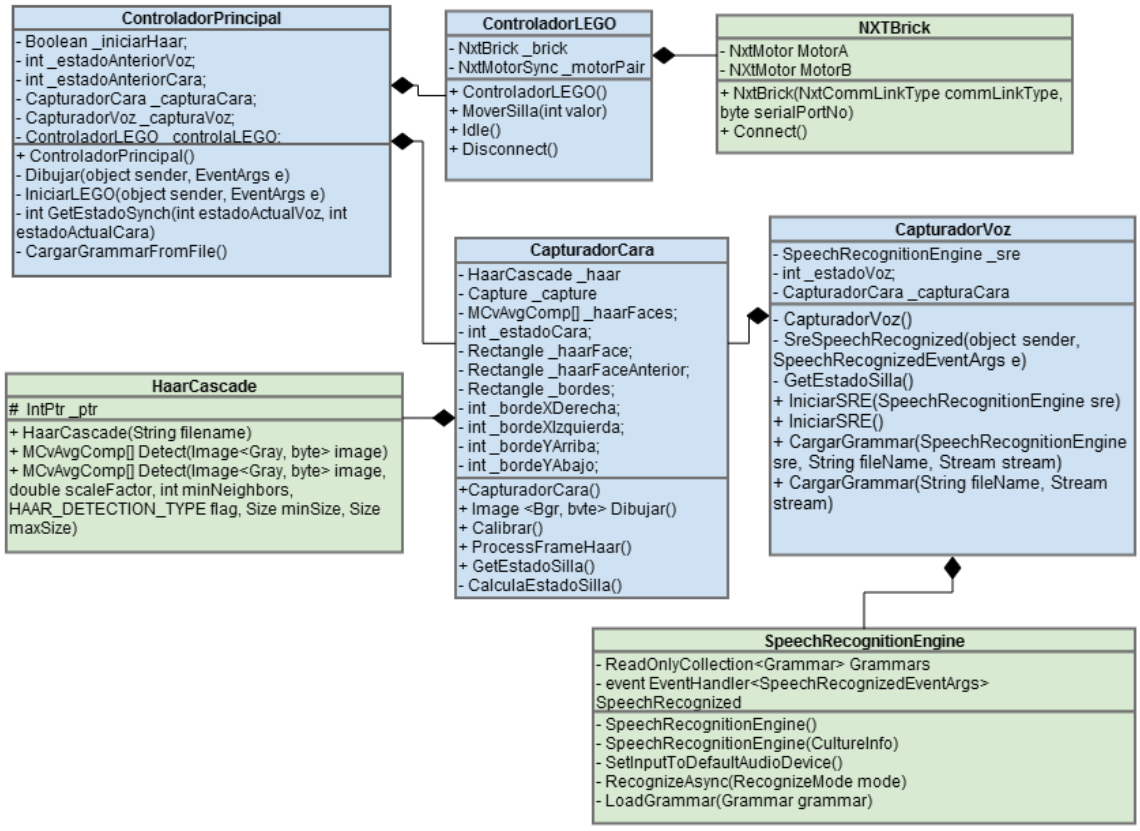


Diagrama 3. Diagrama de Clases del Sistema de Control del Prototipo de Silla de Ruedas

4.2.2. CLASE CONTROLADORLEGO

La clase ControladorLEGO utiliza las clases NXTBrick y NXTMotorSync, a través de ellas es posible la comunicación entre el software alojado en la computadora y el Bloque Inteligente NXT 9841 con el que se construyó el prototipo de silla de ruedas; esto significa que la clase ControladorLEGO fuciona como enlace entre el sistema de control y el prototipo.

En su constructor ControladorLEGO, se inicializa el Bloque Inteligente (NXTBrick brick) y a través de un parámetro se le indica el tipo de conexión que se va a utilizar para la comunicación, puede ser USB o Bluetooth, aunque para utilizar ésta última será necesario hacer la sincronización de los dispositivos (PC-Bloque Inteligente) antes de iniciar el sistema de control del prototipo. También se inicializan en él los dos motores conectados al Bloque Inteligente y un objeto NXTMotorSync *motorPair* que sirve para sincronizar los motores cuando éstos estos avanzan en la misma dirección a la misma velocidad, finalmente se hace un llamado al método Connect del NXTBrick para establecer la conexión.

El método MoverSilla recibe un valor que indica hacia donde se mueve el prototipo (puede ser obtenido del método que se describió en el apartado anterior: GetEstadoSynch).

El método Idle es llamado cuando el prototipo de silla de ruedas está en espera de una instrucción de movimiento y para desconectar al prototipo se llama al método Disconnect.

4.2.3. CLASE CAPTURADORCARA

La clase CapturadorCara contiene métodos que permiten iniciar el reconocimiento facial y determinar en el estado en que se encuentra la cara para mover el prototipo de silla de ruedas.

En su constructor se inicia la captura de la cámara y luego se inicializan las variables de instancia. En particular la variable HaarCascade _haar se inicializa con un archivo xml que contiene las características de la cara como valores numéricos y con la cual es posible el reconocimiento facial.

En el método Dibujar, se dibuja la captura obtenida de la cámara y encima de ella un rectángulo que indica la detección del rostro.

En el método Calibrar se hace un ajuste de los límites de movimiento de la cara detectada, es decir si el rostro sale de estos límites se le asignará un valor numérico a la variable de instancia _estadoCara en el método CalculaEstadoSilla.

ProcessFrameHaar hace uso de otro método, llamado Detect del objeto HaarCascade, éste encuentra regiones rectangulares en la imagen dada que probablemente contienen objetos para el que ha sido entrenado el HaarCascade o en otras palabras, que cumplen con las características que están en el archivo xml.

La función escanea la imagen varias veces a diferentes escalas, superponiendo regiones de la imagen y aplicando los clasificadores a las regiones. Es posible que aplique heurísticas para reducir el número de regiones analizadas, como CannyPruning, con la cual el detector se salta las regiones de la imagen en las que es menos probable que exista un rostro usando un detector de bordes. Los rectángulos candidatos son agrupados y la función regresa una secuencia de rectángulos.

4.2.4. CLASE CAPTURADORVOZ

La clase CapturadorVoz contiene los métodos necesarios para iniciar el motor de Speech Recognition, cargar un grammar (o una gramática) y capturar eventos de habla e interpretarlos para darles un estado numérico.

En el constructor CapturadorVoz se inicializan las variables int _estadoVoz, y CapturadorCara _capturaCara y luego el motor de Speech Recognition, SpeechRecognitionEngine _sre con el lenguaje Español (o RuntimeLanguage Español), además se configura como dispositivo de entrada de audio al que está por default y se asigna a un manejador de eventos el método SreSpeechRecognized.

Cada RuntimeLanguage para Speech Recognition incluye el modelo del lenguaje, el modelo acústico y otros datos necesarios para proveer un motor de Speech Recognition capaz de desempeñarse satisfactoriamente en un lenguaje en particular.

El método SreSpeechRecognized recibe como parámetros un object sender y un SpeechRecognizedEventArgs e, de éste se obtiene una cadena de texto que contiene lo que el motor de SpeechRecognition obtuvo del flujo de audio, ésta se compara con los comandos conocidas y se le da un valor numérico; debido a que al motor de Speech Recognition se le ha cargado una gramática, dentro del dominio de palabras que se recuperan del SpeechRecognizedEventArgs sólo estarán las que pertenecen a ella, como se muestra en la Imagen 14.

Al momento en que se ejecuta este evento, también se ejecuta el método ProcessFrameHaar del CapturadorCara, pues hasta este momento es necesario saber su estado para que en el ControladorPrincipal se hagan las gestiones necesarias con ControladorLEGO para mover el prototipo.

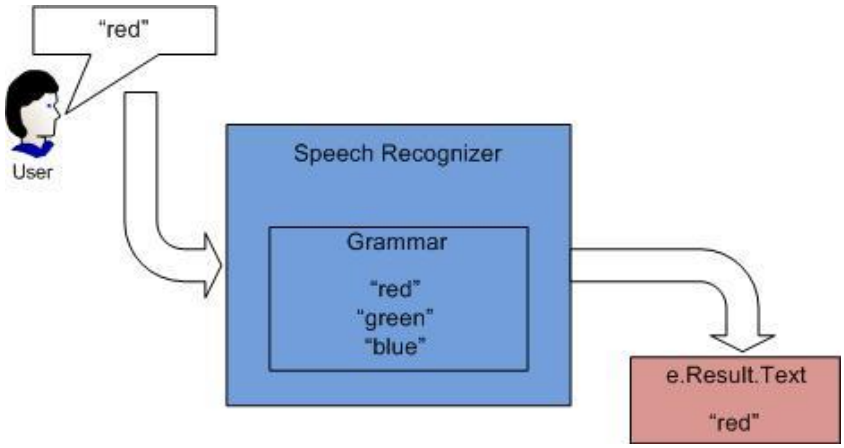


Imagen 14. Ejemplo de Funcionamiento de la Gramática con el Motor de Speech Recognition

El método GetEstadoSilla, regresará el estado numérico que se obtiene en el método anterior.

El método IniciarSRE está sobrecargado debido a que se puede iniciar la variable SpeechRecognitionEngine de instancia o una que se pase como parámetro.

Para inciar el motor de Speech Recognition es necesario hacer un llamado al método RecognizeAsync que ejecuta la operación de reconocimiento asíncrono, lo que significa que el hilo actual en el que se hace la llamada inicia la ejecución del método en otro hilo y luego la llamada regresa inmediatamente. Durante la llamada a este método, el motor de Speech Recognition puede levantar los siguientes eventos:

- SpeechDetected. Sucede cuando el reconocedor detecta una entrada que puede identificar como discurso (habla).
- SpeechHypothesized. Sucede cuando la entrada crea una concordancia ambigua con una de las gramáticas activas.
- SpeechRecognitionRejected o SpeechRecognized. Sucede cuando el reconocedor finaliza una operación de reconocimiento.
- RecognizeCompleted. Sucede cuando la operación del RecognizeAsync termina.

Un reconocimiento asíncrono puede fallar por dos razones, el habla no se detecta antes de que se terminen los tiempos de espera que están establecidos en las propiedades del reconocedor o si el motor detecta un flujo de audio pero no encuentra coincidencias en ninguna de las gramáticas cargadas.

Si a la función RecognizeAsync se le pasa como parámetro el modo Múltiple (RecognizeMode.Multiple) que es el caso actual, la diferencia es que existe la opción detener las operaciones de reconocimiento con una llamada al método RecognizeAsyncStop.

El método CargarGrammar también está sobrecargado, pues al igual que en el método IniciarSRE puede cargarse una gramática para la variable de instancia SpeechRecognitionEngine o para una que se pase como parámetro. Dentro del método se hace una llamada a la función LoadGrammar que pertenece al motor de SpeechRecognition.

El reconocedor lanzará una excepción si la gramática ya fue cargada o si ha fallado su carga en algún reconocedor. No es posible cargar el mismo grammar en varias instancias de SpeechRecognitionEngine. En lugar de eso debe crearse un grammar para cada instancia de SpeechRecognitionEngine.

Al momento de cargar la gramática, ésta se habilita por defecto, por ello si el reconocedor está ejecutándose y se desea deshabilitar, descargar, habilitar una gramática o cargar una nueva, el reconocedor debe pausarse y luego ejecutarse nuevamente.

CAPÍTULO V. RESULTADOS

Se realizó una interfaz como parte del prototipo de silla de ruedas, el botón *Cargar Grammar*, sirve para cargar cualquier gramática que se tenga guardada en un archivo. La separación entre los archivos con la gramática y la aplicación proporciona mayor flexibilidad en la modificación o evolución de las gramáticas, pueden liberarse nuevas versiones de gramáticas por parte del desarrollador y seguirá siendo posible que la aplicación las abra y cargue satisfactoriamente sin necesidad de cambios en el código.

Luego de que la gramática ha sido cargada, se inicia al motor de Speech Recognition con el botón *Iniciar Speech*. La calibración con el botón calibración puede realizarse en cualquier momento.

Con el botón *IniciarLEGO* se inicia la comunicación con el prototipo y se pueden empezar a ejecutar comandos combinados con voz y movimiento de la cabeza (a través de la detección facial).

Es posible realizar este proceso de forma más rápida presionando el botón *Automático* que permite cargar una gramática por default, iniciar el motor de Speech Recognition, hacer una calibración al momento en el que se presiona y conectar el LEGO MINDSTORMS NXT; con el objetivo de que personas que desconocen el funcionamiento del sistema y el orden de presionado de botones puedan probarlo.

Es importante mencionar que al igual que el robot esta interfaz es un prototipo, pues si se tuviera el sistema funcionando en un ambiente real no habría pantalla para que el usuario pudiera observar lo que va sucediendo y el acuse de cada evento en el cuadro de texto, en este momento se usa para mayor apreciación de las características y procesos de la misma.

El sistema cumple satisfactoriamente con los objetivos establecidos, al finalizar la codificación de los sub-módulos de control por Speech Recognition y por MoCap, se realizaron pruebas de funcionamiento y reconocimiento de los comandos. Luego se probó el sistema de forma integral.

El sistema es capaz de controlar satisfactoriamente el prototipo de silla de ruedas a través de las directivas establecidas (Tabla 3).

Capítulo V. Pruebas y Resultados

CAPÍTULO VI. CONCLUSIONES Y TRABAJO A FUTURO

Es posible desarrollar sistemas de control que sirvan como base tecnológica para proyectos de mayor envergadura, tal como el sistema de control por habla y movimiento para una silla de ruedas real manejada por una persona con lesión medular.

Se ha visto que al utilizarse algoritmos que funcionan con cámaras uno se enfrenta con un gran desafío ante la necesidad de ser usados en diversos entornos con diferentes características de iluminación y de ruido. Sin embargo, el algoritmo de reconocimiento facial obtuvo un buen desempeño ante diferentes situaciones de iluminación. Cabe mencionar que dichos subsistemas e incluso el sistema principal, podrían ser empleados no sólo para esta aplicación, sino también para otras que se relacionen con diversos problemas de locomoción.

Durante la fase de pruebas se observó que la coordinación para ejecutar ambos comandos a la vez (habla y movimiento) fue adquirida con relativa facilidad para el sujeto de estudio. Aunque este resultado es prometedor, es necesario hacer pruebas con más sujetos y analizar si efectivamente la metodología a modo de confirmación de los comandos es viable.

Se planea que el sistema pueda ser probado por personas con lesión medular a través de videos pregrabados, como se hizo con anterioridad a través de la red social www.lesionmedular.org en la cual, la gente se mostró dispuesta a participar.

Aunque este sistema está terminado de acuerdo a los alcances y limitaciones establecidos, aún es necesario hacer muchas mejoras y agregar más subsistemas de apoyo para tolerancia a fallos.

Como se mencionó, la pantalla no formará parte de la implementación en tamaño real, por lo que se está realizando un módulo que contiene un sintetizador para que el sistema comunique al usuario lo que hará, solicite confirmación de comandos críticos o solicite completar comandos.

Capítulo VI. Conclusiones

REFERENCIAS

[1] Instituto Nacional de Estadística y Geografía, «INEGI Censo de Población y Vivienda 2010: Tabulados del Cuestionario Ampliado,» 11 Mayo 2011. [En línea]. Available: http://www3.inegi.org.mx/sistemas/TabuladosBasicos/LeerArchivo.aspx?ct=27502&c=27303&s=est&f=1. [Último acceso: 15 Enero 2013].

[2] J. López Chicharro y L. M. López Mojares, Fisiología Clínica del Ejercicio, España: Editorial Médica Panamericana, 2008.

[3] R. Velázquez Guerrero, Dinámica y Control de Sillas de Ruedas Robóticas, México D.F.: UAM-UdG, 2010, pp. 151-164.

[4] L. F. Salcedo Hernández, C. R. Torres San Miguel, G. Urriolagoitia Sosa y B. Romero Ángeles, «Rediseño para la Optimización de una Silla de Ruedas de Dos Posiciones,» de *10° Congreso Nacional de Mecatrónica*, Puerto Vallarta, Jalisco, México., 2011.

[5] M. Mazo, F. J. Rodríguez, J. L. Lázaro, J. Ureña, J. C. García, E. Santiso, P. Revenga y G. J. Jesús, «Wheelchair for Physically Disabled People with Voice, Ultrasonic and Infrared Sensor Control,» de *Autonomous Robots*, Madrid, España, Kluwer Academic Publishers, Boston, 1995, pp. 203-224.

[6] J. Alcubierre, J. Minguez, L. Montesano, L. Montano, O. Saz y E. Lleida, «Silla de Ruedas Inteligente Controlada por Voz,» Universidad de Zaragoza, España, Zaragoza, España, 2005.

[7] K. Arai y R. Mardiyanto, «Autonomous Control of Eye Based ElectrWheel Chair with Obstacle Avoidance and Shortest Path Findings Based on Dijkstra Algorithm,» *International Journal of Advanced Computer Science and Applications,* vol. 2, nº 12, pp. 19-25, 2011.

[8] K. Arai y R. Mardiyanto, «Eyes Based Electric Wheel Chair Control System,» *International Journal of Advanced Computer Science and Applications,* vol. 2, nº 12, pp. 98-105, 2011.

[9] L. Chern-Sheng, H. Chien-Wa, C. Wen-Chen, C.-C. Chiu y Y. Mau-Shiun, «Powered Wheelchair Controlled by Eye-Tracking System,» *Optica Applicata,* vol. 36, nº 2-3, pp. 401-412, 2006.

[10] Y. Matsumoto, T. Ino y T. Ogasawara, «Development of Intelligent Wheelchair System with Face and Gaze Based Interface,» de *10th IEEE International Workshop on Robot-Human Interactive Communication*, París, Francia., 2001.

[11] Q. X. Nguyen y S. Jo, «Electric Wheelchair Control Using Head Pose Free Eye-Gaze Tracker,» *Electronics Letters,* vol. 48, nº 13, pp. 750-752, 2012.

[12] D. Purwanto, R. Mardiyanto y K. Arai, «Electric Wheelchair Control with Gaze Direction and Eye Blinking,» de *14th International Symposium on Artificial Life and Robotics*, Oita, Japón, 2009.

[13] C. D. Rigano, «Sistema de Control de una Silla de Ruedas Motorizada para Personas Cuadrípléjicas.,» Universidad Tecnológica de Argentina, Argentina, 2000.

[14] I. Iturrate, J. M. Antelis, A. Kübler y J. Minguez, «Dispositivos Robóticos de Rehabilitación basados en Interfaces Cerebro-Ordenador: Silla de Ruedas y Robot para Teleoperación,» de *Proceedings of the III Congreso Internacional sobre Domótica, Robótica y Teleasistencia para todos.*, Barcelona, España, 2009.

[15] L. Jzau-Sheng y Y. Win-Ching, «Wireless Brain-Computer Interface for Electric Wheelchairs with EEG and Eye-Blinking Signals,» *International Journal of Innovative Computing, Information and Control,* vol. 8, nº 9, pp. 6011-6024, 2012.

[16] M. Kitawa y B. Windsor, MoCap for Artists: Workflow and Techniques for Motion Capture, USA: Elsevier/Focal Press, 2008.

[17] STT Engineering and Systems, «SST-Systems,» [En línea]. Available: http://www.stt-systems.com/en/. [Último acceso: 13 03 2013].

[18] XSENS, «XSENS Motion Capture,» 2009. [En línea]. Available: www.xsens.com/Motion-Capture. [Último acceso: 16 03 2013].

[19] F. Besse, N. Heynen, B. Jamet y P. Lacour, «Cheap Motion Capture,» 2007. [En línea]. Available: http://phraides.free.fr/team.html. [Último acceso: 17 03 2013].

[20] E. Muybridge, Animals in Motion, USA: Dover Pictorial Archive Series, 1899.

[21] E. Muybridge, The Human Figure in Motion, USA: Dover Publications, 1901.

[22] Universidad Internacional de Andalucía, «Sistemas HMC ópticos basados en marcadores pasivos,» OpenCurseWare-UNIA, 11 2012. [En línea]. Available: http://ocw.unia.es/ciencias-tecnologicas/tecnologia-del-ocio/materiales-basicos-folder/html/B1_UD05/sistemas_hmc_pticos_basados_en_marcadores_pasivos.html/skinless_view. [Último acceso: 16 06 2013].

[23] MOVA, «MOVA CONTOUR,» 2004. [En línea]. Available: http://www.mova.com/flash/. [Último acceso: 18 Marzo 2013].

[24] Polhemus, «Polhemus Innovation in Motion,» [En línea]. Available: http://www.polhemus.com/?page=Motion_G4. [Último acceso: 05 07 2013].

[25] P. Viola y M. Jones, «Rapid Object Detection Using a Boosted Cascade of Simple Features,» de *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kawai, Hawaii, USA, IEEE Computer Society, 2001, pp. 511-518.

[26] Itseez, «OpenCV,» Itseez, 2013. [En línea]. Available: http://www.opencv.org. [Último acceso: 30 Julio 2013].

[27] EmguCV, «EmguCV,» EmguCV, 16 Septiembre 2012. [En línea]. Available: http://www.emgu.com. [Último acceso: 30 Julio 2013].

[28] S. Furui, Digital Speech Processing, Synthesis and Recognition, New York, USA: Marcel Dekker, Inc., 2001.

[29] H. G. Tillmann, «Speech Recognition,» 2004. [En línea]. Available: http://www.speech-recognition.de/. [Último acceso: 18 03 2013].

[30] Microsoft, «MSDN Microsoft Speech Platform,» Microsoft, 2013. [En línea]. Available: http://msdn.microsoft.com/en-us/library/hh361572(v=office.14).aspx. [Último acceso: 05 04 2013].

[31] N. K. Handest, «MindSqualls,» Handest, Niels K, 2011. [En línea]. Available: http://www.mindsquals.net. [Último acceso: 30 Julio 2013].

[32] T. Richards, «Health Informatics International, Inc.,» RelayHealth, 17 09 2012. [En línea]. Available: http://actilean.healthinformatics.net/docs/english/aha/spineinj.crs.asp. [Último acceso: 31 04 2013].

[33] W. Young, «Spinal Cord Injury Levels & Classification,» 2013. [En línea]. Available: http://www.sci-info-pages.com/levels.html. [Último acceso: 25 05 2013].

[34] American Spinal Injury Association, «Asia SpinalInjury,» 2013. [En línea]. Available: http://www.asia-spinalinjury.org/. [Último acceso: 15 04 2013].

[35] LesiónMedular, «Lesión Medular,» 2012. [En línea]. Available: http://www.lesionmedular.org. [Último acceso: 27 05 2013].

[36] Organización Mundial de la Salud, «Informe Mundial Sobre la Discapacidad,» Ediciones de la OMS, Malta, 2011.

[37] LEGO, «LEGO MINDSTORMS,» LEGO Group, 2012. [En línea]. Available: http://mindstorms.lego.com/en-us/default.aspx. [Último acceso: 25 05 2013].

[38] R. Lienhart y J. Maydt, «An Extended Set of Haar-Like Features for Rapid Object Detection,» de *Proceedings 2002 International Conference on Image Processing*, Rochester, New York, USA, IEEE, 2002, pp. 900-903.

APÉNDICE A

OpenCV proporciona las funciones necesarias para la detección facial pero también nos brinda programas que ellos han usado para entrenar a los clasificadores para el algoritmo de Viola-Jones. Así que es posible la creación de objetos clasificadores propios usando dichos programas.

Antes de comenzar se debe contar con una base de datos de entrenamiento de imágenes positivas del objeto de interés e imágenes negativas (de preferencia en formato PNG).

Se pueden crear ejemplos o muestras positivas para el entrenamiento y para pruebas con la utilería *createsamples*. Este es un proceso largo que puede consultarse en la documentación de OpenCV.

ENTRENAMIENTO

HAAR TRAINING

El siguiente paso, después de crear ejemplos o muestras es entrenar al clasificador, y esto se hace con la utilería *haartraining*.

Su uso por línea de comandos es el siguiente:

```
./haartraining
  -data <dir_name>
  -vec <vec_file_name>
  -bg <background_file_name>
  [-npos <number_of_positive_samples = 2000>]
  [-nneg <number_of_negative_samples = 2000>]
  [-nstages <number_of_stages = 14>]
  [-nsplits <number_of_splits = 1>]
  [-mem <memory_in_MB = 200>]
  [-sym (default)] [-nonsym]
  [-minhitrate <min_hit_rate = 0.995000>]
  [-maxfalsealarm <max_false_alarm_rate = 0.500000>]
  [-weighttrimming <weight_trimming = 0.950000>]
  [-eqw]
  [-mode <BASIC (default) | CORE | ALL>]
  [-w <sample_width = 24>]
  [-h <sample_height = 24>]
```

```
[-bt <DAB | RAB | LB | GAB (default)>]
[-err <misclass (default) | gini | entropy>]
[-maxtreesplits <max_number_of_splits_in_tree_cascade = 0>]
[-minpos <min_number_of_positive_samples_per_cluster = 500>]
```

Y los argumentos representan:

- Data <dir_name> es el directorio donde el clasificador entrenado es guardado
- Vec <vec_file_name> es el nombre del archivo con las muestras positivas (creado por *trainingsamples* y otros medios)
- Bg <background_file_name> es el nombre del archivo con las muestras negativas.
- Npos <number_of_positive_samples> es el número de muestras positivas usadas en el entrenamiento en casa etapa de clasificación. Valores razonables son npos = 7000
- Nneg <number_of_negative_samples> es el número de muestras negativas usadas en el entrenamiento en casa etapa de clasificación. Valores razonables son npos = 3000.
- Nstages <number_of_stages> es el número de etapas de entrenamiento
- Nsplits <number_of_splits> determina el clasificador débil usado en cada etapa, puede ser 1 o 2.
- Mem <memory_in_MB> es la memoria disponible en MB para los cálculos, mientras más memoria se tiene más rápido será el proceso de entrenamiento.
- Sym (default), nonsym específica si la clase en entrenamiento tiene simetría vertical o no. La simetría vertical acelera el proceso de entrenamiento. Por ejemplo, las caras frontales muestran simetría vertical.
- Minhitrate <min_hit_rate> Tasa de éxito deseada para cada etapa de clasificación.
- Maxfalsealarm <max_false_alarm_rate> Máxima tasa de falsa alarma deseada para cada etapa del clasificador.
- Weighttrimming <weight_trimming> Especifíca si se hace y qué tan grande será el corte de pesos usados. Una elección deseable es 0.90.
- Eqw habilita pesos iguales para positivos y negativos.
- Mode <BASIC (default) | CORE | ALL> selecciona el tipo del conjunto de características haar usado en el entrenamiento. Basic usa sólo características verticales, mientras que ALL usa el conjunto completo de características desde verticales hasta rotadas 45°.
- W <simple_width> y h<simple_height> son el tamaño de las muestras de entrenamiento en pixeles, deben ser los mismos valores usados durante la creación de muestras de entrenamiento.

- Bt <DAB | RAB | LB | GAB (default)> sirve para seleccionar el clasificador, DAB es Discrete Ada Boost, RAB es Real Ada Boost, LB es Logit Boost, Gab es Gentle AdaBoost.
- Err <misclass (default) | gini | entropy> es el tipo de error disponible sólo cuando se utiliza DAB.
- Maxtreesplits <max_number_of_splits_in_tree_cascade = 0> construye árboles de clasificación en lugar de cascadas. Teóricamente, una cascada debería ser suficiente pero empíricamente una estructura de árbol podría ayudar en algo.

La herramienta *haartraining* arroja una salida como la que se muestra en la Tabla 4, donde N es el número de la iteración, %SMP es el porcentaje de las muestras originales restantes, F indica que la característica se ha volteado y está relacionado con el parámetro –sym, ST.THR es el umbral por etapa, HR es el hit rate, FA es el false alarm rate y EXP.ERR es el error esperado.

```
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.857040| 1.000000| 1.000000| 0.082075|
+----+----+-+---------+---------+---------+---------+
|   2|100%|+|-1.702127| 1.000000| 1.000000| 0.102168|
+----+----+-+---------+---------+---------+---------+
```

Tabla 4. Salida del *haartraining*

La utilidad *haartraining* crea un archivo <dir_name>.xml cuando el entrenamiento ha finalizado completamente.

APÉNDICE B

Se toma en cuenta la siguiente gramática G:

$G = (V^*, S, v_0, P) donde$:

$N = \{< Verb\_Direccion >, < id\_Verb >, < id\_Direccion >, < item >\}$

$V = S \cup N$

$v_o = < Verb\_Direccion >$

$S = \{muévete, retrocede, gira, detener, a, hacia, ahora, adelante, la\ izquierda, la\ derecha\}$

Las reglas de producción P son:

$< Verb\_Direccion > := < id\_Verb > < item > < id\_Direccion >$

$< id\_Verb > := muévete | retrocede | gira | detener$

$< id\_Direccion > := adelante | la\ izquierda | la\ derecha$ | ahora

$< item > := hacia | | a$

Para escribir la gramática que se utiliza por el motor de Speech Recognition se siguió el Speech Recognition Grammar Specification Version 1.0 de la W3C []. En esa especificación se definen varias reglas, Verb_Direccion es la más importante, pues en ella se define la estructura del texto que se reconocerá a través del SpeechRecognitionEngine. La regla indica que lo primero que se debe recibir es una palabra bajo otra regla llamada id_Verb, después un conector y finalmente una palabra bajo otra regla llamada id_Dirección. Aunque existen combinaciones como: gira ahora, éstas no son tomadas en cuenta.

GRAMÁTICA

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar version="1.0" xml:lang="es-ES" mode="voice"
root="Verb_Direccion" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar">

      <rule id="Verb_Direccion" scope="public">

            <ruleref uri="#id_Verb"/>
            <tag> $.Verb = $$; </tag>

            <one-of>
                  <item repeat="0-1"> hacia </item>
                  <item repeat="0-1"> a </item>
                  <item repeat="0-1">  </item>
            <one-of>

            <ruleref uri="#id_Direccion"/>
            <tag> $.Direccion = $$; </tag>

      </rule>

      <rule id="id_Verb" scope="private">
            <one-of>
                  <item> muévete </item>
                  <item> retrocede </item>
                  <item> gira </item>
                  <item> detener </item>
            </one-of>
      </rule>

      <rule id="id_Direccion" scope="private">
            <one-of>
                  <item> ahora </item>
                  <item> adelante </item>
                  <item> atrás </item>
                  <item> la izquierda </item>
                  <item> la derecha </item>
            </one-of>
      </rule>


</grammar>
```