



UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

Patrocinante: Prof. Sebastián E. Godoy , Ph.D.

Comisión: Prof. Jorge E. Pezoa, Ph.D., Prof. Pablo A. Coelho, M.Sc.

INFORME PROYECTO DE HABILITACIÓN PROFESIONAL.

Detección de rostros en tiempo real usando plataforma Raspberry Pi.

Felipe Alfonso Seguel Mora.

Concepción, 24 de septiembre de 2015

Índice general.

1. Introducción.	1
2. Revisión bibliográfica.	3
2.1. Memorias de título.	3
2.2. Tesis de postgrado.	4
2.3. Publicaciones científicas.	4
2.4. Libros.	6
2.5. Otros.	7
2.6. Discusión de la Bibliografía	8
3. Definición del problema	9
3.1. Problema a resolver	9
3.2. Objetivos.	10
3.2.1. Objetivo general.	10
3.2.2. Objetivos específicos.	10
3.3. Alcances y limitaciones.	11
3.4. Metodología de trabajo.	12
4. Avance de la memoria de título.	13
4.1. Introducción.	13
4.2. Teoría, modelos y métodos.	13
4.2.1. <i>Eigenfaces</i>	13
4.2.2. Distancia Euclíadiana	16
4.2.3. Introducción a Support vector machine	17
4.2.4. Evaluación de algoritmos de reconocimiento facial propuestos por Inostroza	20
4.3. Simulaciones preliminares.	24
4.3.1. Simulación distancia Euclíadiana en Matlab	24
4.3.2. Simulación Eigenface en Matlab	25

4.4. Raspberry Pi	27
4.4.1. Hardware Raspberry Pi	27
4.4.2. Pi Cam	28
4.4.3. Instalación de OpenCV en Raspberry Pi	29
4.4.4. Problemas con la instalación OpenCV	31
4.4.5. Implementación de Eigenface con OpenCV en la Raspberry Pi	32
4.4.6. Segmentación de Rostros	34
5. Conclusiones y trabajo futuro.	36
5.1. Conclusiones.	36
5.2. Trabajo futuro.	36
Bibliografía	37
A. Apéndices	40
A.1. Raspberry Pi y Cámara	40

Índice de figuras

1.	Hiperplano definido por la ecuación $w \cdot x + b = 0$ donde w es la normal y b es el offset, [21].	17
2.	SVM linealmente separable con sus planos H_1 y H_2 , [21].	18
3.	Plantillas usadas para la detección de rostros, [20].	20
4.	Errores al implementar distintos tamaños de plantillas o máscaras, [20].	21
5.	Base de datos YaleFace con 3 sujeto en 3 distintas situaciones.	24
6.	a) Imagen de Prueba b)Sujeto detectado con mínima distancia Euclidiana.	24
7.	Cálculo de rostro promedio.	25
8.	Resta del rostro promedio a cada una de las imágenes.	25
9.	Eigenfaces.	26
10.	a) Imagen de Prueba b)Sujeto detectado con algoritmo <i>Eigenface</i>	26
11.	Hardware de Raspberry Pi utilizada para realizar este proyecto	27
12.	Todos los periféricos que se pueden conectar a la Raspberry Pi Modelo B, de dos puertos USB,[22].	28
13.	a)Raspberry Pi con carcasa y Pi Cam montada b)c)d)Imágenes de cámara Raspberry Pi que se utiliza en el proyecto, [23]	29
14.	Update solo chequea repositorios.	29
15.	Upgrade ejecuta la actualizacion efectiva del sistema.	29
16.	Configuración OpenCV.	31
17.	Creación y construcción del directorio con el que se trabaja.	33
18.	a) Ejecución del programa luego de construir y compilar b)Devolución de un video a 30 FPS capturado en Laboratorio de Redes de Datos.	33
19.	Resultado esperado al término del tutorial	34
20.	Segmentación de rostros utilizando los ejemplos de la librería OpenCV	35
21.	a)DataSheet Raspberry Pi Cam b) c)Esquema reducido de Raspberry Pi modelo B+ v1.2, [24]	40

1. Introducción.

En la actualidad, la detección de rostros sigue siendo un tema de popularidad en muchas aplicaciones digitales. El rostro contiene bastante información que puede ser procesada a través de diversos algoritmos existentes hoy en día. Nuestro esfuerzo se basará en conocer técnicas para la detección de rostros y en base a eso, comparar y, posiblemente, desarrollar nuevas técnicas de detección de rostros.

Una de las aplicaciones más comunes es la seguridad o vigilancia, siendo una de las ramas más importantes en donde la tecnología ha ganado terreno, en particular en buscar individuos perseguidos por la justicia^[1]. Otro campo de aplicación es la videoconferencia, que tiene que ver con localizar la imagen del individuo en una secuencia de *webcam* para poder hacer seguimiento, etc. Otra aplicación, es el control de acceso con el fin de evitar la usurpación de identidad [3]. Es por esto que este tipo de tecnología proporciona la base para una amplia gama de soluciones de la identificación y alta seguridad. Este tipo de tecnología se vuelve cada vez más precisa y menos costosa a medida que pasan los años [4].

La idea es hacer un sistema de detección de rostro, integrando tecnologías de bajo costo y *open source*, en este caso Raspberry Pi, en el cual existe mucho interés en trabajar para dar soluciones embebidas a diversos problemas en procesamiento digital de imágenes. Las ventajas son claras, el costo de este tipo de sistema y el tamaño (aproximado a una tarjeta de crédito) permite proyectar esta aplicación a otras soluciones embebidas que necesita la industria.

El proyecto es interesante no porque se vaya a proponer un nuevo método de reconocimiento de rostros, si no que se va a ocupar una tarjeta Raspberry Pi disponible en el comercio, abarcando varias disciplinas como el procesamiento de imágenes, álgebra lineal, estadística, programación, reconocimiento de patrones, entre otras [5].

Los objetivos de este proyecto son básicamente tomar distintas ideas e integrarlas en una sola, para solucionar un problema en particular. Primeramente se estudiarán algoritmos de selección de análisis de componentes principal (PCA), para posteriormente pasar al concepto de *Eigenface* como un sistema base para la descripción de rostros, finalizado esto, se identificará un algoritmo moderno y será comparado con *Eigenface/PCA* para determinar un desempeño real. Luego se implementará en algún lenguaje de programación el algoritmo seleccionado, para finalmente realizar un método de segmentación, con tal que separe el rostro del resto de la imagen, siendo la mayor complejidad del proyecto.

2. Revisión bibliográfica.

2.1. Memorias de título.

- Edwin Arturo Vega Aquino, “Detección y seguimiento de rostros,” Proyecto fin de carrera, Grado en Ingeniería de Telecomunicaciones, Depto. de Ingeniería Telemática, Universidad Carlos III de Madrid, España, 2011 [1].

En esta memoria, Vega presenta un desarrollo de nuevas técnicas para la implementación en la detección y seguimiento de rostros y objetos. En particular se hará uso de las discusiones de las principales técnicas para la detección de rostros, las ventajas y desventajas, así como también sus diferencias, de cada una de ellas, tomando mayor interés *Eigenface* en esta parte del proyecto.

El resultado de esta memoria es importante y será utilizado en la propuesta de implementar un nuevo algoritmo usando librerías de *OpenCV*, en caso de escogerlas. Este proyecto también servirá para la parte de segmentación del rostro, ya que se propone un método de seguimiento de rostros en vídeos de distintos tamaños.

- Enrique Patricio Inostroza Cáceres “Reconocimiento de rostros evaluación de la aplicación de tres algoritmos al reconocimiento facial”, informe de memoria de título de Ingeniería Civil Electrónica, Depto. Ing. Eléctrica, Universidad de Concepción, 2001. [2]

En esta memoria, Inostroza presenta la evaluación de tres algoritmos de clasificación facial, tanto a lo referente a porcentajes de reconocimiento, como a cantidad de información requerida y tiempo empleado.

Los resultados son interesantes desde el punto de vista que se pretende evaluar un conjunto de técnicas de reconocimiento facial totalmente automáticas, que sean aplicables en el desarrollo de un sistema de identificación personal rápida que trabaje con una base de datos. El trabajo de Inostroza se limita a realizar pruebas de reconocimiento, es decir clasificación de patrones de entrada con base en los almacenados, pero no se contempla la implementación de algoritmos de detección de rostros ni la verificación de identidad.

Se comparan tres algoritmos, mapas auto-organizativos, bases características no ortogonales y reconocimiento de patrones mediante comparación con plantilla. Este último se ocupará para comparar con una técnica clásica llamada Eigenface, debido a que posee mejores porcentajes de reconocimiento, según Inostroza, considerando el costo computacional y de procesamiento.

2.2. Tesis de postgrado.

- Marcelo J. Armengot Iborra. “Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras,” Disertación Ph.D., Depto. d’Informàtica, Universitat de València , España, 2006 [3].

Armengot realiza un recorrido en las técnicas más importantes de la literatura revisando el reconocimiento, la detección y ajuste de una cara en una foto. Además, desarrolla, implementa, prueba y compara uno de los método más reciente de *vector común y vectores comunes discriminantes*(DCV) haciendo pruebas de clasificación de 40 clases. El trabajado es muy completo ya que se compara los nuevos métodos con *Eigenface* y *Fisherface* en situaciones en donde se agrega ruido. El problema es que en este trabajo no se busca optimizar los algoritmos ni códigos, algo importante a la hora de hablar de limitaciones de hardware si se piensa implementar en la Raspberry Pi.

2.3. Publicaciones científicas.

- M. Turk and A. Pentland “ Eigenfaces for recognition.J. Cogn. Neurosci,” **3** (1991) 72–86 [8]

En el presente paper, Turk y Pentland buscan implementar un sistema eficiente, sencillo y preciso de reconocimiento facial en un entorno restringido. La clasificación se lleva a cabo usando una combinación lineal de rasgos característicos (*Eigenfaces*). El análisis reduce la dimensionalidad del conjunto de entrenamiento, dejando solo aquellas características que son críticas para el reconocimiento facial. La importancia de este trabajo son los

resultados ya que en los experimentos se llevaron a cabo con distintas iluminaciones, escala y orientación. La base de datos experimental fue de 2.500 imágenes de cara. El problema es que el método no cubre el efecto zoom, o la variación en el tamaño de la cara, pues se producen errores dificultando el reconocimiento, además no contempla si el fondo de la imagen cambia.

- Md. Kawser Jahan Raihan *et al.*. “Raspberry Pi Image Processing Based Economical Automated Toll System,” Global Journal of Researches in Engineering Electrical and Electronics Engineering **13** (2013) 34-41 [9].

Este trabajo se vincula de forma indirecta con el proyecto, debido a que se da una solución al problema de detectar la matrícula de los vehículos en los peajes de pago automático, usando como anfitrión a la Raspberry Pi. La gracia del proyecto es que también es un problema de procesamiento digital de imágenes en el cual se conecta una cámara WiFi al módulo de Raspberry Pi. Este artículo nos sirve para analizar los circuitos que se fabrican y dar una idea que cuanto se pueden reducir los costos de un problema en específico. Lo interesante también es que se discute sobre la memoria limitada que hace difícil almacenar datos.

- Paul Viola and Michael J. Jones “ Robust Real-Time Face Detection,” International Journal of Computer Vision **57(2)**, (2004) 137–154 [10]

Se aborda el tema de como se es capaz de procesar imágenes extremadamente rápido mientras se alcance altos niveles de detección. Los resultados que se obtienen nos dicen que el sistema de detección de caras es aproximadamente quince veces más rápido que otros enfoques. La importancia de este trabajo es que reúne nuevos algoritmos y percepciones que son bastante genéricas usando una técnica agresiva y eficaz el cual es muy útil cuando se tienen limitaciones de hardware.

2.4. Libros.

- Alan C. Bovik, “Unconstrained Face Recognition from a Single Image”, en *The Essential Guide to Image Processing*, Academic Press, 2009 , Capítulo 24, pp. 667-713. [4].

Este libro es uno de los textos de referencia a la hora de hablar de procesamiento digital de imágenes. Abarca todos los aspectos necesarios para el aprendizaje del tema, orientándolo a usuarios novatos y avanzados en el tema. El texto se encuentra dividido en 28 capítulos de los cuales sólo interesa el capítulo 24.

Cap.24 - Se explica como el reconocimiento facial se centra principalmente en verificación, identificación y por último la lista de observación. El capítulo se centra en la tarea de identificación. Luego prueba el protocolo de FERET [6]. Finalmente se habla de 3 enfoques sobre patrones: el primer patrón es donde se explica el método de subespacios (PCA y LDA), en segundo lugar el patrón visual donde se habla de problemas fotométricos (iluminación) y geométricos (propias de la cámara), explicando el modelo de reflectancia Lambertiana. Por último el patrón facial donde se estudia temas de expresión y deformación, envejecimiento, superficie de la cara, auto similitud, maquillaje y cosmética, enfocándose en el reconocimiento en baja iluminación, variaciones de expresión y envejecimiento. El libro es de gran importancia ya que se hablan problemas del reconocimiento de rostros generales y genera acotaciones en las líneas de investigación de este proyecto.

- Jian Guo Liu, Philippa Mason, “Image Classification”, en *Essential Image Processing and GIS for Remote Sensing*, John Wiley & Sons, 2013, Capítulo 8, pp. 91-102. [7]

El libro está claramente dividido en tres partes, la primera es la introducción de técnicas de procesamiento de imágenes esenciales para la teledetección. La segunda parte analiza SIG y comienza con una visión general de los conceptos, las estructuras y los mecanismos por los que opera el SIG. Por último, la tercera parte presenta aplicaciones de la teledetección. La parte que interesa del libro es el capítulo 8 de clasificación de imágenes.

Cap.8 Estudian métodos de clasificación de imágenes más utilizados, a través de métodos de clasificadores esencialmente estadísticos. Se torna una discusión en torno a la precisión

del clasificador dependiendo del tipo de usuario, esto quiere decir el usuario es de alto o escaso conocimiento de los datos y también se discute de las limitaciones de cada método. El libro pasa a ser un complemento ya que aporta poco al aprendizaje relacionado al tema.

2.5. Otros.

- Josep Arnal García, Sergio Cuenca Asensi, Higinio Mora Mora, Francisco Moreno Seco (2010). *Actas del 1^{er} Workshop en Tecnologías de la Informática* [Online]. Available url: http://www.iuui.ua.es/master_ti/actas2010.pdf File: actas2010.pdf, [11].

En este documento toma interés solamente el capítulo 9 de aplicación de redes neuronales al reconocimiento de caras mediante PCA en color. Se pretende simular un dispositivo doméstico que detecte la posición en el hogar de una persona. El documento comienza explicando en breve las principales técnicas utilizadas haciendo una diferenciación entre los métodos holísticos y los métodos basados en características locales. Partiendo por la base de datos particular que usan, en la cual agregan imágenes de objetos, determinan un umbral para detectar si es o no una cara la imagen a reconocer. Además trabajan y calculan el parecido entre caras trabajando con distancias Euclídea, Mahalanobis y Pearson comprobando la efectividad de cada una de ellas direcciones similares a la línea de investigación que sigue este proyecto.

- Víctor Garcés, Pablo Gleisner. “*Sistemas de visión en dispositivos embebidos sobre linux*”, Univ. de Concepción, Concepción, Chile, 2013 [12].

Este documento se relaciona de forma indirecta, pero es un interesante proyecto que busca reconocer personas a través de una Raspberry Pi. Sirve como guía y punto de inicio en esta etapa del proyecto de habilitación profesional, en donde existe mucha confusión. Los autores se basan en un tutorial disponible en <https://thinkrpi.wordpress.com>, el cual reconoce a personas usando el algoritmo de Eigenface. Una desventaja del proyecto es que se utilizó una WebCam USB la cual disminuye la velocidad de procesamiento. Los códigos se modifican completamente si se quiere usar el módulo *Pi cam*.

2.6. Discusión de la Bibliografía

El tema no es nuevo y siempre ha sido un área de investigación de gran interés desde los años 90, lo que ha provocado un crecimiento en la literatura, en especial en la última década de parte de distintas disciplinas [4]. Hasta el momento existe un algoritmo que es el “padre” y el más estudiado, Eigenfaces, ya que a partir de este nace muchas perceptivas del problema de reconocimiento de rostros. A partir de esto podemos decir que hay proyectos que se acercan a lo que pretende llegar esta memoria de título como por ejemplo, lo propuesto por Paul Viola and Michael J. Jones que pretenden hacer un sistema robusto en tiempo real de detección de rostros (sección 2.3) acotando los tiempos de detección y pensando en hardware limitado. Sin lugar de dudas existe mucha información para el estudio del proyecto y variadas perceptivas al problema de reconocimiento de rostros.

3. Definición del problema

Hoy en día las aplicaciones de reconocimiento de rostro son muy amplias, y quedan sujetas a nuevas técnicas y algoritmos con mayor precisión que se mejoran a través de los años. Otro punto importante, es que también en el ámbito de los avances tecnológicos, estos aumentan a un ritmo acelerado, por ejemplo, la densidad de transistores en un procesador aumenta al orden de 30 % por año, la memoria DRAM aumenta en el orden de 60 % por año según Medina, [13]. La vigilancia y todos sus derivados pasaron a ser una razón importante para investigaciones y avances respecto al tema, después de los actos de terrorismo vividos en el mundo, en especial el ataque a las torres gemelas en 2001.

Se persigue en el proyecto dar soluciones personalizadas a problemas específicos mediante la programación de tarjetas comerciales, en específico la tarjeta Raspberry Pi, que con un costo muy accesible, busca dar soluciones de alta eficiencia y precisión. El trabajo a resolver requiere la implementación en hardware, del algoritmo seleccionado, lo que no es trivial, pues hay que adaptar el algoritmo a las limitaciones de la tarjeta.

Como primer alcance, este trabajo se limita a usar la base de datos existente llamada *yalefaces* que se puede encontrar en la página web *UCSD computer vision*, [14]. Como posible trabajo de la memoria de título queda diseñar, implementar y evaluar un método de adquisición de base de datos.

3.1. Problema a resolver

En esta memoria de título se trabajará en la comparación del desempeño en clasificar rostros de los algoritmos basados en mínima distancia Euclidiana, Eigenface y el algoritmo de Support Vector Machine (SVM). Esta comparación se realizará mediante su probabilidad de detección y su probabilidad de falsa alarma, implementados en hardware embebido sobre Linux.

En la segunda parte se cambiarán las condiciones como la iluminación, la perceptiva, rotación del rostro, etc., con tal de determinar el algoritmo más robusto que pueda funcionar en distintas condiciones, para la implementación en hardware embebido, específicamente en la Raspberry Pi.

3.2. Objetivos.

3.2.1. Objetivo general.

El objetivo es implementar un sistema de detección de rostros en tiempo real, que permita enfocar a un sujeto y identificar quien es. Deberá ser configurable de forma fácil para el usuario y/o de forma remota, ya que si estamos en el caso de aplicación de control de acceso y se integra un nuevo sujeto a las dependencias, tiene que ser fácilmente configurable.

3.2.2. Objetivos específicos.

Para proyecto de memoria de título se han planteado los siguientes objetivos:

- Seleccionar los tres mejores algoritmos de la literatura.
- Comparar el desempeño de estos algoritmos.
- Establecer el algoritmo más robusto para distintas condiciones de operación.
- Implementar estos algoritmos en la Raspberry Pi.

3.3. Alcances y limitaciones.

Los alcances definen lo que se va a realizar en el dispositivo embebido sobre *Linux*. El enfoque de este estudio va a pasar por estos tres algoritmos: Distancia Euclíadiana, Eigenface y Support Vector Machine (SVM). Se va a ocupar distancia Euclíadiana porque se parece mucho al método de comparación con plantillas estudiado por Inostroza [2] y por otra parte utilizar distancia Euclíadiana es natural. *Eigenface* es una tradición utilizarlo debido a que es un método robusto y todo el mundo utiliza. Finalmente SVM por que es un algoritmo sólido.

Dentro de las limitaciones se especifica utilizar la implementación de estos tres algoritmos los cuales se encuentran programados en distintos lenguajes. Support vector machine puede necesitar mayores modificaciones en cuanto a implementación en caso de no encontrarse implementado para rostros, para lo cual la solución es ocupar otra implementación y adaptarla a los requerimientos.

El aporte no pasa por elaborar e implementar los algoritmos, sino esta en ver bajo en que condiciones fallan estos algoritmos, caracterizando a través de curvas de desempeño. La idea es obtener porcentajes medios de acierto, analizar la *performance* si cambia la iluminación, si se cambia la rotación del rostro, siendo este tipo de cosas interesantes de estudiar. Entonces para lograr esto se fabricará una batería de posibles cambios lo cual sería el primer paso una vez iniciada la memoria de título.

La aplicación se desarrollará usando el modelo Raspberry Pi B+ v1.2 con el hardware descrito en detalle en la sección 4.4.1 Hardware Raspberry Pi. Además el sistema operativo utilizado será *RASPBIAN* Kernel version 3.18 con fecha de lanzamiento 5 de mayo de 2015 y la versión de Open CV utilizada es la 2.4.9.

3.4. Metodología de trabajo.

La metodología de trabajo consiste en recurrir a los fundamentos de la detección de rostros, enfocado en el estudio diferentes algoritmos. La metodología se divide principalmente en los siguientes pasos:

- Estudio de la literatura
 - Estudios de sistemas de reconocimiento de rostros, en particular revisar en profundidad algoritmos de *Eigenface*, distancia Euclidiana y *support vector machine*.
 - Realizar ejemplos de técnicas de detección de rostros implementadas en entornos de Linux para su posterior ejecución en Raspberry Pi.
- Base de datos
 - Definir como fabricar una base de datos propia.
 - Realizar la base de datos en base a voluntarios que quieran participar en el proyecto.
 - Realizar ajustes y pruebas de la base de datos.
- Implementar a lo menos tres algoritmos en Raspberry Pi
 - Realizar preprocessamiento a las imágenes, entrenando en caso que el algoritmo lo requiera.
 - Realizar pruebas de detección con la base de datos antes desarrollada.
- Desarrollo de una set de pruebas cambiando patrones en la detección de rostros con el fin de obtener una caracterización a través de curvas de desempeño.
- Evaluación de los resultados y conclusiones extraídas.

Cabe destacar que se requiere un computador de uso general con software Ubuntu con versión 14.04 LTS. No se descarta el uso de otras versiones posteriores, o la migración a otro sistema operativo. Respecto a la Raspberry Pi por el momento se ocupará el modelo B+ v1.2 no descartándose el uso de un modelo superior en desempeño que se encuentre en el mercado.

4. Avance de la memoria de título.

4.1. Introducción.

Después de la investigación realizada se decide empezar por estudiar métodos pioneros en el reconocimiento de rostros, en específico, *Eigenface*, ya que es un sistema de implementación relativamente sencilla, con la que se compara todo actualmente, siempre obteniendo buenos resultados. Luego se hará un enfoque al reconocimiento facial, con el fin de comprobar el parecido entre dos caras. Se calculará solo la distancia Euclidiana ya que para implementar la distancia de Mahalanobis se obtuvieron problemas al estimar la matriz de covarianza, ya que se necesitan más observaciones que dimensionalidad de los datos, es decir, si las imágenes son de $N \times N$ hay que tener $N \times N + 1$ imágenes de entrenamiento para estimar bien la matriz de covariancia. Continuando se hará una introducción al concepto de support vector machine, dejando en claro como funciona este de forma general.

Finalmente se mostrará todo el hardware de la Raspberry Pi y las simulaciones obtenidas hasta el momento tanto en Matlab, como en la Raspberry Pi.

4.2. Teoría, modelos y métodos.

4.2.1. *Eigenfaces*

La idea de aplicar análisis de componentes principales (PCA) para representar imágenes de caras en una dimensión baja fue introducida por Lawrence Sirovich y Robert M. Kirby en 1987 [15]. Comenzando con un conjunto original de imágenes de caras calculaban el mejor sistema de coordenadas para comprimir la imagen, donde cada coordenada es una imagen que ellos llaman *eigenpictures*¹ «imágenes propias». Esta idea fue tomada después por Matthew Turk y Alex Pentland [8] quienes desarrollaron el método de reconocimiento de Sirovich y Kirby y la re-formularon con la denominación de *Eigenfaces*, «caras propias». En la literatura se habla de PCA y *Eigenfaces* indistintamente, lo cual no es del todo correcto: El método de PCA es un

¹El término eigen lo introdujo por primera vez en este contexto David Hilbert en 1904, significa «característico» o «individual», en alemán. Los eigen vectors son los vectores propios.”

método general de análisis de datos, por otra parte, *Eigenfaces* es un método de reconocimiento que utiliza PCA con alguna variación.[\[3\]](#).

La idea es tomar de un rostro una secuencia de imágenes, con distintos ángulos e iluminación similar, lo cual forma una matriz en donde cada imagen de $N \times N$ se representa en una columna. Esto nos dice que se necesita una gran cantidad de imágenes de entrenamiento.

Luego se descompone esta matriz, ya que se busca encontrar la base que mejor exprese la distribución de imágenes de caras dentro de un espacio completo, siendo la parte de álgebra lineal. Estos vectores describen la base del subespacio de las imágenes de cara de dimensión $d = N^2$ que describe una imagen de $N \times N$, y es una combinación lineal de los vectores base del subespacio. Estos vectores son los vectores propios de la matriz de covarianza. Los valores propios contienen las características faciales principales que definen los rostros de una persona ordenándolos en forma decreciente. Es decir tomando algunos valores propios se puede representar una cantidad de variación entre los rostros. Esto en términos generales es lo que se conoce como *Eigenface*.

4.2.1.1 Algoritmo

Según Turk, [\[8\]](#), un algoritmo típico de reconocimiento de caras utilizando las *Eigenfaces* seguiría:

1. Tomar un conjunto inicial de imágenes (el conjunto de entrenamiento). Este conjunto debería incluir un número de imágenes para cada persona, con variaciones en la expresión y en la iluminación.
2. Representar cada imagen o sujeto como un vector Γ_i de dimensión $N^2 \times 1$
3. Calcular el promedio del conjunto de vectores Γ_i .
4. Subtraer el promedio a cada rostro.
5. Calcular la matriz de covarianza C.
6. Cálculo de los autovectores, conservando los k autovalores más altos.

7. Reconocimiento de la imagen desconocida Γ de dimensión $N^2 \times 1$
8. Se normaliza y se proyecta la imagen normalizada en el espacio característico.
9. Se calcula la distancia Euclídea

4.2.1.2 Cálculos de Eigenfaces

Según Chichizola, [16] los pasos y cálculos más importantes son:

1. Cada imagen Γ_i con $i = 1, 2, \dots, M$ es reorganizada como un vector de dimensión N^2 cuyo valor es construido como la concatenación de cada una de las filas de la imagen, formando así una matriz de $N^2 \times M$.
2. Se obtiene el rostro promedio

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad (1)$$

3. El rostro promedio Φ obtenido es restado a cada una de las imágenes Γ_i con i entre $1 \dots M$ obteniendo un nuevo conjunto de vectores

$$\Phi_i = \Gamma_i - \Psi \quad (2)$$

que conforman la matriz $\Lambda = [\Phi_1, \Phi_2, \dots, \Phi_M]$ de $N^2 \times M$.

4. En este punto se buscan los autovectores de la matriz de covarianza de Λ

$$C = \frac{1}{N^2} \Lambda \Lambda^T \quad (3)$$

de dimensión $N^2 \times N^2$. Estos vectores propios son los vectores ortonormales usados para construir la representación de las imágenes. El tamaño de la matriz C hace intratable este paso (por el espacio y tiempo requerido). Para solucionar este problema, se obtiene una aproximación de dichos vectores.

- a) Se obtienen la matriz de covarianza reducida

$$L = \frac{1}{M} \Lambda^T \Lambda \quad (4)$$

de dimensión $M \times M$

b) Se obtienen los autovalores de L , los que ordenados de mayor a menor según sus correspondientes autovalores, conforman la matriz v .

c) Se aproximan los autovalores C

$$u = \Lambda v \quad (5)$$

donde cada columna de u representa un vector propio.

5. Se obtiene el patrón

$$\Omega_i^T = [\omega_1, \omega_2, \dots, \omega_M] \quad (6)$$

con $i = 1, 2, \dots, M$ donde

$$\omega_k = u_k^T (\Gamma_i - \Psi) \quad (7)$$

con $k = 1, 2, \dots, M$

Dada la imagen de un rostro como entrada del sistema, el proceso de **reconocimiento** intenta encontrar en la base de imágenes, aquella que se corresponde con el rostro dado, para lo cual se calcula su patrón Ω utilizando el mismo procedimiento anteriormente descrito, y busca la distancia mínima[16].

$$\min(\|\Omega - \Omega_i\|^2) \text{ con } i = 1, 2, \dots, M$$

Una vez encontrada la distancia mínima se indica cuál es la imagen.

4.2.2. Distancia Euclíadiana

La distancia Euclíadiana se define como

$$\epsilon = \sqrt{(y_1 - x_2)^2 + (y_2 - x_2)^2 + \dots + (y_n - x_n)^2} \quad (8)$$

Sirve para comprobar el parecido entre dos caras.

Sea el conjunto de imágenes de caras x_1, x_2, \dots, x_m la cara media se define como el cálculo de \bar{x} . Puede entenderse que el cálculo representa los valores del “individuo medio” del conjunto de imágenes.

4.2.3. Introducción a Support vector machine

El algoritmo *Support vector machine* en adelante SVM, contiene mucha matemática detrás, pero el desempeño de SVM una vez completada la etapa de entrenamiento es muy bueno. Primero hay que aclarar que es un algoritmo binario por lo tanto requerirá de árboles de decisión.

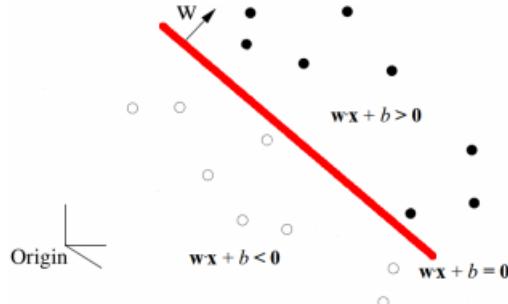


Figura 1: Hiperplano definido por la ecuación $w \cdot x + b = 0$ donde w es la normal y b es el offset, [21].

Como se observa en la Figura 1 existen dos clases y la forma natural de separar estas clases es hacer una línea la cual tiene una pendiente y un *offset*. Lo que propone el método de SVM es definir esta línea de forma óptima, para encontrar los parámetros w y b , tal que la distancia entre los vectores más cercanos a esta línea de cada clase sea la máxima posible. Esto permite que el plano este lo más al medio posible. SVM lo que busca entonces es definir estos dos vectores de forma óptima tal se satisfaga la condición.

Se observa además que dos clases diferentes son linealmente separables. La clasificación mediante SVM se obtiene de la siguiente manera a modo de ejemplo.

$$w \cdot x + b > 0 \Rightarrow \text{dato pertenece a clase 0} \quad (9)$$

$$w \cdot x + b < 0 \Rightarrow \text{dato pertenece a clase 1} \quad (10)$$

Se observa que la clasificación pasa a ser más sencilla para cada nuevo dato, lo que se tiene que hacer es calcular el producto punto, sumarle el offset y ver el signo del resultado para determinar a qué clase pertenece el dato. Ahora la pregunta más lógica es ¿cuál de las líneas que dividen los hiperplanos es la mejor? Para dar solución a esta pregunta se propuso que la

mejor línea era la que maximizaba el margen, que sería la distancia entre las dos clases.

Se llama SVM porque cada uno de los puntos o datos se llaman vectores de soporte, es decir, son los que soportan la definición de la línea. Estos puntos además permiten definir planos H_1 y H_2 y la línea optima es la que pasa justo por entremedio de estos dos planos.

El problema de optimización es encontrar w y b tal que el margen entre ambos planos H_1 y H_2 este maximizado sujeto a la condición de que los vectores deberían estar separados correctamente por el plano tal como se observa a continuación en la Figura 2.

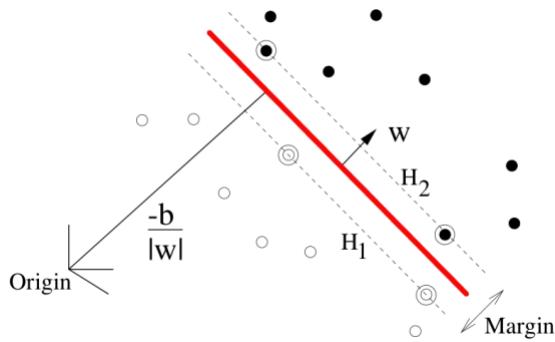


Figura 2: SVM linealmente separable con sus planos H_1 y H_2 , [21].

El margen está determinado por la norma $\frac{1}{\|w\|}$ y se tiene que tener un set de vectores de entrenamiento definido.

SVM además ocupa multiplicadores de Lagrange, esforzándose para encontrar coeficientes tal que:

Buscamos $\alpha_1, \dots, \alpha_n$ tal que

$$Q(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i x_j \text{ este maximizado}$$

Sujeto a

i) $\sum_i \alpha_i y_i = 0$

ii) $\alpha_i \geq 0$ para todo α_i

Finalmente se define una función y se soluciona la problemática por el producto punto de los vectores de entrenamiento con sus etiquetas.

$$f(x) = \sum \alpha_i y_i x_i \cdot x + b \quad (11)$$

Una gran ventaja de este algoritmo es que la implementación se encuentra disponible gracias a *toolbox* en C o en *python* que tienen escritas estas rutinas de optimización.

Otro problema ocurre cuando los datos no son separables, es decir, no se puede definir ningún plano. Este problema se resolvió proyectando los datos a otra dimensión donde probablemente, si existe un plano que permite la separación de las clases. Todo se resume en el producto punto de la ecuación 11 que tiene que satisfacer los teoremas antes mencionados. Si no es posible la separación de las clases, pero si la separación es posible en otra dimensión, existe entonces un plano que permite separarlos. Como consecuencia hay un mapeo Φ que toma el producto punto y lo convierte en el producto punto del mapeo, lo que en realidad define el concepto de kernel que es cualquier función que satisface las condiciones anteriores.

El kernel implícitamente mapea los datos en un mundo de alta dimensión sin conocer Φ . El kernel lineal que se tenía originalmente para el problema de optimización, se puede definir otro tipo de kernel, como los polinómicos o los Gaussianos, los cuales ocupan combinaciones lineales y el problema de optimización finalmente es el mismo.

Entonces lo que se hace es tomar el problema en cierta dimensión resolverlo linealmente en otra dimensión y después mapearlo de vuelta. Sin lugar a dudas SVM es súper poderoso.

SVM aplicado a rostros computacionalmente es costoso en la etapa de entrenamiento, pero una vez que se realiza esta etapa, es el cálculo de multiplicar productos puntos y sumas lo hace sencillo. Otro problema que se ve es que si se ingresa una nueva cara a la base de datos es necesario entrenar de nuevo.

Finalmente como conclusión de este método podemos decir que en términos de aplicabilidad, una vez que ya se está entrenado, es comparable con el algoritmo de Eigenface.

4.2.4. Evaluación de algoritmos de reconocimiento facial propuestos por Inostroza

La idea es explicar de manera más detallada cada algoritmo estudiado por Inostroza [2]. Primero aclarar que el conjunto de imágenes utilizado para realizar las pruebas de reconocimiento corresponde a una base de datos que contiene cuatrocientos rostros de cuarenta individuos distintos, y que pertenece a la base de datos *Olivetti Research Laboratory*.

4.2.4.1 Comparación con plantilla (Template Matching)

Este método busca a partir de un conjunto de imágenes, tener definidas distintas clases, en otras palabras, una clase es un *set* de imágenes de un individuo. A partir de cada clase entonces se creará a lo menos una plantilla o máscara para que una vez que llegue la nueva imagen a clasificar, se mida la distancia entre la plantilla de la nueva imagen, con cada plantilla de las diferentes clases antes obtenidas. La forma de crear una plantilla es usar el promedio entre todas las imágenes o usar la características principales de la clase y se obtiene algo como la Figura 3.

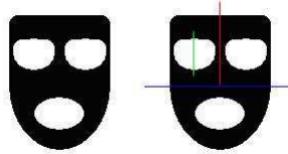


Figura 3: Plantillas usadas para la detección de rostros, [20].

La comparación con una plantilla entonces se refiere a medir directamente la distancia entre un patrón de entrada y aquellos almacenados en la base de datos, para permitir la correspondencia de cada patrón presentado. Los patrones son considerados como vectores y la correspondencia de cada patrón de entrada, con otra base de datos, se realiza con base en la menor distancia encontrada entre estos [17].

El método de comparación con una plantilla es un sistema de clasificación de patrones que consiste en determinar cuál de las clases almacenadas en la base de datos se encuentra más cercana, utilizando una medida dada, al patrón de entrada presentado. Dado un vector de entrada x y un conjunto de vectores de referencia (plantillas) m_i , se busca el vector m_c tal que

$$\|x - m_c\| = \min_i \|x - m_i\| \quad x, m_i \in \Re \quad (12)$$

La norma utilizada para la medición entre los patrones es arbitraria, pudiendo usarse la norma euclidiana. A priori se puede ver que el éxito radica en que los patrones presentados en la entrada, tengan la mayor similitud posible con su respectiva base de datos, tanto en lo que se refiere a tamaño, ubicación, orientación, intensidad, dirección de la iluminación. El método tiene fallos, por ejemplo, si el tamaño de la plantilla es diferente al de la imagen nueva el método no funciona (ver Figura 4) o si cambia el fondo de la imagen a detectar. Tampoco funciona si se cambia la rotación del rostro o hay cambios en la iluminación en la imagen. Debido a todo lo mencionado no es factible utilizar este método e implementarlo en la Raspberry Pi debido a lo limitado del algoritmo.

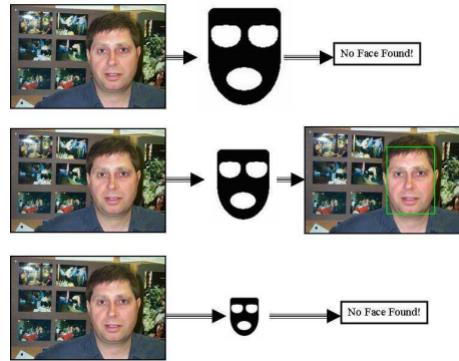


Figura 4: Errores al implementar distintos tamaños de plantillas o máscaras, [20].

4.2.4.2 Mapas Auto-Organizativos

Es un tipo de red neuronal que se caracteriza por preservar las similitudes en los patrones de entrada como similitudes en la localización topológica de la salida.

Se asimila mucho a la idea de Eigenface ya que se tienen estas métricas buscando la similitud entre los rostros a través de las características principales. En el mapa auto-organizativo, cada una de las células se sintoniza específicamente a cierto patrón o clase de patrón de entrada. Este es presentado a todas las células, y la magnitud de la respuesta que una de ellas entregue dependerá de cuanta similitud exista entre el vector de referencia de la célula en particular con

el patrón de entrada. La célula más sintonizada con una entrada determinada(denominada célula “ganadora”), emitirá la mayor señal de salida, mientras que las células vecinas presentarán señales de salida menores, disminuyendo su magnitud a medida que se encuentran más alejadas de la célula “ganadora” [2] . Este tipo de algoritmo es especialmente útil en la clasificación de datos que presentan a un gran número de clases.

La utilización de mapas auto-organizativos en el reconocimiento de rostros pretende que el algoritmo aprenda la distribución de probabilidad de un conjunto de imágenes de entrenamiento y ordene los vectores de referencia de los nodos, de tal forma que permita que imágenes no utilizadas en el entrenamiento, pero que pertenecen a individuos conocidos, puedan ser identificadas de la mejor forma posible. La principal desventaja haciéndolo poco aplicable al problema del proyecto es que el entrenamiento puede durar hasta un día y medio usando la base de datos *Olivetti Research Laboratory*.

4.2.4.3 Bases de características no ortogonales

Este algoritmo se basa en mapas auto-organizativos, que tiene como finalidad crear una base de características no ortogonales para la representación de un conjunto de datos vectoriales. Se pretende crear una base no orgonal de características, que represente en forma adecuada un cierto conjunto de vectores. Al igual que el algoritmo anterior su desventaja es la etapa de entrenamiento.

4.2.4.4 Resultados obtenidos de los tres algoritmos

Cabe destacar que las imágenes trabajadas por Inostroza [2], son de fondo controlado, en las que el rostro se encuentra medianamente normalizado en cuanto a orientación y posición. La comparación de rostros con plantilla, ha sido el que ha arrojado los mejores porcentajes de reconocimiento, llegando a reconocer en los mejores casos, 199 imágenes de las 200 que le han sido presentadas para la clasificación. Las imágenes pertenecen a 40 individuos diferentes (utilizándose como referencia 5 imágenes de cada uno).

Incluso al utilizar imágenes de baja resolución (23x28 píxeles), y el conjunto seleccionado manualmente, se obtiene un buen porcentaje de reconocimiento con un promedio de 95.25 %. La

velocidad de reconocimiento relativamente alta (0.515 segundos por imagen en el caso más lento cuando se utiliza una resolución de 23x28 píxeles).

Para el segundo algoritmo, la utilización de mapas auto-organizativos, presenta un desempeño menor al algoritmo de comparación de plantillas en casi todos los sentidos. La necesidad de este método de que los vectores de referencia sean creados mediante entrenamiento, es su principal desventaja, ya que tarde demaciado tiempo volviendo el método impráctico.

Para el tercer algoritmo bajo evaluación por Inostroza [2] arrojan resultados que están por debajo de los obtenidos por los dos algoritmos anteriores. La desventaja es que el tiempo de entrenamiento es aproximadamente 7.15 veces superior al mapa auto-organizativo. En resumen se obtuvo los resultados que se muestran en la siguiente tabla 1

Porcentaje de reconocimiento			
Método	Comparación con plantilla	Mapa auto-organizativo	Base características no ortogonales
%Máximo	99 %	97 %	87.5 %
Tiempo de reconocimiento(200 imágenes)			
Método	Comparación con plantilla	Mapa auto-organizativo	Base características no ortogonales
Segundos	1 a 713 [seg]	2 a 4 [seg]	4 [seg]

Tabla 1: Desempeños de algoritmos

4.3. Simulaciones preliminares.

4.3.1. Simulación distancia Euclidiana en Matlab

Para la base de datos elegida se implemento la clasificación de mínima distancia Euclidiana (en adelante lo referimos como clasificador Euclidianos).

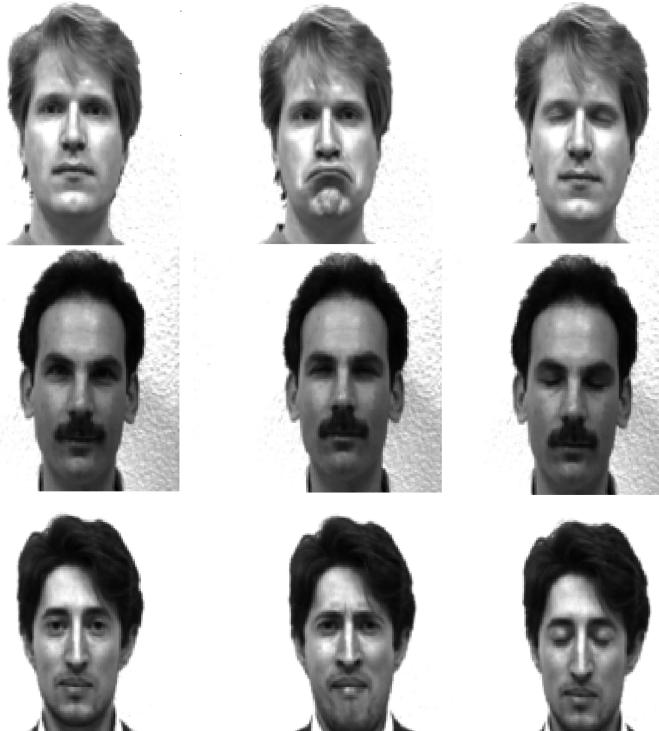


Figura 5: Base de datos YaleFace con 3 sujeto en 3 distintas situaciones.



Figura 6: a) Imagen de Prueba b)Sujeto detectado con mínima distancia Euclidiana.

Entonces, observe la Figura 5. En esta figura, se muestra la pequeña base de datos correspondientes a yaleface de 3 sujetos. De los resultado se esperaba un peor comportamiento, ya que la distancia Euclídea solo dice a cual sujeto se parece más la imagen de prueba. Es decir si tenemos dos sujetos que se parecen la distancia Euclídea no necesariamente va a elegir al sujeto correcto.

4.3.2. Simulación Eigenface en Matlab

Se trabaja con la misma base de datos.



Figura 7: Cálculo de rostro promedio.



Figura 8: Resta del rostro promedio a cada una de las imágenes.

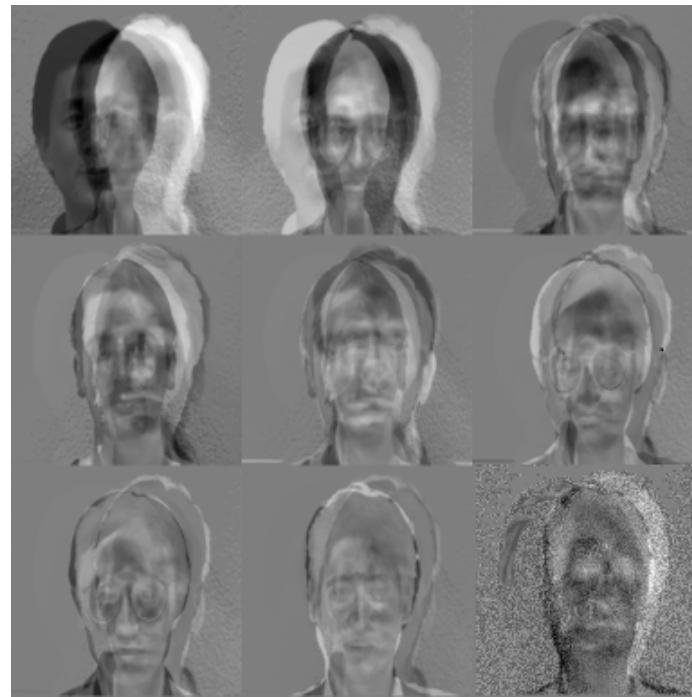


Figura 9: Eigenfaces.

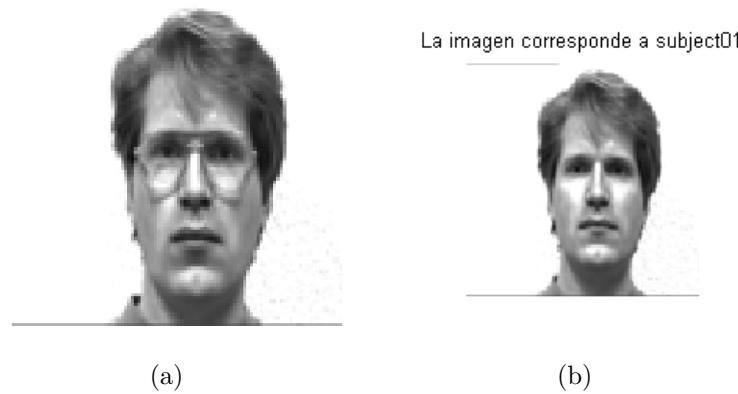


Figura 10: a) Imagen de Prueba b)Sujeto detectado con algoritmo *Eigenface*.

Como se observa en la simulación de *Eigenface* se aplicaron los algoritmos antes descritos y sus respectivos cálculos matemáticos.

4.4. Raspberry Pi

4.4.1. Hardware Raspberry Pi

Debido a que se ha comentado del hardware limitado de la tarjeta comercial Raspberry Pi se expondrá cual son sus componentes en hardware.

El diseño incluye un *system-on-a-chip Broadcom BCM2835*, que permite hacer overclocking hasta velocidad Turbo; 1000 MHz ARM, 500 MHz core, 600 MHz SDRAM, 6 overvolt. de forma segura. Contiene un procesador central de 700MHz *ARM1176JZF-S core* con una *GPU Broadcom VideoCore IV*, *OpenGL ES 2.0*, *OpenVG 1080p30 H.264 / MPEG-4 AVC* de alto perfil decodificador y una memoria *SDRAM* de 512 Megabytes. La primera Raspberry al ser lanzada tenia solamente 256 Megabytes de SDRAM. El modelo que se utilizó para este proyecto incluye cuatro puertos USB con un puerto de red Ethernet tal como se observa a continuación en la Figura 11.

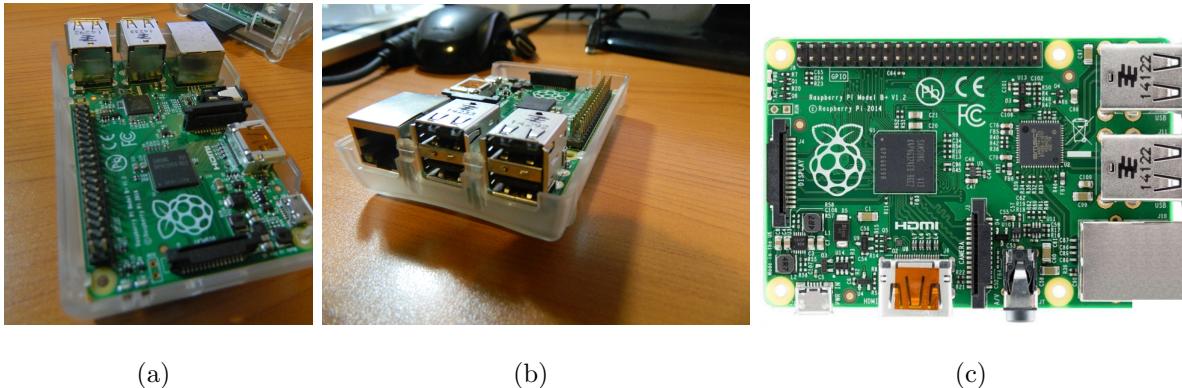


Figura 11: Hardware de Raspberry Pi utilizada para realizar este proyecto

El diseño no incluye un disco duro o una unidad de estado sólido, ya que usa una tarjeta micro SD para el almacenamiento permanente; tampoco incluye fuente de alimentación[12].

Respecto a los periféricos, la tarjeta se puede visualizar en una TV o monitor gracias a la conexión HDMI @ 1920x1200 píxeles. Se puede conectar una cámara, lo cual puede ser una WebCam o la Raspberry Pi Cam. También posee una conexión ethernet 10/100 Mbps para proveer de una conexión a internet o se puede obtener un Wi-Pi que es una adaptar USB de WiFi. Por último los periféricos más utilizados con un teclado y mouse USB que permiten el

manejo y darle instrucciones a las Raspberry Pi. En la Figura 12 se puede observar los periféricos más relevantes.

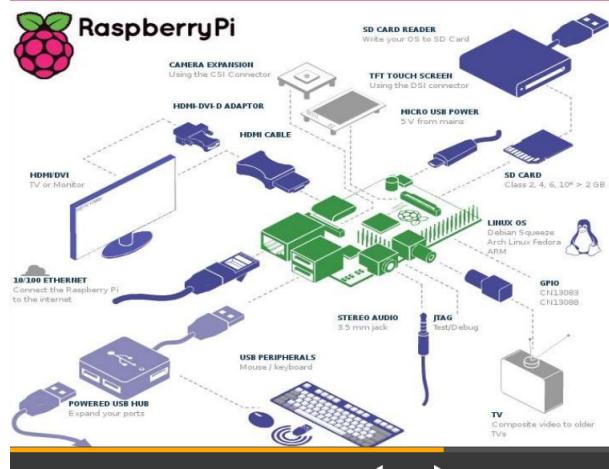


Figura 12: Todos los periféricos que se pueden conectar a la Raspberry Pi Modelo B, de dos puertos USB,[22].

El tamaño de esta placa es de $85,60 \times 56,5\text{mm}$ con un peso de 45 g y un consumo de energía de $5v, 600mA$.

4.4.2. Pi Cam

La instalación de la Pi cámara esta muy bien explicado por el CEO en el sitio oficial de Raspberry Pi <http://www.raspberrypi.org/archives/3890>.

Gracias a la carcasa de la Raspberry Pi, la cámara queda bien montada en la tarjeta y la cinta flex queda protegida, disminuyendo los riesgos por estática. El último paso para montar la cámara es atornillada a la carcasa dejando bien asegurada. La Pi cam y como quedó montada se pude observar en la Figura 13.

Para activar la cámara basta con entrar a la configuración de la Raspberry Pi y navegar hasta cámara para seleccionar “enable”.

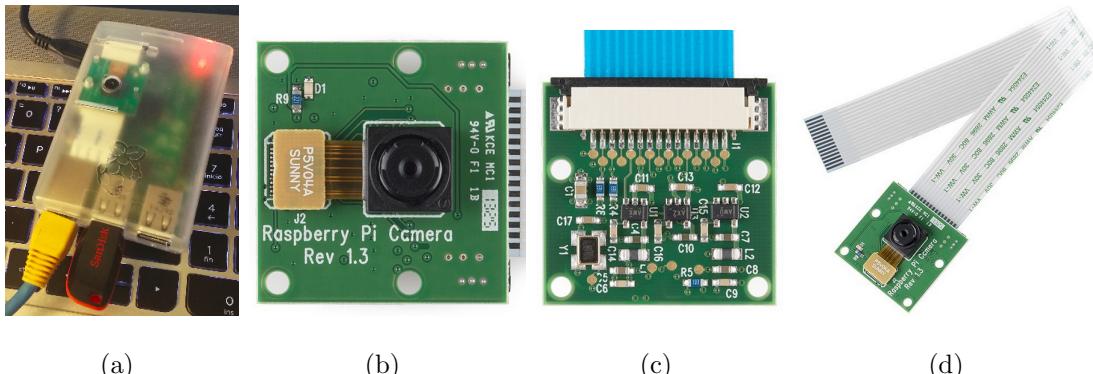


Figura 13: a)Raspberry Pi con carcasa y Pi Cam montada b)c)d)Imágenes de cámara Raspberry Pi que se utiliza en el proyecto, [23]

4.4.3. Instalación de OpenCV en Raspberry Pi

El proceso para instalar OpenCV tomó alrededor de 10 horas. Esto se debe a la creación de un código OpenCV configuración y construcción [19].

Como con cualquier instalación en el Raspberry Pi, se comienza con la actualización de está con el siguiente comando:

```
sudo apt-get update
```

```
pi@raspberrypi ~ $ sudo apt-get update
Hit http://repository.wolfram.com stable Release.gpg
Hit http://archive.raspberrypi.org wheezy Release.gpg
Get:1 http://mirrordirector.raspbian.org wheezy Release.gpg [490 B]
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Hit http://repository.wolfram.com stable Release
Hit http://archive.raspberrypi.org wheezy Release
Get:2 http://mirrordirector.raspbian.org wheezy Release [14.4 kB]
```

Figura 14: Update solo chequea repositorios.

```
sudo apt-get upgrade
```

```
pi@raspberrypi ~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages have been kept back:
  fake-hwclock libbfm-data libgail-3.0 libgcc1 libgfortran3 libgomp1 libgtk-3-0
  libgtk-3-bin libgtk-3-common libjavascriptcoregtk-3.0-0 libstdc++6
  libwebkitgtk-3.0-0 lxpanel omxplayer pcmanfm wolfram-engine

```

Figura 15: Upgrade ejecuta la actualizacion efectiva del sistema.

Luego se ejecutan los dos comandos siguientes:

```
sudo apt-get -y install build-essential cmake cmake-curses-gui \
pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 \
libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev pngtools \
libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools \
libeigen3-dev
```

```
sudo apt-get -y install libjpeg8 libjpeg8-dev libjpeg8-dbg \
libjpeg-progs ffmpeg libavcodec-dev libavcodec53 libavformat53 \
libavformat-dev libgstreamer0.10-0-dbg libgstreamer0.10-0 \
libgstreamer0.10-dev libxine1-ffmpeg libxine-dev libxine1-bin \
libunicap2 libunicap2-dev swig libv4l-0 libv4l-dev python-numpy \
libpython2.6 python-dev python2.6-dev libgtk2.0-dev
```

Después de que las bibliotecas hayan terminado de instalar. Luego se descarga la versión de OpenCV 2.4.9 utilizando el comando wget para descargar el archivo zip

```
wget -O openCV-2.4.9.zip \
http://sourceforge.net/projects/opencvlibrary/files/opencv \
-unix/2.4.9/opencv-2.4.9.zip/download
```

Después de finalizada la descarga se tendrá openCV-2.4.9.zip, el paso siguiente es utilizar el comando unzip para descomprimir el archivo zip

```
unzip openCV-2.4.9.zip
```

Esto creará una carpeta OpenCV-2.4.9. Se creó la carpeta release que es donde vamos a compilar OpenCV.

```
cd openCV-2.4.9
mkdir release
cd release
```

Luego es el momento de configurar OpenCV. Se trata de una larga lista, la cual se puede configurar o usar como viene por defecto. Para entrar al menú de configuración escribimos lo siguiente.

```
sudo ccmake .. /
```

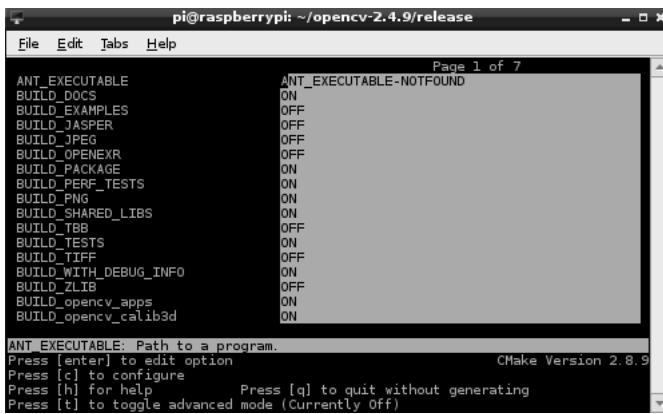


Figura 16: Configuración OpenCV.

Una vez terminada la configuración y guardada, se comenzó a ejecutar el siguiente comando: (estos comandos tomaron alrededor de 10 horas).

```
make
sudo make install
```

En este punto se ha terminado exitosamente de instalar OpenCV.

4.4.4. Problemas con la instalación OpenCV

Una problemática en la instalación de OpenCV es la capacidad del la tarjeta micro SD. Esta es de 8GB y al terminar de instalar OpenCV queda ocupado alrededor de un 85 % de la capacidad total de memoria de la Raspberry Pi.

En una primera instancia, el Makefile generado con los archivos de proyecto, los archivos objeto y binarios de salida ocuparon el 100 % de capacidad de la Raspberry Pi. Se intentó con comandos liberadores de espacio `sudo apt-get clean` y `sudo apt-get autoremove`. La consecuencia de utilizar estos comandos es que eliminan todos los paquetes del cache y borra los paquetes

huérfanos produciendo un error ya que la Raspberry no funcionó correctamente y el espacio liberado no fue significativo. La solución al problema fue reinstalar todo desde el sistema operativo Raspbian desde cero hasta OpenCV y montar un pendrive de 16GB a la Raspberry con la finalidad de evitar problemas de espacio.

4.4.5. Implementación de Eigenface con OpenCV en la Raspberry Pi

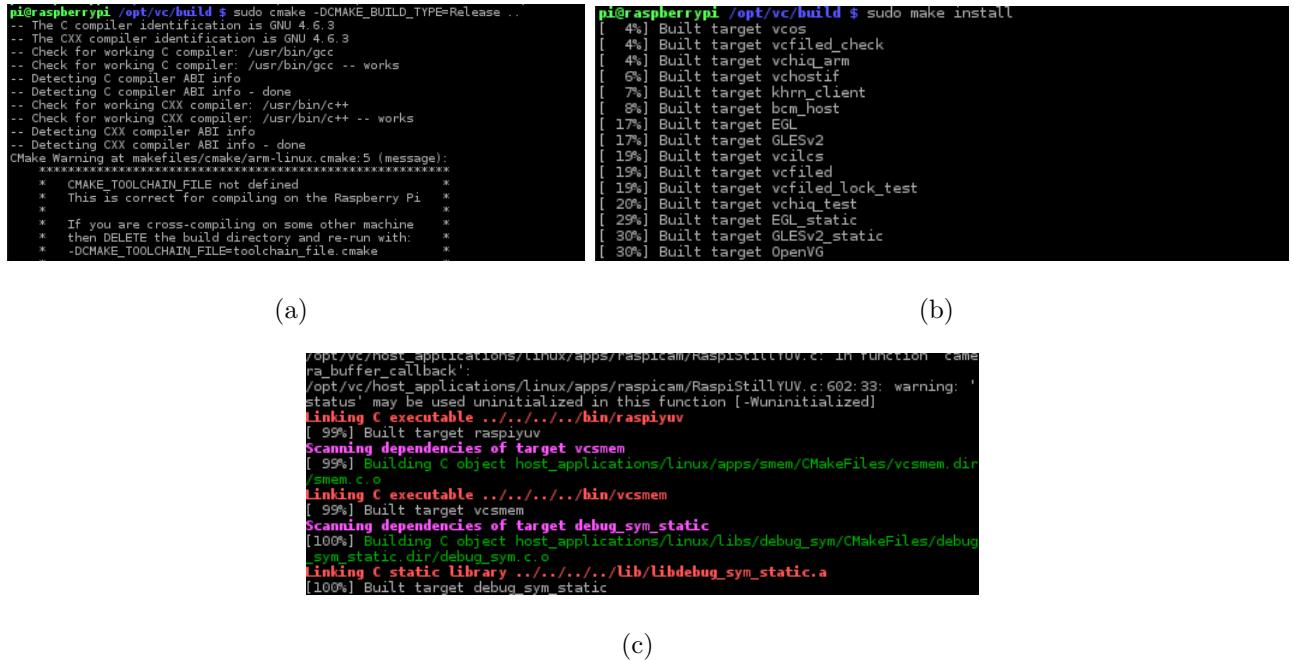
Se siguió con dificultades el tutorial disponible en <https://thinkrpi.wordpress.com/opencv-and-pi-camera-board/>.

El proyecto consiste en desarrollar un sistema de visión embebido, capaz de detectar rostros de sujetos presentes en la base de datos. Este tutorial aún no se termina y lo que hace básicamente es modificar el código fuente de una implementación anterior realizada por el mismo autor, pero utilizando una WebCam.

Es necesario descargar la API de reconocimiento facial que está implementada en OpenCV de <https://github.com/raspberrypi/userland>. Se extrae y se compila con los siguientes comandos:

```
sudo cmake .
sudo make
```

Los resultados luego de terminar el paso 2 del tutorial son:



(a)

```
pi@raspberrypi:~/opt/vc/build$ sudo cmake -DCMAKE_BUILD_TYPE=Release ..
-- The C compiler identification is GNU 4.6.3
-- The CXX compiler identification is GNU 4.6.3
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
CMake Warning at makefiles/cmake/arm-linux.cmake:5 (message):
*****
*   CMAKE_TOOLCHAIN_FILE not defined
*   This is correct for compiling on the Raspberry Pi
*   If you are cross-compiling on some other machine
*   then DELETE the build directory and re-run with:
*   -DCMAKE_TOOLCHAIN_FILE=toolchain_file.cmake
*****
```

(b)

```
pi@raspberrypi:~/opt/vc/build$ sudo make install
[ 4%] Built target vcos
[ 4%] Built target vcfled_check
[ 4%] Built target vchiq_arm
[ 6%] Built target vhostif
[ 7%] Built target khrn_client
[ 8%] Built target bcm_host
[ 17%] Built target EGL
[ 17%] Built target GLESv2
[ 19%] Built target vcilcs
[ 19%] Built target vcfled
[ 19%] Built target vcfled_lock_test
[ 20%] Built target vchiq_test
[ 29%] Built target EGL_static
[ 30%] Built target GLESv2_static
[ 30%] Built target OpenVG
```

(c)

```
[dpt/vc/host_applications/linux/apps/Raspicam/RaspistillYUV.c: In function 'claim_ra_buffer_callback':
/opt/vc/host_applications/linux/apps/raspicam/RaspistillYUV.c:602:33: warning: 'status' may be used uninitialized in this function [-Wuninitialized]
Linking C executable ../../../../bin/raspiyuv
[ 99%] Built target raspiyuv
Scanning dependencies of target vcsmem
[ 99%] Building C object host_applications/linux/apps/smem/CMakeFiles/vcsmem.dir/vsmem.c.o
Linking C executable ../../../../bin/vcsmem
[ 99%] Built target vcsmem
Scanning dependencies of target debug_sym_static
[100%] Building C object host_applications/linux/libs/debug_sym/CMakeFiles/debug_sym_static.dir/debug_sym.c.o
Linking C static library ../../../../lib/libdebug_sym_static.a
[100%] Built target debug_sym_static
```

Figura 17: Creación y construcción del directorio con el que se trabaja.

En el paso 6 la capturación del vídeo se obtuvo el siguiente resultado:

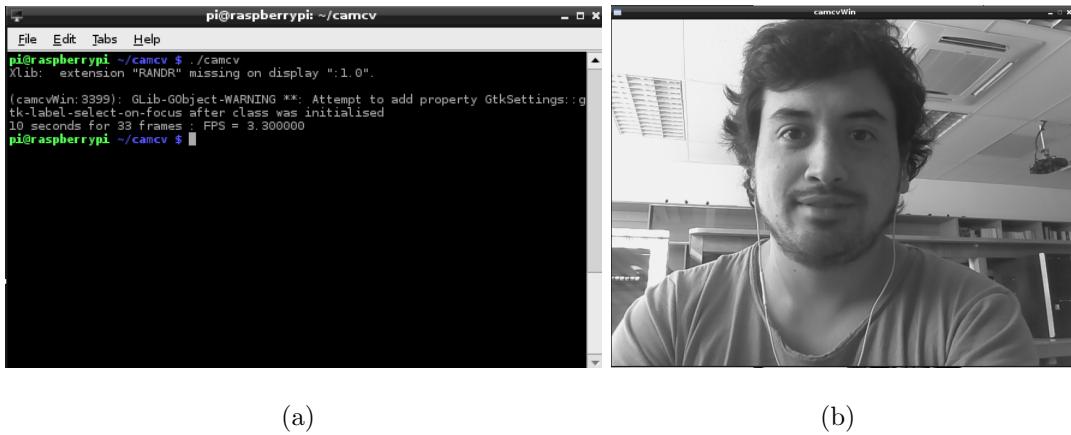


Figura 18: a) Ejecución del programa luego de construir y compilar b) Devolución de un video a 30 FPS capturado en Laboratorio de Redes de Datos.

Lamentablemente con cuatro días de intento para dar la soluciones de los errores de compilación del paso 7 no se pudo terminar este tutorial completamente. Lo que se busca es el siguiente resultado.



Figura 19: Resultado esperado al término del tutorial

4.4.6. Segmentación de Rostros

El rostro humano es muy dinámico y puede tomar distintas apariencias. A un determinado personaje le puede crecer el pelo, la barba, usar anteojos, la iluminación puede cambiar entre una infinidad de opciones.

Como un objetivo, se había planteado segmentar el rostro del resto de la imagen para lo cual se usaron los ejemplos que trae OpenCV. Los resultados se puede observar en la Figura 20.



Figura 20: Segmentación de rostros utilizando los ejemplos de la librería OpenCV

5. Conclusiones y trabajo futuro.

5.1. Conclusiones.

- El proyecto desarrollado analiza y describe lo que otros autores han realizado hasta el momento, las técnicas y algoritmos que existen para la detección de rostros.
- Se acotó los algoritmos a utilizar en la memoria de título, los cuales serán: distancia Euclidiana, Eigenface y Support vector machine.
- Se a escogido apoyarse en las librerías OpenCV para el desarrollo de este proyecto, a pesar de que en un principio se pensó en trabajar en Matlab debido a las facilidades de implementación y a la conexión disponible con la Raspberry Pi. Sin lugar a dudar OpenCV es más sólido y tiene bastantes códigos de guía.
- El reconocimiento facial es un tema muy estudiado y de grandes aplicaciones requeridas en tiempos actuales. Si esto se le mezcla con un sistema embebido de código abierto y bajo costo puede llegar a ser una gran innovación.
- Si es posible dar una solución a este problema, el tema ahora pasa ser eficientes y precisos en la implementación de la solución. Por eso la importancia a la que pueden llegar a ser las curvas de desempeño.

5.2. Trabajo futuro.

Lamentablemente quedaron varios temas pendientes a desarrollar. En general son temas ligados a la programación.

- Primero es terminar de implementar OpenCV en la Raspberry Pi a pesar de los errores de compilación.
- Terminar la implementar el algoritmo de Mahalanobis, queda como tarea pendiente pensando como posible nuevo algoritmo a caracterizar.
- Mejorar y estudiar en más profundidad en la segmentación de rostros.
- Implementar SVM en la Raspberry Pi.
- Crear la base de datos definitiva con la cual se trabajará el resto del proyecto.

- Crear una batería de pruebas para poder caracterizar bajo distintas condiciones y obtener curvas de desempeño.

Referencias

- [1] Edwin Arturo Vega Aquino, “Detección y seguimiento de rostros”, Informe proyecto de fin de carrera, Universidad Carlos III de Madrid, 2011.
- [2] “Reconocimiento de rostros evaluación de la aplicación de tres algoritmos al reconocimiento facial”, informe de memoria de título de Ingeniería Civil Electrónica, Depto. Ing. Eléctrica, Universidad de Concepción, 2001.
- [3] Marcelo J. Armengot Iborra, “Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras”, Vniversitat De València, 2006.
- [4] Alan C. Bovik, “*The Essential Guide to Image Processing*”, Academic Press, 2009 , Capítulo 24, pp. 667-713.
- [5] R. Chellappa, C. L. Wilson, and S. Sirohey. “ Human and machine recognition of faces: a survey”. Proc. IEEE, 83:705–740, 1995.
- [6] P. J. Phillips, H. Moon, S. Rizvi, and P. J. Rauss. The FERET evaluation methodology for facerecognition algorithms.IEEE Trans. Pattern Anal. Mach. Intell.,22:1090–1104, 2000.
- [7] Jian Guo Liu, Philippa Mason, “*Essential Image Processing and GIS for Remote Sensing*” ,John Wiley & Sons, 2013, Capítulo 8, pp. 91-102.
- [8] M. Turk and A. Pentland. “Eigenfaces for recognition” .J. Cogn. Neurosci., 3:72–86, 1991.
- [9] Md. Kawser Jahan Raihan *et al.*. “Raspberry Pi Image Processing Based Economical Automated Toll System”. Global Journal of Researches in Engineering Electrical and Electronics Engineering **13** (2013) 34-41.
- [10] Paul Viola and Michael J. Jones “ Robust Real-Time Face Detection” .,International Journal of Computer Vision **57(2)**, (2004) 137–154.

- [11] Josep Arnal García, Sergio Cuenca Asensi, Higinio Mora Mora, Francisco Moreno Seco (2010). *Actas del 1^{er} Workshop en Tecnologías de la Informática* [Online]. Available url: http://www.iuui.ua.es/master_ti/actas2010.pdf
File: actas2010.pdf
- [12] Víctor Garcés, Pablo Gleisner. “Sistemas de visión en dispositivos embebidos sobre linux”, Univ. de Concepción, Concepción, Chile, 2013.
- [13] Mario Medina C. Sistemas (2015). *Introducción a Sistemas Computacionales* [Online]. Avaiable url: <http://mondrian.die.udec.cl/~mmedina/Clases/SisComp/2014-2/SisComp-01-Intro.pdf>
File: SisComp-01-Intro.pdf
- [14] Página Web de Yale Face Database url<http://vision.ucsd.edu/content/yale-face-database>.
- [15] L. Sirovich and M. Kirby “Low-dimensional procedure for the characterizacion of human faces”, Journal of the Optical Society of America A, 4(3): 519-524,1987.
- [16] Franco Chichizola, Armando De Giusti, and Marcelo Naiouf. “Eigenfaces de Imagen Reducida para el Reconocimiento Automático de Rostros”. RedUNCI, pag. 812-823, 2003.
- [17] Tijmen Majoor (2000). “Face detection using color based region of interest selection”[Online].Avaiable url:
<http://www.science.uva.nl/research/ias/masterProjects/MajoorThesis.pdf>
File: MajoorThesis.pdf
- [18] Raspberry Pi(2015). *¿Qué es una Raspberry Pi?* Avaiable url: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>
- [19] RoboPapa (2014). *Install Open CV on Raspberry Pi*[Online]
Avaiable url: <http://www.robopapa.com/Projects/InstallOpenCVOnRaspberryPi>
- [20] Amir Faizi. “Robust Face Detection Using Template Matching Algorithm”, Thesis for the degree of Masters of Applied Science, Dep. of Electrical Engineering, University of Toronto, 2008.

- [21] S. E. Godoy. “Introduction to Support Vector Machines (SVM) Classifier”. Application to MS data, ECE - CHTM University of New Mexico Albuquerque, NM, USA, 22 July, 2011.
- [22] Xataka(2015) *Raspberry Pi 2 Model B: más potente y con soporte Windows 10* Avaiible url: <http://www.mamaluchona.com/sistemas-it/raspberry-pi-pi-pi-pi-en-palabras-sencillas/>
- [23] Bricogeek(2015) *Cámara de 5MP para Raspberry Pi*, Avaiible url: <http://tienda.bricogeek.com/raspberry-pi/569-camara-de-5mp-para-raspberry-pi.html>
- [24] Especificaciones técnicas Raspberry Pi(2014), Avaiible url:http://www.element14.com/community/servlet/JiveServlet/previewBody/68376-102-1-295702/RPI-BPLUS-V1_2-SCHEMATIC-REDUCED.pdf

A. Apéndices

A.1. Raspberry Pi y Cámara

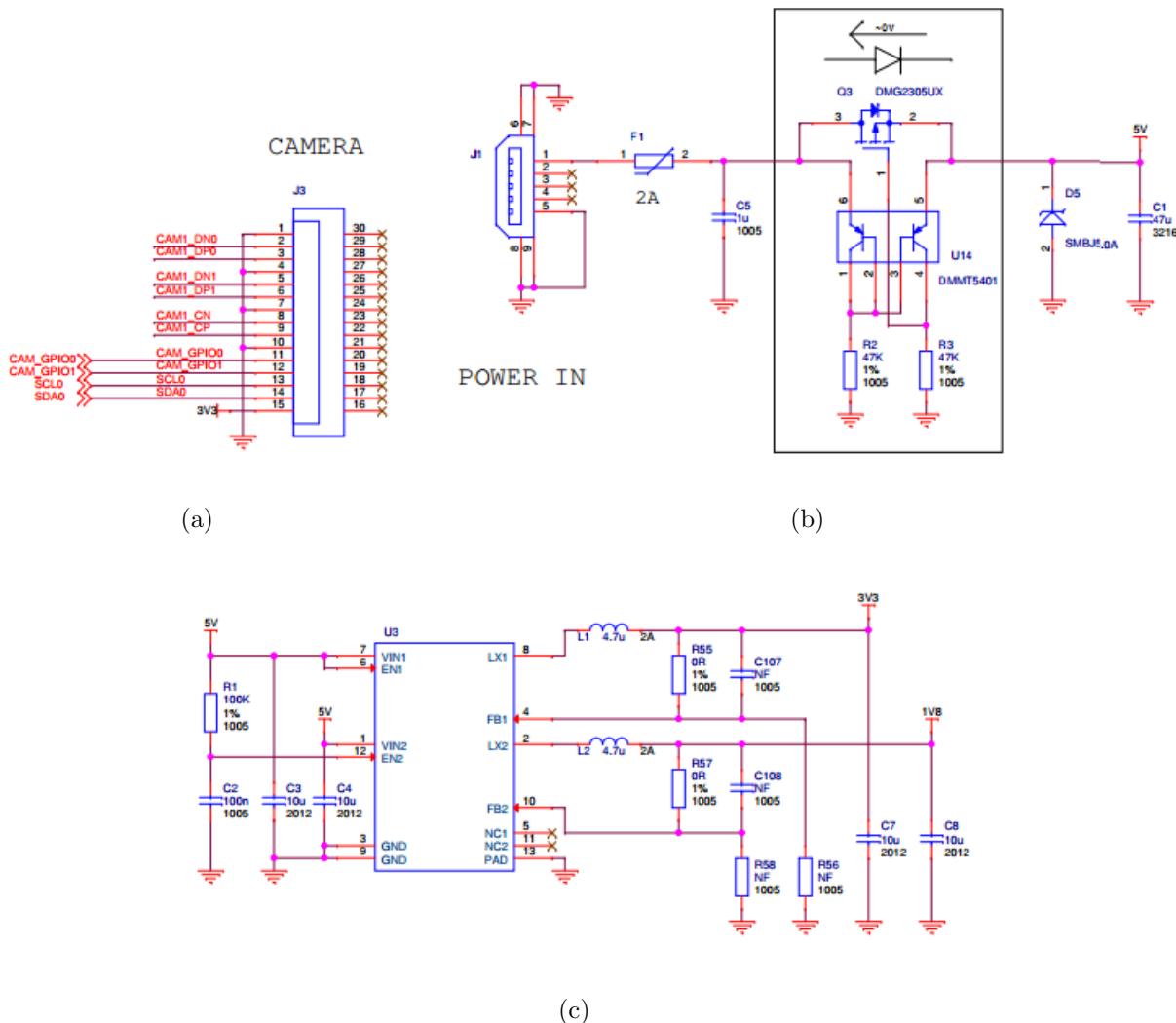


Figura 21: a)DataSheet Raspberry Pi Cam b) c)Esquema reducido de Raspberry Pi modelo B+ v1.2, [24]