ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICO E INSTRUMENTACIÓN

TEMA: “DISEÑO DE UNA APLICACIÓN INTERACTIVA QUE PERMITA ASISTIR AL CONDUCTOR EN EL RECONOCIMIENTO DE SEÑALES DE TRÁNSITO MEDIANTE PROCESAMIENTO DE IMÁGENES UTILIZANDO SOFTWARE LIBRE Y TECNOLOGÍA BEAGLEBONE”

AUTOR: CAIZA ANDRANGO LUIS ERNESTO

DIRECTOR: ING. EDDIE GALARZA Z.

LATACUNGA

2016

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"DISEÑO DE UNA APLICACIÓN INTERACTIVA QUE PERMITA ASISTIR AL CONDUCTOR EN EL RECONOCIMIENTO DE SEÑALES DE TRÁNSITO EN LA VÍA MEDIANTE PROCESAMIENTO DE IMÁGENES UTILIZANDO SOFTWARE LIBRE Y TECNOLOGÍA BEAGLEBONE"** realizado por el señor **LUIS ERNESTO CAIZA ANDRANGO**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor **LUIS ERNESTO CAIZA ANDRANGO** para que lo sustente públicamente.

Latacunga, 10 de marzo del 2016

ING. EDDIE EGBERTO GALARZA ZAMBRANO

DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

Yo, **LUIS ERNESTO CAIZA ANDRANGO**, con cédula de identidad N° 050378652-7, declaro que este trabajo de titulación **"DISEÑO DE UNA APLICACIÓN INTERACTIVA QUE PERMITA ASISTIR AL CONDUCTOR EN EL RECONOCIMIENTO DE SEÑALES DE TRÁNSITO EN LA VÍA, MEDIANTE PROCESAMIENTO DE IMÁGENES, UTILIZANDO SOFTWARE LIBRE Y TECNOLOGÍA BEAGLEBONE"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 10 de marzo del 2016

LUIS ERNESTO CAIZA ANDRANGO

C.C.:050378652-7

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Yo, **LUIS ERNESTO CAIZA ANDRANGO**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"DISEÑO DE UNA APLICACIÓN INTERACTIVA QUE PERMITA ASISTIR AL CONDUCTOR EN EL RECONOCIMIENTO DE SEÑALES DE TRÁNSITO EN LA VÍA, MEDIANTE PROCESAMIENTO DE IMÁGENES, UTILIZANDO SOFTWARE LIBRE Y TECNOLOGÍA BEAGLEBONE"** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 10 de marzo del 2016

LUIS ERNESTO CAIZA ANDRANGO

C.C.:050378652-7

DEDICATORIA

CAIZA ANDRANGO LUIS ERNESTO

Dedico este trabajo a mi madre y hermano, quienes me brindaron el apoyo y fortaleza necesarios para seguir a lo largo de esta carrera y el desarrollar de este proyecto para así concluirlo satisfactoriamente.

También dedico este proyecto al Ingeniero Eddie Galarza que supo guiarme con sus conocimientos y consejos para culminar con el proyecto, y por último dedico este trabajo a todas las personas que de cierto modo colaboraron para cumplir con este objetivo en mi vida

AGRADECIMIENTO

CAIZA ANDRANGO LUIS ERNESTO

Agradezco de una manera muy especial a mi madre María y a mi hermano Edison por su ejemplo de ser personas luchadores que siempre buscan cumplir sus planes y metas trazadas, mi agradecimiento a estas dos personas es por ser más que familia, es porque son mis amigos y de los más importantes en mi vida. A otra persona muy importante mi hermana Silvia.

Agradezco de manera especial al Ingeniero Eddie Galarza por su apoyo, guía y amistad que me brindó a lo largo de este proyecto de investigación desde el inicio hasta su culminación ayudándome a formar académicamente. Gracias también a los maestros que supieron brindar su invaluable conocimiento técnico por medio de sus consejos y experiencias.

ÍNDICE DE CONTENIDOS

PORTADA.....**i**
CERTIFICACIÓN.....ii
AUTORÍA DE RESPONSABILIDAD.....iii
AUTORIZACIÓN.....iv
AGRADECIMIENTO.....vi
ÍNDICE DE CONTENIDOS.....vii
ÍNDICE DE TABLAS.....xi
ÍNDICE DE FIGURAS.....xv
RESUMEN.....xvii
ABSTRACT.....xviii
INTRODUCCIÓN.....xix

CAPÍTULO I

1. PROBLEMA.....1
1.1. Prólogo.....1
1.2. Planteamiento del problema.....1
1.3. Antecedentes.....2
1.4. Justificación e importancia.....3
1.5. Objetivos.....4
1.5.1. Objetivo general.....4
1.5.2. Objetivos específicos.....4
1.6. Hipótesis.....4
1.7. Delimitación.....5

CAPÍTULO II

2. GENERALIDADES.....6
2.1. Señales de tránsito.....6
2.2. Procesamiento de imágenes.....8
2.2.1. Visión artificial.....8
2.2.2. Procesamiento digital de imágenes.....10

a. Tipos de imagen 12
b. Tipos de procesamiento de imágenes 13
c. Algoritmo SURF 15
2.3. Software libre OpenCV 24
2.4. Beaglebone Black card 26
2.5. Cámara para procesamiento de imágenes 28
2.5.1. El sensor 29
2.5.2. Tecnología de los sensores 29
2.5.3. Cámara Logitech HD Pro Webcam C920 30
a. Características 31
2.6. Reconocimiento de objetos 32

CAPÍTULO III

3. DESARROLLO 35
3.1. Lectura de imágenes con OpenCV y C++ 35
3.1.1. Bibliotecas para adquisición de imágenes 36
3.1.2. Variables para lectura de imágenes 37
3.1.3. Presentación de imágenes 42
3.1.4. Ejemplos 43
3.2. Adquisición de imágenes en tiempo real con OpenCV y C++ 46
3.2.1. Inicialización de la cámara 47
3.2.2. Lectura de la captura 48
3.2.3. Presentación de la captura 49
3.2.4. Captura de imágenes con OpenCV 50
3.3. Algoritmo SURF con OpenCV y C++ 53
3.3.1. Bibliotecas de encabezado y namespace 53
3.3.2. Clases utilizadas para detección de objetos 54
a. Detección de puntos característicos (kps) 54
b. Emparejamiento de puntos característicos 56
c. Emparejamiento de puntos característicos que coinciden 57
d. Dibujo de puntos característicos que coinciden 58

3.3.3. Algoritmo para el reconocimiento de señales de tránsito 59
a. Diseño de la base de datos 60
b. Captura de Imágenes y lectura de patrones 60
c. Comparación entre patrones 61
d. Programa para el reconocimiento de señales de tránsito en tiempo real 61
3.4. Implementación de la aplicación utilizando la tarjeta Beaglebone . 64
3.4.1. Características de Software y Hardware 64
3.4.2. Instalación del sistema operativo y las librerías de OpenCV en la tarjeta Beaglebone Black 64
3.4.3. Control de pines GPIO de la tarjeta Beaglebone Black con C++ ... 69
3.4.4. Compilación y ejecución 71
3.4.5. Prototipo para ejecutar aplicación 72

CAPÍTULO IV

4. PRUEBAS Y RESULTADOS EXPERIMENTALES 74
4.1. Determinación del tamaño de la imagen patrón y umbral Hessiano para búsqueda de puntos característicos 74
4.1.1. Detección de puntos característicos (kps) 75
4.2. Captura de video y tiempos de ejecución del algoritmo SURF 78
4.2.1 Pruebas en el computador 81
4.2.2 Pruebas en la tarjeta Beaglebone Black 85
4.3. Consumo de recursos de memoria del algoritmo SURF 87
4.4. Distancia para el reconocimiento de las señales de tránsito de velocidad en la Tarjeta Beaglebone Black 91
4.4.1. Distancias para detección con capturas de 640x480 pixeles 92
4.4.2. Distancias para detección con capturas de 1024x748 pixeles 108
4.4.3. Influencia de la inclinación y rotación del objeto en el reconocimiento 125
4.5. Análisis de los resultados obtenidos en las diferentes pruebas ... 139

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES 152
5.1. Conclusiones 152
5.2. Recomendaciones 154

REFERENCIAS BIBLIOGRÁFICAS 156
ANEXOS 159

ANEXO A: PASOS PARA INSTALAR OPENCV EN EL COMPUTADOR ALGORITMO PARA
ANEXO B: PASOS PARA INSTALAR OPENVC EN LA TARJETA BEAGLEBONE BLACK
ANEXO C: ALGORITMO SURF PARA EL RECONOCIMIENTO DE OBJETOS EN EL COMPUTADOR
ANEXO D: CONTROL DE LOS PUERTOS DE BEAGLEBONE BLACK
ANEXO E: ALGORITMO PARA EL RECONOCIMIENTO DE OBJETOS EN LA TARJETA BEAGLEBONE BLACK

ÍNDICE DE TABLAS

Tabla 1: Valores de puntos caraterísticos para la señal de 50km/h 76
Tabla 2: Valores de puntos caraterísticos para la señal de 80km/h 76
Tabla 3: Valores de puntos caraterísticos para la señal de 100km/h 76
Tabla 4: Valores de puntos caraterísticos para la señal de 50km/h 77
Tabla 5: Valores de puntos caraterísticos para la señal de 80km/h 77
Tabla 6: Valores de puntos caraterísticos para la señal de 100km/h 78
Tabla 7: Tiempos de ejecución (30fps) 83
Tabla 8: Tiempos de ejecución (2fps) 84
Tabla 9: Tiempos de ejecución (1fps) 84
Tabla 10: Tiempos de ejecución RAM saturada 84
Tabla 11: Tiempos de ejecución (30fps) 85
Tabla 12: Tiempos de ejecución (2fps) 86
Tabla 13: Tiempos de ejecución (1fps) 86
Tabla 14: Detección de la señal de 50km/h 93
Tabla 15: Detección de la señal de 80km/h 93
Tabla 16: Detección de la señal de 100km/h 94
Tabla 17: Detección de la señal de 100km/h 94
Tabla 18: Detección de la señal de 80km/h 95
Tabla 19: Detección de la señal de 100km/h 96
Tabla 20: Detección de la señal de 50km/h 96
Tabla 21: Detección de la señal de 80km/h 97
Tabla 22: Detección de la señal de 50km/h 97
Tabla 23: Detección de la señal de 100km/h 98
Tabla 24: Detección de la señal de 800km/h 99
Tabla 25: Detección de la señal de 50km/h 100
Tabla 26: Detección de la señal de 50km/h 101
Tabla 27: Detección de la señal de 80km/h 102
Tabla 28: Detección de la señal de 100km/h 102
Tabla 29: Detección de la señal de 100km/h 103
Tabla 30: Detección de la señal de 80km/h 103

Tabla 31: Detección de la señal de 100km/h 104
Tabla 32: Detección de la señal de 50km/h 104
Tabla 33: Detección de la señal de 80km/h 105
Tabla 34: Detección de la señal de 50km/h 105
Tabla 35: Detección de la señal de 100km/h 106
Tabla 36: Detección de la señal de 80km/h 106
Tabla 37: Detección de la señal de 50km/h 107
Tabla 38: Detección de la señal de 50km/h 108
Tabla 39: Detección de la señal de 80km/h 109
Tabla 40: Detección de la señal de 100km/h 110
Tabla 41: Detección de la señal de 100km/h 110
Tabla 42: Detección de la señal de 80km/h 111
Tabla 43: Detección de la señal de 100km/h 112
Tabla 44: Detección de la señal de 50km/h 113
Tabla 45: Detección de la señal de 80km/h 113
Tabla 46: Detección de la señal de 50km/h 114
Tabla 47: Detección de la señal de 100km/h 115
Tabla 48: Detección de la señal de 80km/h 116
Tabla 49: Detección de la señal de 50km/h 116
Tabla 50: Detección de la señal de 50km/h 117
Tabla 51: Detección de la señal de 80km/h 118
Tabla 52: Detección de la señal de 100km/h 118
Tabla 53: Detección de la señal de 100km/h 119
Tabla 54: Detección de la señal de 80km/h 120
Tabla 55: Detección de la señal de 100km/h 121
Tabla 56: Detección de la señal de 50km/h 121
Tabla 57: Detección de la señal de 80km/h 122
Tabla 58: Detección de la señal de 50km/h 123
Tabla 59: Detección de la señal de 100km/h 123
Tabla 60: Detección de la señal de 80km/h 124
Tabla 61: Detección de la señal de 50km/h 125

Tabla 62: Inclinación de 10° 127
Tabla 63: Inclinación de 20° 127
Tabla 64: Inclinación de 30° 128
Tabla 65: Inclinación de 10° 128
Tabla 66: Inclinación de 20° 128
Tabla 67: Inclinación de 30° 128
Tabla 68: Inclinación de 10° 129
Tabla 69: Inclinación de 20° 129
Tabla 70: Inclinación de 30° 129
Tabla 71: Inclinación de 10° 129
Tabla 72: Inclinación de 20° 130
Tabla 73: Inclinación de 30° 130
Tabla 74: Inclinación de 10° 130
Tabla 75: Inclinación de 20° 130
Tabla 76: Inclinación de 30° 131
Tabla 77: Inclinación de 10° 131
Tabla 78: Inclinación de 20° 131
Tabla 79: Inclinación de 30° 131
Tabla 80: Inclinación de 10° 132
Tabla 81: Inclinación de 20° 132
Tabla 82: Inclinación de 30° 132
Tabla 83: Inclinación de 10° 132
Tabla 84: Inclinación de 20° 133
Tabla 85: Inclinación de 30° 133
Tabla 86: Inclinación de 15° 134
Tabla 87: Inclinación de 30° 134
Tabla 88: Inclinación de 15° 134
Tabla 89: Inclinación de 30° 134
Tabla 90: Inclinación de 15° 135
Tabla 91: Inclinación de 30° 135
Tabla 92: Inclinación de 15° 135

Tabla 93: Inclinación de 30° 135
Tabla 94: Inclinación de 15° 136
Tabla 95: Inclinación de 30° 136
Tabla 96: Inclinación de 15° 136
Tabla 97: Inclinación de 30° 136
Tabla 98: Inclinación de 15° 137
Tabla 99: Inclinación de 30° 137
Tabla 100: Inclinación de 15° 137
Tabla 101: Inclinación de 30° 137
Tabla 102: Captura 640x480 con objeto rotado 139
Tabla 103: Captura 1024x748 con objeto rotado 139
Tabla 104: Tiempos de ejecución computador 140
Tabla 105: Tabla 105 Tiempos de ejecución Beaglebone Black 141
Tabla 106: Detección de la señal de 50 km/h con iluminación baja 142
Tabla 107: Detección de la señal de 80 km/h con iluminación baja 142
Tabla 108: Detección de la señal de 100 km/h con iluminación baja 143
Tabla 109: Detección de la señal de 50 km/h con iluminación alta 143
Tabla 110: Detección de la señal de 80 km/h con iluminación alta 144
Tabla 111: Detección de la señal de 100 km/h con iluminación alta 144
Tabla 112: Detección de la señal de 50 km/h con iluminación baja 145
Tabla 113: Detección de la señal de 80 km/h con iluminación baja 145
Tabla 114: Detección de la señal de 100 km/h con iluminación baja 146
Tabla 115: Detección de la señal de 50 km/h con iluminación alta 146
Tabla 116: Detección de la señal de 80 km/h con iluminación alta 147
Tabla 117: Detección de la señal de 100 km/h con iluminación alta 147

ÍNDICE DE FIGURAS

Figura 1: Señales de tránsito. 7
Figura 2: Representación en diagrama de bloques de la visión artificial. 9
Figura 3: Espectro electromagnético según longitud de onda. 11
Figura 4: Imagen para formar la diferencia de escalas. 17
Figura 5: Derivadas parciales de segundo orden de un filtro gaussiano..... 18
Figura 6: Representación gráfica de las longitudes de los lados del filtro..... 19
Figura 7: Funciones de Haar empleadas en el detector SURF. 20
Figura 8: Asignación de la orientación. 21
Figura 9: Construcción del descriptor 22
Figura 10: Sumatoria para diferentes contrastes 23
Figura 11: Beaglebone Black Card. 28
Figura 12: Cámara Logitech HD Pro Webcam C920. 30
Figura 13: Sistema de reconocimiento. 33
Figura 14: Lectura de una imagen con OpenCV, clase IplImage. 44
Figura 15: Lectura de una imagen utilizando Opencv, clase Mat. 45
Figura 16: Captura en tiempo real en RGB. 51
Figura 17: Captura en tiempo real en escala de grises. 52
Figura 18: Detección de los puntos característicos de una imagen. 56
Figura 19: Emparejamiento de los puntos característicos de una imagen. ... 57
Figura 20: Emparejamiento de puntos característicos que coinciden. 59
Figura 21: Base de datos para reconocimiento de señales de tránsito..... 60
Figura 22: Búsqueda de señal de tránsito por medio del Algoritmo SURF ... 62
Figura 23: Resultado de búsqueda de señal de tránsito por medio del Algoritmo SURF, presenta falsos positivos. 62
Figura 24: Búsqueda de señal de tránsito por medio del Algoritmo SURF. .. 63
Figura 25: Resultado de búsqueda de señal de tránsito por medio del Algoritmo SURF. 63
Figura 26: Prototipo 72
Figura 27; Imagen patrón de velocidad máxima cincuenta Km/h. 75
Figura 28: Captura con resolución 320x240 pixeles 79

Figura 29: Captura con resolución 480x320 pixeles 80
Figura 30: Captura con resolución 640x480 pixeles 80
Figura 31: Captura con resolución 1024x748 pixeles 81
Figura 32: Procesamiento con resolución 640x480 82
Figura 33: Procesamiento con resolución 1024x748 83
Figura 34: Recursos de memoria antes de ejecutar la aplicación con el algoritmo SURF..... 88
Figura 35: Recursos de memoria después de ejecutar la aplicación con el algoritmo SURF..... 89
Figura 36: Recursos utilizados cuando se satura la memoria RAM. 89
Figura 37: Recursos de memoria usados por el algoritmo SURF. 90
Figura 38: Recursos de Memoria en Beaglebone Black 91
Figura 39: Recursos de Memoria en Beaglebone Black al ejecutar la aplicación 91
Figura 40: Señal de tránsito con inclinación horizontal a la derecha 126
Figura 41: Señal de tránsito con inclinación horizontal a la izquierda..... 127
Figura 42: Objeto capturado con inclinación frontal 133
Figura 43: Señal rotada 270° sexagesimales 138
Figura 44: Señal rotada 270° sexagesimales 138
Figura 45: Tendencia del tiempo de ejecución durante el reconocimiento . 140
Figura 46: Tendencia del tiempo de ejecución durante el reconocimiento . 141
Figura 47: Tendencia de reconocimiento de objetos con iluminación baja . 143
Figura 48: Tendencia de reconocimiento de objetos con iluminación alta .. 144
Figura 49: tendencia de reconocimiento de objetos con iluminación baja .. 146
Figura 50: tendencia de reconocimiento de objetos con iluminación alta ... 148
Figura 51: Inclinación lateral del objeto..... 149
Figura 52: Inclinación frontal del objeto 150

RESUMEN

En el presente trabajo se analizará la operación de la tarjeta Beaglebone Black como dispositivo para implementar el algoritmo SURF en conjunto con C++ bajo una distribución de software libre (Linux), con el fin de realizar procesamiento de imágenes en tiempo real. El proyecto de investigación se enfoca en el reconocimiento de señales de tránsito de velocidad en tiempo real por medio de una cámara web de alta velocidad, la tarjeta Beaglebone Black como unidad para el procesamiento de imágenes y un módulo de audio para tener una señal sonora que informe al conductor cuando una señal se haya detectado. El trabajo en si consiste en realizar una aplicación que permita reconocer los objetos (señales de tránsito) tomando en cuenta la distancia, la adquisición de datos (calidad de capturas de imágenes), los factores ambientales y los recursos necesarios en hardware y software de la tarjeta. Las mismas pruebas se realizaron en el computador para de este modo definir rendimientos. El estudio finaliza con tablas de recopilación de datos dónde claramente se presentan los resultados obtenidos en las pruebas (pruebas realizadas sin implementar el sistema en un automóvil) que dependen mucho de la resolución de las imágenes, la distancia, el ambiente y el tiempo de procesamiento al momento de ejecutar la aplicación.

PALABRAS CLAVE:

- **BEAGLEBONE BLACK**
- **PROCESAMIENTO DE IMÁGENES**
- **ALGORITMO SURF**
- **SEÑALES DE TRÁNSITO DE VELOCIDAD**

ABSTRACT

In these work the Beaglebone Black card is analyzed as a device to implement the algorithm SURF in conjunction with the C language under a free software distribution (Linux), in order to perform image processing in real time. The research project focuses on the recognition speed traffic signals in real time using high speed web camera, The Beaglebone Black card is used as the main element for image processing and the system includes an audio module for a beep to report the driver when a traffic signal is detected. The work itself is to make an application that can recognize objects (traffic signs speed) taking into account the distance, data acquisition (quality of scanned images) and the necessary hardware and software resources of the card. The same tests were performed on the computer to thereby define yields. The study finished with data collection tables where we present the results of the tests (tests without deploying the system in a car) that depend of the resolution of the images, the distance, the environment and the processing time are presented when run the application.

KEYWORDS:

- **BEAGLEBONE BLACK**
- **IMAGE PROCESSING**
- **SURF ALGORITHM**
- **SPEED TRAFFIC SIGNALS**

INTRODUCCIÓN

El presente proyecto se ha desarrollado en cinco capítulos, cada uno de estos define de manera puntual los temas tratados y las actividades realizadas, con el fin de dar a conocer de dónde parte el proyecto y hasta dónde finaliza. Se han dividido los capítulos de la siguiente manera:

CAPÍTULO I

Se detalla el proyecto de investigación, planteamiento del problema, antecedentes, justificación e importancia.

CAPÍTULO II

Describe los conceptos generales sobre los temas tratados para comprender y familiarizarse con el tema, para de esta manera tener idea del tema, de cómo funciona la tarjeta Beaglebone Black y el algoritmo SURF para el reconocimiento de las señales de tránsito.

CAPÍTULO III

Describe el desarrollo para la aplicación desde la adquisición de imágenes por medio de la cámara web, el procesamiento de imágenes en la tarjeta Beaglebone Black al implementar el algoritmo SURF como método de reconocimiento de objetos.

CAPÍTULO IV

Presenta los resultados de las pruebas para el reconocimiento de los objetos a distintas distancias, ambientes físicos y resoluciones de captura, y análisis de los datos obtenidos.

CAPÍTULO V

Finalmente se tiene las conclusiones y recomendaciones del proyecto de investigación.

CAPÍTULO I

1. PROBLEMA

1.1. Prólogo

En la actualidad existe un enfoque e interés muy marcado en cuanto a procesar imágenes, con la finalidad de reemplazar a varios sensores convencionales y en lugar de ellos utilizar cámaras cómo dispositivos de entrada. Dentro de varias opciones de procesar imágenes aparece el algoritmo SURF, es un algoritmo poderoso que permite el reconocimiento de objetos a partir de una imagen patrón cargada en la base de datos.

En la actualidad se poseen vías de primer orden en dónde los conductores se movilizan a altas velocidades sobrepasando los límites establecidos, esto sumado al gran número de vehículos, generan que no se cumplan la leyes de tránsito en especial los límites de velocidad, provocando un alto número de infracciones y también un número elevado de accidentes. Por esta razón se ha planteado el diseñar una aplicación mediante procesamiento de imágenes que permita asistir al conductor en la identificación de señales de tránsito de velocidad en la vía.

1.2. Planteamiento del problema

En la actualidad las sanciones y accidentes de tránsito tienen elevado número de incidencias, esto es debido al descuido y la imprudencia de los conductores al no tomar atención las señales de tránsito. Este tipo de situación ha venido presentando polémica en los últimos años puesto que a medida que las vías se amplían y se mejoran, de igual manera la cantidad de accidentes vehiculares van en aumento pese a que las vías presentan señalización de primer orden para facilitar la movilización de sus habitantes.

Un modo de reducir los accidentes fue la implementación de radares para reducir el exceso de velocidad, pero para complementar esto se busca la

manera de alertar al conductor de las diferentes señales que pueden pasar desapercibidas al momento de trasladarse por la vía.

El conductor de un vehículo no siempre puede prestar atención a todas las señalizaciones de tránsito que se presentan en la vía, es por esto que se busca una solución a este problema y que mejor con la utilización de las tecnologías actuales, mediante una aplicación para asistir al conductor, y por ende evitar infracciones o accidentes.

1.3. Antecedentes

El procesamiento de imágenes permite mejorar el aspecto de las imágenes y hacer evidente en ellas ciertos detalles que se deseen hacer notar. Este es un proceso que pretende extraer información visual de una imagen o un conjunto de imágenes para un propósito concreto. El procesamiento de las imágenes se puede en general hacer por medio de métodos ópticos, o bien por medio de métodos digitales, en una computadora.

Los primeros algoritmos de reconocimiento y procesamiento de imágenes fueron basados en técnicas heurísticas y antropométricas. Estos algoritmos tenían una tasa de fallo elevado y muy sensible a los cambios. Mediante investigaciones fueron evolucionando los métodos de reconocimientos de objetos y además las tecnologías evolucionaron y permitieron la implementación de nuevos métodos para el procesamiento de imágenes.

Hoy en día el procesamiento de imágenes para la detección de objetos a través de la captura de una imagen puede llevar desde pocas milésimas hasta varios segundos, según el tamaño de la imagen, la fiabilidad del algoritmo, etc. El interés por hacer interfaces hombre-máquina al momento de conducir es cada vez más inteligentes, que ha propiciado el desarrollo de aplicaciones para asistir a los conductores, como las aplicaciones GPS, de radio frecuencia, de cámaras y muchas más.

A medida que su sector automotriz y las tecnologías avanzan, aumentan las prestaciones de los vehículos, para de esta manera brindar mayor confort, eficiencia y seguridad de un automotor. Pero no todo es beneficio, porque los conductores al momento de manejar el vehículo exceden límites de velocidad, no prestan atención a la señalización de la vía e incluso quebrantan la ley de tránsito, teniendo como resultado accidentes en las vías.

A esto se suma el gran número de vehículos que existen en la actualidad y con el pasar del tiempo, al ver que existen conductores imprudentes, se ve la necesidad de crear señales visuales en las vías, las mismas que pueden estar ubicadas de forma horizontal y vertical, dependiendo de lo que se quiera dar a conocer al conductor, todo este tipo de señalización se la ha determinado con el objetivo de dar seguridad a los conductores como a los peatones dentro y fuera de la ciudad.

1.4. Justificación e importancia

Con el pasar de los años y el avance de nuevas tecnologías, el procesamiento de imágenes se ha convertido en un tema de gran importancia y aplicaciones para el beneficio de la sociedad, los campos de aplicación son la industria, la seguridad, etc.

El proyecto sirve como base para el desarrollo de proyectos futuros que se relacionen con aplicaciones para asistir a conductores vehiculares, para que ayuden a solucionar problemas como no estar atento en la vía, no prestar atención a las señales de tránsito y ser imprudente, lo mismo que ocasiona sanciones y accidentes de tránsito.

La aplicación notificará al conductor cuando en la vía se presentan señales de tránsito, para de esta manera alertarlo y hacer que preste más atención en la vía. Brindando así una ayuda al conductor y también disminuyendo el riesgo de un imprevisto que pueda presentarse.

1.5. Objetivos

1.5.1. Objetivo general

Diseñar de una aplicación interactiva que permita asistir al conductor en el reconocimiento de señales de tránsito en la vía, mediante procesamiento de imágenes, utilizando software libre y tecnología BEAGLEBONE.

1.5.2. Objetivos específicos

- Investigar sobre el procesamiento de imágenes, los diferentes métodos de procesamiento, algoritmos, y aplicaciones.
- Investigar sobre el procesamiento de imágenes en OpenCV.
- Estudiar el algoritmo SURF para el procesamiento de imágenes.
- Estudiar sobre el funcionamiento de la tarjeta Beaglebone Black como unidad de procesamiento de imágenes y cómo tarjeta de adquisición de datos.
- Obtener en tiempo real imágenes por medio de la cámara con ayuda de OpenCV, para procesarlas mediante el algoritmo SURF.
- Realizar pruebas con la aplicación implementada.
- Determinar cuáles son las ventajas, desventajas y problemas al utilizar la tarjeta Beaglebone Black como sistema embebido.
- Determinar soluciones con las que se podría mejorar el rendimiento de la aplicación.

1.6. Hipótesis

El Diseñar una aplicación interactiva para el reconocimiento de señales de velocidad de tránsito, mediante procesamiento de imágenes, utilizando software libre y la tecnología Beaglebone Black, será factible implementarla en los vehículos y permitirá que el conductor preste atención en la vía al momento de conducir.

1.7. Delimitación

Analizar la eficiencia y rendimiento de hardware y software que presenta la tarjeta Beaglebone Black al implementar una aplicación basada en el algoritmo SURF, mediante pruebas bajo distintos factores que se pueden presentar en los ambientes dónde se pretende utilizar la aplicación, de esta manera determinar si es conveniente la construcción de un prototipo y su implementación en los vehículos.

CAPÍTULO II

2. GENERALIDADES

En el presente capítulo se analizan los conceptos como son: las señales de tránsito, procesamiento de imágenes dentro de la visión artificial, la librería OpenCV, el algoritmo SURF y la tecnología Beaglebone Black. Se tratarán los mencionados temas en forma puntal ya que se pretende implementar una aplicación que permita la identificación de señales de tránsito de velocidad en tiempo real en la tarjeta Beaglebone Black utilizando OpenCV y el algoritmo SURF.

2.1. Señales de tránsito

Las señales de tránsito son “imágenes, letras y números de diferentes colores, tamaños y formas” que se ubican en carteles en la vía pública, cuya simbología tienen un significado para alertar al peatón o conductor a tomar “precauciones” o alertar de las situaciones que se dan en la carretera y vía pública. (Instituto Ecuatoriano de Normalización, 2011)

El objetivo y función de las señales de tránsito como se indica en la figura 1, es contribuir con el orden en las carreteras y vías para de este modo evitar infracciones por parte de los peatones y conductores, además reducir a lo mínimo los accidentes. Se pueden clasificar en “tres grupos principales” las señales de tránsito. (Ministerio de Educación Presidencia de la Nación)

- **Informativas**

Las “señales informativas” tienen como finalidad servir de guía y encaminar al conductor en la vía sobre sitios de interés, de esta manera las señales de tránsito para el conductor se convierte en objetos muy útiles para su normal traslado y arribo al destino. Son “carteles de forma rectangular, de fondo verde y letras blancas o fondo azul y letras blancas”. (Vasquez. Arthur, 2014)

- **Preventivas**

Las “señales preventivas” advertir a los conductores de la “presencia de riesgos en la vía”, siendo los peligros por causa de la geografía y de condiciones naturales. Los carteles tienen “forma de rombo, de fondo amarillo y los símbolos de color negro”, por lo general son las que con más frecuencia se las localiza en las carreteras. (Vasquez. Arthur, 2014)

- **Prohibitivas y Reguladores**

Son utilizadas para “alertar” a los conductores sobre la “existencia de limitaciones, prohibiciones y restricciones” en la carretera y vía pública. Los carteles tienen “forma rectangular con fondo blanco y letras en color rojo y negro”. Las señales de tránsito se alinean de manera vertical u horizontal, las señales que están montadas en placas y se colocan en postes son de señalización vertical y las señales de palabras, símbolos y objetos que se sitúan sobre el pavimento son de señalización horizontal. (Vasquez. Arthur, 2014)

Figura 1: Señales de tránsito.

Fuente: (Comisión de tránsito, 2011)

El presente proyecto se centrará la atención en las señales de tránsito de velocidad ya que los conductores por lo general no prestan cuidado a los límites de velocidad que se tiene establecidos por ley, de esta manera incrementan los riesgos en la carretera no solo para el conductores, sino también para el peatón, y llegando incluso a accidentes mortales.

2.2. Procesamiento de imágenes

El “procesamiento de imágenes permite realizar el análisis e interpretación de una imagen”, este es un campo importante en lo que corresponde a la visión artificial, haciendo posible que un ordenador sea capaz de procesar capturas de imágenes bidimensionales o tridimensionales como también fotogramas de vídeo, para someterlos a un procesado de extracción de información de determinadas características o parámetros, o bien para elaborar nuevas imágenes procesadas como salida. “El PDI tiene la finalidad de procesar, buscar, comprender e interpretar las características de una imagen con un objetivo claro y concreto.” (Jose Esqueda, 2005)

2.2.1. Visión artificial

La “visión artificial abarca de una manera más genérica lo que es el PDI, y con la aplicación de procesos apropiados”, permite obtener, procesar y analizar información de cualquier tipo que sea proveniente de imágenes digitales, facilitando así la implementación de diversas aplicaciones. (Prof. Dr. Nicolás L)

La visión artificial es un conjunto de varios procesos que se realizan con el fin de analizar imágenes como se muestra en la figura 2. Los Procesos son; captura de imágenes o fotogramas, almacenar la información que se ha obtenido durante el procesado y el análisis e interpretación del resultado después del proceso. (Visión Artificial)

La visión artificial permite la “identificación de objetos, determinar posicionamiento en el espacio” mediante la determinación coordenadas de uno o varios objetos en un espacio determinado, mediciones y posicionamientos

bidimensionales y tridimensionales. Permite también la automatización de procesos repetitivos, para el “control de calidad, analizar e inspeccionar objetos sin la necesidad del contacto físico, etc”. (Visión Artificial)

En la visión artificial la imagen es capturada en forma digital y se debe tomar en cuenta los aspectos como; “contraste, brillo, resolución, color”, con que se trabaja, el “nivel en escala de grises”, etc. En este proceso para captura, procesamiento y análisis de imágenes se “presentan varios bloques que conforman en si el sistema completo”. (Visión Artificial)

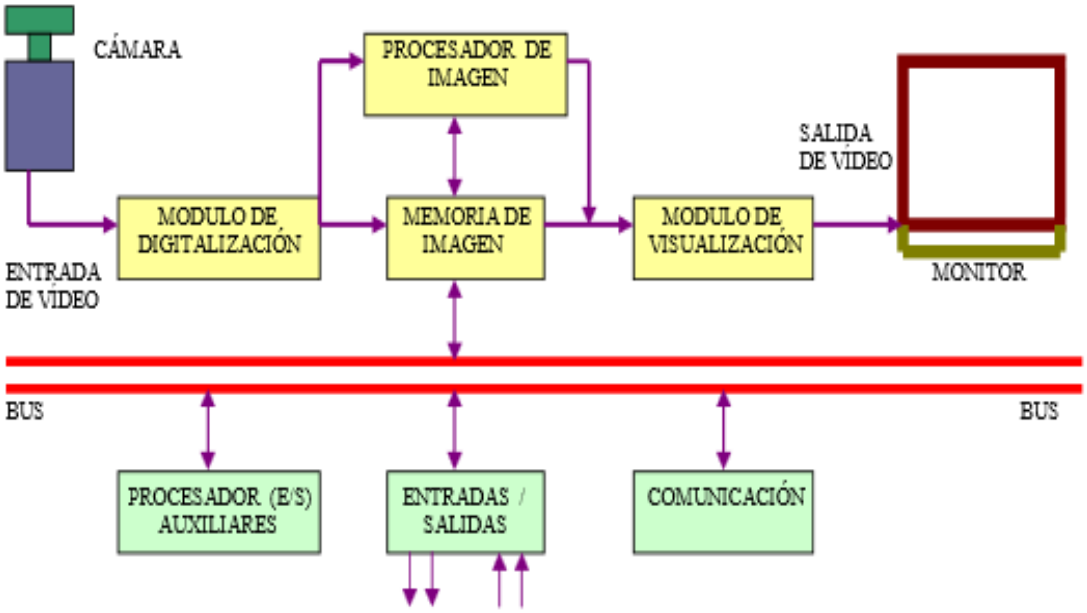

Figura 2: Representación en diagrama de bloques de la visión artificial.

Fuente: (Visión Artificial)

Según el artículo (Visión Artificial) se presenta a continuación cada uno de los bloques y las etapas que comprende un sistema de visión artificial tal como se presentó en la figura 2.

- Módulo de digitalización. Convierte la señal proveída por la cámara que es analógica a una señal digital.
- Memoria de imagen. Se encarga de almacenar la señal digital del módulo anterior.

- Módulo de visualización. Permite la presentación de la información en un monitor, para esto se debe realizar una conversión de señal digital a señal analógica para poder ser presentada.
- Procesador de imagen. Etapa en la que se realiza el procesamiento dependiendo de la aplicación.
- Módulo de entradas y salidas. El módulo de entrada es responsable de la inicialización y sincronización para capturar la imagen. Mientras que el módulo de salida se encarga del funcionamiento de los dispositivos externos que se controlaran en un sistema.
- Comunicaciones. Se realiza por las entradas y salidas, Ethernet, RS232, etc.

2.2.2. Procesamiento digital de imágenes

El “Procesamiento Digital de Imágenes”, se realiza por medios electrónicos y su aplicación “incluye diversas áreas como, la medicina, la manufactura, microscopía, análisis de imágenes satelitales, sistemas de seguridad, astronomía, etc.” Dentro del procesamiento de imágenes se puede obtener la información de diversas fuentes para su procesamiento. (Visión Artificial)

El “proceso consiste en trabajar sobre la señal de información de una imagen en dos dimensiones, posterior se aplican técnicas estándar para realizar el procesamiento en una dimensión”, dentro de las técnicas más comunes se encuentran las transformaciones geométricas que permiten realizar el reajuste, rotación, ampliación, etc., “también se tiene las técnicas de corrección de color para el contraste, brillo y nitidez, y otras técnicas como la alineación, división, intercalación, identificación de patrones, etc”. (Mendoza Dario, 2012)

Un aspecto relevante de PDI, es que la información (imágenes) con que se trabaja “no proviene únicamente de capturas de luz visible que corresponde al ojo humano, sino que las imágenes pueden provenir de varias secciones del espectro electromagnético.” Esto representa una gran ventaja, ya que los sistemas que incorporan la fase de PDI no están limitados únicamente a las

imágenes que puedan ser capturadas por el ojo humano. (Mendoza Dario, 2012)

En la actualidad los sistemas que realizan procesamiento digital de imágenes realizan la adquisición e interpretación de información de gran parte del espectro electromagnético, este espectro se puede visualizar en la figura 3. "El rango de la luz visible es solo una pequeña parte del espectro electromagnético, convirtiéndole al PDI en una herramienta de trabajo." (Mendoza Dario, 2012)

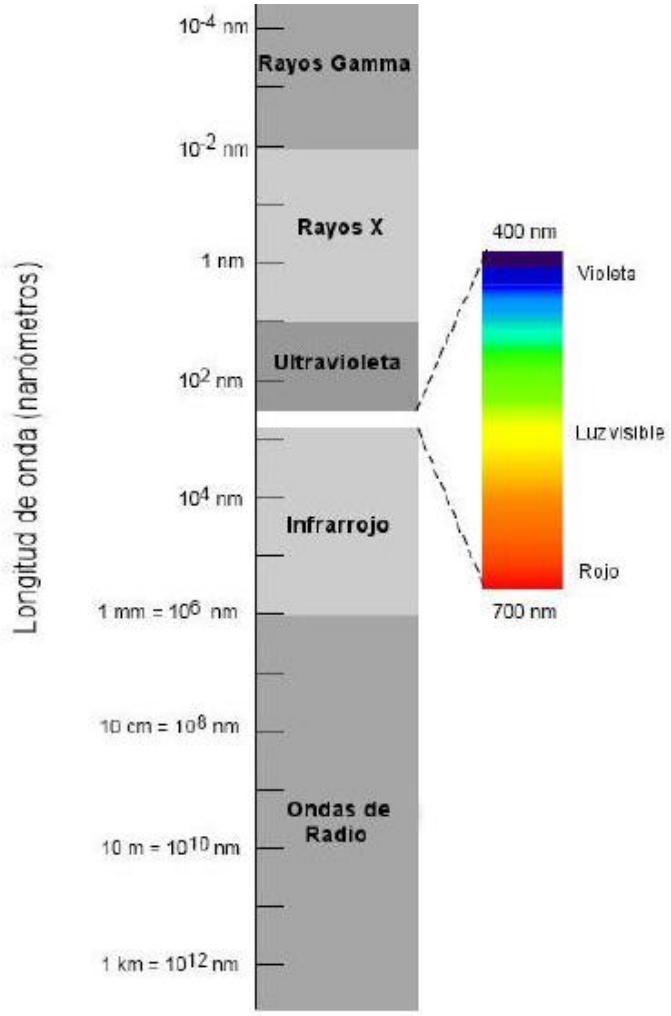

Figura 3: Espectro electromagnético según longitud de onda.

Fuente: (Mendoza Dario, 2012)

a. Tipos de imagen

- **Imágenes de tipo RGB**

Este tipo de imágenes se representan por un arreglo de pixeles que contienen en cada elemento un valor para “el color rojo, verde y azul, por lo tanto la imagen es una matriz de tipo 3xMxN” y de esta manera se tiene una imagen a color, este modelo es el más utilizado ya que se ve presente en monitores, celulares, paneles táctiles, etc. (Gonzales. G)

En este modelo se puede decir que “se tienen tres diferentes imágenes una para cada color y a una intensidad definida” para que cuando sea presentada por un monitor el ojo humano pueda percibirla como una imagen a color, la combinación de estos tres colores a diferentes intensidades de luminosidad producen el resto de colores. (Gonzales. G)

- **Imágenes tipo escala de intensidades**

Este tipo de imagen es la conocida “escala de grises o escala monocromática”, este tipo de imágenes son “arreglos matriciales de una componente en cada pixel con un distinto nivel de luminosidad”, dentro del procesamiento de imágenes se utiliza este tipo de imágenes cuando no se pretende obtener mucha información. (Dra. Flores Leticia)

La escala de intensidades se caracteriza por medio del matiz o tono que define el color, pudiendo variar en esta los valores de luminosidad y saturación. Para realizar la conversión de una imagen a color en una imagen a escala de grises se puede hacer por varios métodos pero siempre con el fin de mantener la información y eliminar los niveles de tono y saturación. Conociendo que una imagen RBG tiene valores de intensidad luminosa para los colores, rojo, verde y azul, “la aproximación para la conversión de la imagen está dada por la ecuación GRAY”, donde el valor de “menor intensidad luminosa está dado por cero” y es el “color negro” y el valor “de mayor intensidad luminosa está dado por uno” y es el” color blanco”. (Gonzales. G)

$$GRAY = (0.30 * R) + (0.59 * G) + (0.11 * B) \qquad (1)$$

La ecuación define las intensidades de luminosidad de los colores rojo, verde y azul, en valor de gray comprendido entre uno y cero se multiplica por la intensidad de cada color para mantener la información de la imagen.

Según el documento (Gonzales. G) se tiene que:
0.30*R = la intensidad del color rojo
0.59*G = intensidad de color verde
0.11*B = intensidad del color azul

- **Imágenes tipo indexadas**

Este tipo de imágenes es “una manera práctica de representar las imágenes a color, pero presenta algunas limitaciones”. Con este tipo de imágenes lo que se hace es “almacenar la información de la imagen en dos matrices” diferentes. La “primera matriz contiene la información en lo que respecta al tamaño” con el número de pixeles, mientras que “la segunda matriz corresponde al mapa de colores que se utilizarán en la imagen” y el “tamaño de esta depende del número de colores que tendrá la nueva imagen”. (Dra. Flores Leticia)

Este tipo de imagen “se utiliza para ahorrar memoria, porque comprime la imagen y con esto logra disminuir el tiempo de procesamiento”. En el proceso, lo que se hace es pasar de un arreglo RBG que contiene la información de las tres componentes por pixel a una matriz de una sola componente por pixel conjuntamente con otra matriz que indica el mapa de colores de la imagen obtenida. (Dra. Flores Leticia)

b. Tipos de procesamiento de imágenes

- **Procesos de imágenes a bajo nivel**

Este tipo de procesamiento no va más allá que realizar en las imágenes la “disminución de ruido”, elevar o “disminuir contraste” y aplicación de filtros de enfoque, estos procesos no necesitan de muchos recursos para realizarlos.

Este proceso se caracteriza por tener que trabajar en las “entradas y salidas únicamente con imágenes” y no otra información. (Cruz, 2013)

- **Procesos de imágenes a nivel medio**

En este tipo de procesamiento las tareas que se realizan son; segmentación, descripción de los objetos mediante el procesamiento en la computadora y clasificación de objetos. “Este tipo de procesamiento es caracterizado porque en las entradas por lo general son imágenes”, pero a “las salidas se tiene atributos extraídos de estas entradas”, esos atributos pueden ser; bordes, contornos y la identificación de algún objeto. (Cruz, 2013)

- **Proceso de imágenes a alto nivel**

En el procesamiento de alto nivel se implica la obtención de algún significado de un conjunto de objetos que son “reconocidos y analizados”, para finalmente realizar las “funciones cognitivas asociadas con la vista”. Es decir un ordenador puede “discriminar entre uno o varios objetos en un contorno”, además se pueden someter a entrenamientos. (Cruz, 2013)

La caracterización de este procesamiento es que por medio de la visión artificial puede determinar características y extraer puntos de interés, descriptores y aplicar algunas técnicas de reconocimiento de objetos, con la visión artificial no se pretende únicamente variar propiedades de la imagen como brillo, contraste, color, tamaño, etc., sino que pretende extraer información detallada de un escenario, la misma que permite la comparación con el patrón y de este modo emular la inteligencia humana para el reconocimiento de cosas.

Dentro del presente proyecto se puede indicar que se utilizará un procesamiento de alto nivel debido a que se empleará el método SURF, este es un algoritmo capaz de extraer información específica y detallada de una imagen, esta información sirve para realizar el reconocimiento de objetos, rostros en escenas bidimensionales y para el posicionamiento, seguimiento y

fijación de trayectorias de un objeto en escenas tridimensionales. SURF es un algoritmo de inteligencia artificial que puede ser entrenado para la interpretación de imágenes y determinar el contenido de estas.

c. Algoritmo SURF

Desarrollado por Herbert Bay, como un detector y descriptor de puntos de interés para el reconocimiento de objetos. Se considera una evolución del algoritmo SIFT. Siendo una de sus ventajas sobre otros métodos, el mejoramiento de la velocidad de cálculo y la robustez ante transformaciones de las imágenes. Estas mejoras se consiguen mediante la reducción de la dimensión y complejidad en el cálculo de los vectores de características de los puntos de interés obtenidos, mientras continúan siendo suficientemente particulares e igualmente repetitivos. [9]. El algoritmo SURF consta de tres etapas y se muestran a continuación de manera detallada.

- **Detección de puntos de interés**

El algoritmo “SURF” utiliza un detector de tipo BLOB el mismo que permite tener una base de datos de imágenes, audio y archivos multimedia. De este modo se permite leer y escribir de manera binaria la información que se dispone en la base. Una desventaja de utilizar el detector BLOB es que no todos los sistemas permiten gestionar bases de datos de ese tipo. (Anqi Xu, 2008)

En este primer paso se hace la detección de puntos de interés denominados keypoints, este proceso está basado en la matriz Hessiana mediante una aproximación muy básica de la misma. Sabiendo que la matriz Hessiana es un arreglo MxN de derivadas parciales de segundo orden de una función evaluada con un escalar o un campo escalar sirve para determinar si la función es convexa o cóncava para de este modo conocer los valores mínimos y máximos de los puntos críticos por medio del determinante. (Herbeert B., Andreas E., Tunne T., & Luc V.,l, 2008)

El motivo de usar la matriz Hessiana se encuentra en su buen rendimiento en cuanto a la velocidad de cálculo y la precisión. Sin embargo, este detector, al contrario del método empleado por otros detectores, en lugar de usar una medida diferente para elegir la posición como la escala, lo que hace es emplear el determinante de la matriz Hessiana en ambos casos. Por tanto, si se tiene un punto $p = (x, y)$ de la imagen I, se determina la matriz Hessiana $H(x, \sigma)$ para el punto p a una escala σ está definida por la ecuación (2):

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix} \qquad (2)$$

Donde $L_{xx}(\rho, \sigma)$ es la convolución de la derivada parcial de segundo orden de la Gaussiana $\frac{\theta^2}{\theta x^2} g(\sigma)$ con la imagen $I(x, y)$ en el punto p. Lo mismo ocurre para $L_{xy}(p, \sigma)$ y $L_{yy}(p, \sigma)$. Para aumentar la velocidad en el cálculo del espacio, el descriptor SURF utiliza una aproximación a los filtros Gaussianos, conocidos como los filtros de caja. Estos filtros aproximan los valores de las derivadas parciales de segundo orden y pueden ser calculados de manera muy eficiente utilizando imágenes integrales. Donde una imagen integral $I_\Sigma(p)$ de un punto $p = (x, y)^T$ representa la suma de todos los píxeles del rectángulo dentro de la imagen $I(x, y)$ formado por el origen y el punto p.

$$I_\Sigma(p) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j) \qquad (3)$$

A pesar de que los filtros Gaussianos son óptimos para el análisis del espacio en cada escala, también presentan una serie de limitaciones como la necesidad de ser discretizados, el efecto aliasing, etc. Se ha probado en el detector SURF una alternativa a los filtros gaussianos*:* los filtros de caja que se visualizan en la figura 4, indican que; un "filtro de caja a más de aumentar la velocidad permite calcular y re-escalar la imagen de forma progresiva mediante la implementación de un suavizado a la imagen para de esta manera disminuir el ruido mediante una ventana traslucida", obteniendo de esta manera un escenario previo para el procesamiento y mejorando la imagen en diferentes

escalas, la ventaja de usar este tipo de filtros es que una imagen en una caja es más fácil de integrar. (Herbeert B., Andreas E., Tunne T., & Luc V.,l, 2008)

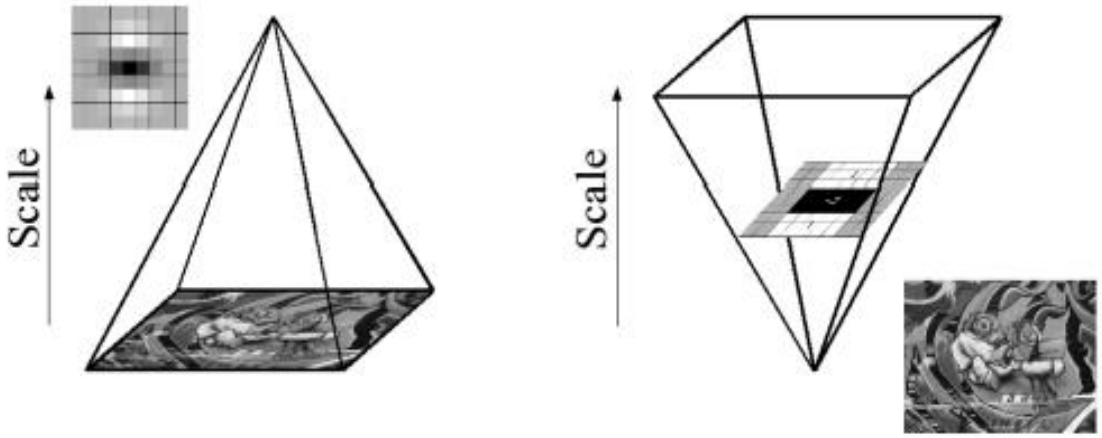

Figura 4: (Izquierda) Algoritmo SIFT, reduce el tamaño de la imagen para formar la diferencia de escalas. (Derecha) Algoritmo SURF aumenta el tamaño del filtro para formar la diferencia de escalas.

Fuente: (Herbeert B., Andreas E., Tunne T., & Luc V.,l, 2008)

Las aproximaciones de las *derivadas parciales* se denotan como $Dxx, Dxy\ y\ Dyy$. En cuanto al determinante de la matriz Hessiana, este queda definido de la siguiente manera:

$$Det(Haprox) = D_{xx}D_{yy} - \left(\omega D_{xy}\right)^2 \qquad (4)$$

Los espacios de escala son a menudo implementados como pirámides de imágenes, en las que estas son suavizadas repetidamente con un filtro Gaussiano y posteriormente submuestreadas para alcanzar un nivel más alto en la pirámide. En la Figura 5 se observa cómo se va aplicando el filtro Gaussiano, primero aplicando el suavizado y luego aproximando a un valor la vecindad que ha sido filtrada. El fin de utilizar el filtro es fijar a un solo valor los valores que componen una región en donde se localiza el punto característico, para de este modo obtener ya un valor promedio de toda la región. (Pederson J., 2011)

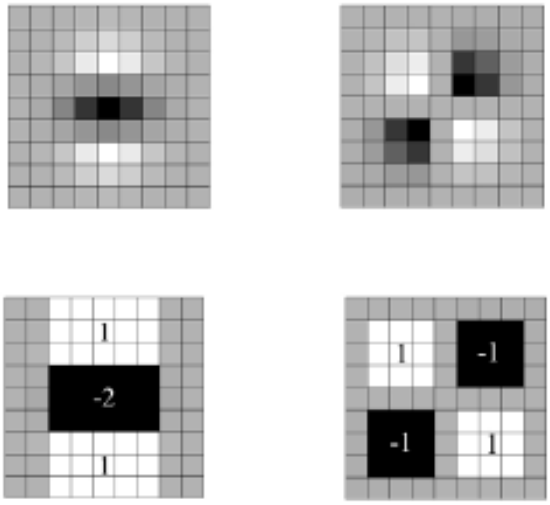

Figura 5: Derivadas parciales de segundo orden de un filtro gaussiano. Aproximación utilizada en SURF.
Fuente: (Pederson J., 2011)

El detector SURF, debido al uso de filtros de caja e imágenes integrales, no se tiene que aplicar iterativamente el mismo filtro a la salida de una capa filtrada previamente, sino que se pueden aplicar dichos filtros de cualquier tamaño a la misma velocidad directamente en la imagen original. De este modo el espacio escala es analizado por medio de ir elevando el tamaño del filtro, en vez de ir reduciendo el tamaño de la imagen. La salida obtenida de aplicar el filtro de dimensión 9x9 es la considerada como escala inicial (s = 1.2, correspondiente a una Gaussiana con σ = 1.2). Las sucesivas capas se van obteniendo como consecuencia de aplicar gradualmente filtros mayores. La figura 6 muestra como el filtro varía en cada una de las octavas incrementando el tamaño del mismo dando como resultado doblar la octava anterior:

Octava inicial: $9x9 \xrightarrow{6} 15x15 \xrightarrow{6} 21x21 \xrightarrow{6} 27x27$

Siguiente octava: $15x15 \xrightarrow{12} 27x27 \xrightarrow{12} 9x39 \xrightarrow{12} 51x51$

Siguiente octava: $27x27 \xrightarrow{24} 51x51 \xrightarrow{24} 75x75 \xrightarrow{24} 99x99$

Y así sucesivamente.

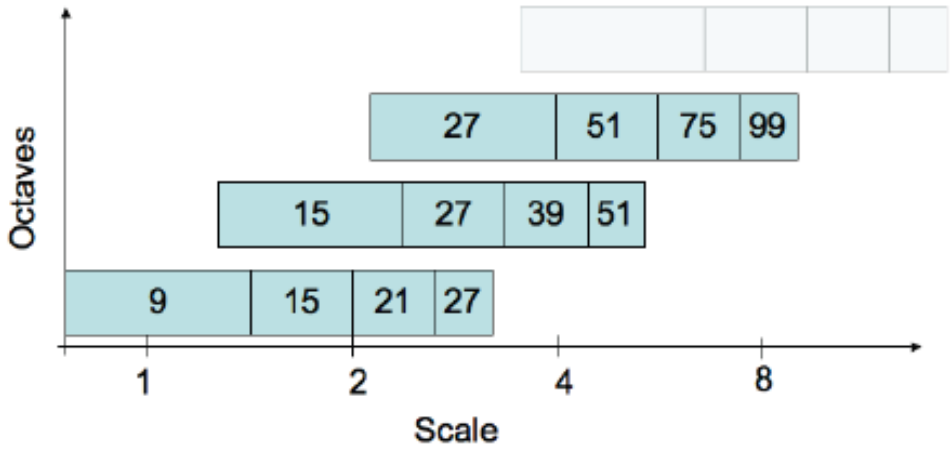

Figura 6: Representación gráfica de las longitudes de los lados del filtro de tres diferente octavas. El eje horizontal representa la escala logarítmica.
Tenga en cuenta que las octavas se superponen con el fin de cubrir todas las posibles escalas.
Fuente: (Jacob T, 2011)

Por último, una vez que se ha calculado el determinante de la aproximación de la matriz Hessiana se obtienen las localizaciones y escalas de los puntos de interés. Los puntos de interés serán los que cumplan la condición de máximo valor de la aproximación con respecto a sus ocho píxeles vecinos, sus nueve píxeles vecinos de una escala mayor y sus nueve píxeles vecinos de una escala menor, interpolando su valor con el de los píxeles cercanos para mejorar su exactitud. Los puntos de interés en una imagen se los debe de localizar a diferentes escalas, es por este motivo que se le aplica la forma de pirámide a la imagen, para así dar por concluida la etapa de detección de puntos. (Oscar B. Gracia, 2011)

- **Asignación de la orientación**

La orientación es un paso previo a la obtención del descriptor, para darle robustez e invarianza ante la rotación, luminosidad y orientación de la imagen

otorgándole a cada punto de interés una única orientación y un único descriptor. Para esto, primero se debe calcular la respuesta de la dirección *x* e *y* como se ve en la figura 7, que se la hace por medio de la respuesta Haar, esta respuesta no es más que el reajuste de funciones de forma cuadrada, en un radio 6s alrededor del punto de interés, formando de esta manera una vecindad sobre el punto de interés, donde *s* es la escala del punto de interés detectado.

La etapa de muestreo también depende de la escala y se toma como valor *s* como referencia, a mayor valor de escala, mayor es la dimensión de las respuestas. Una vez hecho esto, se usan nuevamente imágenes integrales para un filtrado rápido. Para obtener la respuesta en la dirección $x\ o\ y$ se necesita únicamente realizar una segmentación en seis partes a la vecindad de radio 6s. (Herbeert B., Andreas E., Tunne T., & Luc V.,l, 2008)

Figura 7: Funciones de Haar empleadas en el detector SURF.
Fuente: (Cruz, 2013)

Una vez calculadas las respuestas de Haar, éstas son ponderadas por una gaussiana centrada en el punto de interés, para dar mayor importancia a las respuestas más cercanas al punto. Por último, se divide el área circular alrededor del punto de interés en 6 secciones, cubriendo un ángulo de $\frac{\pi}{3}$ cada una, las áreas divididas como lo muestra la figura 8 tienen un objetivo, y es determinar la orientación dominante y la dirección en la que están las mayores características del punto, esta es una de las partes más importantes ya que de esta depende que existan o no falsos positivos cuando se hace el

emparejamiento de puntos de la práctica. Sumando todas las direcciones dentro de cada sección se obtiene un vector de mayor longitud. (Cruz, 2013)

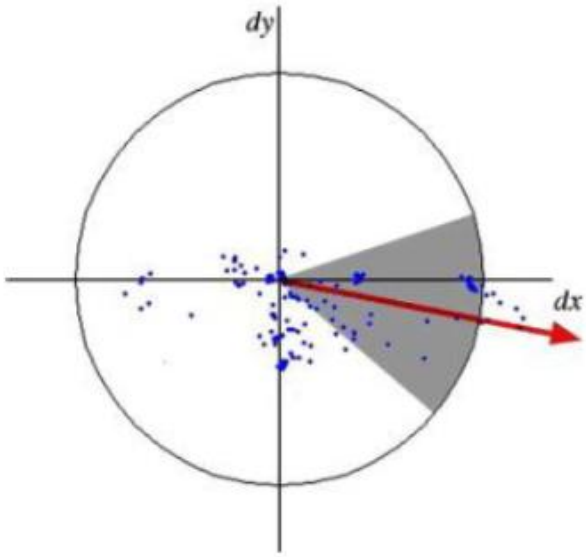

Figura 8: Asignación de la orientación de cada sector del área circular alrededor del punto, cubriendo un ángulo de π/3.
Fuente: (Cruz, 2013)

- **Extracción de los descriptores**

En este paso lo primero es realizar la construcción de una región cuadrada alrededor del punto de interés y orientarla en relación a la dirección obtenida en el paso anterior. El tamaño de la región cuadrada es 20s. Entonces, la región es reducida progresivamente en regiones más pequeñas 4x4. Para cada nueva subregión se calculan características en puntos de muestra separados por una región 5x5. (Oscar B. Gracia, 2011)

La respuesta de Haar en la dirección horizontal es llamada dx, mientras que en la dirección vertical es llamada dy (la dirección horizontal y vertical se definen en función de la orientación del punto de interés seleccionado). Para darle mayor robustez ante deformaciones geométricas y errores de posicionamiento, las respuestas dx y dy son ponderadas con una Gaussiana

de $\sigma = 3{,}3s$ centrada en el punto de interés. En la figura 9 se observa la sumatoria de una subregión donde se encuentra en punto característico. (Herbeert B., Andreas E., Tunne T., & Luc V.,l, 2008)

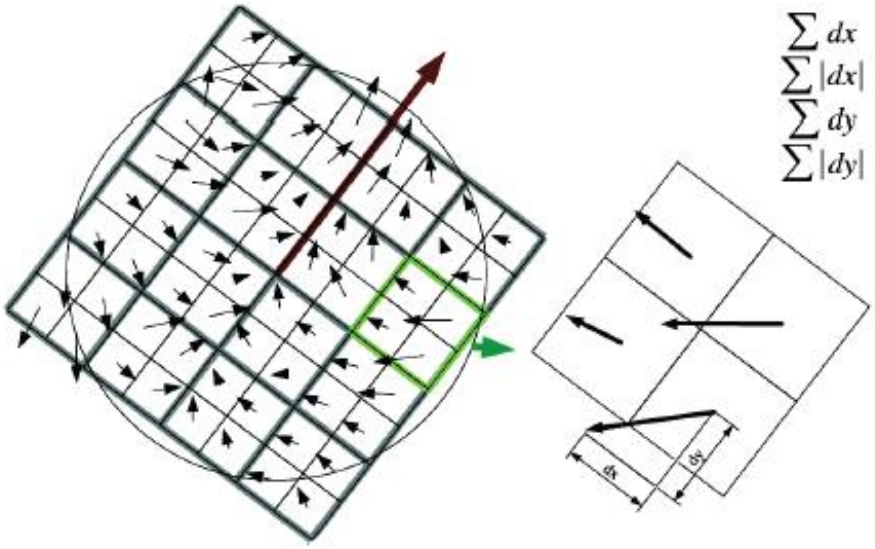

Figura 9: Para construir el descriptor, una red cuadrática orientada con 4x4 subregiones cuadradas se colocan sobre el punto de interés (a la izquierda). Para cada cuadrado, las respuestas wavelet se calculan. Las subdivisiones 2x2 de cada cuadrado corresponden a los campos reales del descriptor. Estas son las sumas $\boldsymbol{dx}, \boldsymbol{dy}, |\boldsymbol{dx}|$ y $|\boldsymbol{dy}|$, calculadas con relación a la orientación de la rejilla.

Fuente: (Herbeert B., Andreas E., Tunne T., & Luc V.,l, 2008)

A continuación, se suman las respuestas de cada subregión. También, se suman los valores absolutos de las respuestas $|dx|$ y $|dy|$ de cada subregión para obtener información sobre los cambios de intensidad. Cada subregión se representa por el vector resultante de la integral con la siguiente estructura:

$$v = (\Sigma dx. \Sigma dy. \Sigma|dx|, \Sigma|dy|) \qquad (5)$$

Por lo que, se obtiene el descriptor SURF de sesenta y cuatro elementos dado por las cuatro dimensiones del vector de las subregiones. El resultado de las

diferentes áreas o contrastes se aprecia de mejor manera en la figura 10 que indica la sumatoria en los diferentes ejes, y de este modo se puede obtener el valor del descriptor. (Cruz, 2013)

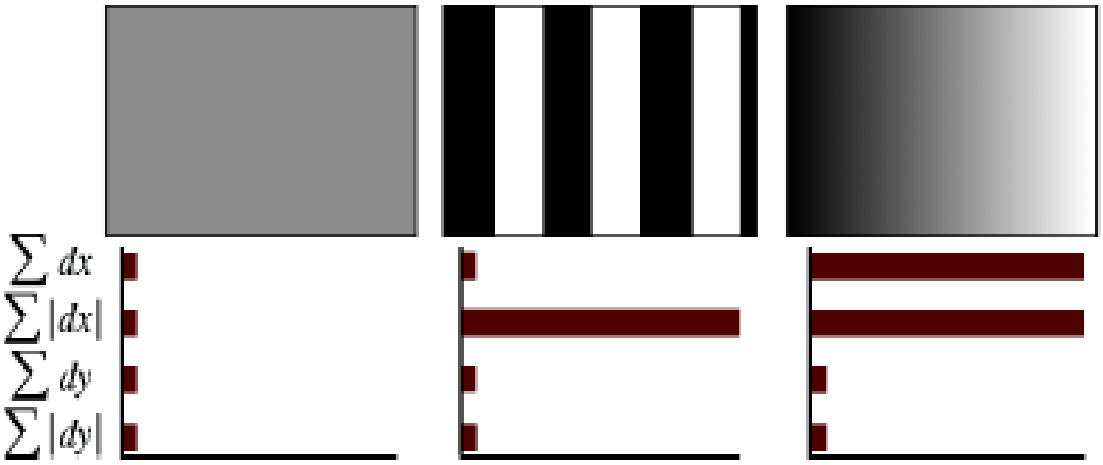

Figura 10: Sumatoria para diferentes contrastes; Izquierda: En el caso de una región homogénea, todos los valores son relativamente bajos. Centro: En presencia de frecuencias en la dirección x, el valor de $\boldsymbol{\Sigma|dx|}$| es alta, pero todos otros permanecen bajo. Derecha. Si la intensidad está aumentando gradualmente en la dirección x, ambos valores $\boldsymbol{\Sigma dx}$ y | $\boldsymbol{\Sigma|dy|}$| son altos.

Fuente: (Cruz, 2013)

La última etapa del detector, consiste en enlazar los puntos de interés hallados en dos imágenes consecutivas. Cada punto de interés en la imagen uno (instante t) será comparado con los puntos de interés de la imagen dos (instante t + 1) por medio del cálculo de la distancia Euclidena entre sus descriptores. De este modo, un emparejamiento es detectado en caso de que la distancia relativa entre ambos puntos sea menor a 0.7 veces la distancia respecto al segundo vecino más cercano. Esta estrategia de emparejamiento se conoce como la del vecino más próximo. Además, añadiendo restricciones geométricas adicionales se disminuye el riesgo de incurrir en falsos emparejamientos

- **Matching**

El matching es el paso que se realiza después de buscar los puntos de interés o puntos característicos, para de este modo poder comparar los descriptores de una imagen patrón con una imagen capturada. Este es el punto final, pero depende mucho del descriptor ya que si no se obtuvo de manera correcta el descriptor no se emparejaran los puntos y no se reconocerá el objeto.

2.3. Software libre OpenCV

Es una biblioteca open source para C, C++, Python y Java, siendo compatible con Windows, Linux, Mac OS, iOS y Android. Da la posibilidad de realizar procesamiento de imágenes y visión por computador en tiempo real. Dicha biblioteca fue desarrollada inicialmente por Intel y su primera versión estable fue liberada en 2006. En octubre de 2009, se liberó la segunda versión conocida con el nombre de OpenCV v2. (OpenCV)

Opencv posee estructuras básicas de datos para operaciones con matrices y procesamiento de imágenes, permitiendo visualizar datos muy sencillamente y extraer información de imágenes y videos. Tiene también diversas funciones que permiten la captura y presentación de imágenes y sus principales módulos son: (OpenCV.org)

- Cv: Contiene las funciones principales de la biblioteca.

- Core. Es un módulo compacto que define las estructuras de datos básicos, incluyendo los arreglos y matrices multidimensionales.

- Imgproc: Es un módulo de procesamiento de imagen que incluye filtrado lineal para imágenes, transformaciones de imagen geométricos (cambiar el tamaño, la perspectiva de deformación, reasignación genérico basado en tablas), la conversión de espacio de color, histogramas, etc.

- Video: Es un módulo de análisis de vídeo que incluye la estimación de movimiento, sustracción de fondo y los algoritmos de seguimiento de objetos.

- Calib3d: Módulo que se utiliza en algoritmos básicos para el enfoque y captura de un escenario en dos dimensiones o tres dimensiones, durante el procesamiento permite la reconstrucción de objetos o escenarios en tres dimensiones.

- Features2d: Módulo que sirve para los distintos detectores de rasgos sobresalientes, descriptores y comparadores descriptores.

- Objdetect: Sirve para la detección de objetos e instancias de las clases predefinidas (por ejemplo, caras, ojos, las tazas, las personas, vehículos, etc.).

- Highgui: Sirve para crear una interfaz fácil de usar para la captura de vídeo, imagen y codecs de vídeo, así como capacidades simples de interfaz de usuario.

- Gpu: Son módulos para algoritmos acelerados por GPU de diferentes módulos OpenCV.

Contienen también módulos auxiliares, como; Flann, Python y otros.

Algunas de las aplicaciones que se pueden realizar con las librerías de OpenCV descritas son; captura de imágenes en tiempo real, captura de imágenes de un archivo de vídeo, tratamiento de imagen básica (brillo, contraste, umbral, etc.), detección de objetos (cara, cuerpo, etc.), reconocimiento de regiones, reconstrucción de escenarios, etc.

2.3.1. Librerías de dependencia para procesamiento de imágenes

OpenCV instala consigo varios paquetes y para realizar esto necesita librerías para al momento de la ejecución hacerlo de manera eficiente, estos paquetes se los conoce también como dependencias y sirven para el soporte, lectura, escritura de archivos de imagen, dibujar en la pantalla, generar

pantallas y varias herramientas necesarias para procesar imágenes. Además de forma conjunta se debe de instalar el compilador con el que se trabajará para el desarrollo de las aplicaciones.

2.4. Beaglebone Black card

La tarjeta Beaglebone Black está diseñada para hacer frente a la Comunidad Open Source y cualquier persona interesada en un procesador de bajo costo basada ARM Cortex-A8. Se ha equipado con un conjunto mínimo de funciones para que el usuario pueda experimentar el poder del procesador y no pretende ser una plataforma de desarrollo completa ya que muchas de las características y las interfaces proporcionadas por el procesador son de recursos limitados. La Beaglebone Black se apega más a la experimentación y el aprendizaje de cómo programar el procesador y el acceso a los periféricos por la creación de su propio software y hardware. (Beagleboard.org)

A continuación se detallan los componentes y la distribución de la *BeagleBone Black* y se aprecia en la figura 11.

- DC Power.- Es la entrada principal de voltaje para alimentar a la placa, esta acepta 5V con una corriente de hasta 2A.

- Puerto Ethernet.- Permite realizar conexiones a redes LAN con velocidades de 10/100MBps conexión RJ-45.

- Serial Debug.- Puerto que permite realizar comunicación serial.

- USB Client.- Es un conector mini-USB que permite conectar la tarjeta al computador y puede también servir como fuente de alimentación alternativa, brindando 5v y 1A de corriente.

- BOOT switch.- Este botón permite arrancar a la tarjeta en uno de los varios modos como por la micro-SD, serial o USB.

- LEDS.- Permiten visualizar el estado de funcionamiento de la tarjeta.

- Reset Button.- Permite inicializar al procesador.

- Micro-SD slot.- Ranura donde se coloca la micro-SD.

- Micro-HDMI.- Conector donde el monitor o display es conectado

- USB Host.- Permite conectar diferentes dispositivos USB como teclado, Wi-fi, etc.

- Sitara AM3359AZCZ100.- En esta versión la placa utiliza un procesador SitaraXAM3359AZCZ que opera a 1GHz de velocidad.

- Micron 512MB DDR3L.- Memoria Ram dual data rate, trabaja de forma óptima con 1.5V.

- TPS65217C PMIC.- Integrado que provee voltajes y corrientes de alimentación a los diversos componentes de la placa, también se encarga de proporcionar la tensión de 3.3V para los rieles donde se encuentran las entradas y salidas digitales y analógicas.

- SMSC Ethernet PHY.- Interfaz física que permite realizar conexiones Ethernet.

- Micron eMMC.- Es un tipo de memoria permite almacenar hasta 2GB de información.

- HDMI.- Circuito integrado que permite gestionar la conexión con pantallas y adaptadores.

NOTA: Esta descripción rápida de la Beaglebone Black permite conocer las características más relevantes del Hardware de la tarjeta, pero hay que tener presente que no únicamente es un ordenador, ya que es una tarjeta de instrumentación y esto significa que se puede utilizar para el nivel entradas y salidas, de campo y en el nivel de control .

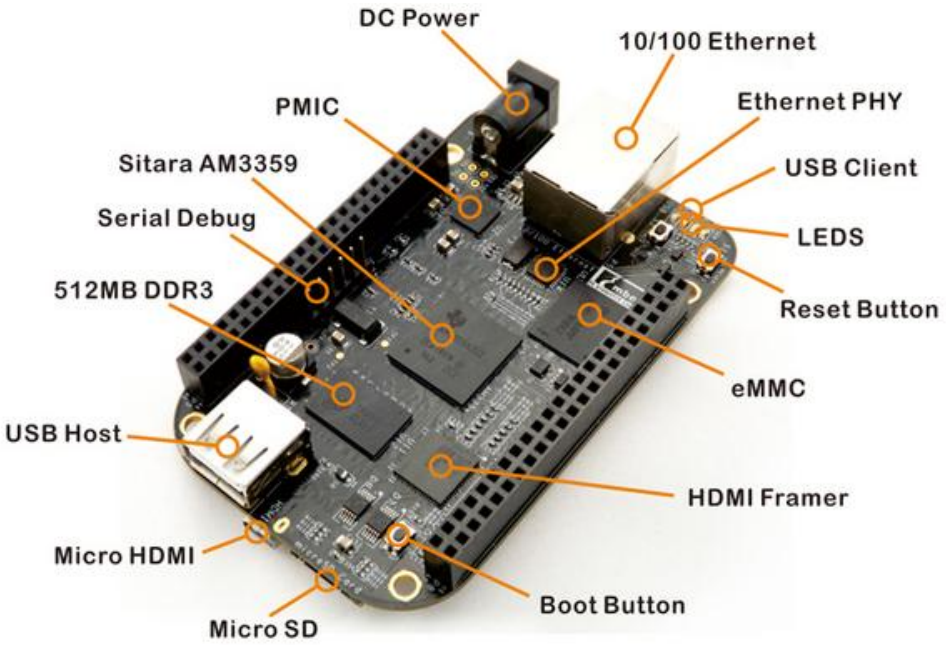

Figura 11: Beaglebone Black Card.

Fuente: (Adafruit, 2013)

El BeagleBone Black es compatible con algunas distribuciones de Linux como Angstrom, Debian, Ubuntu, como también con Android, aunque sea un ordenador de recursos limitados.

2.5. Cámara para procesamiento de imágenes

En los sistemas de visión artificial, la función de las cámaras de visión es capturar la imagen proyectada. Las cámaras de video han tenido una rápida evolución en los últimos años, desde las primeras cámaras de video que iban equipadas con tubos Vidicon hasta las más modernas cámaras provistas de sensores CCD (Charge Coupled Devide) y CMOS que se incorporan en los sistemas de visión artificial en la actualidad. Las cámaras que se utilizan en visión artificial requieren una serie de características específicas, como el control del disparo de la cámara para capturar las piezas que pasan por delante de la cámara exactamente en la posición requerida. Estas características son básicas para su utilización en un sistema de visión artificial en la industria. (Mendoza Dario, 2012)

Según el documento (Típos de cámara). Las cámaras de visión artificial son más sofisticadas que las convencionales, ofreciendo un completo control de los tiempos y señales, de la velocidad de obturación, de la sensibilidad y de otros factores fundamentales a la hora de integrarlas en un sistema de visión artificial tanto en aplicaciones científicas como industriales. Hay múltiples tipos de cámaras que se han separado según sus características de utilización.

Cámaras Matriciales.
Cámaras Lineales.
Cámaras Alta Velocidad.
Cámaras 3D.
Cámaras Inteligentes.
Cámaras Infrarrojas / Cámaras Térmicas.
Sistemas espectrales y sistemas Multiespectrales.

2.5.1. El sensor

"El sensor es la parte fundamental de la cámara**,** dependiendo de este, la capturar es óptima" o no, el proceso de captura consiste en encaminar la luz del exterior hasta "el sensor de la cámara para de esta manera realizar las capturas del punto de interés que se enfoca". Mientras mejor sea el sensor mejor será la captura. (INFAIMON.S.L)

2.5.2. Tecnología de los sensores

Los sensores en las cámaras se pueden clasificar según su tecnología:

- CCD y Super CCD
- CCD RGBE
- CMOS
- Foveon X3

Aunque en realidad los 2 tipos de sensores más expandidos o populares son el CCD y el CMOS**.**

Según (Mendoza Dario, 2012). Los sensores del tipo CCD fueron los primeros en usarse, pero en la actualidad la mayoría de las cámaras están usando sensores CMOS, pues se descubrió que esta nueva tecnología llamada CMOS permitía la creación se sensores que consumían mucha menos batería y que a la vez permitían un procesamiento de la imagen mucho más rápido. Por otro lado, a las fábricas les resulta mucho más económico fabricar un sensor CMOS que uno CCD. En cuanto a calidad de la imagen, en el pasado los CCD ofrecían mejor calidad de imagen, pero con el tiempo los CMOS alcanzaron ya esa calidad.

2.5.3. Cámara Logitech HD Pro Webcam C920

En el presente proyecto se utilizará la cámara C920, debido a que la aplicación a implementar en la detección de señales de tránsito en tiempo real, teniendo en cuenta los vehículos se desplazan a una velocidad moderada o alta y que las señales se encuentran a una distancia alejada, la cámara se ajusta de manera adecuada a la aplicación por su alta velocidad de captura y alta resolución de imágenes. “La cámara Logitech HD Pro Webcam C920” que se indica en la figura 12 presenta las siguientes características. (Logitech, 2012)

Figura 12: Cámara Logitech HD Pro Webcam C920.

Fuente: (Logitech, 2012)

a. Características

Norma de vídeo H.264

Esta norma tiene la característica de hacer que el vídeo sea más rápido y fluido, así como mayor calidad de imagen y menor exigencia al ordenador gracias a la codificación H.264, la norma del sector para vídeo en alta definición. Todo esto es posible gracias al hardware y software que presenta de la cámara

Grabación Full HD 1080p

A más de presentar varias resoluciones al momento de capturar imágenes la más relevante es que puede grabar imágenes excepcionales en pantalla panorámica con calidad Full HD 1080p y hasta 30 cuadros por segundo. Además, con la codificación H.264, el ordenador no tiene que esforzarse tanto para obtener vídeo de calidad.

Enfoque automático

El enfoque automático de 20 pasos tiene mayor capacidad de respuesta, sensibilidad e inteligencia. Tanto si los niños no paran como si quiere demostrar su estilo de baile, C920 ofrece imágenes de nitidez extraordinaria (a una distancia de 10 cm o mayor) para todas las ocasiones.

Instantáneas de 15 megapíxeles

Puede enviar brillantes instantáneas de 15 megapíxeles (mejora por software).

Especificaciones: La cámara Logitech HD Pro Webcam C920 presenta las siguientes especificaciones técnicas.

- Videoconferencias Full HD 1080p (hasta 1920 x 1080 píxeles)
- Videoconferencias HD 720p (1280 x 720 píxeles)

- Grabaciones de vídeo Full HD (hasta 1920 x 1080 píxeles
- Tecnología Logitech Fluid Crystal™ Compresión de vídeo H.264
- Lente Carl Zeiss® con enfoque automático de 20 pasos
- Micrófonos estéreo integrados con reducción de ruido automática
- Corrección automática de iluminación escasa
- Certificación USB 2.0 de alta velocidad (compatible con USB 3.0)
- Clip universal compatible con trípodes para monitores LCD, CRT o portátiles.

Software de cámara Web Logitech:

- Grabación de vídeo: captura de vídeo de hasta Full HD 1080p
- Captura de fotografías: hasta 15 megapíxeles (mejora por software)

Requisitos del sistema:

- Windows® XP (SP3), Windows Vista® y Windows® 7 (32 bits o 64 bits)

Requisitos recomendados para videoconferencias Full HD 1080p y 720p

- Carga/descarga de 1 Mbps para 720p
- Carga/descarga de 2 Mbps para 1080p

Para grabación de vídeo HD 1080p:

- Procesador Intel® Core 2 Duo a 2,4 GHz
- GB de RAM o más
- Espacio en el disco duro para los vídeos grabados
- Puerto USB 2.0 (preparado para USB 3.0)

2.6. Reconocimiento de objetos

El reconocimiento de objetos es un campo amplio en dónde se involucran la ingeniería, computación y matemática, para que de este modo una máquina

pueda relacionar objetos físicos de un entorno y reconocerlos, mediante varias etapas, cada una de estas etapas ayudan al ordenador a que tome decisiones.

Según (Mendoza Dario, 2012). El sistema se divide en bloques, cada uno ejecutará una tarea específica, por lo general se puede identificar las siguientes etapas; la adquisición de datos, la extracción de características y la toma de decisiones. Dónde el punto más importante de todo es el reconocimiento de objetos. Para lograr esta tarea es necesario tener que realizar un sistema que sea capaz de captar y observar objetos del entorno de forma autónoma, someterse a una etapa de entrenamiento y aprendizaje dónde se llevará a cabo la determinación de características relevantes que permitirán por último el reconocimiento de objetos, el diagrama de la figura 13 muestra un sistema de reconocimiento en general.

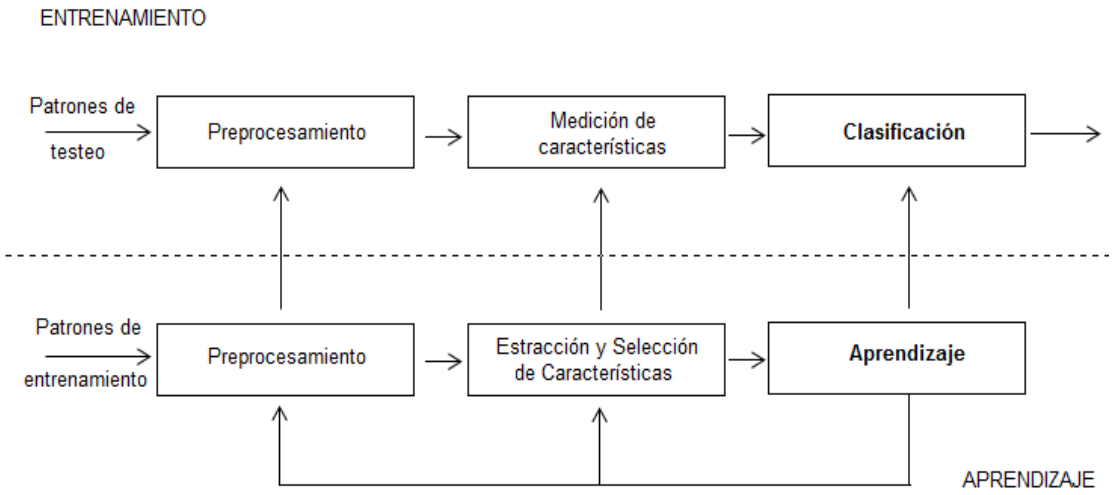

Figura 13: Sistema de reconocimiento.
Fuente: (Mendoza Dario, 2012)

"El módulo encargado de realizar la extracción de características transforma los patrones de entrada" que en este caso sería una captura, para que se almacene a manera de arreglos de dos dimensiones o tres dimensiones. Lo más importante de esta transformación es el tener datos que sean más fáciles de procesar por la máquina, además darle algunas propiedades como

“invarianza de los patrones de entrada y obtener características” que permitan discriminar entre los patrones y tener un buen rendimiento al momento de reconocerlo. Para un sistema de reconocimiento completo se tienen los siguientes componentes. (Seijas M., 2011)

- **Sensor**

El mismo que proporciona los elementos del entorno (variables físicas) al sistema y de este dependerán los límites en el rendimiento de todo el sistema, es la parte primordial en el sistema ya que sin este elemento no se puede realizar ninguna de las siguientes etapas, es la nica entrada para este proyecto (en las pruebas se puede hacer con otras fuentes).

- **Extracción de Características**

En esta etapa se puede implementar un sinnúmero de métodos, técnicas y algoritmos, todas estas llevan a un mismo fin, que es extraer la información que permita a la máquina ir discriminando si la información que obtuvo es suficiente como para lograr la distinción de objetos físicos y posterior se elimina la información redundante e irrelevante de los objetos que no se deseen reconocer.

- **Clasificador**

Esta etapa es donde se prioriza la información válida sin falsos positivos (detección de objetos o características de una imagen patrón en una imagen capturada del entorno donde no las existe) que puedan aparecer en el proceso de extracción de características, para poder tomar la decisión correcta.

CAPÍTULO III

3. DESARROLLO

Este capítulo explica de manera detallada el proceso para el reconocimiento de objetos, partiendo desde la captura de imágenes en tiempo real, procesamiento y posterior reconocimiento. Se revisarán desde las librerías, tipos de variables, clases e instancias que se emplearán en mencionado proceso.

3.1. Lectura de imágenes con OpenCV y C++

La lectura de imágenes en el reconocimiento de objetos es la parte primordial dentro de este proceso, para facilitar este trabajo de visión por computadora se utiliza el conjunto de librerías de OpenCV en conjunto con C++. Para la lectura de imágenes hay que tener presente el formato del archivo en el que se encuentran las imágenes ya que no todos son compatibles con OpenCv.

Antes de realizar la lectura y presentación de una imagen, se debe tener presente cuales son las bibliotecas estándares de C++ y las bibliotecas de OpenCV que son bibliotecas privadas. Tener presente que las bibliotecas de cabecera son una recopilación de clases y funciones que están listas para utilizar.

Estos dos tipos de bibliotecas se las declara con *#include* al inicio, lo qué hay que tener presente es que las bibliotecas estándares al declarar tienen la siguiente sintaxis *#include <biblioteca-estándar>* y las bibliotecas privadas en caso de OpenCV tienen la sintaxis ***#include** “biblioteca-OpenCV”,* además se debe tener presente el uso del *namespace*, qué no son nada más que los espacios de nombres de identificadores o instancias únicas que pueden existir en el programa. Para definir una instancia de manera global se lo hace después de las librerías de cabecera, para de ese modo no tener que declarar con cada instrucción que se vaya a utilizar, se lo define como; *namespace*

instancia; y en C++ es fácil reconocer los *namespace* ya que vienen definidos por dos puntos antes de la instrucción.

Una vez conocido como definir las bibliotecas y los *namespace*, ahora se debe conocer el nombre de las bibliotecas de cabecera, los *namespace* y el tipo de variables que se utilizarán para cargar una imagen y presentarla, hay que tener en cuenta que las variables a usar son objetos de ciertas clases e instancias que son facilitadas por las librerías de OpenCV.

3.1.1. Bibliotecas para adquisición de imágenes

Para cargar o leer una imagen desde el ordenador en C++ y OpenCV, se necesita de la biblioteca *OpenCV.hpp* y/o la biblioteca *highgui.hpp* dependiendo del formato de la variable que almacenará la información. Las mencionadas bibliotecas en este caso se utilizarán para trabajar en la lectura y presentación de una imagen a través de una unidad de interfaz gráfica (GUI).

Las bibliotecas se las define de la siguiente manera.

#include "OpenCV2/OpenCV.hpp"

#include "OpenCV2/highgui/highgui.hpp"

Al momento de definirlas se debe especificar la dirección en donde se encuentran instaladas las librerías para no tener inconvenientes al momento de compilar y ejecutar un programa. En caso de que se tenga instalado de manera correcta y de error al momento de compilar se debe exportar un PATH para llamar a las librerías de Opencv. Conociendo las bibliotecas que permiten la lectura de imágenes, lo siguiente es conocer dónde y cómo se almacenará la información de la imagen, para esto se tienen dos maneras que son las más usadas y se las describe más adelante.

3.1.2. Variables para lectura de imágenes

OpenCV al momento de trabajar con imágenes en dos dimensiones facilita hacerlo por medio de la clase *IplImage* y la clase *Mat.* Las variables vienen a ser objetos de mencionadas clases.

Clase IplImage

Esta clase presenta una estructura de un único arreglo qué contiene la información de la imagen y las características de la misma, el ancho, el alto, el tipo de formato de los pixeles, el número de canales dependiendo si es una imagen en escala de grises o en formato RGB, el apuntador al arreglo que contiene la imagen (depende del formato de pixeles DEPTH), y el número de bytes qué ocupa cada fila del arreglo.

Los formatos de pixeles soportados en una *IplImage* son:

- IPL_DEPTH_8U - entero sin signo de 8 bits. Equivalente a CV_8U en los tipos de matriz.

- IPL_DEPTH_8S - entero de 8 bits. Equivalente a CV_8S en los tipos de *matriz*.

- IPL_DEPTH_16U - entero sin signo de 16 bits. Equivalente a CV_16U en los tipos de matriz.

- IPL_DEPTH_16S - entero de 8 bits. Equivalente a CV_16S en los tipos de matriz.

- IPL_DEPTH_32S - entero de 32 bits. Equivalente a CV_32S en los tipos de matriz.

- IPL_DEPTH_32F - precisión simple de número en coma flotante. Equivalente a CV_32F en los tipos de matriz.

- IPL_DEPTH_64F - de doble precisión el número de coma flotante. Equivalente a CV_64F en los tipos de matriz.

Esta clase permite enviar los colores por matrices separadas. Por ejemplo en una imagen a escala de grises se tiene un solo canal, mientras que en una imagen RGB se utiliza tres canales para eso.

Una de las ventajas de trabajar con esta estructura es que se puede seleccionar regiones de interés para procesarlas mediante la Ip1ROI*, sobreponer imágenes con el *maskROI*,* revisar información mediante la función *void*tileinfo* y asignar espacio en la memoria para una nueva imagen mediante *void*ImageID*. (Dr. Alfonso Alba Cadena, 2001) (OpenCV.org)

Para definir una estructura tipo *IplImage* y cargar una imagen, se hace de dos maneras como se indica a continuación:

*IplImage *nombre_variable;*

Nombre_variable = cvLoadImage("Dirección y nombre de la imagen. Banderas");

IplImage nombre_variable = cvLoadImage("Dirección y nombre de la imagen". Banderas);*

Al momento de cargar la imagen se lo hace por medio de función *cvLoadImage.* Lo que hace es cargar una imagen desde una clase *IplImage* y presenta los siguientes parámetros; Dirección y nombre del archivo que se cargará, banderas *CV_LOAD_IMAGE_COLOR* la imagen se carga en formato *RGB, CV_LOAD_IMAGE_GRAYSCALE* la imagen se carga en escala de grises. Actualmente los siguientes formatos de archivo son compatibles:

Mapas de bits de Windows - BMP, DIB

JPEG - JPEG, JPG, JPE

Portable Network Graphics – PNG

Formato de imagen Portable - PBM, PGM, PPM

Raster Sun - SR, RAS

Archivos TIFF - TIFF, TIF

Uno de los mayores inconvenientes al momento de utilizar esta clase es que se debe gestionar la memoria de manera manual y asi reservar ese espacio específicamente para esa variable. (OpenCV.org)

Clase Mat.

La clase Mat a diferencia de la clase *IplImage* que contiene un solo arreglo, presenta dos matrices, una que es de cabecera y contiene la dirección de almacenamiento, el tamaño de la matriz, los métodos que utiliza para almacenar o leer la matriz y otros parámetros, y una segunda matriz que contiene los valores de pixeles y esta depende del método con el que se lee o almacena.

Esta clase permite almacenar una matriz en un objeto que puede ser representada por un solo canal o varios canales dependiendo de las dimensiones de la imagen, puede almacenar vectores de números reales o complejos, imágenes a color o en escala de grises, campos vectoriales, nubes de puntos, histogramas, etc.

Un objeto de tipo Mat almacena matrices de dos dimensiones a manera de un arreglo de filas y columnas y almacena una matriz de tres dimensiones plano por plano. Al poder almacenar los valores a manera de arreglo le hace compatible con la clase *IplImage, CvMat, Numpy (ndarrays),* mapas de bits (Win32) y con otros tipos de matrices que se utilizan para calcular la posición de pixeles.

Los tipos de formatos de pixeles que soporta esta clase son los siguientes:

CV_8U entero sin signo de 8 bits cero canales utilizado para procesar matrices pequeñas

CV_8S entero con signo de 8 bits 1 canal.

CV_16U entero sin signo de 16 bits 2 canales.

CV_16S entero con signo de 16 bits 3 canales.

CV_32S entero con signo de 32 bits 4 canales.

CV_32F punto flotante de 32 bits cinco canales.

CV_64F punto flotante de 16 bits seis canales.

Para definir un objeto (variable) de tipo *Mat* se utiliza el método créate (nrows, ncols, tipo). La mayor ventaja de utilizar esta estructura es que al momento de cargar una imagen se tiene un constructor que ajusta de manera automática los valores de la matriz de cabecera y la matriz de formato de pixeles según se haya definido, la gestión de memoria es igualmente realizada de manera automática (OpenCV.org)

A continuación se indica la manera de definir un objeto tipo *Mat* de forma práctica.

Mat nombre_variable(nrows, ncols, tipo[Formato de pixeles]);

Mat matriz(1000,1000,CV_32F);

En el caso de que se quiera ajustar los valores de manera automática.

Mat matriz;

Para cargar una imagen en un objeto de tipo Mat se hace de la siguiente manera.

Mat immage= imread(“Dirección y nombre del archivo”, bandera);

Para cargar una imagen se lo hace por medio de la función *imread,* lo que hace es leer el archivo especificado y lo devuelve. En caso de que la imagen no se puede leer es porque no se encuentra o es de un formato distinto a los que soportan el objeto *Mat* y el resultado es una matriz vacía *(Mat:: datos == NULL).*

- Las banderas lo que hacen es especificar el tipo de formato de pixeles y en este caso se puede trabajar con las siguientes banderas.

- La bandera *CV_LOAD_IMAGE_COLOR,* lo que hace es cargar una imagen con el formato de pixeles RGB.

- La bandera *CV_LOAD_IMAGE_GRAYSCALE,* con esta se carga una imagen con el formato de pixeles en escala de grises.

- La clase Mat soporta los formatos de archivo enlistados a continuación.

Mapas de bits de Windows - * .bmp, * .dib.

JPEG - * .jpeg, * .jpg, * .jpe.

JPEG 2000 archivos - * .jp2.

Portable Network Graphics - * .png.

Formato de imagen Portable - * .pbm, * .pgm, * .ppm.

Raster Sun - * .sr, * .ras.

Archivos TIFF - * .tiff, * .tif.

Al momento de trabajar con una estructura Mat para cargar una imagen, se puede realizar operaciones aritméticas como si se trataran de matrices normales, lo que no permite realizar *IplImage*. Por esta facilidad que brinda la clase Mat en el presente proyecto se utilizaran objetos de este tipo para trabajar con los diferentes arreglos. (OpenCV.org)

3.1.3. Presentación de imágenes

Para presentar una imagen con OpenCV, se hace por medio de la función *imshow,* y se define de la siguiente manera:

imshow("Etiqueta", variable);

Donde la etiqueta viene siendo el nombre de la ventana y la variable puede ser de tipo *IplImage* o *Mat* para este caso. La función *imshow* muestra una imagen en la ventana especificada con la etiqueta y el tamaño. Para la presentación de las imágenes es necesario conocer cuál es la profundidad con la que trabaja.

Si la imagen es de 8 bits sin signo, se muestra como es.

- Si la imagen es de 16 bits sin signo o entero de 32 bits, los píxeles se dividen por 56. Es decir, el rango de valores [255 * 256] se asigna a [0 255].

- Si la imagen es de 32 bits de punto flotante, los valores de píxel se multiplican por 255. Es decir, el rango de valores [0,1] se asigna a [0 255].

Si la ventana en la que se presentará la imagen no se creó antes de llamar a la función *imshow*, se crea una ventana de manera automática con el parámetro o bandera *CV_WINDOW_AUTOSIZE.* Si la ventana se ha creado con la mencionada bandera, la imagen se muestra con el tamaño original que tiene, sin embargo, si la resolución de la pantalla es menor que el tamaño de la imagen, entonces la ventana que se genera es limitada al tamaño de la resolución de pantalla.

En caso de tener una imagen que sea más grande que la resolución de la pantalla, se define la ventana con la bandera *WINDOW_NORMAL* antes de llamar la función *imshow* y eso genera una ventana estándar de 320 x 240 pixeles. Para crear una ventana previa y colocar la imagen en la misma se realiza de la siguiente manera.

namedWindow(“Etiqueta ”, parámetro o bandera);

Para poder visualizar la imagen en la ventana que fue definida o generada de manera automática se necesita de la función *waitKey.* Esta función lo que hace es esperar por un evento generado por teclado, donde el sistema operativo realiza un retardo mínimo en lo que se está ejecutando en ese momento. El retardo es generado en milisegundos y en caso de poner cero a este parámetro lo que sucede es que nunca esperara el evento y el programa se ejecutará hasta esa línea y no saldrá.

waitKey(retraso);

Esta función es el único método que tiene *highgui* para controlar los eventos, además esta función solo trabaja si hay al menos una ventana *highgui* activa. (OpenCV.org)

NOTA: La función *imshow* solo funciona acompañada de la función *waitKey.*

3.1.4. Ejemplos

Para comprender todo lo mencionado en los puntos anteriores se presentan ejemplos para cada caso.

En este caso donde la variable es una clase de tipo *IplImage* se tiene las siguientes líneas de instrucciones.

```
#include “OpenCV2/OpenCV.hpp”

int main( int argc, char** argv ) {

IplImage* imagen = cvLoadImage("/home/usuario/Escritorio/cincuenta.jpg");

//cvNamedWindow( "CAPTURA", CV_WINDOW_AUTOSIZE );

cvShowImage("CAPTURA", imagen );

cvWaitKey(0);
```

cvReleaseImage(&imagen); //se libera la memoria donde está la imagen.

cvDestroyWindow("CAPTURA");

}

El resultado se visualiza en la figura 14 donde se ha cargado una imagen en formato RGB y el resto de parámetros de forma automática.

Figura 14: Lectura de una imagen con OpenCV, mediante la clase *IplImage* en formato de pixeles RGB.

Para presentar una imagen con el formato de pixeles en escala de grises basta con colocar la bandera *CV_LOAD_IMAGE_GRAYSCALE* al momento de cargarla. Para este caso se aplica un objeto de clase *Mat*, como se conoce que *Mat* es una instancia que necesita de un espacio de nombre de la librería OpenCV se definido un *namespace*.

#include “OpenCV2/highgui/highgui.hpp”
#include “OpenCV2/imgproc/imgproc.hpp”

```
using namespace cv;
int main( int argc, char** argv ) {
Mat imagen = imread("/home/luis/Escritorio/cincuenta.jpg",
CV_LOAD_IMAGE_GRAYSCALE );  //carga la imagen indicada y la devuelve
en una matriz
        imshow("CAPTURA",imagen); //Muestra la imagen en la ventana
indicada
        waitKey(0);      //Espera a que se presione una tecla
        destroyWindow( "CAPTURA" );   //destruye la ventana
}
```

En la figura 15 se muestra el resultado de cargar una imagen con clase *Mat* y en escala de grises.

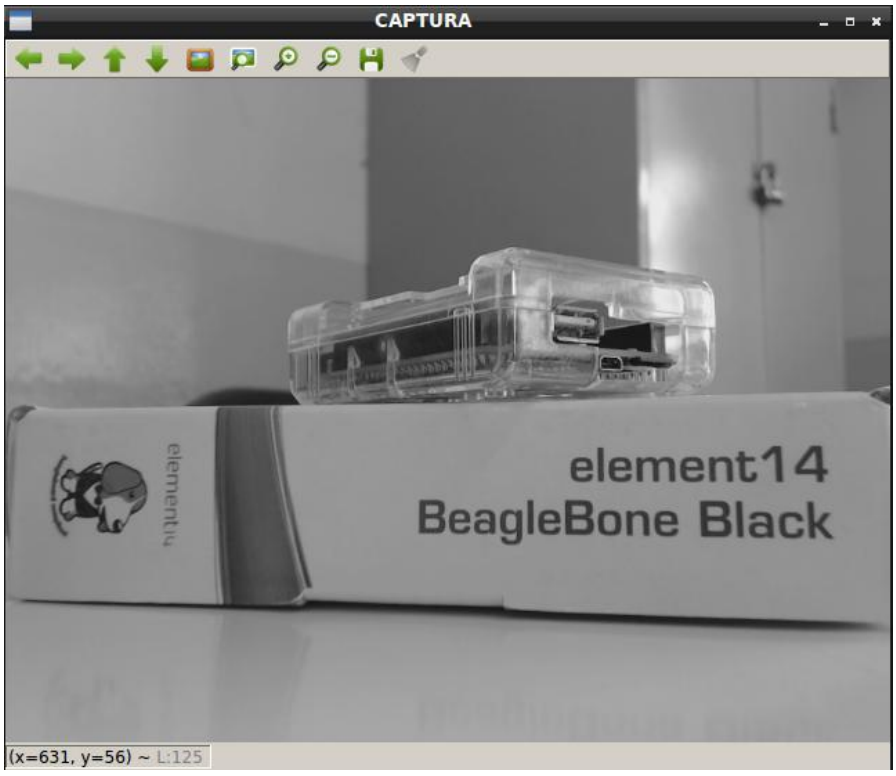

Figura 15: Lectura de una imagen utilizando Opencv, mediante la clase Mat en formato de pixeles GRAYSCALE.

En este caso, para lograr cargar una imagen en escala de grises se necesita de una librería de cabecera extra que es *imgproc.hpp,* permite el cambio de formato de pixeles, más adelante se explicará. Para el caso de cargar y visualizar una imagen sin posterior procesamiento, los resultados de hacer con una clase *IplImge* o una clase *Mat* presentan resultados similares.

Los programas están basados en los ejemplos de la documentación de OpenCV 2.4.9.

3.2. Adquisición de imágenes en tiempo real con OpenCV y C++

Para la adquisición de imágenes en tiempo real en el presente proyecto se lo realizó mediante una cámara digital ya que es la manera más común de hacerlo, en el presente trabajo no se requiere de imágenes capturadas bajo otras fuentes ya que las imágenes que se procesaran son imágenes que presenten la luz visible.

Al momento de realizar la captura lo que se hace es tener una señal analógica en el dispositivo, el proceso de conversión para que la imagen sea digitalizada es llevada a cabo por el software de la cámara y de esta manera tener a la salida un arreglo de valores numéricos donde cada valor pertenecerá a un pixel que compone a la imagen en su totalidad.

El procedimiento que se lleva a cabo es similar al cargar una imagen desde el ordenador, salvo que la diferencia es que ahora la fuente de la información de la imagen es la cámara digital y no un archivo guardado con anterioridad. Para la lectura de imágenes en tiempo real se utilizan las mismas librerías de cabecera que se utilizan en la lectura de imágenes desde un ordenador.

En los siguientes puntos se conocerá las formas de inicializar el dispositivo así como también las clases y funciones que se utilizaron en el proceso.

3.2.1. Inicialización de la cámara

Para inicializar la cámara y leer una captura desde la misma, se puede hacer de dos modos diferentes que son los más comunes, se lo hace por medio de la función *CvCaptureFromCam* y la clase *VideoCapture.*

En el caso de la función *CvCaptureFromCAM* trabaja con un objeto *CvCapture* el cual es un constructor en donde se almacenará la captura hasta cuando esta sea leída y para leer se utiliza un objeto de tipo *IplImage.* El procedimiento de inicialización y captura se lo define de la siguiente manera en forma práctica.

```
CvCapture *captura = NULL;

captura = cvCaptureFromCAM(0);
```

o simplemente

```
CvCapture *captura =  cvCaptureFromCAM(0);
```

Como se observa, se trabaja con punteros de memoria, lo que aquí llama la atención es que en la primera forma inicializa y limpia la variable y en el segundo caso lo que hace es cargar la imagen de manera directa sin limpiar la variable y sin importar los valores previos que tenga ese espacio de memoria, aunque esto puede generar ruido en la imagen.

En el caso de la función *VideoCapture* permite trabajar con objetos te tipo Mat para almacenamiento y presentación de las capturas, la ventaja de trabajar con estos objetos es que se limpian e inicializan de manera automática además facilitando el procesamiento. Para inicializar y capturar desde una cámara se lo hace de la siguiente manera.

```
Mat captura;

captura= VideoCapture(0);
```

O simplemente

VideoCapture captura(0);

La función *VideoCapture* facilita trabajar con la clase Mat, esto permite definir de manera directa la variable.

En ambas funciones el índex del dispositivo será cero, porque solo se tiene un dispositivo de captura. Cuando se trabaja con adquisición de imágenes desde una cámara o algún archivo de video lo que se puede hacer es obtener y fijar algunos parámetros como: Altura, ancho, formato de pixeles, modo de captura, contraste, saturación, brillo y formato del archivo, en lo que corresponde a los parámetros al utilizar una cámara. (OpenCV.org)

3.2.2. Lectura de la captura

Para la lectura de un frame que se ha capturado con la cámara, lo que se debe hacer en primer lugar es conocer la clase del objeto que contiene la imagen, para de este modo crear objetos compatibles con la clase que se utiliza para la captura y con la clase de objetos que pueden ser presentados en una ventana GUI.

Para leer una captura realizada con la función *cvCaptureFromCAM,* no es posible hacerlo de manera directa ni es posible liberar ese espacio de memoria porque se perderá la información y además al hacerlo se genera un break y se termina la ejecución del programa, posterior a la lectura si se desea liberar el dispositivo se usa la función *release.*

Para leer se utiliza otra función denominada *cvQueryFrame,* este es un método que se encarga de devolver una imagen que se ha almacenado en el buffer de video durante la captura, posterior a la extracción se almacena la imagen en un objeto de tipo *IplImage.* Para realizar este proceso en forma práctica se hace de la siguiente manera:

*IplImage *frame = NULL;*

frame = cvQueryFrame(captura);

Ahora ya se tiene una imagen que pueda ser procesada.

En el caso de usar la función *VideoCapture* existen dos métodos para hacer la lectura de una captura, pero llegan a un mismo final que es almacenar la imagen en un objeto tipo Mat. La primera consiste en hacer una lectura de directa de la variable que contiene la imagen y se define de la siguiente manera.

captura.read(Objeto tipo Mat);

El segundo método consiste en asignar de manera directa la variable que contiene la captura a un objeto tipo *Mat* para su posterior utilización.

captura >> objeto tipo Mat;

Posterior a tener ya el objeto con la imagen que se tenía en el buffer se continúa con el respectivo procesamiento o presentación. (OpenCV.org)

3.2.3. Presentación de la captura

Para presentar la imagen en una ventana gráfica se procede de la misma manera que se hace con imágenes cargadas desde el ordenador. Salvo con una diferencia al usar la función *waitKey,* ahora obligadamente el retardo debe ser mayor a cero y presentar de manera permanente las capturas, sabiendo que el retardo al ser cero no espera el evento el programa se ejecuta hasta esa línea y el resto de código nunca se ejecutará y el resultado será tener en la pantalla la primera imagen que se capturó.

Para llevar a cabo esa acción en forma práctica se define de la siguiente manera.

imshow("CAPTURA", captura); if (waitKey(10) == 'a') return 0;

Más adelante se presentan ejemplos de cómo realizar la captura de imágenes en tiempo real.

3.2.4. Captura de imágenes con OpenCV

Las siguientes instrucciones presentan la captura de imágenes en tiempo real por medio de una cámara web.

```
#include <OpenCV2/highgui/highgui.hpp>

using namespace cv;

int main( int argc, char** argv ) {

CvCapture *captura = NULL; IplImage *fotograma = NULL;

captura = cvCaptureFromCAM(0);

while(1)   {

  fotograma = cvQueryFrame(dispositivo);

  imshow("Captura", fotograma);

  if (waitKey(10) == 'a')  return 0;

  }

destroyAllWindows();

 }
```

La figura 16 muestra la captura que se realiza en tiempo real en formato de pixeles RGB.

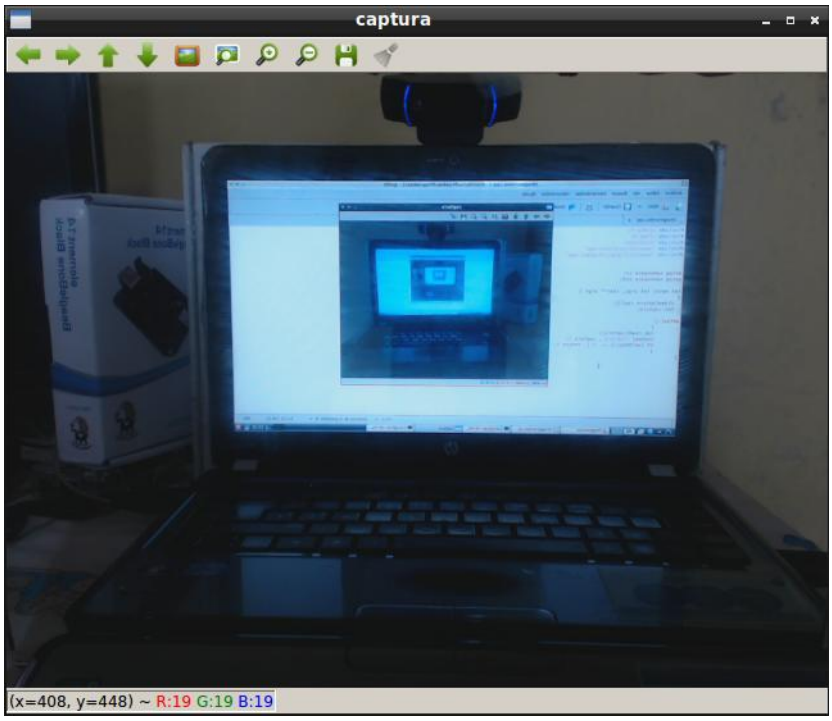

Figura 16: Captura en tiempo real en RGB.

Para tener la captura en escala de grises se necesita de una función extra la que permite realizar la conversión de formato de pixeles. Ya que el procesamiento de imágenes se puede hacer en escala de grises o en imágenes RGB. En la siguiente secuencia de instrucciones se muestra la captura de video en tiempo real y en escala de grises.

```
#include <OpenCV2/highgui/highgui.hpp>

#include <OpenCV2/imgproc/imgproc_c.h>

using namespace cv;

using namespace std;

int main( int argc, char** argv ) {

Mat Frame, Frame1; VideoCapture captura(0);

while(1)    {
```

```
        captura >> Frame;    //captura.read(Frame);

        cvtColor(Frame, Frame1, CV_RGB2GRAY);

        imshow("captura", Frame1);

        if (waitKey(10) == 'a')  return 0;

    }

destroyAllWindows(); }
```

Para realizar la conversión de formato de pixeles de RGB a escala de grises es necesario utilizar otro objeto de tipo *Mat* para no perder la información, además de la función *cvtColor* que se encarga de la conversión de formatos de pixeles. La figura 17 muestra la captura en escala de grises que se realiza en tiempo real.

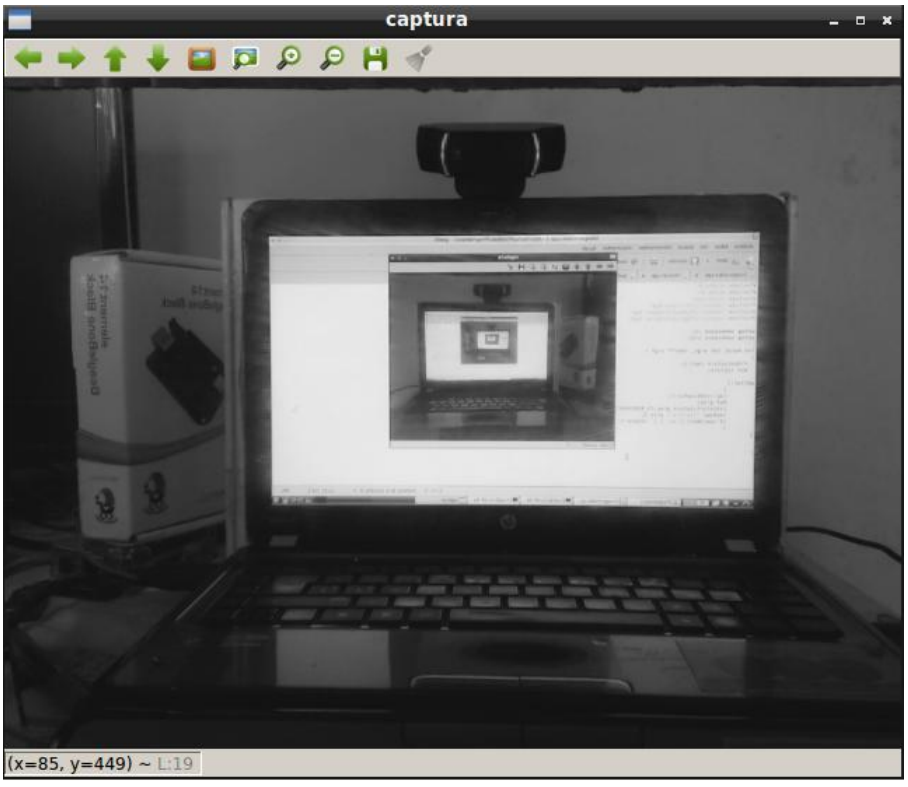

Figura 17: Captura en tiempo real en escala de grises.

3.3. Algoritmo SURF con OpenCV y C++

El algoritmo SURF para la identificación de objetos se lo puede implementar con imágenes almacenadas previamente en el ordenador así como también con imágenes capturadas en tiempo real. Para la implementación de este algoritmo se necesita conocer las clases, funciones, métodos e instancias que se utilizarán para cumplir con su fin.

3.3.1. Bibliotecas de encabezado y namespace

EL algoritmo SURF necesita de las siguientes bibliotecas de cabeceras para su implementación.

#include <stdlib.h> Biblioteca para la gestión de memoria dinámica, control de procesos y otras funciones.

*#include <unistd.h>*Biblioteca específica de Unix para trabajar con sockets de entrada y salida.

#include <cv.h> Biblioteca general de OpenCV

#include <stdio.h> Biblioteca estándar de entrada y salida

#include "OpenCV2/core/core.hpp" Biblioteca que permite trabajar con estructuras básicas, estructuras dinámicas y arreglos para el SURF

#include "OpenCV2/highgui/highgui.hpp" Biblioteca para presentar imágenes en una interface de usuario, lectura y escritura de imágenes.

#include "OpenCV2/imgproc/imgproc.hpp" Biblioteca que sirve para realizar el filtrado de la imágenes y transformación geométrica de imágenes al implementar SURF.

#include "OpenCV2/features2d/features2d.hpp" Biblioteca que se utiliza para la detección de puntos característicos mediante descriptores y extractores, dibujar los puntos de interés, interface para buscar coincidencias y forzar valores.

#include "OpenCV2/calib3d/calib3d.hpp" Complementa algunas funciones de la biblioteca features2d.

#include "OpenCV2/nonfree/nonfree.hpp" Define las características de detección y de los descriptores de SURF.

Estas tres últimas bibliotecas de cabecera son las más importantes para la implementación del algoritmo SURF. (OpenCV.org)

Definidas ya las cabeceras también se pueden definir los espacios de nombre que se utilizarán en el algoritmo y son los siguientes. *using namespace cv*, este nombre de espacio se define para todas las funciones e instancias de las librerías de OpenCV para captura, conversión, procesamiento y presentación de imágenes. *using namespace std;* Este nombre de espacio se define para todos las instancias y funciones de OpenCV como también para las de C++ ya que se trabajará con vectores, arreglos y matrices de distintos tipos.

Con las bibliotecas de cabecera y los *namespace* definidos, ahora se debe conocer las clases, funciones y métodos que se utilizan en el algoritmo SURF.

3.3.2. Clases utilizadas para detección de objetos

En el reconocimiento de objetos mediante el algoritmo SURF, se utilizan clases que permitan inicializar de manera directa o automática sus objetos, además se necesita conocer los métodos que se utilizan, para qué se utilizan y cómo se utilizan, para de este modo entender el funcionamiento del Algoritmo SURF.

En los siguientes puntos se conocerán todas las clases, instancias y métodos que se utilizan en cada etapa del algoritmo SURF.

a. Detección de puntos característicos (kps)

El primer paso para implementar el algoritmo SURF después de cargar las imagines es la detección de puntos característicos, para la obtención de los

mismos se necesita de las funciones *SurfFutureDetector y SurfDescriptorExtractor.* La primera función detecta los puntos de la imagen basándose en el principio de la matriz Hessiana para determinar si un pixel o conjunto de pixeles en la imagen se consideran como un punto característico.

Al momento de definir el objeto con esta función se debe especificar el valor umbral con el que se va a trabajar en la obtención de puntos característicos. Hay que conocer que en la práctica mientras mayor sea el valor del *minHessian* menos puntos clave se obtendrán y se espera que esos puntos clave sean repetitivos, por lo tanto más útil. Por otra parte, cuanto menor es el *minHessian,* más puntos clave se obtendrán, teniendo más ruido en el proceso, y por lo tanto genera falsos positivos al momento de compararlas. (Stackoverflow)

La segunda función se encarga de la extracción de las características locales de cada uno de los puntos que han sido detectados en el paso anterior.

Para lograr la detección y extracción de los puntos característicos en forma práctica se debe definir las siguientes funciones y objetos.

SurfFeatureDetector detector(5000);

SurfDescriptorExtractor extractor;

detector.detect (image, kp_image);

extractor.compute (image, kp_image, des_image);

NOTA: El valor de 5000 especifica que se trabaja con 50 puntos característicos.

Todos los valores que fueron encontrados en este paso son almacenados en arreglos tipo vector. La figura 18 muestra la detección de puntos característicos de una imagen.

Figura 18: Detección de los puntos característicos de una imagen.

En el caso de utilizar una imagen cargada con la clase *IplImage* se debe realizar una conversión a tipo Mat.

b. Emparejamiento de puntos característicos

Una vez que se obtengan los puntos característicos lo que se hace es emparejar los puntos de la imagen patrón con los puntos de la imagen de la captura. Este es el paso previo más importante a la detección del objeto ya que de este emparejamiento se seleccionarán los puntos que de verdad coincidan y correspondan a los puntos de la imagen patrón y los que no son.

Para realizar esto de forma práctica se define de la siguiente manera.

FlannBasedMatcher matcher;

matcher.knnMatch(des_object, des_image, matches, 2);

En la figura 19 se puede ver el resultado del emparejamiento de puntos característicos.

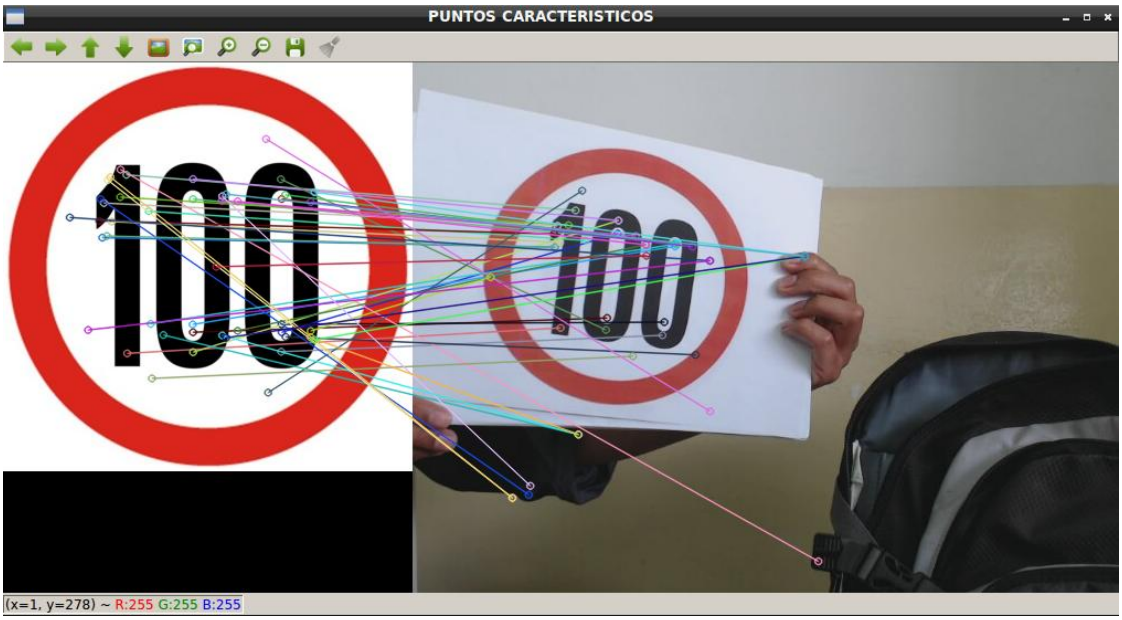

Figura 19: Emparejamiento de los puntos característicos de una imagen.

c. Emparejamiento de puntos característicos que coinciden

Para realizar el emparejamiento de puntos característicos que coinciden del patrón hacia la captura, se trabaja con el tamaño de los vectores del descriptor de la imagen que se captura, con el tamaño del arreglo de los puntos encontrados en la captura y de la distancia que tiene cada punto. Para eso se implementa el siguiente código:

```
for(int i = 0; i < min(des_image.rows-1,(int) matches.size()); i++)

{

if((matches[i][0].distance < 0.6*(matches[i][1].distance)) && ((int) matches[i].size()<=2 && (int) matches[i].size()>0))

{ good_matches.push_back(matches[i][0]);}

}
```

Aquí lo que se hace es ir comparando los valores del arreglo del patrón con los de la captura ya que los dos valores se guardaron en un mismo arreglo,

primero que nada hay que comparar la distancia entre cada punto. Lo otro es comparar los valores de las vecindades ya que cada vecindad entrega un valor diferente.

d. Dibujo de puntos característicos que coinciden

Los pasos anteriores permitieron obtener los puntos característicos de la captura y emparejarlos con los puntos característicos que coincidan en la imagen patrón, ahora para poderlos visualizar en una misma pantalla a la imagen patrón y la imagen de captura con sus respectivos puntos característicos y emparejamiento se lo hace por medio de en un solo arreglo.

Para esto se tiene la función *drawMatches,* esta función permite combinar los objetos de tipo *Mat* de n dimensiones con objetos de tipo vector, la diferencia entre estos dos objetos es que los *Mat* se organizan como ancho x alto x canales de colores, mientras que los objetos tipo vector se organizan por filas x columnas x dimensiones, es por esa similitud que presentan en su estructura que se pueden dibujar de manera conjunta. De manera práctica se define de la siguiente manera:

drawMatches(object, kp_object, frame, kp_image, good_matches, img_matches, Scalar::all(-1), Scalar::all(-1), vector<char>(), DrawMatchesFlags::NOT_DRAW_SINGLE_POINTS);

La bandera que más importa en esta función es la de no dibujar los puntos que hayan quedado sueltos por no coincidir. La figura 20 muestra el resultado del emparejamiento de los puntos característicos.

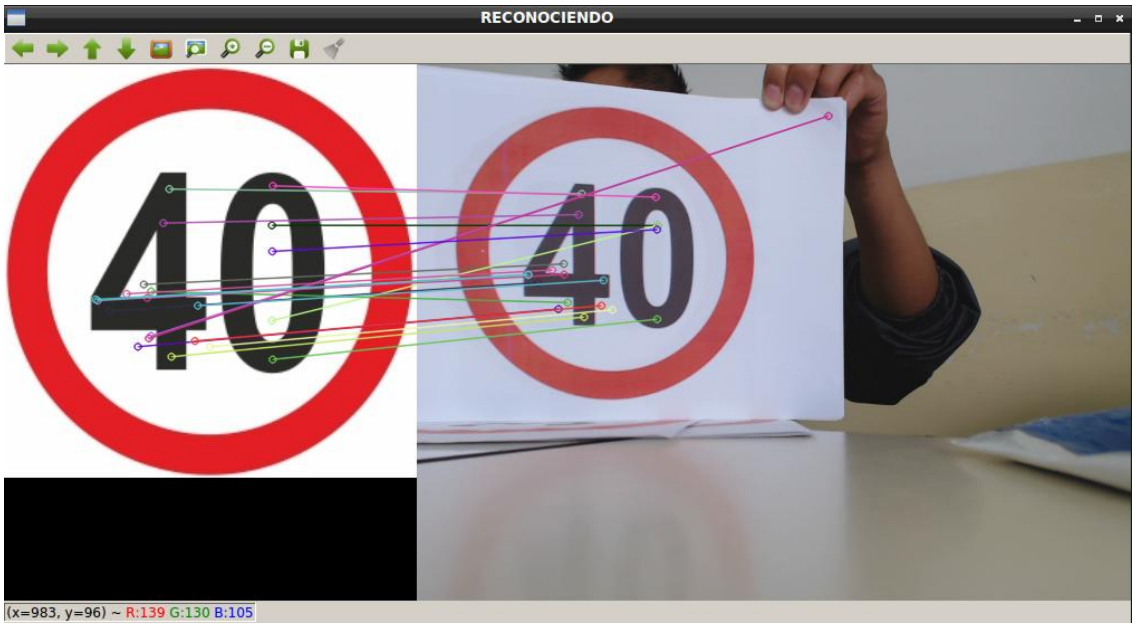

Figura 20: Emparejamiento de puntos característicos que coinciden.

3.3.3. Algoritmo para el reconocimiento de señales de tránsito

El algoritmo en general consiste en capturar una imagen del entorno, por medio del SURF realizar el procesamiento para extracción de puntos característicos de dicha captura, compararlo con cada uno de los patrones de la base de datos que se tiene hasta que coincidan. Para discriminar que una señal ha sido encontrada, se ha tomado en cuenta que el valor de los puntos característicos de la imagen capturada deben ser mayor o igual al setenta por ciento del valor de los puntos característicos que tiene la imagen patrón, de ser así, se notificará que el objeto se ha detectado, caso contrario se notificará que el objeto no existe.

Se elige el valor de setenta ya que en este caso se van a detectar las señales tránsito de velocidad y las imágenes patrón tienen en común el número cero. Las imágenes capturadas al procesar tienen alrededor del cincuenta al sesenta por ciento de puntos característicos qué coinciden con cualquiera de las señales de la base de datos y para no tener inconvenientes al momento de identificarlas se definió el valor de setenta para realizar la parte práctica.

a. Diseño de la base de datos

C++ en conjunto con OpenCV tiene la facilidad de trabajar con base de datos BLOB como se explicó en el capítulo anterior. Esta simplifica el trabajo de crear dicha base, de este modo lo que se hace es guardar imágenes de las señales de tránsito de velocidad que se utilizarán para el reconocimiento en tiempo real. La figura 21 muestra la base de datos para usar el algoritmo SURF.

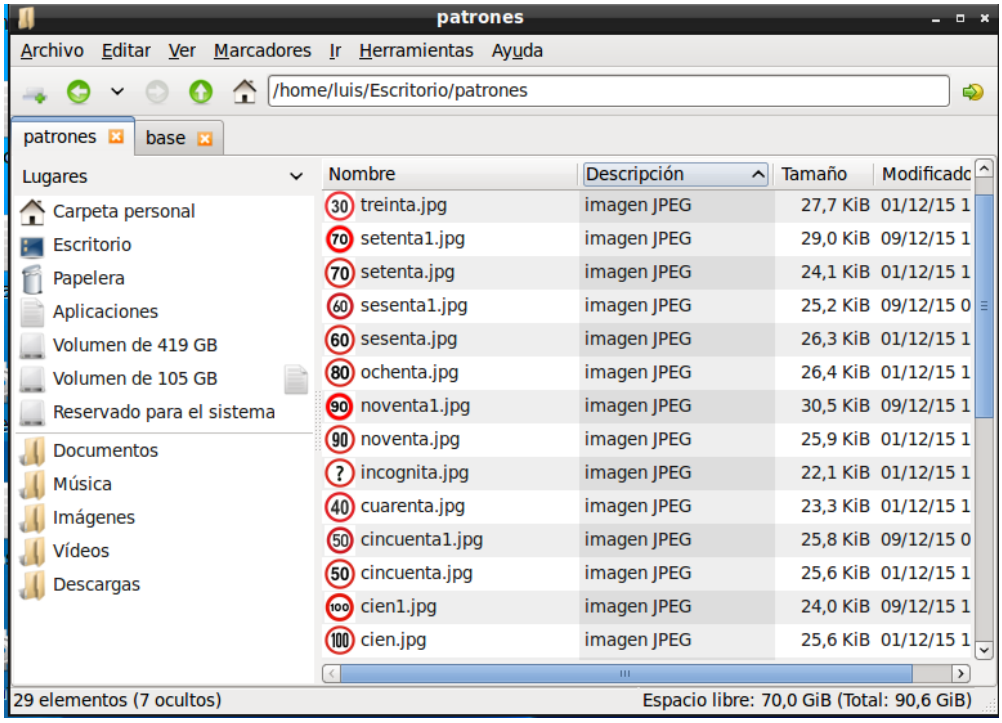

Figura 21: Base de datos para reconocimiento de señales de tránsito de velocidad.

b. Captura de Imágenes y lectura de patrones

En esta etapa lo que se hace es inicializar la cámara y la lectura de los diferentes patrones de la base de datos para una posterior búsqueda. Este punto es muy importante ya que al no direccionar de manera adecuada simplemente se presentaran arreglos vacíos y no se podrá realizar los pasos siguientes a éste.

c. Comparación entre patrones

En este paso lo que se hace es tener ya extraídos los puntos característicos de los patrones para que se realice la comparación y emparejamiento de los puntos característicos de cada uno de los patrones con los puntos característicos de la captura hasta encontrar el objeto, al momento de realizar el emparejamiento lo que se obtiene es el valor de la región de cada punto característico.

d. Programa para el reconocimiento de señales de tránsito en tiempo real

El programa para la detección de señales de tránsito lo que hace es tomar una captura en tiempo real, posterior a ello realiza la extracción de puntos característico y las compara con los de la imagen patrón para ver si existe o no existe dicha señal, en este programa se trabaja con solo una señal de tránsito, ya que para trabajar con más de una señal de tránsito lo que hay que hacer es el mismo proceso que se hizo para con el reconocimiento de una.

Para ver el programa que permite identificar una señales de tránsito en tiempo real dirigirse a los anexos al primer programa.

En las siguientes figuras 22 y 23 se aprecian los resultados de la implementación del algoritmo SURF para reconocimiento de una señal de tránsito en tiempo real, teniendo presentes falsos positivos. Lo que hay que tener presente al momento de reconocer los objetos es que se haya implementado de manera correcta el algoritmo SURF porque de no ser así se identificará objetos donde no los hay. Y en las siguientes figuras 24 y 25 muestran la funcionalidad del algoritmo para reconocer objetos de manera correcta.

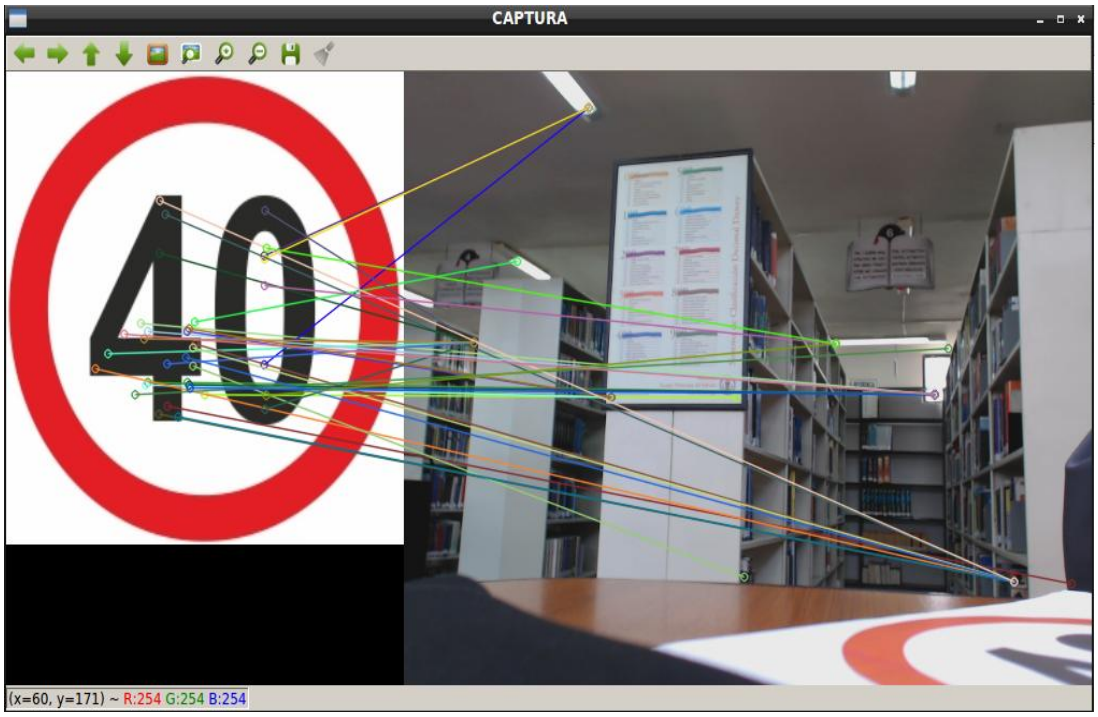

Figura 22: Búsqueda de señal de tránsito por medio del Algoritmo SURF, presenta falsos positivos.

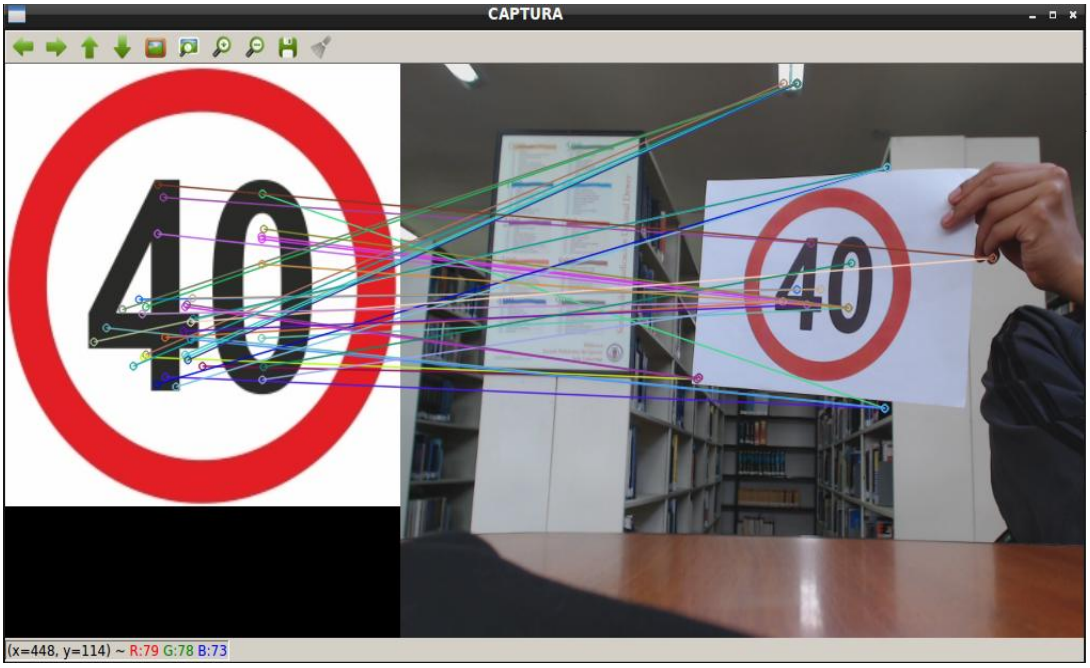

Figura 23: Resultado de búsqueda de señal de tránsito por medio del Algoritmo SURF, presenta falsos positivos.

Figura 24: Búsqueda de señal de tránsito por medio del Algoritmo SURF.

Figura 25: Resultado de búsqueda de señal de tránsito por medio del Algoritmo SURF.

3.4. Implementación de la aplicación utilizando la tarjeta Beaglebone

La aplicación a implementar en la tarjeta Beaglebone consiste en reconocer una señal de tránsito de velocidad e informar al conductor de manera sonora o preventiva que dicha señal está presente. Una parte importante al momento implementar la aplicación es la utilización de la Beaglebone Black como tarjeta de adquisición de datos, ya que se debe activar las salidas para cumplir con mencionado propósito.

3.4.1. Características de Software y Hardware

Como se indicó anteriormente, la tarjeta Beaglebone Black trabaja con un procesador cortex A8, con 512 MB en RAM y para esta aplicación se trabaja sobre el sistema operativo Debian (Jessey) una distribución de Linux de 32 bits, en conjunto con OpenCV y las librerías que permiten controlar los pines de propósito general.

3.4.2. Instalación del sistema operativo y las librerías de OpenCV en la tarjeta Beaglebone Black

A continuación se explican los pasos para realizar la instalación del sistema operativo sobre la tarjeta y la librería OpenCV.

Creación de una unidad de arranque

Para crear una micro-SD de arranque existen varias maneras, en este caso se revisan tres formas por las cuales se puede hacer e instalar una distribución de Linux en la tarjeta Beaglebone Black.

La primera forma y la más común de hacer en caso de no tener instalado una distribución de Linux en el computador es por medio de una aplicación llamada Win32Disimage para grabar el sistema operativo en la micro-SD. Los pasos a seguir son los siguientes:

- Insertar la micro-SD y verificar que se haya reconocido.

- Formatear y hacerlo en formato recomendado.

- Abrir la aplicación y elegir la unidad de disco que corresponda a la micro-SD.

- Seleccionar el archivo que tenga a su final la extensión .img

- Esperar unos minutos hasta que se escriba la imagen en la micro-SD

- Retirar la micro-SD e insertar en la Beaglebone Black.

La segunda manera para crear una micro-SD de arranque es mediante tablas de particiones desde la terminal, esta forma es utilizada por los usuarios que utilizan cualquier distribución de Linux en el ordenador, el procedimiento es el siguiente.

Inserta la micro-SD en el ordenador y revisar cual es la dirección en donde se encuentra montado, para esto desde la terminal se ingresa la instrucción *ls /dev/sd*.* Cuando se haya insertado la micro-SD desmontarla y trabajar únicamente sobre la dirección en la que se encuentra en este caso la dirección es /dev/mmcblk0.

Para iniciar las particiones de la micro-SD se abre una terminal, y se ingresa e ingresamos como súper usuario para tener permisos y privilegios de administrador del sistema operativo. Posterior a eso se sigue la siguiente secuencia de pasos.

- Iniciar una nueva tabla de partición seleccionando la letra **o**, y verificar que la tabla de particiones se encuentre vacía, para seleccionar la letra **p**.

- Crear la partición de arranque seleccionando la letra **n** para nuevo, después la letra **p** para primaria y el número **1** para fijar que es la primera partición. Ahora presionar **enter** y aceptar el primer valor por defecto y en el segundo valor fijar **4095.**

- Ahora se debe cambiar el formato a FAT16 y para esto se selecciona la letra **t** para escoger tipo y la letra **e** para seleccionar el formato W95 FAT16(LBA).

- Fijar la partición de arranque seleccionar la letra **a** y después el **1.**

- El siguiente paso es crear la partición para el sistema de archivos seleccionando la letra **n** para nueva partición, la letra **p** para partición primaria, el número **2** para especificar que es la segunda partición y presionar **enter** dos veces aceptando los valores por defecto

- Luego seleccionar la letra **p** para verificar que las particiones se han creado.

- Finalmente seleccionar **w** para escribir la tabla de particiones. (ARMhf)

Una vez creada la tabla de particiones se procede a formatear cada una de ellas. Para la partición uno se formatea en formato tipo FAT y es donde se generan los archivos que permiten que arranque, y la segunda partición se la formatea como ext4 y es donde va a estar contenido el sistema de archivos de la distribución de Linux que se instalará.

Ahora para que la micro-SD se haga de arranque se instala el e-boot y para esto se digita la siguiente instrucción desde la terminal.

wget http://s3.armhf.com/dist/bone/bone-uboot.tar.xz

Posteriormente posterior se crea una carpeta la cual contendrá los archivos de arranque, se monta la unidad y luego se descomprime el archive añadiendo al final –C boot, que permite hacer que la partición se defina como arranque.

```
mkdir boot
mount /dev/sdX1 boot
tar xJvf bone-uboot.tar.xz -C boot
umount boot
```

La tercera manera consiste en descargarse la distribución de Linux que se desea instalar, buscar la dirección donde se ha montado la micro-SD y luego desde la terminal ingresar a la carpeta donde se tiene el sistema operativo, luego ejecutar la instrucción siguiente: sudo ./setup_sdcard.sh --mmc /dev/sdX --dtb board. Esta instrucción no crea solamente las particiones, sino que directamente instala ya el sistema operativo en la micro-SD.

Instalación del sistema operativo

Para la instalación del sistema operativo se utilizó una micro-SD de 16GB y de clase 10(10Mb/s) con la versión 7.8 de Debian que se puede descargar de la página principal de la Beaglebone Black.

Para la instalación se deben seguir los siguientes pasos:

Primero descargar la imagen desde la página principal de la beagleboard.org (Recomended Debian Images) o desde la página de elinux.org. (BeagleBoard Denbian)

Una vez que se haya descargado, se debe descomprimir la carpeta que contiene la imagen de Debian (jessei). Para esto se puede hacer por medio de la terminal o simplemente buscando la dirección de donde se descargó la carpeta.

El segundo paso es la creación de particiones en la micro-SD, para esto hay dos opciones desde la terminal, la primera es crear una tabla de particiones definiendo las extensiones de cada partición y la segunda es dirigirse directamente a la carpeta de la descarga y ejecutar el *setup* de la imagen en la micro-SD de manera directa, en el punto anterior se detallaron los procedimientos. Este proceso puede tardar unos minutos hasta que se instale la imagen.

El tercer paso es arrancar en la tarjeta Beaglebone Black mediante la micro-SD, esto tarda algunos minutos la primera vez que se arranca, y una vez

finalizado el arranque, se realiza la conexión a internet y se procede a actualizar el sistema.

Instalación de OpenCV2.4.9

Una vez que se tenga instalado y actualizado el sistema operativo se instala OpenCV se debe primero que nada descargarse la versión que sea compatible con el procesador ARM de 32 bits. Una vez que se tenga una versión de OpenCV compatible con el procesador se procede a instalar los requerimientos y las dependencias.

Entre los requerimientos están las *librerías build-essential, cmake y pgk-config,* estas librerías se encarga de la generación de meta-paquetes, la compilación para instalar programas y la configuración de banderas en Debian. Además en conjunto se instala el compilador GCC y C/C++.

Las dependencias que se instalan son las de interfaces gráficas (GUI), dependencias de entrada y salida de video, dependencias para lectura de diferentes formatos de imágenes y otras más que se requieren para compilar OpenCV.

Una vez instaladas las dependencias hay que dirigirse hasta la carpeta de OpenCV 2.4.9, ahí se creará una nueva carpeta con el nombre que se desee (compilar por ejemplo), donde se compilará y posterior a ello se instalará OpenCV.

En este punto al hacer el cmake (es un conjunto de herramientas diseñadas para construir, testear y empaquetar software) es muy importante ya que de este paso depende la instalación de soportes para lectura y escritura de imágenes, compiladores adicionales e interfaces gráficas extra. En esta parte de la instalación se indicará si se instaló de manera correcta los requerimientos y dependencias que necesita OpenCV para su correcto funcionamiento. Se procede de la siguiente manera:

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_ PREFIX=/usr/local .. (banderas que se desea activar).

El siguiente paso es el make (es una herramienta que sirve para la gestión de las dependencias con los archivos que componen el código del programa que se compilará) y posterior el make install, ese proceso tardará algunas minutos mientras compila.

Tras la instalación lo que se debe hacer es configurar la librería OpenCV, reiniciar el ordenador y trabajar en ello. (debian)

Librería BeagleBoneBlack-GPIO-master

Esta librería permite trabajar a la beaglebone desde C++ y únicamente sirve para controlar los pines de entrada y salida de propósito general. Para trabajar con esta librería lo único que se debe hacer es descargarse desde la página oficial, descomprimir y trabajar dentro de esa carpeta para al momento de compilar llamar a las clases que permiten compilar y ejecutar la aplicación.

3.4.3. Control de pines GPIO de la tarjeta Beaglebone Black con C++

Para controlar desde C++ los pines GPIO (puestos de entradas y salidas de propósito general) de la tarjeta Beaglebone Black se debe conocer de manera previa el manejo de los puertos de manera directa, ya que en esta parte se conocerá de manera específica la funcionalidad de los puertos además como las corrientes y voltajes que utilizarán. Para probar el funcionamiento se lo hace desde la terminal ingresando como root (administrador).

Lo que se hace es exportar un pin, definirlo como salida y fijar el valor de uno o cero lógico. Si se lo hace desde la terminal se utilizan las instrucciones siguientes.

- Exportar el pin: echo “pin” > */sys/class/gpio/export*

- Definir el pin como entrada o salida: *echo “in/out” > /sys/class/gpio/gpiopin/ direction*

- Fijar valor lógico: *echo “1/0” > /sys/class/gpio/gpiopin/value*

- Para controlar los pines desde C++ se deben fijar las bibliotecas de cabecera *GPIO/GPIOManager.h y GPIO/GPIOConst.h.* Para el encendido y apagado de un pin de manera intermitente se tiene las siguientes líneas de instrucciones.

```
int main() {
  GPIO::GPIOManager* gp = GPIO::GPIOManager::getInstance();
  int pin = GPIO::GPIOConst::getInstance()->getGpioByKey("P9_16");
  int contador=0;
  gp->setDirection(pin, GPIO::OUTPUT);
  while(1)  {
              gp->setValue(pin, GPIO::HIGH);      sleep(1);
              gp->setValue(pin, GPIO::LOW);      sleep(1);
              }
   gp->~GPIOManager();   return 0;
 }
```

En este conjunto de instrucciones que permite en control de los pines GPIO con C++ se tiene la clase *GPIOmanager* que permite realizar el control sobre el o los pines que se deseen emplear según la necesidad y la clase *GPIOconst* que permite direccionar el o los pines con los que se trabajarán.

- La función exportPin – Export exporta un pin y equivale a tener la instrucción *echo “pin” > /sys/class/gpio/export* desde la terminal.

- La función setDirection – Set direction define al pin como entrada o salida y equivale a tener la instrucción *echo “in/out” > /sys/class/gpio/gpiopin/direction* desde la terminal.

- La función setValue – Set value fija un valor lógico al pin y equivale a la instrucción *echo “1/0” > /sys/class/gpio/gpiopin/value* desde la terminal.

3.4.4. Compilación y ejecución

Para realizar la compilación de un programa en C++ en conjunto con OpenCV se lo hace desde la terminal con la instrucción; g++ ejemplo.cpp –o ejemplo `pkg-config --cflags --libs OpenCV`. Lo que se hizo es crear un ejecutable que es el objeto del código fuente y para que las librerías de OpenCV se compilen y ejecuten se configuran todas las banderas de esta librería.

Para el caso de compilar una aplicación para controlar los puertos de entrada y salida de la tarjeta Beaglebone Black, lo que se debe hacer es guardar la aplicación con extensión .cpp en la carpeta que contiene a la librería, para la compilación desde la terminal se debe definir de la siguiente manera; g++ ejemplo.cpp –o ejemplo GPIO/GPIOConst.cpp GPIO/GPIOManager.cpp, se debe compilar de forma conjunta el GPIOconst el mismo que permite direccionar a los puertos de la tarjeta. Con GPIOManager lo que se hace es definir al puerto como entrada o salida y controlarlo.

Para compilar la aplicación capaz de trabajar en C++ en conjunto con OpenCV y BeagleBoneBlack-GPIO-master, desde la terminal únicamente hay que definir la activación de banderas de OpenCV y compilar los archivos que permiten controlar a los puertos. Esto queda de la siguiente manera:

g++ ejemplo.cpp –o ejemplo GPIO/GPIOConst.cpp GPIO/GPIOManager.cpp `pkg-config --cflags --libs OpenCV`.

Una vez que se conoce la instalación del sistema operativo en conjunto con las librerías de opencv y las librerías para controlar los puertos de entrada y salida de propósito general en la tarjeta Beaglebone Black, conociendo también cómo adquirir imágenes en tiempo real por medio de la cámara web y los pasos para la implementación del algoritmo SURF se necesita conocer los diferentes componentes en un sistema o prototipo que tenga la funcionalidad y finalidad del reconocimiento de las señales de tránsito en tiempo real.

3.4.5. Prototipo para ejecutar aplicación

La aplicación en su defecto necesita de un prototipo para su funcionamiento, en el que se dispone de una etapa de entrada, una de procesamiento y una de salida. En este caso la estrada viene siendo la cámara la misma que proveerá de las imágenes necesarias, la etapa del procesamiento es realizada por la tarjeta Beaglebone Black y la etapa de salida es el módulo de audio que se activa por medio de la los pines de la tarjeta. En la siguiente figura 26 se aprecia el prototipo que se utiliza para implementar la aplicación.

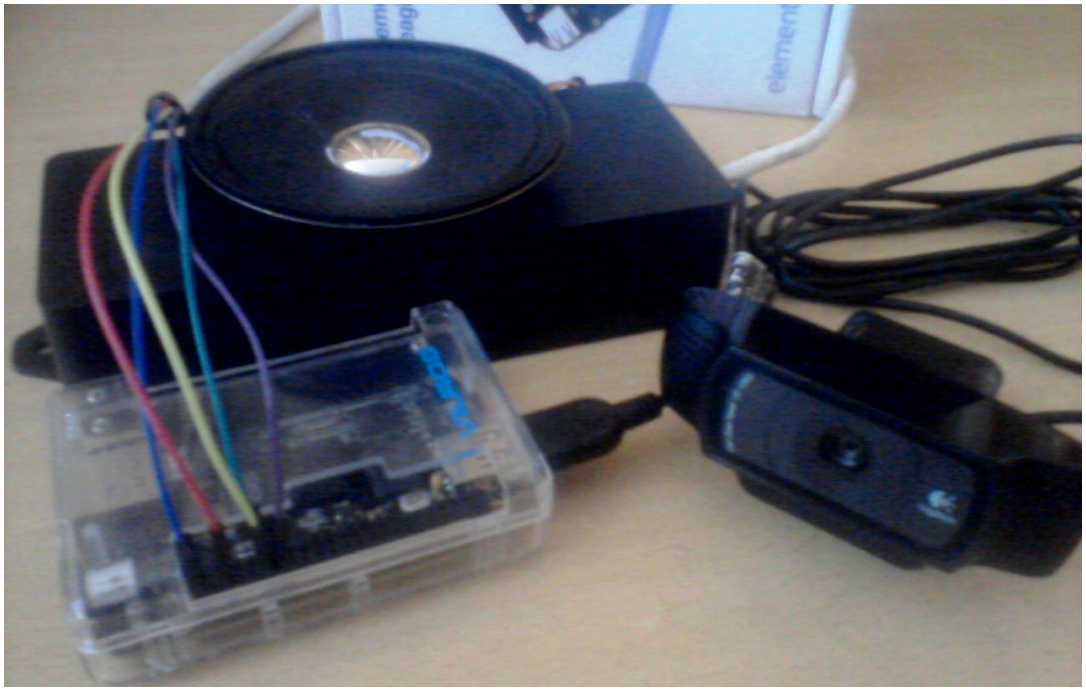

Figura 26: Prototipo

Con este prototipo lo que se pretende es ver si es posible asistir al conductor cuando se encuentre en las vías y se presente una señal de transito de velocidad y el sistema de manera autónoma sea capaz de reconocer las señales de tránsito y de manera audible dar a conocer al conductor que se ha detectado una señal.

CAPÍTULO IV

4. PRUEBAS Y RESULTADOS EXPERIMENTALES

En este capítulo se presentan las diferentes pruebas que se realizaron, desde la captura de imágenes en tiempo real, la implementación del Algoritmo SURF y detección de objetos en diferentes condiciones, las pruebas se llevaron a cabo en el computador y en la tarjeta Beaglebone Black. La parte adicional que se implementó para las pruebas con la tarjeta Beaglebone Black es la activación de los puertos de entrada y salida de propósito general, los mismos que controlan a un módulo de audio con la finalidad de alertar al conductor cuando una señal de tránsito de velocidad ha sido detectada.

4.1. Determinación del tamaño de la imagen patrón y umbral Hessiano para búsqueda de puntos característicos

Para determinar el valor umbral para el Hessiano hay que tener en cuenta la resolución (tamaño de la imagen en pixeles) de la imagen patrón y el objeto (señal de velocidad de tránsito) que se pretende reconocer, estas dos características son las más importantes de una imagen patrón ya que de esto dependerá la cantidad de puntos característicos que se obtengan y la suma de los fics (suma de los valores de la región del punto característico) de cada imagen. En una imagen pequeña no es tan fácil extraer la información correcta ya que por no tener buena resolución se puede tener falsos negativos al momento de realizar el reconocimiento de los objetos y además genera más puntos característicos repetidos, si la imagen del objeto en análisis representa un tamaño significativo con respecto a la totalidad de la imagen no hay inconveniente con los puntos característicos pero al momento de emparejarlos genera un poco más de retardo en el procesamiento ya que en esta imagen se puede obtener mayor información que en una imagen pequeña.

En las imágenes patrón, deben estar incluidos únicamente el objeto que se pretende reconocer, en este caso como los objetos a identificar son de forma

circular, se adecuó para que mencionado objeto ocupe el área de la región cuadrada en su mayoría, para de este modo tener información puntual de mencionado patrón. La figura 27 muestra una de las imágenes patrón utilizadas en diferentes tamaños de resolución para la obtención del valor de los puntos característicos.

Figura 27; Imagen patrón de velocidad máxima cincuenta Km/h.
Izquierda patrón completo, derecha patrón incompleto
Fuente: (Comisión de tránsito, 2011)

Para seleccionar el tamaño (resolución en pixeles) de las imágenes de los patrones para el presente trabajo se realizaron las pruebas con tres señales de tránsito de velocidad máxima de cincuenta, ochenta y cien kilómetros por hora, los tamaños variaron entre; 250x250, 370x370, 480x480 y 640x640 pixeles de resolución.

4.1.1. Detección de puntos característicos (kps)

Esta prueba se realizó con el fin de obtener el valor de los puntos característicos en base al tamaño de la imagen, se pueden revisar los valores obtenidos de manera detallada en las tablas 1, 2 y 3. En forma adicional se

realizó la misma prueba con las imágenes patrones de las señales de tránsito incompletos y a diferentes tamaños de resolución obteniendo los resultados que se presentan en las tablas 4, 5 y 6. Para las pruebas de la tarjeta se detallan más adelante.

Tabla 1

Valores de puntos característicos para la señal de 50km/h.

Valor del Hessiano	Resolución de la imagen en pixeles			
	250X250	370X370	480X480	640X640
1000	903	2485	3160	3180
2000	741	1378	1378	1728
3000	730	981	1225	1225
4000	403	735	1176	1225
5000	266	503	820	990

Tabla 2

Valores de puntos característicos para la señal de 80km/h.

Valor del Hessiano	Resolución de la imagen en pixeles			
	250X250	370X370	480X480	640X640
1000	903	3403	4095	5253
2000	666	1378	1770	1740
3000	624	946	1485	1378
4000	601	746	931	978
5000	295	341	676	796

Tabla 3

Valores de puntos característicos para la señal de 100km/h.

Valor del Hessiano	Resolución de la imagen en pixeles			
	250X250	370X370	480X480	640X640
1000	780	946	965	1716
2000	666	561	580	1485
3000	564	528	542	739
4000	435	478	528	560
5000	270	308	388	492

Los datos obtenidos muestran el resultado de la suma de los valores de la

región donde se encuentra el punto característico, los valores obtenidos muestran que los puntos varían dependiendo de la resolución de la imagen o del valor umbral del Hessiano, los valores bajos significan que se trabajará con mayor número de puntos característicos, mientras que los valores altos significa que trabajara con un número menor de puntos de característicos, en este caso si se trabaja con un Hessiano de 1000 se tiene 200 puntos característicos, mientras que si se trabaja con Hessiano de 5000 se tiene 50 puntos característicos. Esto es debido a que el valor del Hessiano sirve para calcular y evaluar al determinante en ese valor, obteniendo como resultado el número de los puntos característicos y la suma de sus regiones.

En las pruebas con las imágenes de patrones incompletas, se procede a variar la resolución y el valor del Hessiano para de este modo determinar el valor de los puntos característicos.

Tabla 4

Valores de puntos característicos para la señal de 50 km/h.

Valor del Hessiano	Resolución de la imagen en pixeles			
	250X250	370X370	480X480	640X640
1000	321	351	455	512
2000	297	300	452	423
3000	286	300	378	390
4000	286	300	330	378
5000	226	276	286	286

Tabla 5

Valores de puntos característicos para la señal de 80 km/h.

Valor del Hessiano	Resolución de la imagen en pixeles			
	250X250	370X370	480X480	640X640
1000	321	595	455	512
2000	297	351	452	423
3000	286	300	378	390
4000	286	300	330	378
5000	226	300	286	286

Tabla 6

Valores de puntos característicos para la señal de 100 km/h.

Valor del Hessiano	Resolución de la imagen en pixeles			
	250X250	370X370	480X480	640X640
1000	120	120	157	190
2000	72	78	132	140
3000	72	78	118	127
4000	70	68	98	112
5000	48	55	67	98

Tomando en cuenta que los valores varían de manera notable cuando se cambian las resoluciones como el valor del Hessiano y el tipo del patrón, se optó por trabajar con las imágenes patrones completos y tamaño 370x370 pixeles de resolución.

Se eligió esa resolución primero porque es una imagen que brinda la suficiente información como para realizar la búsqueda de los objetos y en segundo lugar porque si se utiliza una imagen demasiado pequeña no hay información suficiente y de ser demasiado grande los retardos en el momento de procesar se incrementan.

4.2. Captura de video y tiempos de ejecución del algoritmo SURF

Para el presente trabajo se realizaron capturas de 320x240 pixeles, 480x320 pixeles, 640x480 pixeles y 1024x748 pixeles para realizar la búsqueda de los objetos en tiempo real y teniendo las imágenes patrón de resolución 370x370 pixeles, durante este proceso se tomaron los tiempos de ejecución para cada caso de resolución. Más adelante se aprecia de manera detallada los datos obtenidos tanto en el computador como en la tarjeta Beaglebone Black.

Los tiempos de ejecución obtenidos son un promedio de varias muestras que se tomaron, además hay que tener presente que cuando la memoria se satura los tiempos de ejecución se incrementan. La memoria se satura debido

a que los bloques de memoria cache que quedan libres al momento de la ejecución comienzan a realizar lectura del disco (en este caso se realiza lectura de las capturas de la cámara). Cuando el buffer se satura se debe liberarlo por medio del comando *sync* esto obliga a que los datos que se cargan en la cache se graben de manera automática en el disco (se escribe en el SWAP que es la memoria de intercambio, la cual tiene un rendimiento menor por ser una memoria virtual ya que se emplea para esto la unidad de almacenamiento) y se libere la memoria.

Esta saturación es debido a que se tienen demasiadas imágenes capturadas en el buffer y no se pueden procesar de manera inmediata. La ventaja de utilizar cualquier distribución de Linux es que cada 30 segundos se realiza un “flushes” (purgado) del buffer de manera automática aunque si se sobrecargan las entradas y salidas el uso del sync no mejoran las cosas.

Para determinar el tiempo ejecución durante el proceso de reconocimiento de imágenes con el algoritmo SURF, se han tomado capturas en diferentes tamaños de resolución en pixeles para de este modo ver cómo afecta le tamaño al momento de ejecutar la aplicación, las figuras 28, 29, 30 y 31 muestran las imágenes capturadas en tiempo real.

Figura 28: Captura con resolución 320x240 pixeles

Figura 29: Captura con resolución 480x320 pixeles

Figura 30: Captura con resolución 640x480 pixeles

Figura 31: Captura con resolución 1024x748 pixeles

En las capturas presentadas, se puede observar que las imágenes a diferentes resoluciones cambian y de manera muy notable, por lo tanto para realizar las pruebas de medición de tiempo de ejecución del algoritmo SURF se realizaron con las resoluciones de 640x480 y 1024x748 ya que con estas resoluciones se tiene una buena información para realizar el reconocimiento de las señales de tránsito de velocidad.

4.2.1 Pruebas en el computador

Esta prueba tiene el fin de medir los tiempos de ejecución del algoritmo SURF cuando se ejecuta en un computador de procesador i5, con 8 GB en memoria RAM y una versión de 64 bits para la distribución de Linux. Para determinar estos tiempos de ejecución se hicieron pruebas capturando imágenes de manera continua (30fps), cada segundo y cada medio segundo.

En esta prueba se realizaron las mediciones de los tiempos de ejecución del algoritmo SURF teniendo una imagen patrón de resolución 370x370 pixeles y con las imágenes capturadas utilizando la cámara con resolución de 640x480

y 1024x748 pixeles respectivamente.

La prueba es llevada a cabo independiente del ambiente, la distancia, el objeto patrón y dejando de lado el reconocimiento de las señales de tránsito de velocidad, esta prueba tiene la finalidad de medir los tiempos de ejecución del algoritmo SURF al implementarse en el computador y posteriormente en la tarjeta, para de este modo establecer tablas comparativas donde se tengan los datos del rendimiento de ambos equipos.

En las figura 32 y 33 se puede apreciar las capturas de la imagen del entorno. En el primer caso es una captura en resolución 640x480 y en segundo caso es una imagen capturada a una resolución de 1024x748 en diferentes ambientes.

Figura 32: Procesamiento con resolución 640x480

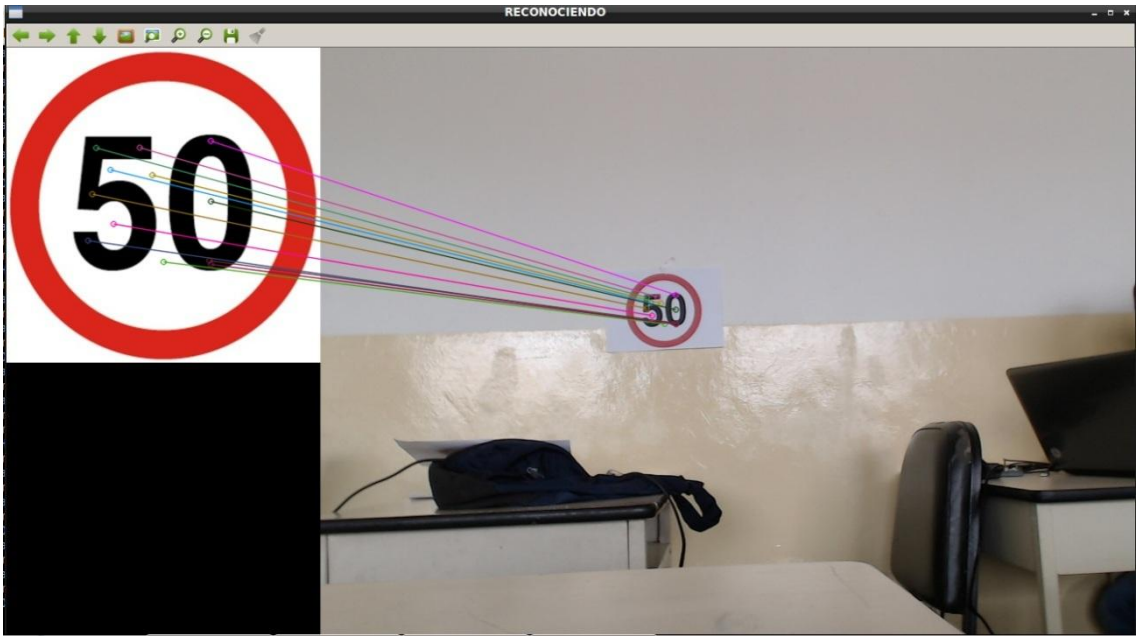

Figura 33: Procesamiento con resolución 1024x748

Las tablas 7, 8, 9 y 10 presentan los resultados de la medición de tiempos de ejecución del algoritmo SURF cuando se capturan las imágenes en diferentes tamaños de resolución en pixeles.

Tabla 7

Tiempos de ejecución (s) para 30fps

Valor del Hessiano	Tiempo de ejecución (s)	
	640x480 pixeles	1024x748 pixeles
1000	0,132652	0,134703
2000	0,131127	0,133626
3000	0,123734	0,126474
4000	0,117586	0,120969
5000	0,114219	0,118686

Los datos presentados en la tabla 7 son un promedio de cinco mediciones que se realizaron durante la ejecución del algoritmo SURF y se puede notar que los tiempos de ejecución si varían dependiendo de la resolución y del valor umbral del Hessiano.

Tabla 8

Tiempos de ejecución (s) para 2fps

Valor del Hessiano	Tiempo de ejecución (s)	
	640x480 pixeles	1024x748 pixeles
1000	0,124841	0,139802
2000	0,124161	0,126581
3000	0,115113	0,122071
4000	0,112579	0,116558
5000	0,104536	0,114396

Los datos del tiempo de ejecución obtenidos muestran una disminución al momento de hacer la lectura de 2fps y posterior procesamiento.

Tabla 9

Tiempo de ejecución (s) para 1fps

Valor del Hessiano	Tiempo de ejecución (s)	
	640x480 pixeles	1024x748 pixeles
1000	0,121842	0,128824
2000	0,121361	0,127369
3000	0,112113	0,122101
4000	0,108559	0,115059
5000	0,105436	0,109369

Los datos del tiempo de ejecución obtenidos al igual que en la tabla 9 muestran una variación mínima en los tiempos de procesamiento.

Tabla 10

Tiempos de ejecución (s) con RAM saturada

Valor del Hessiano	Tiempo de ejecución (s)	
	640x480 pixeles	1024x748 pixeles
1000	0,240652	0,395073
2000	0,213127	0,334626
3000	0,236734	0,323474
4000	0,258586	0,313969
5000	0,248219	0,294686

Los datos obtenidos para cada caso indican que el tiempo de ejecución varía de una manera notable en cuanto al valor umbral del Hessiano (para determinar el número de puntos característicos con los que se trabaja) y la resolución de la captura con la que se trabaja. Lo más relevante que se puede notar son los valores obtenidos cuando la memoria RAM se satura, en ese momento los tiempos de ejecución se incrementan de una manera muy notable, llegando ese tiempo de ejecución a ser casi el triple que cuando se ejecuta un par de veces de acuerdo a la aplicación.

4.2.2 Pruebas en la tarjeta Beaglebone Black

Para la medición de los tiempos de ejecución del algoritmo SURF cuando se ejecuta en la tarjeta Beaglebone Black que tiene un procesador cortex-A8, memoria RAM de 512 MB y una versión de 32 bits para la distribución de Linux, se realizaron pruebas capturando y leyendo imágenes de manera continua (30fps), cada dos segundos y cada segundo. Las tablas 11, 12 y 13 muestran los datos obtenidos en la prueba de medición de tiempo de ejecución.

Tabla 11

Tiempo de ejecución (s) para 30fps

Valor del Hessiano	Tiempo de ejecución (s)	
	640x480 pixeles	1024x748 pixeles
1000	4,850736	4,950736
2000	4,706269	4,842569
3000	3,534743	3,493743
4000	2,739696	2,823696
5000	2,306864	2.463864

Los datos obtenidos al momento de realizar la captura de 30fps y medir el tiempo de ejecución del algoritmo SURF al igual que en el computador no varían mucho en base a la resolución, pero si en base al valor del Hessiano con el que se trabaje para realizar el procesamiento.

Tabla 12

Tiempo de ejecución (s) para 2fps

Valor del Hessiano	Tiempo de ejecución (s)	
	640x480 pixeles	1024x748 pixeles
1000	4,650736	4,750736
2000	4,626269	4,742569
3000	3,434743	3,493743
4000	2,636961	2,823696
5000	2,306864	2.264864

Tabla 13

Tiempo de ejecución (s) para 1fps

Valor del Hessiano	Tiempo de ejecución (s)		
	640x480 pixeles		1024x748 pixeles
1000	4,450736	4,650736	4,450736
2000	4,086269	4,642569	4,086269
3000	3,534743	3,693743	3,534743
4000	2,639696	2,823696	2,639696
5000	2,306864	2,463864	2,306864

Los datos obtenidos indican que el tiempo de ejecución no depende del número de frames capturados al momento de realizar el procesamiento de imágenes con el algoritmo SURF, lo que si influye de manera directa es el valor del Hessiano y la resolución en pixeles de las capturas con las que se trabaja.

Una observación muy importante en esta prueba, es que los tiempos de ejecución del algoritmo SURF en la tarjeta Beaglebone Black en comparación con el computador se incrementan de una manera elevada, revisando los datos de las tablas presentadas anteriormente se puede decir que esta variación es de unas veinte veces más, esto es debido a los recursos de hardware y software que presenta la tarjeta y el computador.

Una caso particular es que si se lee cada uno de los 30fps capturados se va a tener un retraso en el procesamiento y en la presentación de las

imágenes ya que lo que se hace es procesar de manera continua cada uno de los frames capturados y luego presentarlos, es por esta razón que cuando se realiza la captura por defecto (captura con los parámetros definidos a 30 fps a una resolución de 640x480) se demora alrededor de 15 a 20 segundos para presentar un frame distinto que se ha tenido almacenado en la memoria, pero no es el actual. Para solucionar ese problema lo que se hace es mandar a leer de manera continua para en un determinado tiempo, seleccionar el último frame y procesarlo, esto genera únicamente el retraso al procesar la imagen con el algoritmo SURF.

Temando en cuenta los datos obtenidos en base a la resolución y en base al número de frames al momento de realizar la captura, se realizarán las diferentes pruebas para el reconocimiento de las señales de tránsito de velocidad en tiempo real con capturas de imágenes de escena de 640x480 y 1024x748 pixeles de resolución, ya que con este tamaño se puede obtener la información necesaria para el reconocimiento de objetos y se pretende establecer una distancia adecuada para tener un reconocimiento óptimo.

NOTA: No se puede definir los valores de 1 fps ni 2 fps, lo que se hace es capturar los frames que según la resolución especificada de la cámara permita capturar, pueden ser, quince, treinta o sesenta frames por segundo. Todos los frames se almacenan y posterior a eso se manda a leer únicamente el último frame que se ha obtenido, de esta manera se trata de tener un reconocimiento de objetos en tiempo real. Para ajustar el número de capturas y de hecho el rendimiento se fija el número de frames requeridos por software mediante la función *set.cap(CV_PROP_FRME_RATE, número de fps),* el número de frames que se capturaran dependen de la cámara.

4.3. Consumo de recursos de memoria del algoritmo SURF

El consumo de los recursos de memoria al momento de ejecutar el algoritmo SURF no se ven afectados por la resolución de captura, el número

de capturas y el valor del Hessiano, esos parámetros son irrelevantes al momento de ejecutar la aplicación (no influyen porque el consumo de memoria al momento de procesar se mantiene en un valor que casi no varía, pero se ven afectados los tiempos de ejecución) el factor más importante y que influye de manera directa es la adquisición de datos (capturas que se almacenan en el buffer), ya que al tener demasiados datos se sobrecargan las entradas y consecuente a eso la memoria RAM comienza a saturarse.

Consumo de recursos de memoria en el computador

Las figuras 34, 35 y 36 muestran el consumo de los recursos de la memoria RAM. Para ello se tienen los datos del estado de la memoria RAM previo a la ejecución de la aplicación, mientras se ejecuta y cuando la memoria se ha saturado debido a que se sobrecargan las entradas.

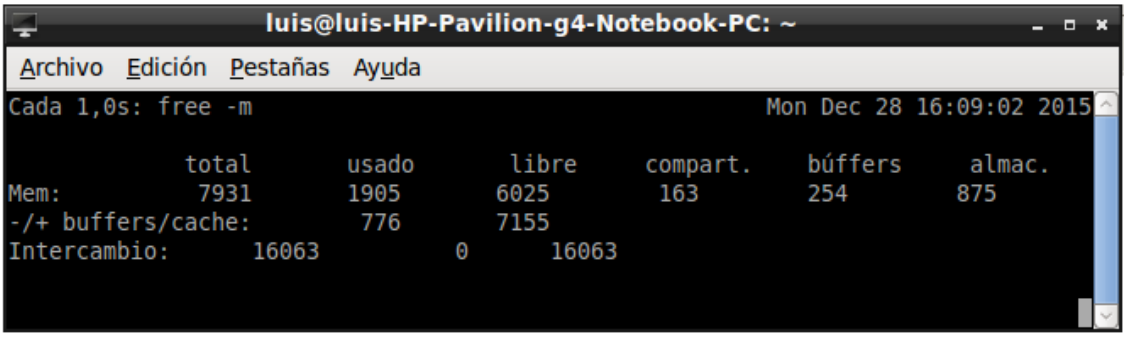

Figura 34: Recursos de memoria antes de ejecutar la aplicación con el algoritmo SURF.

En la figura 34 se puede observar que en la memoria RAM se está usando 1905 MB y se tienen libres 6025 MB, de igual manera los bloques que se están utilizando del buffer son 254 y los de almacenamiento (lectura de entradas) son 875, se debe tener muy en cuenta estos valores previos ya que al momento de ejecutar la aplicación estos valores varían y posterior a ello se debe establecer el consumo de la memoria RAM.

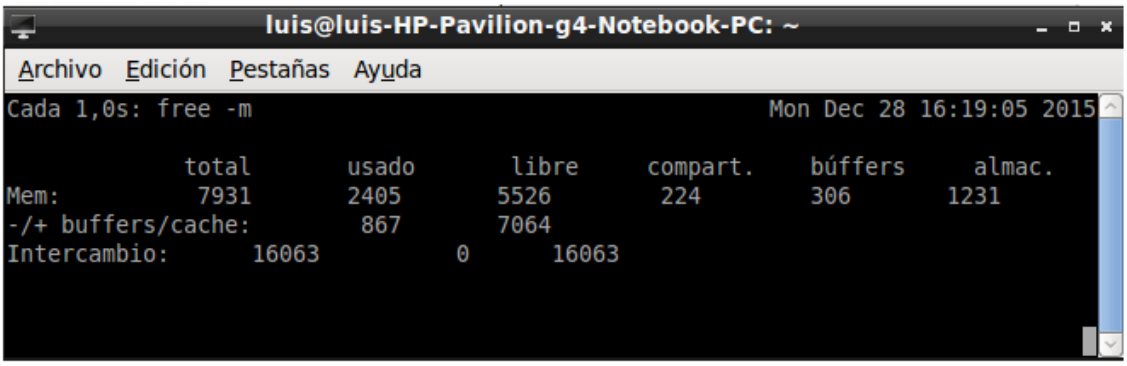

Figura 35: Recursos de memoria después de ejecutar la aplicación con el algoritmo SURF.

Las figuras 34 y 35 que se presentaron anteriormente indican que al momento de ejecutar el algoritmo SURF los valores de la memoria RAM cambian, ahora el valor de memoria en uso es de 2405 MB, esto quiere decir que el consumo de la memoria es de alrededor de 500 MB y de igual manera los bloques del buffer cambian a 306 es decir ocupan alrededor de 50 bloques más, y los de almacenamiento cambian a 1231 ocupando alrededor de 350 bloques más para las capturas de las imágenes.

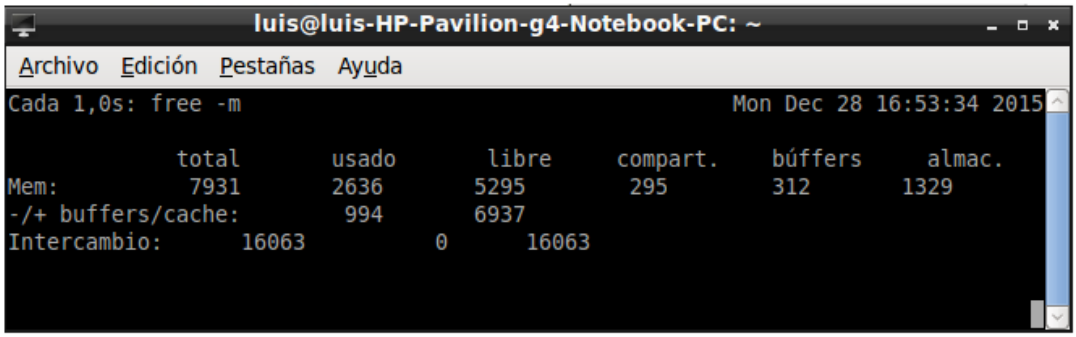

Figura 36: Recursos utilizados cuando se satura la memoria RAM.

Para el caso cuando la RAM se satura, los estados en uso de la RAM son de 2636 MB a diferencia de los 1905MB que se usaban antes de ejecutar la aplicación, esto quiere decir que el consumo es de alrededor de 700 MB, de igual manera los bloques del buffer que se ocupan son 312 y los bloques de almacenamiento para la lectura de imágenes es de 1329. Esto indica un

crecimiento notable en el uso de los recursos de la memoria. El estado de saturación dura alrededor de unos 10 segundos mientras el sistema operativo hace un flushes y libera la memoria.

Como se mencionó, el consumo de memoria para el procesamiento de imágenes no depende de la resolución ni del tipo de imagen. En la figura 37 se muestra los recursos de RAM que se consumen al procesar las imágenes mediante el algoritmo SURF.

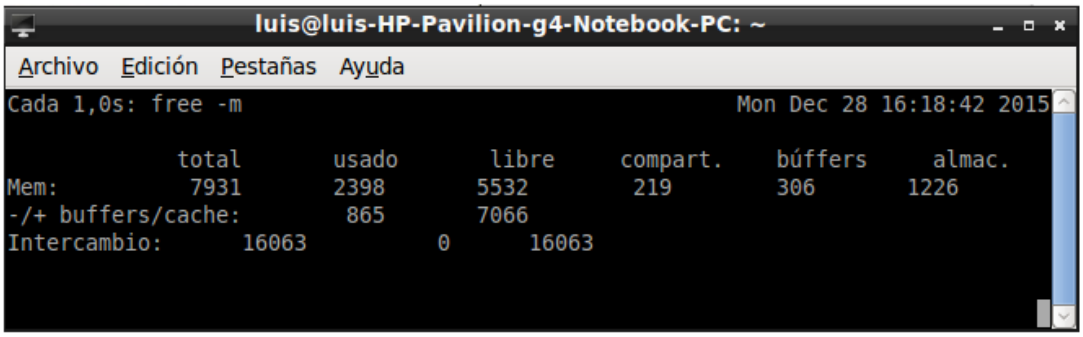

Figura 37: Recursos de memoria usados por el algoritmo SURF.

Los recursos utilizados no varían mucho, es decir se sigue con el consumo de memoria de más o menos 500 MB.

Consumo de recursos en la tarjeta Beaglebone Black

Una cosa muy importante al momento de implementar el algoritmo SURF en la tarjeta Beaglebone Black es que al tener recursos de hardware limitados y muy bajos la memoria se satura de entrada, en las figuras 38 y 39 se puede ver los recursos de memoria antes y durante la ejecución del algoritmo SURF.

Algo muy importante que se debe notar es el uso de los bloques de la cache, ya que es la que más se satura, en la figura 38 se ven los valores en bytes y en la figura 39 se aprecian los valores en kilobytes.

Figura 38: Recursos de Memoria en Beaglebone Black

Figura 39: Recursos de Memoria en Beaglebone Black al ejecutar aplicación

Una observación muy puntual al momento de revisar el consumo de los recursos de la memoria RAM de la tarjeta Beaglebone Black, es que se satura la memoria cache y el buffer, ocupando más bloques para la lectura de las entradas y para el procesamiento, esto ocasiona que el procesamiento sea más lento.

4.4. Distancia para el reconocimiento de las señales de tránsito de velocidad en la Tarjeta Beaglebone Black

Para el reconocimiento de las señales de tránsito de velocidad en tiempo real se trabajó con las señales de tamaño 55 cm x 55 cm (tamaño similar al real de la señal de tránsito de velocidad de clase B que se ubican en carreteras secundarias y dentro de la ciudad), y además se varia la distancia a razón de 1 metro con el fin de determinar la distancia mínimo y máximo en el que puede estar el objeto para el reconocimiento óptimo de las señales de tránsito de velocidad.

Otro parámetro a variar es la resolución y el ambiente en dónde se va realizar el reconocimiento de las señales de tránsito.

4.4.1. Distancias para detección con capturas de 640x480 pixeles

La captura a utilizar en esta prueba es de aproximadamente 0.3 mega pixeles ya que se va a trabajar con una imagen de tamaño 648x480, y se pretende obtener una buena distancia para el reconocimiento de los objetos ya que las señales de tránsito se encuentran ubicadas a una distancia moderada tomando en cuenta que el vehículo circula por la vía y la señal se encuentra en una vereda en el caso de la ciudad y tras la banqueta en las vías fuera de la ciudad.

Para cada una de las distancias en el reconocimiento se varía de metro a metro para ubicar al objeto y para cada variación se ha tomado una muestra de diez valores para determinar la incidencia de coincidencias que puedan existir, este procedimiento se ha realizado para cada una de las tres señales de tránsito de velocidad que se desean identificar.

Esta prueba se ha realizado en dos condiciones ambientales un poco diferentes, en el primer ambiente se tiene un escenario muy iluminado (intemperie, señales sin obstrucción luminosa) y en el otro ambiente se tiene un lugar con una luminosidad un tanto limitada (ambiente con sombra u obstrucción luminosa).

- **Primer ambiente**

Para este ambiente se llevó a cabo pruebas de reconocimiento de objetos cuando se tienen en la base de datos una, dos y tres señales a reconocer, de este modo se pretende conocer el rendimiento de la aplicación al momento de buscar objetos en base a una distancia y a la resolución de la captura.

En las tablas 14, 15 y 16 se presentan los resultados de las pruebas cuando se utiliza únicamente una señal de tránsito de velocidad en la base de datos y se la desea reconocer.

Tabla 14

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos (%)
1	10	198	100%
2	10	156	100%
3	10	210	100%
4	10	231	100%
5	10	206	100%
6	10	210	90%
7	10	187	90%
8	10	124	50%
9	10	69	0%
10	10	24	0%

En esta tabla los datos obtenidos indican que se puede tener un reconocimiento de objetos hasta los ocho metros, pero no de una manera tan óptima

Tabla 15

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos (%)
1	10	116	100%
2	10	124	100%
3	10	156	100%
4	10	212	100%
5	10	183	100%
6	10	137	90%
7	10	119	90%
8	10	78	60%
9	10	38	0%
10	10	2	0%

De igual manera se puede tener un reconocimiento de objetos hasta los ocho metros de distancia de donde se encuentra el objeto.

Tabla 16

Detección de la señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos (%)
1	10	106	100%
2	10	127	100%
3	10	131	100%
4	10	127	100%
5	10	98	90%
6	10	121	90%
7	10	125	90%
8	10	69	50%
9	10	17	0%
10	10	0	0%

Los resultados presentados en las tablas 14, 15 y 16 son casi similares, esto es debido a que se está trabajando con una única señal de tránsito para realizar el reconocimiento. Se puede decir que tiene una manera óptima de detectar hasta los cuatro metros de distancia y teniendo un alcance máximo de siete metros pero con una taza de incidencias al reconocer muy baja.

Para la siguiente prueba se tiene ya dos señales de tránsito a ser reconocidas a diferencia que en la prueba anterior que se tenía únicamente una señal de tránsito en la base de datos, se pretende tener una buena taza para el reconocimiento de los objetos y no existan falsos positivos y conlleven a un reconocimiento erróneo y para ello se han ajustado los umbrales en el reconocimiento. En las tablas 17 y 18 se muestran los resultados cuando se va a reconocer un objeto, con dos imágenes patrón en la base de datos. En este caso se han utilizado las señales de velocidad de cien y ochenta km/h.

Tabla 17

Detección de la señal de tránsito de 100km

Distancia (m)	# Muestras	Promedio Kps100Km/h	Promedio Kps80 km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	124	15	100%	0%

CONTINÚA

2	10	181	12	100%	0%
3	10	154	25	100%	0%
4	10	237	16	100%	0%
5	10	131	17	90%	0%
6	10	104	39	80%	10%
7	10	74	38	60%	10%
8	10	16	1	10%	0%
9	10	6	0	0%	0%
10	10	0	0	0%	0%

En esta prueba los umbrales para la detección se han aumentado comparando con el caso de que se tuviera una sola señal. Los resultados obtenidos de igual manera permiten un reconocimiento óptimo hasta los cuatro metros y defectuoso hasta los ocho metros.

Tabla 18

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps100Km/h	Promedio Kps80 km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	43	102	10%	90%
2	10	52	148	10%	90%
3	10	25	137	0%	100%
4	10	16	103	0%	100%
5	10	31	95	0%	90%
6	10	24	87	0%	80%
7	10	18	71	0%	70%
8	10	1	24	0%	10%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

En la tabla 18 se aprecia de manera clara que al momento de reconocer la señal de tránsito de ochenta km/h se generan varios falsos positivos, y esto es debido a que los patrones tienen en común el número cero y la circunferencia. Si se desea tener un mejor reconocimiento de las señales de tránsito de velocidad se debe incrementar el valor de umbral para el reconocimiento, pero con esto también se disminuye la distancia para el reconocimiento, en este caso para tener un reconocimiento óptimo se debe de fijar un umbral de más del setenta por ciento de la suma de los puntos característicos, pero al hacer

esto se tiene un reconocimiento únicamente en un rango que va de dos metros hasta los cinco metros. Para mejorar la distancia del reconocimiento se ha fijado que si se tiene un valor superior al cincuenta por ciento del valor de puntos característicos sea detectado.

De igual manera en las tablas 19 y 20 se muestran los resultados cuando se desea reconocer un objeto con dos imágenes en la base de datos. En este caso se han utilizado las señales de velocidad de cien y cincuenta km/h.

Tabla 19

Detección de la señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Promedio Kps100Km/h	Promedio Kps50 km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	113	35	100%	0%
2	10	121	48	100%	0%
3	10	118	67	90%	10%
4	10	125	73	90%	10%
5	10	119	36	90%	0%
6	10	102	28	80%	0%
7	10	98	21	70%	0%
8	10	66	9	10%	0%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

Los datos obtenidos muestran que la señal de tránsito de velocidad de cien km/h genera falsos positivos con la señal de transito de velocidad de cincuenta km/h, por este motivo se reconocen objetos de manera errónea. De igual manera, el umbral para identificar las señales, se incrementó con el fin de disminuir errores y con ello se afectó a la distancia para el reconocimiento.

Tabla 20

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps100Km/h	Promedio Kps50 km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	33	142	0%	100%
2	10	32	158	0%	100%

CONTINÚA

3	10	86	167	10%	90%
4	10	103	183	20%	80%
5	10	31	125	0%	80%
6	10	24	102	0%	80%
7	10	14	89	0%	70%
8	10	7	83	0%	20%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

De igual manera sucede que a señal de cincuenta km/h genera falsos positivos y se reconoce la señal de tránsito de cien km/h cuando no la hay.

En las tablas 21 y 22 se muestran los resultados. En este caso se han utilizado las señales de velocidad de 80 km/h y 50 km/h.

Tabla 21

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps80Km/h	Promedio Kps50 km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	253	39	100%	0%
2	10	297	45	100%	0%
3	10	203	77	90%	10%
4	10	168	83	80%	10%
5	10	114	36	90%	0%
6	10	125	28	80%	0%
7	10	96	21	70%	0%
8	10	87	67	20%	10%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

Como se puede observar los resultados obtenidos indican que existen falsos positivos al momento de realizar el reconocimiento de los objetos.

Tabla 22

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps80Km/h	Promedio Kps50 km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	13	125	0%	100%
2	10	21	137	0%	100%

CONTINÚA

3	10	18	152	0%	100%
4	10	25	121	10%	90%
5	10	26	109	10%	90%
6	10	67	102	10%	80%
7	10	18	89	0%	70%
8	10	8	23	0%	30%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

En la tabla 22 de igual manera se observa que al reconocer la señal de tránsito de cincuenta km/h genera falsos positivos y da un reconocimiento erróneo con la señal de ochenta Km/h.

En las tablas 23, 24 y 25 se presentan los datos obtenidos al reconocer un objeto, salvo que en esta ocasión, la base de datos presenta tres imágenes patrón y ahora existe mayor riesgo de tener más falsos positivos al momento de realizar el reconocimiento de las señales de tránsito de velocidad.

Un aspecto muy importante a tener presente en esta prueba es la similitud de los patrones, ya que se pueden confundir. Si se trabaja con el valor umbral de reconocimiento del setenta por ciento como valor mínimo de puntos característicos para el reconocimiento de objetos, se obtuvo valores para un reconocimiento a una distancia de entre dos metros a cinco metros, los demás valores no alcanzaban el porcentaje necesario para establecer el reconocmiento. Pero para mantener una buena distancia para el reconocimiento pero con algunos falsos positivos se decidió trabajar con el valor del cincuenta por ciento del valor umbral para el reconocimiento.

Las tablas 23, 24 y 25 presentan los datos obtenidos al momento de realizar el reconocimiento con las señales de cien, ochenta y cincuenta Km/h.

Tabla 23

Detección de la señal de tránsito de 100k/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	127	44	82	80%	0%	20%

CONTINÚA

2	10	128	76	70	80%	10%	10%
3	10	107	24	76	90%	0%	10%
4	10	112	21	46	90%	10%	0%
5	10	128	13	50	90%	0%	10%
6	10	122	15	25	80%	0%	10%
7	10	135	11	14	70%	0%	0%
8	10	89	47	34	20%	10%	20%
9	10	0	0	0	0%	0%	0%
10	10	0	0	0	0%	0%	0%

Al momento de realizar el reconocimiento de la señal de transito de velocidad de cien km/h se tiene una incidencia de valores erróneos con las otras dos señales de tránsito que se tiene en la base de datos. Los valores de aciertos con la señal que se está buscando ya no alcanzan el cien por ciento y es debido a que los objetos son similares y tiene mucha información que coincide.

En la tabla 23 los resultados obtenidos en comparación con las tablas 21 y 22 presentan de manera clara que las incidencias de falsos positivos se generan con mayor frecuencia en la señal de tránsito de cincuenta km/h que con las señal de ochenta Km/h cuando se está buscando e intentando reconocer la señal de tránsito de cien km/h.

Tabla 24

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	49	117	23	0%	100%	0%
2	10	40	108	24	0%	100%	0%
3	10	45	142	41	0%	90%	10%
4	10	33	131	36	0%	90%	10%
5	10	23	141	67	0%	90%	10%
6	10	17	145	73	0%	80%	20%
7	10	22	112	67	0%	70%	20%
8	10	11	78	78	0%	40%	30%
9	10	0	0	0	0%	0%	0%
10	10	0	0	0	0%	0%	0%

La tabla 24 muestra de manera clara que al momento de buscar la señal de velocidad de ochenta km/h no se ha generado reconocimientos erróneos con la señal de cien km/h ya que los valores que genera para esa señal no superan el umbral para dar error, pero los errores se generan con la señal de tránsito de cincuenta km/h.

Tabla 25

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	30	24	109	0%	0%	100%
2	10	34	12	126	0%	0%	100%
3	10	61	32	128	10%	0%	90%
4	10	25	36	109	10%	0%	80%
5	10	45	23	115	0%	0%	80%
6	10	36	14	97	0%	0%	70%
7	10	65	12	89	10%	0%	70%
8	10	13	45	67	0%	10%	40%
9	10	0	0	0	0%	0%	0%
10	10	0	0	0	0%	0%	0%

Los resultados obtenidos en las tablas 24 y 25 indican que la señal de tránsito de velocidad de cincuenta km/h es la que más falsos positivos genera al momento de realizar la búsqueda cuando se trabaja con tres señales para la búsqueda en la base de datos.

Uno de los aspectos más importantes al momento de realizar la detección de objetos cuando se trabaja con más de una señal de tránsito, se debe considerar el valor umbral de la suma de los puntos característicos ya que de este umbral depende si se reconoce o no la señal de tránsito de velocidad que se tenga presente.

En esta prueba se demostró que con una captura de 640x480 se tiene una distancia para el reconocimiento de objetos de hasta siete metros y además depende del número de imágenes patrón con las que se trabajan en la base de datos para realizar el reconocimiento.

- **Segundo ambiente**

En este ambiente las pruebas para realizar el reconocimiento de objetos se llevó a cabo de la misma manera que en el ambiente con iluminación a la intemperie, es decir se trabajó con una, dos y tres señales a la vez. Los datos obtenidos se presentan más adelante en las tablas respectivas según la prueba realizada.

El fin de esta prueba es la de determinar la distancia para el reconocimiento de objetos y también determinar cómo afecta la luz en el proceso de reconocimiento. En las tablas 26, 27 y 28 se presentan los datos obtenidos al reconocer un objeto, esta prueba es la misma que en el caso anterior para un ambiente iluminado.

Tabla 26

Detección de señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	292	80%
2	10	287	90%
3	10	150	90%
4	10	174	80%
5	10	144	80%
6	10	104	70%
7	10	79	60%
8	10	62	30%
9	10	17	0%
10	10	9	0%

Los valores obtenidos indican que ahora el reconocimiento de objetos no llega al cien por ciento, pero de igual manera la distancia para realizar el reconocimiento tiene un alcance de entre siete a ocho metros, aunque no sea un reconocimiento óptimo. Pero hay que tener presente que los reconocimientos logrados a ocho metros de distancia pueden ser ocasionados por falsos positivos.

Tabla 27

Detección de señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	131	80%
2	10	224	90%
3	10	239	80%
4	10	209	80%
5	10	131	80%
6	10	142	70%
7	10	71	60%
8	10	45	40%
9	10	9	0%
10	10	1	0%

Tabla 28

Detección de señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	150	90%
2	10	121	80%
3	10	180	80%
4	10	150	80%
5	10	155	80%
6	10	64	70%
7	10	46	50%
8	10	17	30%
9	10	5	0%
10	10	1	0%

En las tablas 27 y 28 se tienen resultados similares que en la tabla 26, esto es debido a que la iluminación si afecta en el reconocimiento de los objetos.

En las tablas 29 y 30 se muestran los resultados cuando se desea reconocer un objeto, pero ahora se tiene dos imágenes patrón en la base de datos a diferencia de la prueba anterior. En este caso se han utilizado las señales de velocidad de 100 km/h y 80 km/h.

Tabla 29

Detección de señal de tránsito de 100km

Distancia (m)	# Muestras	Promedio Kps 100 Km/h	Promedio Kps 80 km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	115	16	80%	0%
2	10	76	26	70%	0%
3	10	69	39	70%	10%
4	10	82	26	70%	0%
5	10	73	35	60%	0%
6	10	56	23	40%	0%
7	10	31	10	20%	0%
8	10	9	3	0%	0%
9	10	1	0	0%	0%
10	10	0	0	0%	0%

Tabla 30

Detección de señal de tránsito de 80 km/h

Distancia (m)	# Muestras	# Acierto 100 Km/h	# Acierto 80 km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	6	87	0%	80%
2	10	16	98	0%	70%
3	10	27	66	0%	70%
4	10	13	53	10%	60%
5	10	12	40	10%	50%
6	10	11	32	10%	40%
7	10	3	33	0%	20%
8	10	0	1	0%	0%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

En las tablas 29 y 30 los valores indican que en un ambiente menos iluminado el reconocimiento de objetos disminuye así como las detecciones erróneas con la otra señal.

En las tablas 31 y 32 se muestran los resultados cuando se desea reconocer un objeto, pero ahora se tienen dos imágenes en la base de datos. En este caso se han utilizado las señales de velocidad de 100 km/h y 50 km/h.

Tabla 31

Detección de la señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Promedio Kps 100 Km/h	Promedio Kps 50 km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	142	14	90%	0%
2	10	156	25	80%	0%
3	10	195	26	80%	0%
4	10	113	16	70%	10%
5	10	78	15	80%	0%
6	10	68	13	70%	0%
7	10	46	11	70%	0%
8	10	9	1	30%	0%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

Tabla 32

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio kps 100Km/h	Promedio Kp 50km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	23	80	0%	80%
2	10	45	98	10%	70%
3	10	56	87	10%	70%
4	10	16	65	0%	80%
5	10	18	78	0%	80%
6	10	15	56	0%	70%
7	10	8	12	0%	50%
8	10	1	0	0%	30%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

En las tablas 31 y 32 se parecía notablemente una disminución de aciertos errados con la otra señal en la base de datos, pero de igual manera el número de aciertos correctos disminuyó.

En las tablas 33 y 34 se muestran los resultados cuando se desea reconocer un objeto, pero ahora se tienen dos imágenes en la base de datos. En este caso se han utilizado las señales de velocidad de 80 km/h y 50 km/h.

Tabla 33

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps 80 Km/h	Promedio Kps 50 km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	102	17	80%	0%
2	10	106	21	90%	0%
3	10	95	29	80%	0%
4	10	78	28	80%	0%
5	10	61	31	80%	0%
6	10	45	17	70%	0%
7	10	39	4	50%	0%
8	10	6	1	30%	0%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

Tabla 34

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio kps 80 Km/h	Promedio Kps 50 km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	14	102	0%	90%
2	10	25	86	0%	90%
3	10	36	89	10%	80%
4	10	27	78	10%	80%
5	10	37	51	0%	70%
6	10	15	45	0%	60%
7	10	3	34	0%	50%
8	10	1	5	0%	40%
9	10	0	0	0%	0%
10	10	0	0	0%	0%

Los datos de la tablas 33 y 34 indican que la señal de tránsito de ochenta km/h no genera tantos falsos positivos con la señal de cincuenta km/h al momento de reconocerlos en comparación que la prueba anterior al identificar señales de cien y ochenta km/h en forma conjunta.

En las tablas 35, 36 y 37 se presentan los datos obtenidos al reconocer un objeto, salvo que en esta ocasión la base de datos presenta tres imágenes

patrón y existe riesgo de tener falsos positivos con otra señal de la base de datos.

Tabla 35

Detección de señal de tránsito de 100k/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	125	69	82	80%	20%	0%
2	10	129	71	60	80%	10%	10%
3	10	117	56	66	90%	10%	0%
4	10	119	45	56	90%	10%	0%
5	10	131	47	51	90%	10%	0%
6	10	132	57	35	80%	10%	0%
7	10	137	42	18	70%	0%	0%
8	10	93	73	44	20%	20%	10%
9	10	0	0	0	0%	0%	0%
10	10	0	0	0	0%	0%	0%

En la tabla 35 los datos obtenidos indican que al momento de realizar la detección de señales de tránsito de velocidad, al capturar una imagen con la señal de velocidad de cien km/h puede generar falsos positivos con la señal patrón de ochenta km/h al momento de la comparación y esto disminuye el nivel de aciertos para el objeto que se desea detectar.

Tabla 36

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	89	217	23	10%	90%	0%
2	10	61	318	24	10%	90%	0%
3	10	85	342	41	10%	90%	0%
4	10	63	373	36	10%	90%	0%
5	10	63	347	27	10%	90%	0%
6	10	57	344	33	10%	80%	0%
7	10	52	192	27	10%	60%	0%
8	10	45	78	28	0%	40%	0%
9	10	0	0	0	0%	0%	0%
10	10	0	0	0	0%	0%	0%

Con la señal de cincuenta km/h no existen muchos aciertos erróneos ya que esta señal no presenta muchos puntos característicos que se asemejen a las señales de cien y de 80 km/h.

Tabla 37

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	35	44	149	0%	0%	100%
2	10	64	32	176	0%	0%	100%
3	10	8	34	198	10%	0%	90%
4	10	15	39	149	10%	0%	80%
5	10	35	23	165	0%	0%	80%
6	10	16	24	97	0%	0%	70%
7	10	65	22	89	10%	0%	70%
8	10	3	45	67	0%	10%	40%
9	10	0	0	0	0%	0%	0%
10	10	0	0	0	0%	0%	0%

Los valores de la tabla al momento de buscar la señal de velocidad de cincuenta km/h generan aciertos erróneos con la señal de cien km/h, pero menores en comparación al buscar la señal de 80 km/h.

Al realizar la prueba con una captura de 640x480 pixeles de resolución, los datos obtenidos indican que la luz es un factor que influye en la identificación, es decir, mientras mayor sea la luminosidad, se refleja con mayor intensidad en el objeto y no permite un reconocimiento óptimo, si la iluminación es demasiado baja ocurre algo similar, no se logra capturar de manera adecuada toda la información y por consiguiente no hay información relevante o se generan falsos positivos.

Al tener ambientes con una iluminación moderada, las imágenes capturadas ofrecen la información necesaria para al momento de realizar el reconocimiento de objetos no se generen falsos positivos y así tener un rendimiento óptimo de la aplicación.

4.4.2. Distancias para detección con capturas de 1024x748 pixeles

Esta prueba se la realizó con el objetivo de mejorar el reconocimiento de objetos y con eso aumentar la distancia de reconocimiento. Se debe tener presente que con aumentar la resolución de la imagen de la trama de video aumenta también el tiempo de ejecución. En esta prueba, al igual que la anterior, se varió la distancia y se tomaron diez muestras de medición por cada una de las distancias por objeto. De este modo, a más de establecer la distancia para el reconocimiento de objetos, se pretende conocer el rendimiento del sistema cuando e trabaja con una imagen mucho más grande que en el caso anterior.

Para esto, los objetos de las señales de tránsito se sometieron a dos ambientes diferentes al igual que en la prueba anterior. A continuación se detallan cada una de las pruebas para cada uno de los ambientes.

- **Primer ambiente**

En este ambiente se procede de manera similar que en la prueba anterior, variando la distancia cada metro hasta llegar a una distancia donde ya no se reconozca al objeto. Para esto se tiene en la base de datos una señal, dos señales y tres señales de tránsito de velocidad para realizar el reconocimiento.

En las tablas 38, 39 y 40 se presentan los resultados de las pruebas cuando se utiliza una señal de tránsito de velocidad en la base de datos para su reconocimiento.

Tabla 38

Detección de señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	301	100%
2	10	723	100%
3	10	599	100%
4	10	467	100%
5	10	401	100%

CONTINÚA

6	10	574	100%
7	10	515	100%
8	10	469	100%
9	10	364	100%
10	10	354	90%
11	10	401	90%
12	10	397	60%
13	10	144	30%
14	10	0	0%
15	10	0	0%

Los datos obtenidos en la prueba indican de manera clara, cómo el valor de los puntos característicos se incrementan al igual que la distancia para el reconocimiento cuando se trabaja con una resolución mayor, teniendo también una mejor taza de reconocimiento.

Tabla 39

Detección de señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	362	90%
2	10	876	100%
3	10	784	100%
4	10	570	100%
5	10	630	100%
6	10	745	100%
7	10	863	100%
8	10	679	100%
9	10	848	100%
10	10	543	100%
11	10	253	90%
12	10	147	70%
13	10	123	20%
14	10	0	0%
15	10	0	0%

De igual manera se aprecia un incremento de puntos característicos y de aciertos con la señal de ochenta km/h.

Tabla 40

Detección de señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	238	90%
2	10	420	100%
3	10	350	100%
4	10	326	100%
5	10	349	100%
6	10	299	100%
7	10	342	100%
8	10	282	100%
9	10	204	100%
10	10	180	90%
11	10	188	80%
12	10	167	70%
13	10	132	20%
14	10	0	0%
15	10	0	0%

Los resultados obtenidos en las tablas 38, 39 y 40 indican que al tener un solo patrón en la base de datos cuando se hace reconocimiento de objetos hay menos riesgos de que se generen falsos positivos.

En las tablas 41 y 42 se muestran los resultados cuando hace reconocimiento de objetos y se tiene dos imágenes patrón en la base de datos. En este caso se han utilizado las señales de velocidad de 100 km/h y 80 km/h

Tabla 41

Detección de señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Promedio kps100Km/h	Promedio Kps80km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	291	44	90%	0%
2	10	479	34	100%	0%
3	10	239	107	90%	10%
4	10	725	113	90%	10%
5	10	672	65	100%	0%
6	10	512	23	100%	0%

CONTINÚA

7	10	582	14	100%	0%
8	10	512	24	100%	0%
9	10	466	23	100%	0%
10	10	374	18	90%	0%
11	10	267	11	90%	0%
12	10	114	0	70%	0%
13	10	12	0	40%	0%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

Los datos de la tabla 41 indican que hay aciertos erróneos con la señal de tránsito de ochenta km/h, esto es debido a que al generar el valor de puntos característicos, la señal de cien km/h no generó el valor del umbral suficiente y se generaron errores.

Tabla 42

Detección de señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio kps100Km/h	Promedio Kp80km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	78	245	10%	90%
2	10	87	354	10%	90%
3	10	56	476	0%	100%
4	10	78	783	10%	90%
5	10	54	367	0%	100%
6	10	56	352	0%	100%
7	10	59	358	0%	100%
8	10	98	308	10%	90%
9	10	78	262	10%	90%
10	10	67	187	10%	80%
11	10	23	165	0%	70%
12	10	15	143	0%	70%
13	10	8	78	0%	30%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

Al reconocer la señal de ochenta km/h, los datos indican que con la señal de cien km/h genera resultados erróneos, esto es porque la señal de ochenta km/h tiene información similar a la señal de cien km/h. Debido a esta razón los resultados erróneos se incrementan al momento de buscar la señal de ochenta

km/h en comparación con la de 100km/h. Esto también se debe a que primero se manda a comparar la captura con la señal patrón de cien y luego con la de ochenta km/h.

En las tablas 43 y 44 se muestran los datos obtenidos cuando se hace el reconocimiento de objetos con dos imágenes en la base de datos. En este caso se han utilizado las señales de velocidad de 100 km/h y 50 km/h.

Tabla 43

Detección de la señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Promedio kps100Km/h	Promedio Kps50km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	413	23	100%	0%
2	10	352	12	90%	0%
3	10	370	41	100%	0%
4	10	531	50	100%	0%
5	10	349	62	90%	10%
6	10	203	24	100%	0%
7	10	301	26	100%	0%
8	10	356	13	100%	0%
9	10	220	11	90%	0%
10	10	231	13	90%	0%
11	10	167	9	50%	0%
12	10	123	5	40%	0%
13	10	0	0	0%	0%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

Los datos obtenidos en la tabla 43 indican que con la señal de tránsito de velocidad de 100km/h tiene un número de aciertos muy elevado y casi sin tener falsos positivos respecto a la otra señal de tránsito de velocidad. Una parte muy puntal que hay que recalcar es que se manda a realizar la comparación primero con la imagen patrón de cien km/h y luego con la de cincuenta km/h, y es debido a este orden que se tienen aciertos de manera óptima con la señal de 100 km/h.

Tabla 44

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio kps100Km/h	Promedio Kps50km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	78	356	10%	90%
2	10	34	345	0%	100%
3	10	65	567	0%	100%
4	10	51	487	0%	100%
5	10	57	486	0%	100%
6	10	34	453	0%	100%
7	10	45	423	10%	90%
8	10	68	418	20%	80%
9	10	57	387	10%	70%
10	10	46	282	10%	70%
11	10	13	201	0%	60%
12	10	15	68	10%	30%
13	10	0	0	0%	0%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

De igual manera que al hacer el reconocimiento con la señal de ochenta km/h, se tienen resultados erróneos con la señal de cincuenta km/h, esto se debe primero al orden en el que se realiza la comparación de las capturas con los objetos y segundo porque las imágenes tienen en común ciertas características y esto influye en los errores generados.

En las tablas 45 y 46 se muestran los resultados cuando se desea reconocer un objeto, pero ahora se tienen dos imágenes en la base de datos. En este caso se han utilizado las señales de velocidad de 80 km/h y 50 km/h.

Tabla 45

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio kps80Km/h	Promedio Kps50km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	523	54	90%	0%
2	10	573	62	100%	0%
3	10	685	51	100%	0%

CONTINÚA

4	10	783	67	100%	0%
5	10	832	32	100%	0%
6	10	692	56	100%	0%
7	10	498	53	100%	0%
8	10	582	35	100%	0%
9	10	462	45	90%	10%
10	10	396	67	80%	10%
11	10	297	89	70%	20%
12	10	201	84	50%	20%
13	10	0	0	0%	0%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

La señal de ochenta km/h tiene un gran número de aciertos favorables porque se manda a comparar con la imagen patrón de esta señal antes que mandarla a comparar con la señal de cincuenta km/h.

Tabla 46

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio kps80Km/h	Promedio Kps50km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	81	213	10%	90%
2	10	76	288	10%	90%
3	10	52	358	0%	100%
4	10	42	286	0%	100%
5	10	45	368	0%	100%
6	10	36	314	0%	100%
7	10	79	273	10%	90%
8	10	56	235	10%	90%
9	10	84	196	20%	80%
10	10	98	158	20%	80%
11	10	108	190	20%	70%
12	10	73	85	20%	30%
13	10	0	0	0%	0%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

Al momento de reconocer la señal de cincuenta km/h, ésta genera resultados erróneos con la señal de ochenta km/h, lo que se debe a que al

momento de realizar la búsqueda se comparan primero con la de ochenta y luego con la de cincuenta.

En las tablas 47, 48 y 49 se presentan los datos obtenidos cuando se hace el reconocimiento de objetos, salvo que en esta ocasión la base de datos presenta tres imágenes patrón y el riesgo de tener falsos positivos con otra señal de la base de datos se incrementará, pero para no tener un exceso de aciertos erróneos también se incrementó el umbral de condición de reconocimiento.

Tabla 47

Detección de señal de tránsito de 100k/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	533	292	372	90%	0%	0%
2	10	583	497	197	90%	10%	0%
3	10	563	401	201	90%	10%	0%
4	10	488	596	298	80%	20%	0%
5	10	456	597	290	80%	10%	10%
6	10	476	492	294	80%	10%	10%
7	10	536	583	284	90%	10%	0%
8	10	498	384	284	90%	10%	0%
9	10	332	454	154	70%	20%	0%
10	10	501	426	124	80%	10%	0%
11	10	598	331	235	60%	10%	10%
12	10	237	322	123	50%	10%	0%
13	10	0	112	112	0%	20%	20%
14	10	0	0	0	0%	0%	0%
15	10	0	0	0	0%	0%	0%

La tabla 47 indica que al momento de realizar la detección de la señal de cien km/h se generan falsos positivos en su mayor parte con la señal de ochenta km/h y unos pocos con la señal de cincuenta km/h dando como resultado un índice de aciertos menor que cuando se realizaba la detección con dos imágenes patrón en la base de datos.

Tabla 48

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	143	292	372	0%	90%	10%
2	10	132	497	197	0%	90%	10%
3	10	218	401	201	0%	90%	10%
4	10	196	596	298	10%	80%	0%
5	10	201	597	290	0%	80%	0%
6	10	162	492	294	20%	80%	0%
7	10	142	583	284	0%	90%	0%
8	10	123	384	284	0%	90%	0%
9	10	189	454	154	10%	70%	0%
10	10	132	426	124	0%	80%	0%
11	10	192	331	235	0%	60%	10%
12	10	135	322	123	10%	50%	0%
13	10	4	112	112	0%	20%	0%
14	10	0	0	0	0%	0%	0%
15	10	0	0	0	0%	0%	0%

Al capturar una imagen con la señal de tránsito de ochenta km/h se nota claramente como los aciertos erróneos disminuyen en comparación con los resultados obtenidos en la tabla 47.

Tabla 49

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps 100Km/h	Promedio Kps 80Km/h	Promedio Kps 50Km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	113	143	492	0%	0%	90%
2	10	106	132	497	0%	10%	90%
3	10	130	218	401	0%	10%	90%
4	10	185	196	498	10%	10%	80%
5	10	101	201	390	0%	20%	80%
6	10	209	162	394	20%	0%	80%
7	10	112	142	384	0%	0%	90%
8	10	132	123	384	0%	0%	90%
9	10	165	189	454	10%	10%	70%
10	10	122	132	424	0%	0%	80%
11	10	102	192	435	0%	10%	60%

CONTINÚA

12	10	98	135	423	10%	10%	50%
13	10	81	204	412	0%	20%	20%
14	10	0	0	0	0%	0%	0%
15	10	0	0	0	0%	0%	0%

Al momento de realizar la detección de la señal de tránsito de cincuenta km/h se observa que en una gran parte se generan aciertos erróneos con las dos señales restantes, pero de igual manera es alto el número de aciertos favorables.

- **Segundo ambiente**

En este ambiente, de igual manera, se ha llevado a cabo el reconocimiento de los objetos para cuando se tienen una, dos y tres imágenes patrón en la base de datos. Los datos obtenidos se presentan más adelante de la misma manera que se ha realizado en las pruebas anteriores.

En las tablas 50, 51 y 52 se presentan los datos obtenidos al realizar el reconocimiento de objetos cuando se tiene un patrón en la base de datos.

Tabla 50

Detección de señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	236	100%
2	10	553	100%
3	10	989	100%
4	10	1041	100%
5	10	853	100%
6	10	548	100%
7	10	560	100%
8	10	504	90%
9	10	480	90%
10	10	488	90%
11	10	385	90%
12	10	357	70%
13	10	353	30%
14	10	35	0%

CONTINÚA

15	10	0	0%

Los datos de la tabla 50 indican un reconocimiento óptimo en los valores de distancia cercanos al dispositivo de captura y a medida que se aleja disminuye ese número de aciertos, en promedio se puede decir que la distancia de reconocimiento aceptable es de 12 metros.

Tabla 51

Detección de señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	580	100%
2	10	920	100%
3	10	790	100%
4	10	520	100%
5	10	1100	100%
6	10	9060	100%
7	10	636	100%
8	10	567	90%
9	10	609	90%
10	10	608	90%
11	10	514	70%
12	10	620	60%
13	10	579	30%
14	10	95	0%
15	10	0	0%

De igual manera que con la señal anterior, la señal de ochenta km/h, al momento de realizar la detección, se detecta sin problemas hasta una distancia de doce metros y con un nivel de inciertos aceptable.

Tabla 52

Detección de señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Kps Promedio	Aciertos(%)
1	10	310	100%
2	10	450	100%
3	10	437	100%
4	10	268	100%

CONTINÚA

5	10	250	100%
6	10	280	100%
7	10	251	90%
8	10	223	90%
9	10	270	90%
10	10	242	90%
11	10	234	80%
12	10	198	80%
13	10	132	60%
14	10	95	0%
15	10	95	0%

Los datos obtenidos en las tres tablas 50, 51 y 52 indican un índice alto de aciertos al momento de realizar el reconocimiento de objetos cuando se tiene una única imagen de patrón en la base de datos.

En las tablas 53 y 54 se muestran los resultados cuando se hace el reconocimiento de dos objetos y se tiene dos imágenes patrón en la base de datos a diferencia de la prueba anterior. En ésta se han utilizado las señales de tránsito de velocidad de cien km/h y ochenta km/h.

Tabla 53

Detección de señal de tránsito de 100km

Distancia (m)	# Muestras	Promedio Kps100km/h	Promedio Kps80km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	310	90	90%	10%
2	10	450	150	90%	10%
3	10	437	137	90%	10%
4	10	268	168	90%	10%
5	10	250	150	80%	10%
6	10	280	180	90%	0%
7	10	251	151	90%	0%
8	10	223	123	90%	0%
9	10	270	170	90%	0%
10	10	242	142	90%	0%
11	10	234	134	80%	0%
12	10	198	198	60%	0%
13	10	65	76	0%	0%
14	10	0	0	0%	0%

CONTINÚA

15	10	0	0	0%	0%

Los resultados de la tabla 53 indican un cierto número de resultados erróneos al acertar con la señal de ochenta km/h y no reconocer de forma óptima la señal de cien km/h. De igual manera se tiene un alcance para el reconocimiento de hasta 12 metros de distancia. Pero en esta distancia los resultados obtenidos en su gran parte son por falsos positivos que generan los objetos que se encuentran en el ambiente.

Tabla 54

Detección de señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps100km/h	Promedio Kps80km/h	Aciertos(%) 100km/h	Aciertos(%) 80km/h
1	10	78	245	10%	90%
2	10	87	354	10%	90%
3	10	56	476	0%	100%
4	10	78	783	10%	90%
5	10	54	367	0%	100%
6	10	56	352	0%	100%
7	10	59	358	0%	100%
8	10	98	308	10%	90%
9	10	78	262	10%	90%
10	10	67	187	10%	80%
11	10	23	165	0%	70%
12	10	15	143	0%	70%
13	10	8	78	0%	30%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

De la tabla 54 al igual que en las anteriores, vale resaltar el número de resultados erróneos que se generan al momento de hacer el reconocimiento de objetos es similar, aunque existan errores, el número de resultados correctos se encuentra dentro de un rango aceptable.

En las tablas 55 y 56 se muestran los resultados cuando se hace el reconocimiento de objetos con dos imágenes patrón en la base de datos. En

este caso se han utilizado las señales de velocidad de cien km/h y cincuenta km/h.

Tabla 55

Detección de la señal de tránsito de 100 km/h

Distancia (m)	# Muestras	Promedio Kps100km/h	Promedio Kps50km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	331	64	100%	0%
2	10	314	123	100%	0%
3	10	452	111	100%	0%
4	10	497	132	100%	0%
5	10	670	64	100%	0%
6	10	915	67	100%	0%
7	10	585	47	100%	0%
8	10	468	185	90%	10%
9	10	620	25	90%	0%
10	10	652	23	90%	0%
11	10	384	0	90%	0%
12	10	329	0	80%	0%
13	10	268	0	80%	0%
14	10	112	0	10%	0%
15	10	97	0	0%	0%

Considerando en la base de datos las señales de cien y cincuenta km/h, al reconocer la señal de cien km/h no se tiene muchos resultados erróneos,

Tabla 56

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps 100km/h	Promedio Kps 50km/h	Aciertos(%) 100km/h	Aciertos(%) 50km/h
1	10	100	411	0%	100%
2	10	125	325	0%	100%
3	10	119	432	0%	100%
4	10	102	534	0%	100%
5	10	80	660	0%	100%
6	10	181	468	10&	90%
7	10	135	620	10%	90%
8	10	71	653	0%	90%
9	10	97	394	0%	90%
10	10	130	369	0%	90%

CONTINÚA

11	10	100	358	0%	90%
12	10	67	331	0%	80%
13	10	0	85	0%	80%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

A diferencia de buscar la señal de cien km/h, con la señal de cincuenta km/h se tiene un índice menor de aciertos ya que se generan falsos positivos con la señal de cien km/h provocando resultados erróneos.

En las tablas 57 y 58 se muestran los resultados cuando se hace el reconocimiento de objetos, pero ahora se tienen dos imágenes en la base de datos. En este caso se han utilizado las señales de velocidad de ochenta km/h y cincuenta km/h.

Tabla 57

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio kps 80 Km/h	Promedio Kps 50 km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	245	150	90%	10%
2	10	354	165	90%	10%
3	10	476	179	100%	0%
4	10	783	182	90%	10%
5	10	367	180	100%	0%
6	10	352	281	90%	0%
7	10	358	235	90%	0%
8	10	308	171	90%	10%
9	10	362	197	90%	10%
10	10	287	230	90%	10%
11	10	265	266	80%	0%
12	10	143	267	80%	0%
13	10	78	0	30%	0%
14	10	0	0	0%	0%
15	10	0	0	0%	0%

En la tabla 57 los datos indican como el número de detecciones erróneas se generan cuando se tienen en la base de datos las señales, la de cincuenta y ochenta km/h, esos resultados erróneos son consecuencia de que las señalas

que se tienen en la base de datos presentan características físicas similares entre sí.

Tabla 58

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps 80 Km/h	Promedio Kps 50 km/h	Aciertos(%) 80km/h	Aciertos(%) 50km/h
1	10	100	411	0%	100%
2	10	125	325	0%	100%
3	10	119	432	0%	100%
4	10	102	534	0%	100%
5	10	80	560	0%	100%
6	10	181	468	10&	90%
7	10	135	620	10%	90%
8	10	71	653	0%	90%
9	10	97	394	0%	90%
10	10	130	369	0%	90%
11	10	100	358	0%	90%
12	10	67	331	0%	80%
13	10	0	285	0%	50%
14	10	0	199	0%	0%
15	10	0	135	0%	0%

La diferencia que cuando se desea detectar la señal de ochenta km/h al buscar la señal de cincuenta km/h es que el número de aciertos es menor, esto se debe a que la señal de cincuenta km/h genera menos coincidencias y puntos falsos en comparación a la de ochenta km/h.

En las tablas 59, 60 y 61 se presentan los datos obtenidos al reconocer un objeto, salvo que en esta ocasión la base de datos presenta tres imágenes patrón y ahora existe riesgo de tener falsos positivos con las otras dos señal que no corresponden al objeto que en realidad es.

Tabla 59

Detección de señal de tránsito de 100k/h

Distancia (m)	# Muestras	Promedio Kps 100 Km/h	Promedio Kps 80 km/h	Promedio kps 50 km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h

CONTINÚA

1	0%	392	233	272	90%	0%	10%
2	0%	397	337	197	90%	0%	10%
3	0%	501	263	201	90%	0%	10%
4	10%	496	288	298	80%	10%	0%
5	0%	497	256	190	80%	0%	0%
6	20%	492	276	194	80%	20%	0%
7	0%	583	236	184	90%	0%	0%
8	0%	384	298	184	90%	0%	0%
9	10%	454	232	104	70%	10%	0%
10	0%	426	101	135	80%	0%	0%
11	0%	331	398	265	60%	0%	10%
12	10%	322	237	126	50%	10%	0%
13	0%	112	0	112	20%	0%	0%
14	0%	0	0	0	0%	0%	0%
15	0%	0	0	0	0%	0%	0%

Los datos muestran claramente que al momento de realizar el reconocimiento de la señal de tránsito de velocidad de cien km/h se tiene un alto número de detecciones erróneas con la señal de ochenta y cincuenta km/h.

Tabla 60

Detección de la señal de tránsito de 80 km/h

Distancia (m)	# Muestras	Promedio Kps 100 Km/h	Promedio Kps 80 km/h	Promedio Kps 50 km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	123	392	372	0%	90%	10%
2	10	143	457	197	0%	90%	10%
3	10	212	461	231	0%	90%	10%
4	10	134	536	198	10%	80%	0%
5	10	223	577	190	0%	80%	0%
6	10	136	422	199	20%	80%	0%
7	10	147	513	178	0%	90%	0%
8	10	134	324	188	0%	90%	0%
9	10	154	414	156	10%	70%	0%
10	10	165	426	125	0%	80%	0%
11	10	173	311	235	0%	60%	10%
12	10	145	322	123	10%	50%	0%
13	10	204	112	112	0%	20%	0%
14	10	0	0	0	0%	0%	0%
15	10	0	0	0	0%	0%	0%

De igual manera que al reconocer la señal de cien km/h, la señal de ochenta km/h genera falsos positivos con las dos restantes señales, siempre van a aparecer estas detecciones erróneas debido a que hay mucha información en común entre los patrones y los objetos.

Tabla 61

Detección de la señal de tránsito de 50 km/h

Distancia (m)	# Muestras	Promedio Kps 100 Km/h	Promedio kps 80 km/h	Promedio kps 50 km/h	Aciertos(%) 100 km/h	Aciertos(%) 80 km/h	Aciertos(%) 50 km/h
1	10	113	143	492	0%	0%	90%
2	10	106	132	497	0%	10%	90%
3	10	130	218	401	0%	10%	90%
4	10	185	196	498	10%	10%	80%
5	10	101	201	390	0%	20%	80%
6	10	209	162	394	20%	0%	80%
7	10	112	142	384	0%	0%	90%
8	10	132	123	384	0%	0%	90%
9	10	165	189	454	10%	10%	70%
10	10	122	132	424	0%	0%	80%
11	10	102	192	435	0%	10%	60%
12	10	98	135	423	10%	10%	50%
13	10	81	204	412	0%	20%	20%
14	10	0	0	0	0%	0%	0%
15	10	0	0	0	0%	0%	0%

Los resultados de la tabla 61 de igual manera que en las anteriores, indica un cierto número de errores al buscar el objeto aunque el valor de aciertos aún sigue siendo alto y aceptable. Con esto se comprueba que al momento de realizar el reconocimiento de objetos con el algoritmo SURF la iluminación es de gran importancia ya que varían las propiedades de los objetos y por lo tanto el resultado de la búsqueda.

4.4.3. Influencia de la inclinación y rotación del objeto en el reconocimiento

Conociendo que la luz es un factor que influye de manera directa al momento de reconocer los objetos, mediante estas pruebas se pretende

determinar cómo influye el grado de inclinación y la rotación en la que el objeto a ser reconocido puede ubicarse al momento de realizar la captura para el posterior reconocimiento. Para estas pruebas se han tomado en consideración tres importantes posiciones del objeto y de igual manera las distancias en las que se hacen la toma de datos.

- **Inclinaciones laterales**

En esta prueba se han variado los parámetros de inclinación, distancia y resolución y utilizando cualquiera de los tres objetos al momento del reconocimiento. En las figuras 40 y 41 se muestran las capturas y los datos de aciertos que se han realizado durante este proceso utilizando la tarjeta Beaglebone Black, para este caso se tienen dos imágenes en la base de datos para reconocer los objetos. Se pueden observar claramente la suma de los valores resultantes de esta prueba.

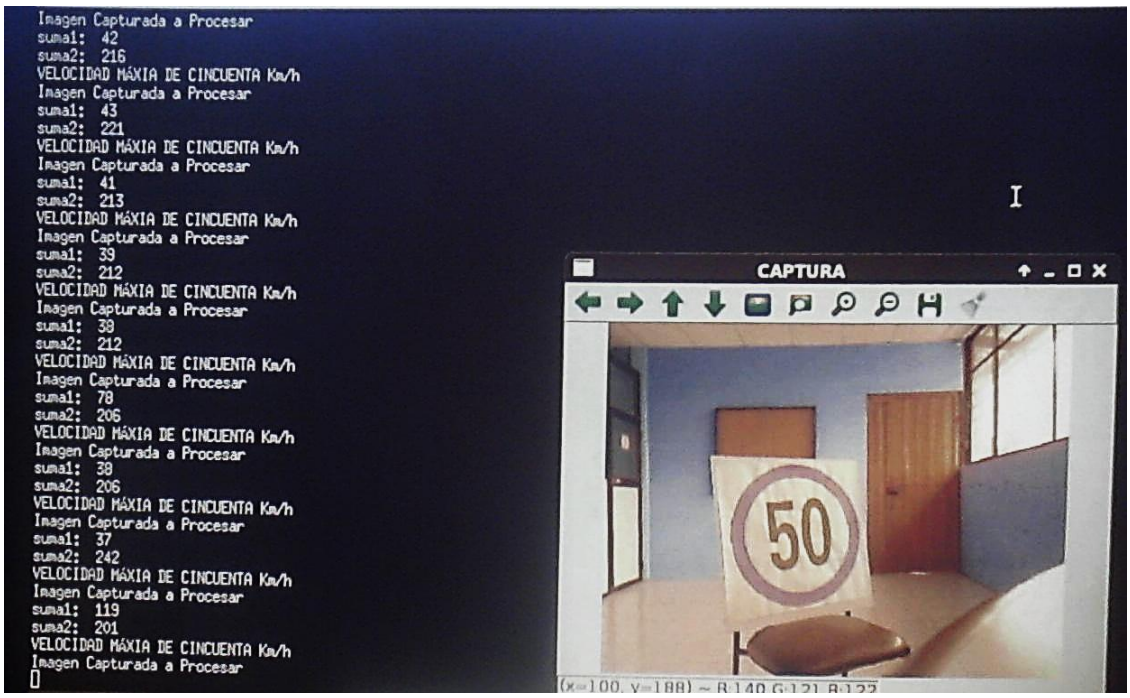

Figura 40: Señal de tránsito con inclinación horizontal a la derecha

Figura 41: Señal de tránsito con inclinación horizontal a la izquierda

Para las capturas en estas primeras pruebas se realizaron considerando una resolución de 640x480 pixeles y se consideran los casos que se indican a continuación.

Primer caso, captura de imagen con inclinación a la derecha y ambiente con iluminación baja. En las tablas 62, 63 y 64 se muestran los datos obtenidos de esta prueba.

Tabla 62

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	10° sexagesimales	102	8
4	10	10° sexagesimales	98	8
6	10	10° sexagesimales	78	7

Tabla 63

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	20° sexagesimales	98	8
4	10	20° sexagesimales	89	7

CONTINÚA

6	10	20° sexagesimales	76	7

Tabla 64

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	98	7
4	10	30° sexagesimales	67	6
6	10	30° sexagesimales	58	6

Segundo caso, captura de imagen con inclinación horizontal a la izquierda e iluminación baja. En las tablas 65, 66 y 67 se muestran los datos obtenidos de esta prueba.

Tabla 65

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	10° sexagesimales	112	8
4	10	10° sexagesimales	97	8
6	10	10° sexagesimales	83	7

Tabla 66

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	20° sexagesimales	102	8
4	10	20° sexagesimales	96	7
6	10	20° sexagesimales	74	7

Tabla 67

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	98	7
4	10	30° sexagesimales	73	6
6	10	30° sexagesimales	69	6

Al igual que se realizó la prueba bajo un ambiente poco iluminado, en esta sección se lleva a cabo la misma prueba en un ambiente con mayor iluminación, pero manteniendo la resolución de 640x480 pixeles.

Tercer caso, captura de imagen con inclinación horizontal a la derecha con iluminación alta. En las tablas 68, 69 y 70 se muestran los datos obtenidos de esta prueba.

Tabla 68

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	10° sexagesimales	111	8
4	10	10° sexagesimales	78	7
6	10	10° sexagesimales	76	7

Tabla 69

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	20° sexagesimales	104	8
4	10	20° sexagesimales	83	7
6	10	20° sexagesimales	71	6

Tabla 70

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	102	7
4	10	30° sexagesimales	97	6
6	10	30° sexagesimales	63	5

Cuarto caso, captura de imagen con inclinación horizontal a la izquierda con iluminación alta. En las tablas de la 71, 72 y 73 se muestran los datos obtenidos de esta prueba.

Tabla 71

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	10° sexagesimales	122	8
4	10	10° sexagesimales	98	7
6	10	10° sexagesimales	78	7

Tabla 72

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	20° sexagesimales	103	8
4	10	20° sexagesimales	95	6
6	10	20° sexagesimales	71	6

Tabla 73

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	98	7
4	10	30° sexagesimales	85	6
6	10	30° sexagesimales	70	6

Para los siguientes casos se ha realizado el reconocimiento de objetos con imágenes capturadas a una resolución de 1024x648 pixeles, para determinar los resultados al momento del reconocimiento de objetos.

Quinto caso, captura de imagen con inclinación Horizontal a la derecha con iluminación baja. En las tablas 74, 75 y 76 se muestran los datos obtenidos de esta prueba.

Tabla 74

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	10° sexagesimales	347	9
8	10	10° sexagesimales	261	8
12	10	10° sexagesimales	126	6

Tabla 75

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	20° sexagesimales	323	9
8	10	20° sexagesimales	216	8
12	10	20° sexagesimales	1163	6

Tabla 76

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	342	8
8	10	30° sexagesimales	256	7
12	10	30° sexagesimales	111	5

Sexto caso, captura de imagen con inclinación horizontal a la izquierda con iluminación baja. En las tablas 77, 78 y 79 se muestran los datos obtenidos de esta prueba.

Tabla 77

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	10° sexagesimales	242	9
8	10	10° sexagesimales	212	9
12	10	10° sexagesimales	119	6

Tabla 78

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	20° sexagesimales	353	9
8	10	20° sexagesimales	271	8
12	10	20° sexagesimales	161	6

Tabla 79

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	243	8
8	10	30° sexagesimales	159	7
12	10	30° sexagesimales	117	5

Al igual que para la prueba bajo un ambiente poco iluminado, en esta sección se presentan los resultados para el caso de un ambiente con más iluminación.

Séptimo caso, captura de imagen con inclinación horizontal a la derecha con iluminación alta. En las tablas 80,81 y 82 se muestran los datos obtenidos de esta prueba.

Tabla 80

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	10° sexagesimales	354	9
8	10	10° sexagesimales	271	8
12	10	10° sexagesimales	156	6

Tabla 81

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	20° sexagesimales	312	8
8	10	20° sexagesimales	209	8
12	10	20° sexagesimales	113	6

Tabla 82

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	296	8
8	10	30° sexagesimales	242	7
12	10	30° sexagesimales	113	5

Octavo caso, captura de imagen con inclinación horizontal a la izquierda con iluminación alta. En las tablas 83, 84 y 85 se muestran los datos obtenidos de esta prueba.

Tabla 83

Inclinación de 10°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	10° sexagesimales	376	9
8	10	10° sexagesimales	213	8
12	10	10° sexagesimales	124	6

Tabla 84

Inclinación de 20°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	20° sexagesimales	317	8
8	10	20° sexagesimales	174	7
12	10	20° sexagesimales	126	6

Tabla 85

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	278	8
8	10	30° sexagesimales	239	8
12	10	30° sexagesimales	131	7

- **Inclinaciones frontales**

Esta prueba se llevó a cabo bajo las condiciones de los dos ambientes ambiente y resoluciones con las que se han trabajado en las pruebas anteriores. En la figura 42 se observa cómo se llevó a cabo la prueba en referencia.

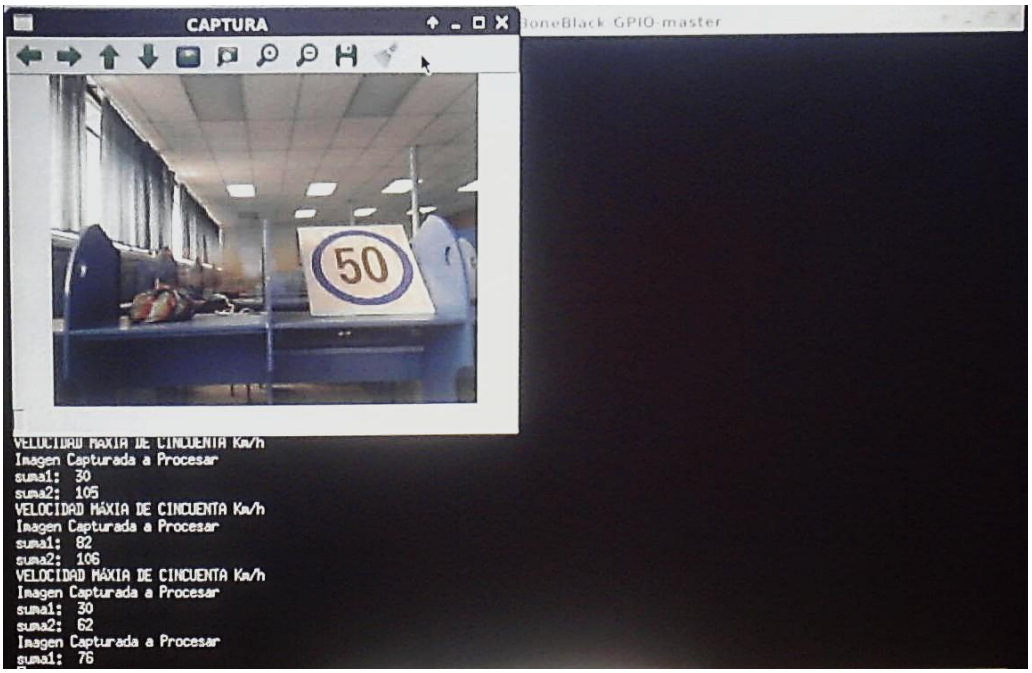

Figura 42: Objeto capturado con inclinación frontal

Primer caso, en este caso se ha realizado una inclinación hacia el frente en un ambiente con iluminación baja. En las tablas 86 y 87 se presentan los datos obtenidos.

Tabla 86

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	15° sexagesimales	112	8
4	10	15° sexagesimales	103	7
6	10	15° sexagesimales	98	6

Tabla 87

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	105	7
4	10	30° sexagesimales	82	6
6	10	30° sexagesimales	76	5

Segundo caso, captura de imagen con inclinación hacia atrás e iluminación baja. En las tablas 88 y 89 se muestran los datos obtenidos de esta prueba.

Tabla 88

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	15° sexagesimales	103	7
4	10	15° sexagesimales	98	6
6	10	15° sexagesimales	77	6

Tabla 89

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	98	6
4	10	30° sexagesimales	78	6
6	10	30° sexagesimales	56	5

Al igual que se realizó con la prueba bajo un ambiente poco iluminado, en esta sección se lleva a cabo la misma prueba en un ambiente con mayor iluminación.

Tercer caso, captura de imagen con inclinación hacia al frente. En las tablas 90 y 91 se muestran los datos obtenidos de esta prueba.

Tabla 90

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	15° sexagesimales	116	8
4	10	15° sexagesimales	108	8
6	10	15° sexagesimales	86	6

Tabla 91

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	112	7
4	10	30° sexagesimales	95	6
6	10	30° sexagesimales	79	5

Cuarto caso, captura de imagen con inclinación hacia atrás. En las tablas 92 y 93 se muestran los datos obtenidos de esta prueba.

Tabla 92

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	15° sexagesimales	107	8
4	10	15° sexagesimales	93	7
6	10	15° sexagesimales	78	6

Tabla 93

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	30° sexagesimales	105	8
4	10	30° sexagesimales	82	6
6	10	30° sexagesimales	77	5

Para los siguientes casos se ha realizado el reconocimiento de objetos con imágenes capturadas a una resolución de 1024x648 pixeles, se realiza el mismo procedimiento que en el caso interior.

Quinto caso, captura de imagen con inclinación hacia el frente e iluminación baja. En las tablas 94 y 95 se muestran los datos obtenidos de esta prueba.

Tabla 94

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	15° sexagesimales	215	8
8	10	15° sexagesimales	167	6
12	10	15° sexagesimales	116	5

Tabla 95

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	203	7
8	10	30° sexagesimales	152	6
12	10	30° sexagesimales	102	5

Sexto caso, captura de imagen con inclinación hacia atrás con iluminación baja. En las tablas 96 y 97 se muestran los datos obtenidos de esta prueba.

Tabla 96

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	15° sexagesimales	236	8
8	10	15° sexagesimales	168	7
12	10	15° sexagesimales	121	5

Tabla 97

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	215	7
8	10	30° sexagesimales	127	6
12	10	30° sexagesimales	106	5

Al igual que se realizó la prueba bajo un ambiente poco iluminado, ahora se lleva a cabo la misma prueba en un ambiente con mayor iluminación.

Séptimo caso, captura de imagen con inclinación hacia el frente con iluminación alta. En las tablas 98 y 99 se muestran los datos obtenidos de esta prueba.

Tabla 98

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	15° sexagesimales	255	7
8	10	15° sexagesimales	167	6
12	10	15° sexagesimales	116	5

Tabla 99

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	215	7
8	10	30° sexagesimales	156	6
12	10	30° sexagesimales	103	4

Octavo caso, captura de imagen con inclinación hacia atrás e iluminación alta. En las tablas 100 y 101 se muestran los datos obtenidos de esta prueba.

Tabla 100

Inclinación de 15°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	15° sexagesimales	285	7
8	10	15° sexagesimales	177	6
12	10	15° sexagesimales	106	5

Tabla 101

Inclinación de 30°

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
4	10	30° sexagesimales	243	7
8	10	30° sexagesimales	171	6
12	10	30° sexagesimales	98	4

- **Rotaciones**

Conociendo cómo influye la inclinación al momento de realizar el reconocimiento, esta prueba pretende determinar cómo influye la rotación del

objeto al momento del reconocimiento y para esto se hicieron pruebas bajo dos condiciones ambientales y dos tipos de resoluciones. En las figuras 43 y 44 se presentan las capturas para el reconocimiento del objeto.

Figura 43: Señal rotada 270° sexagesimales

Figura 44: Señal rotada 270° sexagesimales

Primer caso, en éste caso se realiza considerando una resolución de 640x480 bajo un ambiente con iluminación baja. Los resultados se muestran en la tabla 102.

Tabla 102

Captura 640x480 con objeto rotado

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
2	10	0° sexagesimales	134	9
4	10	90° sexagesimales	179	9
6	10	180° sexagesimales	87	8
7	10	270° sexagesimales	63	3

Segundo caso en este caso la captura tiene una resolución de 1024x748 bajo condiciones de iluminación alta. Los resultados se muestran en la tabla 103.

Tabla 103

Captura 1024x748 con objeto rotado

Distancia(m)	Muestras	° Inclinación	Kps Promedio	# Aciertos
3	10	0° sexagesimales	234	9
6	10	90° sexagesimales	268	9
9	10	180° sexagesimales	159	8
12	10	270° sexagesimales	131	6

Los datos obtenidos en esta prueba indican que al igual que la iluminación, los ángulos de inclinación y rotación influyen de una manera directa al momento de realizar el reconocimiento de objetos.

4.5. Análisis de los resultados obtenidos en las diferentes pruebas

Los primeros resultados obtenidos a lo largo de las diferentes pruebas, presentan los puntos característicos al momento de su detección, en esta prueba los datos obtenidos no varían si se realiza en el computador o en la tarjeta de manera independiente. Los factores que afectan en los valores de los puntos característicos son el umbral del Hessiano y el tamaño de la imagen con la que se trabaja.

Los tiempos de ejecución del algoritmo SURF dependen mucho de los recursos de hardware y software que tenga el equipo dónde se implementa la aplicación. Los resultados obtenidos en el computador y en la tarjeta Beaglebone Black son completamente distintos ya que los equipos y sistemas son de características electrónicas diferentes.

En la tabla 104 se presenta el resumen de los tiempos de ejecución del algoritmo SURF. Los resultados presentados corresponden a los obtenidos en el computador.

Tabla 104

Tiempos de ejecución computador

Valor Hessiano	Tiempo de ejecución (s)	
	pixeles (640x480)	pixeles (1024x748)
1000	0,132652	0,134703
2000	0,131127	0,133626
3000	0,123734	0,126474
4000	0,117586	0,120969
5000	0,114219	0,118686

En la tabla 104 los datos obtenidos del tiempo de ejecución indican que no afecta la información de la imagen al momento de procesarla, lo que afecta es la cantidad de puntos característicos que se buscarán y la resolución con la que se trabaje. En la figura 45 se ve la tendencia del tiempo para el reconocimiento de objetos (eje x valor del Hessiano*1000 eje y tiempo (s)).

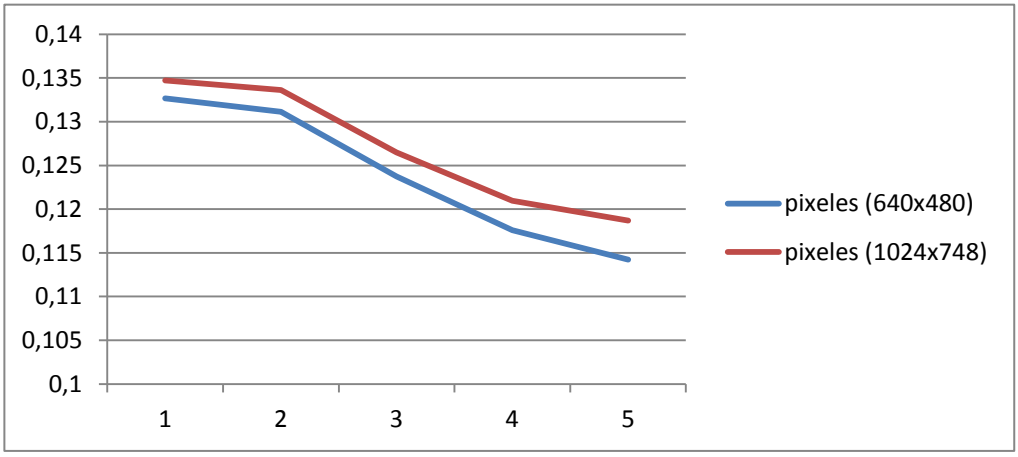

Figura 45: Tendencia del tiempo de ejecución durante el reconocimiento

De igual manera, para obtener los tiempos de ejecución en la tarjeta Beaglebone Black se determina que no importa el tipo de imagen que se utilice para procesar, sino la resolución y el número de puntos característicos con los que se desea trabajar. En la tabla 105 se presentan los datos promedio obtenidos en esta prueba.

Tabla 105

Tiempos de ejecución Beaglebone Black

Valor Hessiano	Tiempo de ejecución (s)	
	pixeles (640x480)	pixeles (1024x748)
1000	4,450736	4,650736
2000	4,086269	4,642569
3000	3,534743	3,693743
4000	2,639696	2,823696
5000	2,306864	2,463864

Los tiempos de ejecución obtenidos en la tarjeta Beaglebone Black son sumamente elevados en comparación como los tiempos de ejecución en el computador, esto se debe a los recursos de hardware y software que posee la tarjeta, ya que el algoritmo SURF consume muchos de ellos al momento de su ejecución. En la figura 46 se muestra la tendencia de los tiempos de ejecución para el reconocimiento de objetos en la tarjeta Beaglebone Black **(**eje x valor del Hessiano*1000 eje y tiempo (s)).

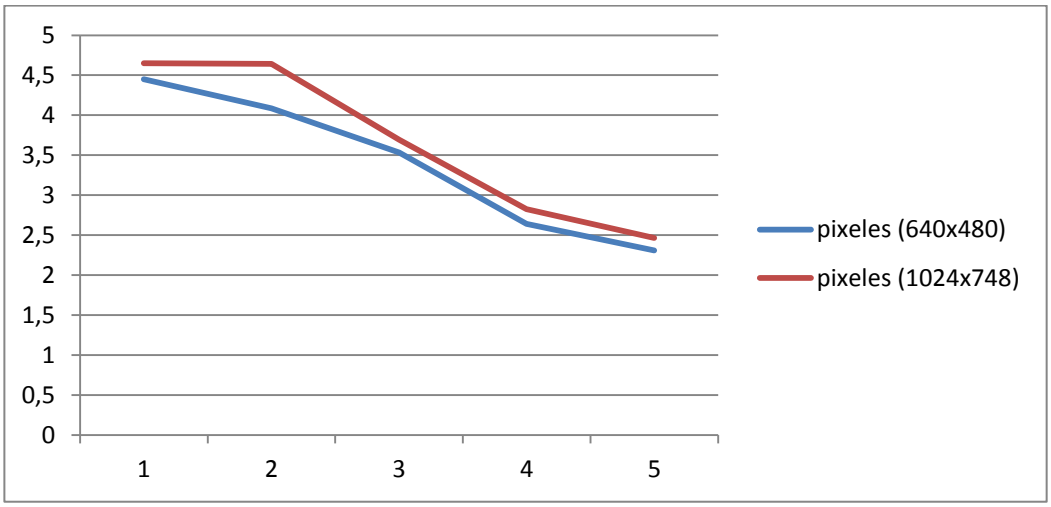

Figura 46: Tendencia del tiempo de ejecución durante el reconocimiento

Una vez determinado el valor de los puntos característicos y el tiempo promedio de ejecución del algoritmo SURF para el reconocimiento de objetos tanto en el computador con en la tarjeta Beaglebone Black, hay que tener presente que el valor de los puntos característicos de la imagen patrón no varían, los que varían son los de la captura. Los resultados que se presentan a continuación son de las pruebas para determinar la distancia óptima de detección de objetos en base a diferentes factores como, iluminación, distancia, inclinación y rotación de los objetos.

- **Resultados de distancia en base a la iluminación**

En las pruebas realizadas para la detección de una, dos y tres señales de tránsito de velocidad a una resolución de 640x480 se han hecho un resumen de los valores promedio que se presentan en las tablas 106, 107, 108, 109, 110 y 111 para cada señal de manera indistinta y así tener valores promedio de la detección.

Tabla 106

Detección de la señal de 50 km/h con iluminación baja

Distancia(m)	Aciertos(%)
1	100%
2	100%
3	95%
4	87,5%
5	87,5%
6	80%
7	75%

Tabla 107

Detección de la señal de 80 km/h con iluminación baja

Distancia(m)	Aciertos(%)
1	97,5%
2	97,5%
3	95%
4	92,5%
5	92,5%
6	82,5%
7	75%

Tabla 108

Detección de la señal de 100 km/h con iluminación baja

Distancia(m)	Aciertos(%)
1	95%
2	95%
3	95%
4	95%
5	90%
6	82,5%
7	72,5%

En la figura 47 se ve la tendencia del reconocimiento de objetos en una iluminación baja y se aprecia como la distancia influye en el reconocimiento de objetos (eje x distancia en metros y eje y porcentaje de reconocimientos).

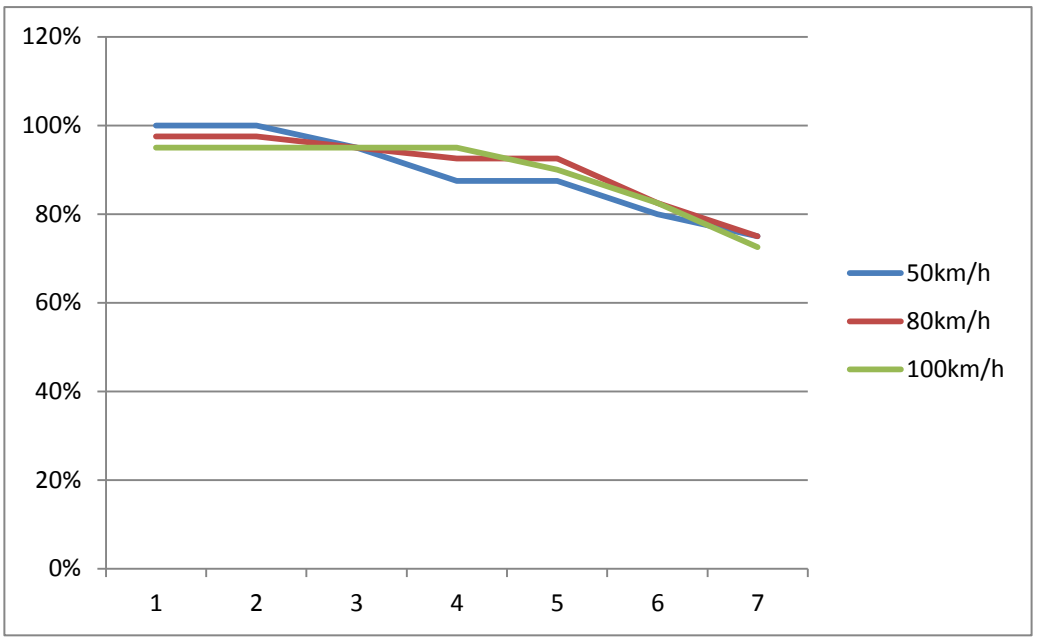

Figura 47: Tendencia de reconocimiento de objetos con iluminación baja

Tabla 109

Detección de la señal de 50 km/h con iluminación alta

Distancia(m)	Aciertos(%)
1	87,5%
2	87,5%
3	82,5%
4	80%
5	77,5%
6	67,5%
7	57,5%

Tabla 110

Detección de la señal de 80 km/h con iluminación alta

Distancia(m)	Aciertos(%)
1	82,5%
2	85%
3	80%
4	77,5%
5	75%
6	65%
7	47,5%

Tabla 111

Detección de la señal de 100 k m/h con iluminación alta

Distancia(m)	Aciertos(%)
1	85%
2	77,5%
3	80%
4	77,5%
5	77,5%
6	65%
7	52,5%

En la figura 48 se ve la tendencia del reconocimiento de objetos en una iluminación alta y se aprecia como la distancia influye en el reconocimiento de objetos (eje x distancia en metros y eje y porcentaje de reconocimientos).

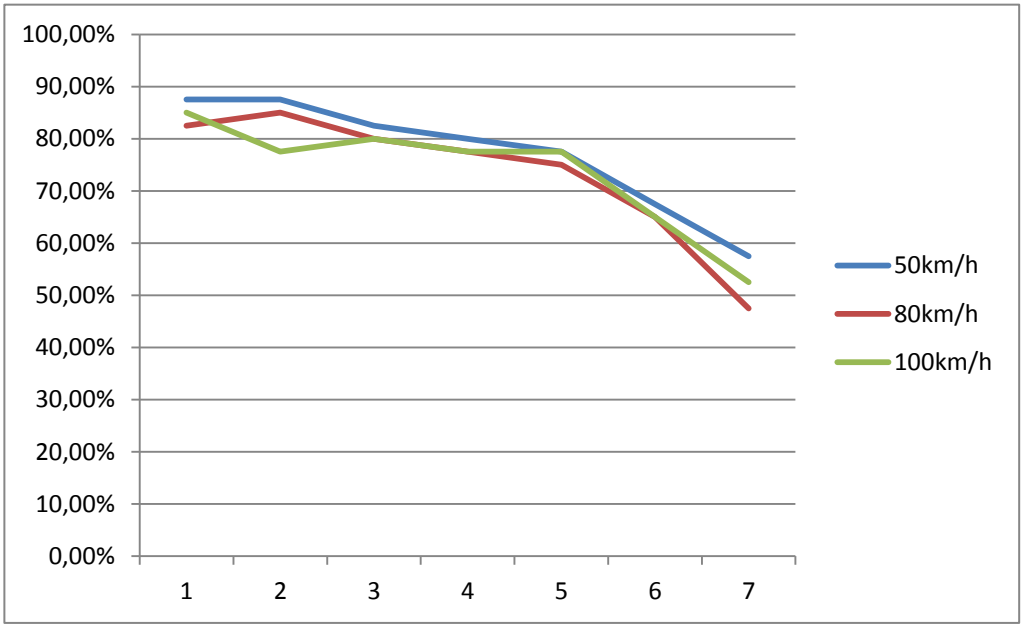

Figura 48: Tendencia de reconocimiento de objetos con iluminación alta

Los datos obtenidos indican que los promedios porcentuales del número de aciertos están en un valor aceptable, tomando en cuenta todas las condiciones durante el reconocimiento.

Al trabajar con una resolución de 1024x748 pixeles se obtuvo de igual manera los datos de todas las tablas en un resumen que expresan el porcentaje de aciertos en función de la distancia, para revisar ese valor de aciertos se tienen las tablas 112, 113, 114, 115, 116 y 117

Tabla 112

Detección de la señal de 50 km/h con iluminación baja

Distancia(m)	Aciertos(%)
1	92,5%
2	95%
3	97,5%
4	95%
5	97,5%
6	95%
7	92,5%
8	90%
9	80%
10	80%
11	70%
12	42,5%

Tabla 113

Detección de la señal de 80 km/h con iluminación baja

Distancia(m)	Aciertos(%)
1	90%
2	95%
3	97,5%
4	92,5%
5	85%
6	95%
7	97,5%
8	95%
9	87,5%
10	85%
11	72,5%
12	60%

Tabla 114

Detección de la señal de 100 km/h con iluminación baja

Distancia(m)	Aciertos(%)
1	92,5%
2	95%
3	95%
4	92,5%
5	92,5%
6	95%
7	97,5%
8	97,5%
9	90%
10	87,5%
11	70%
12	57,5%

En la figura 49 se ve la tendencia del reconocimiento de objetos en una iluminación baja y se aprecia como la distancia influye en el reconocimiento de objetos (eje x distancia en metros y eje y porcentaje de reconocimientos).

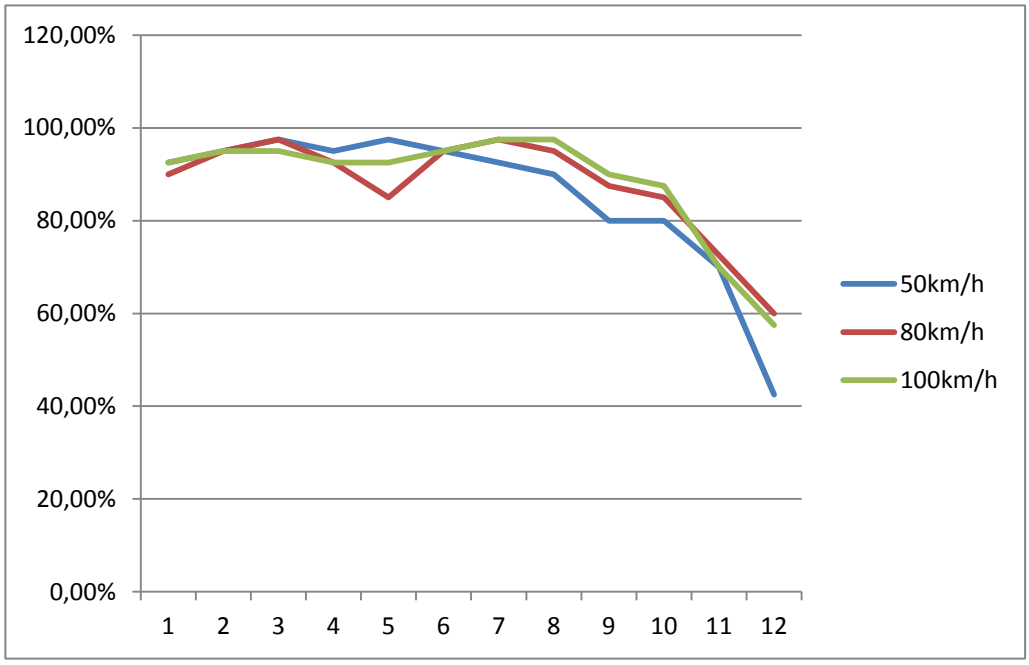

Figura 49: tendencia de reconocimiento de objetos con iluminación baja

Tabla 115

Detección de la señal de 50 km/h con iluminación alta

Distancia(m)	Aciertos(%)
1	97,5%
2	97,5%

CONTINÚA

3	97,5%
4	95%
5	95%
6	90%
7	92,5%
8	90%
9	85%
10	87,5%
11	82,5%
12	70%

Tabla 116

Detección de la señal de 80 km/h con iluminación baja

Distancia(m)	Aciertos(%)
1	92,5%
2	92,5%
3	97,5%
4	90%
5	92,5%
6	95%
7	90%
8	85%
9	85%
10	70%
11	65%
12	55%

Tabla 117

Detección de la señal de 100 km/h con iluminación alta

Distancia(m)	Aciertos(%)
1	95%
2	95%
3	95%
4	92,5%
5	85%
6	92,5%
7	92,5%
8	90%
9	85%
10	87,5%
11	77,5%
12	67,5%

En la figura 50 se ve la tendencia del reconocimiento de objetos en una iluminación alta y se aprecia como la distancia influye en el reconocimiento de objetos (eje x distancia en metros y eje y porcentaje de reconocimientos).

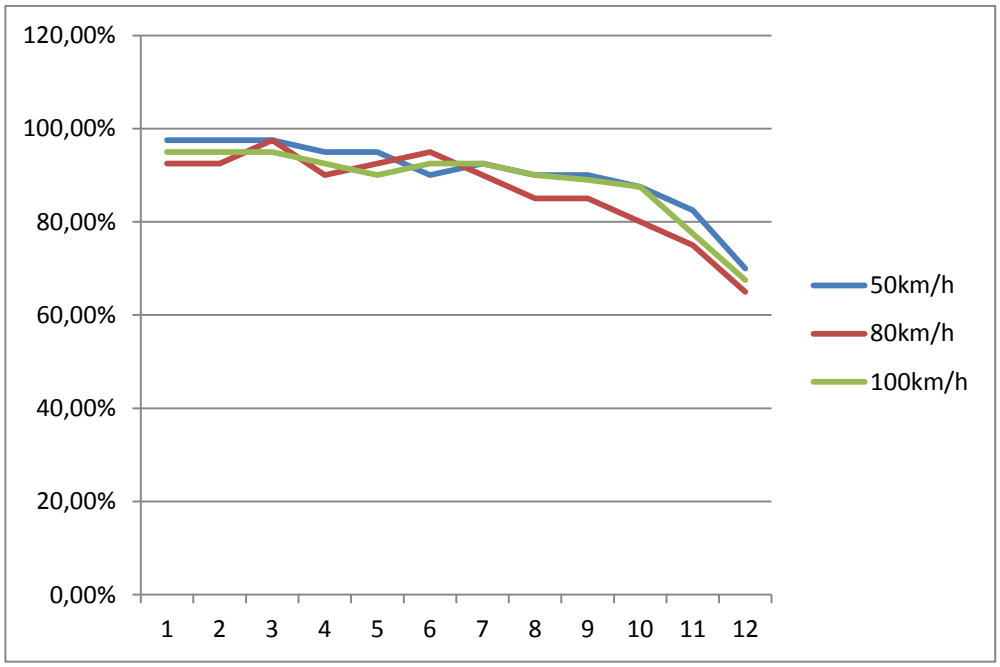

Figura 50: tendencia de reconocimiento de objetos con iluminación alta

Los datos y tendencias obtenidos presentan un resumen con el promedio de aciertos de cada una de las señales de tránsito de velocidad.

- **Resultados de detección en base a la inclinación del objeto**

Al momento de realizar el reconocimiento de objetos, cuando el objeto que se encuentra en la imagen capturada tiene un cierto grado de inclinación sea a la derecha o izquierda, influye en el reconocimiento de objetos. Para una inclinación de 10° sexagesimales en cualquier orientación lateral, los resultados obtenidos no varían demasiado.

Cuando se varía a 20° sexagesimales el ángulo de inclinación horizontal, los resultados obtenidos varían, primero que nada ya no se logran obtener el número de aciertos como si no existiese variación de ángulo, pero de igual manera el número de aciertos aun es bastantemente aceptable para lograr reconocer el objeto.

En un tercer caso cuando el ángulo se lo ha variado a 30° sexagesimales, el número de aciertos ya no es aceptable, porque primero que nada las

características del objeto cambian y pueden generarse falsos positivos con otra señal o simplemente no reconocerlos y segundo porque al inclinar la imagen la luz incide de diferente modo.

Para valores mayores a una inclinación de 30° sexagesimales da como resultados valores que son poco aceptables porque no hay muchos aciertos favorables. En la figura 51 se muestran los grados de inclinación lateral con los que se hicieron las pruebas.

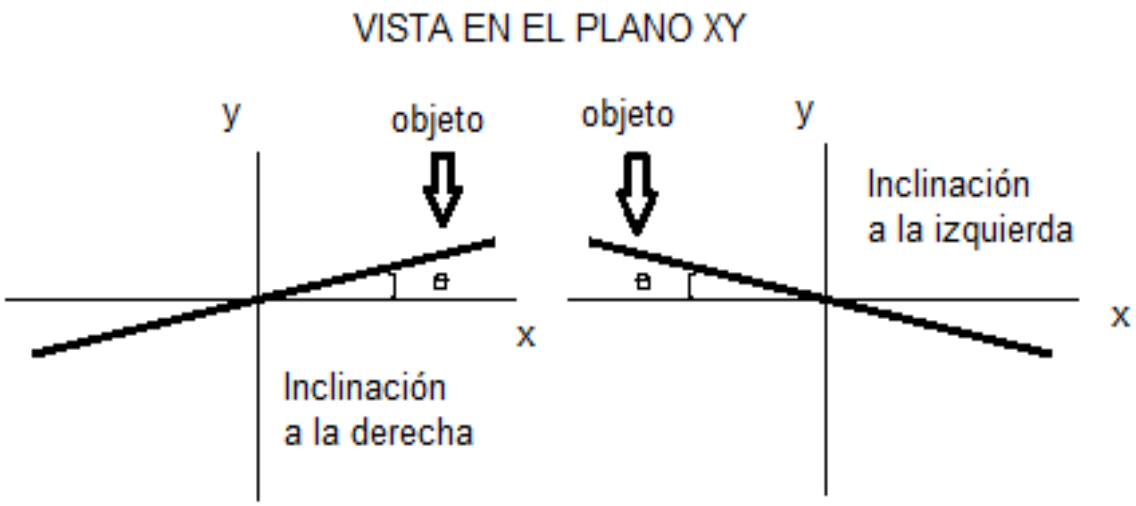

Figura 51: Inclinación lateral del objeto

En el caso de tener la inclinación hacia el frente, es posible realizar un buen reconocimiento de objetos con inclinación de hasta 15° sexagesimales y un reconocimiento deficiente con un ángulo de 30° sexagesimales, esto es debido a que cuando se inclina hacia el frente en 15° sexagesimales, la iluminación disminuye en cierto modo que afecta de manera mínima el reconocimiento de los objetos, pero al hacerlo a 30” sexagesimales esa iluminación disminuye significativamente y el reconocimiento de los objetos es mínimo.

En el caso de tener una inclinación hacia atrás, considerando una inclinación de 15° sexagesimales, los resultados no son adecuados en comparación con la inclinación hacia el frente al inclinarlo hacia el frente, esto

es debido a que hay más iluminación sobre el objeto y no permite un buen reconocimiento de los mismos. En la figura 52 se muestra la inclinación del objeto en forma frontal.

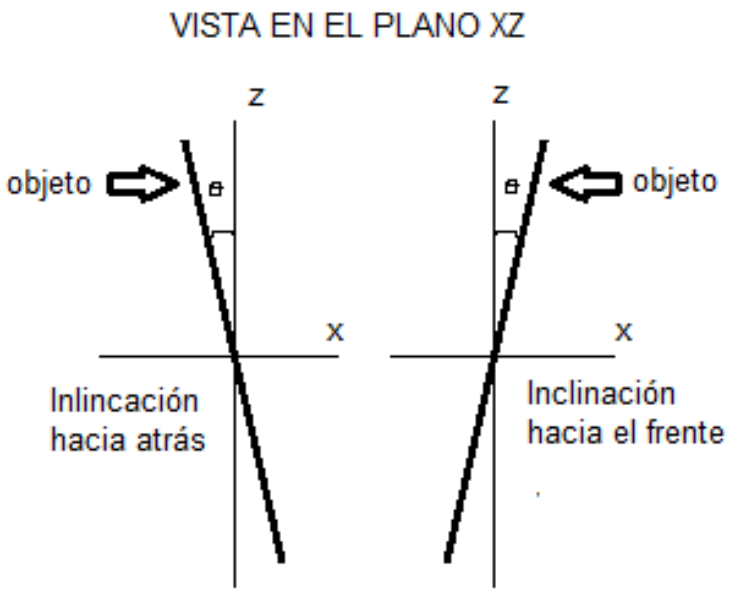

Figura 52: Inclinación frontal del objeto

- **Resultados de distancia en base a la rotación**

Cuando al objeto se lo ha rotado en sentido horario u anti-horario en cualquier valor en el rango de los 360° sexagesimales, los resultados obtenidos en comparación con los resultados obtenidos cuando el objeto mantiene la posición del objeto patrón, varían de una manera mínima, dando como resultado un número alto de aciertos favorables y sin falsos positivos.

Los resultados obtenidos en las pruebas indican que se puede reconocer a un objeto de manera óptima tendiendo una inclinación lateral de hasta 30° sexagesimales, una inclinación frontal de 15°sexageciamlaes hacia el frente y bajo una iluminación moderada (iluminación no tan en exceso por la presencia de la luz solar, ni tan baja como la escases de luz como en una habitación mal iluminada).

Ya se conoce que los promedios de tiempo de ejecución son sumamente elevados al implementar el algoritmo SURF en la tarjeta Beaglebone Black, y esto es debido a los pocos recursos que presenta la misma, una parte muy importante que hay que recalcar es que si se implementa en un vehículo la aplicación, el tiempo que demora en avisar al usuario de la existencia de una señal a depender de manera directa de la velocidad a la que se desplace el móvil.

Todas las pruebas realizadas indican que el algoritmo en si es una buena manera de reconocer objetos, pero como el sistema requiere demasiado tiempo para responder, no es recomendable utilizar para dicha aplicación con la Tarjeta Beaglebone Black.

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

El sistema operativo que se instale en la tarjeta Beaglebone Black influye de manera significativa en el rendimiento de software y hardware y en los tiempos de ejecución, además de la robustez que puede ofrecer.

En el caso de utilizar Ubuntu no se logra un rendimiento bueno ya que el algoritmo SURF hace que no opere la tarjeta de manera continua. Con Debian los rendimientos mejoran de manera notable en comparación al otro sistema operativo y la tarjeta opera sin ningún inconveniente.

La implementación del algoritmo SURF para detección de objetos en tiempo real en la tarjeta Beaglebone Black opera de manera óptima siempre y cuando el proceso para la detección de objetos sea lento, debido a que el algoritmo necesita de muchos recursos de memoria al momento de procesar imágenes lo que produce retardos en el procesamiento de las imágenes.

La detección de puntos característicos y su debido emparejamiento es la base del reconocimiento de objetos en el Algoritmo SURF ya que de este paso depende si se hace un reconocimiento verdadero o un reconocimiento falso.

Se puede obviar la presentación de imágenes con el emparejamiento debido: primero porque para dibujar las líneas de emparejamiento se necesita de más tiempo en la ejecución y segundo porque el GUI se demora en presentar la información y debe ajustar la ventana al patrón indicado cada vez que se ejecute la secuencia para la detección haya o no un reconocimiento de objetos.

Para tener una presentación inmediata de las capturas realizadas por la webcam se debe instalar las librerías para el soporte gráfico, porque de no hacerlo, la presentación de imágenes se retrasa en unas cinco o seis

imágenes, y en el caso de usar Ubuntu esto provoca que se detenga la aplicación y se deba reiniciar a la tarjeta. Por esa razón se trabaja utilizando el sistema operativo Debian.

La base de datos que se incluye en la tarjeta Beaglebone Black no debe presentar más de dos o tres patrones en el peor de los casos para realizar la búsqueda, ya que al momento de ir buscando y comparando la captura con cada uno de los objetos patrón, el tiempo de detección se incrementa y ya no es posible tener reconocimiento de la señales de tránsito en tiempo real.

El consumo de los recursos de la memoria RAM no dependen ni de la resolución de las imágenes ni el formato de las imágenes, sino dependen de los recursos de hardware que presente el dispositivo o máquina en que se ha implementado el mencionado algoritmo, así como también de la frecuencia con la que se está ejecutando la aplicación.

Una manera de optimizar el procesamiento en tiempo real es realizar la lectura de la última imagen capturada rechazando de este modo las demás capturas y procesarla de manera instantánea. Al realizar el procesamiento de todas las imágenes capturadas, se retrasa el proceso de reconocimiento de objetos y con esto saturar la memoria de la tarjeta Beaglebone Black.

En el procesamiento de imágenes mediante el algoritmo SURF, se debe tener presente los factores de iluminación, distancia, inclinación y resolución de la imagen, ya que estos afectan de manera directa al reconocer objetos.

La tarjeta Beaglebone Black presenta un cierto número de librerías y dependencias para el control de los puertos dependiendo de las aplicaciones que se realicen, en este caso se utiliza únicamente una librería que permite utilizar a los puertos de propósito general para por medio de estos controlar el módulo de audio que se incluyó en el sistema.

Al trabajar con los puertos de entradas y salidas de propósito general, se debe exportar los pines a utilizar siempre antes de iniciar la aplicación, porque al no

hacerlo simplemente no se realizará la tarea especificada. Y para hacer esto se requiere de privilegios de administrador y no simplemente de usuario, por lo que se debe trabajar todo bajo una cuenta de *root* desde la terminal.

5.2. Recomendaciones

Si se desea instalar el sistema operativo para trabajar con aplicaciones como en este proyecto, es necesario instalar el sistema operativo en una unidad de almacenamiento extra (micro-SD), de una capacidad mayor o igual a 4GB, ya que las librerías requeridas ocupan alrededor de 4GB sin contar con las dependencias y librerías del sistema, es recomendado trabajar con tarjetas de 8GB y 16GB de capacidad, ya que la tarjeta incluye una memoria de almacenamiento de tan solo 2GB.

Utilizar una micro-SD de alta velocidad de transferencia de datos, para de este modo mejorar el rendimiento de las aplicaciones que se desean implementar en la tarjeta Beaglebone Black.

Utilizar las protecciones debidas en los puertos de propósito general ya que al momento de ejecutar la aplicación, las entradas solo soportan tensiones de 3.3 V. y al ingresar un valor estándar de 0 a 5 voltios o de 1 a 5 voltios se puede quemar los puertos y la tarjeta.

En caso de no poder ejecutarse la aplicación se debe importar un PATH para que las librerías de OpenCV sean reconocidas y no se generen más inconvenientes.

Trabajar con un valor de puntos característicos de un valor inferior a cincuenta puntos como se hizo en el proyecto, de este modo no se tiene valores repetidos y se asegura que el reconocimiento sea mejor.

Es recomendable antes de instalar la librería que permita controlar los puertos de propósito general desde C++, probar desde la terminal si se activan o no los mencionados puertos para de este modo asegurarse de que se ha instalado de manera correcta el sistema operativo.

Trabajar dentro de la carpeta donde se contienen las librerías para el control de los puertos para de este modo facilitar las cosas al momento de la ejecución de la aplicación.

Trabajar con una tarjeta de mayores prestaciones en hardware y software para realizar el procesamiento de imágenes y el reconocimiento de señales de tránsito de tiempo real.

Es recomendable trabajar con lenguaje C para realizar la aplicación ya que es un lenguaje que está a nivel del lenguaje de máquina y esto hace que los tiempos de ejecución sean menores a utilizar un lenguaje interpretado como lo es python.

Al momento de ejecutar la aplicación hay que tener presente como generar el ejecutable en la compilación, porque al no hacerlo no se podrá poner en marcha la aplicación.

Para trabajar como tarjeta de adquisición de datos con la Beaglebone Black se debe tener presente los voltajes y corrientes máximos que soportan a las entradas y salidas, ya que si se trabaja sobre esos umbrales se puede quemar la tarjeta.

Para mejorar el rendimiento de la tarjeta se recomienda instalar únicamente las librerías y bibliotecas con las que se va a trabajar, porque al instalar otras que no sé utilicen se estará consumiendo recursos de memoria innecesariamente.

Se recomienda utilizar una cámara que permita capturar un número bajo de cuadros por segundo, para de este modo no se sobrecargue las entradas.

REFERENCIAS BIBLIOGRÁFICAS

Morales R. Sossa J., (2012). “Procesamiento y análisis digital de imágenes”, Editorial Alfaomega

Pajares G. De La Cruz J., (2002), "Visión por computadora / imágenes digitales y aplicaciones", editorial Rama.

Chacón, M., (2007), "Procesamiento digital de imágenes", editorial Trillas, México, D. F.

Mendoza Dario, R. W. (2012). *Diseño y construcción de un prototipo de sistema de control para monitorear e incrementar la seguridad en el acceso vehicular al parqueadero de la ESPE-L, utilizando procesamiento digital de imágenes.* Latacunga.

Instituto Ecuatoriano de Normalización. (2011). *INEN*. Recuperado el 4 de Noviembre de 2015, de INEN: https://www.usfq.edu.ec/sobre_la_ usfq/servicios/autoclub/Documents/reglamento_tecnico.pdf

Anqi Xu, G. (2008). *Visión artificial.* Recuperado el 30 de Septiembre de 2015, de http://www.vision.ee.ethz.ch/~surf/eccv06.pdf

ARMhf. (s.f.). *ARMhf Linux for ARMhf devices*. Recuperado el 12 de Octubre de 2015, de http://www.armhf.com/boards/beaglebone-black/bbb-sd-install/

Beagleboard.org. (s.f.). Recuperado el 12 de Octubre de 2015, de http://beagleboard.org/latest-images

Beagleboard.org. (s.f.). *Beagleboard*. Recuperado el 12 de Octubre de 2015, de Beagleboard: Available: http://www.farnell.com/datasheets/1685587.pdf

Cruz, L. (2013). Recuperado el 8 de Noviembre de 2015, de http://arantxa. ii.uam.es/~jms/pfcsteleco/lecturas/20130527CarlosLeguaCruz.pdf

debian.*forums.debian.net*. Recuperado el 13 de Octubre de 2015, de http://forums.debian.net/viewtopic.php?f=8&t=118038

Dr. Alfonso Alba Cadena. (2001). Recuperado el 13 de Noviembre de 2015, de http://galia.fc.uaslp.mx/~fac/ssd/OpenCV.pdf

Dra. Flores Leticia. (s.f.). *wordpress.* Recuperado el 6 de Noviembre de 2016, de wordpress: https://aicitel.files.wordpress.com/2011/08/clase3_pdi.pdf

elinux. Recuperado el 15 de Octubre de 2015, de http://elinux.org/BeagleBoardDebian

Gonzales. G. (s.f.). *caratinaudlap.mix.* Recuperado el 7 de Noviembre de 2015, de caratinaudlap.mix: http://catarina.udlap.mx/u_dl_a/tales/documentos /mel/gonzalez_g_ra/capitulo2.pdf

Herbeert B., Andreas E., Tunne T., & Luc V.,l. (2008). *twiki.* Recuperado el 3 de Noviembre de 2015, de http://campar.in.tum.de/twiki/pub/Chair/ TeachingWs09MATDCV/SURF_paper.pdf

INFAIMON.S.L. (s.f.). *pserv.udg.edu*. Recuperado el 12 de Noviembre de 2015, de Available: http://pserv.udg.edu/Portal/Uploads/4103862/CAMARAS _Infaimon.pdf

Jose Esqueda, L. P. (2005). *Fundamento de procesamiento de imagenes.* México: Mexicalia Baja califormia.

Logitech. (s.f.). *pccomponentes*. Recuperado el 12 de Octubre de 2015, de pccomponentes: http://www.pccomponentes.com/logitech_ hd_pro_ webcam_c920.html

Ministerio de Educación Presidencia de la Nación. (s.f.). *Colección educa.ar*. Recuperado el 4 de Noviembre de 2015, de Colección educa.ar: http://coleccion.educ.ar/coleccion/CD16/contenidos/ley/index2.html

Normalización, I. E. (2011). Recuperado el 4 de Noviembre de 2015, de https://www.usfq.edu.ec/sobre_la_usfq/servicios/autoclub/Documents/reglamento_tecnico.pdf

OpenCV. (s.f.). *OpenCV*. Recuperado el 14 de Octubre de 2015, de OpenCV: http://ubaa.net/shared/ processing/OpenCV/

OpenCV.org. (s.f.). *OpenCV*. Recuperado el 25 de Octubre de 2015, de OpenCV: http://OpenCV.org/ documentation.html

OpenCV.org. (s.f.). *OpenCV Documentation*. Recuperado el 10 de Noveimbre de 2015, de http://docs.OpenCV.org/2.4.9/modules/core/doc/old_basic _structures.html#IplImage

OpenCV.org. (s.f.). *OpenCV Documentation*. Recuperado el 24 de Noviembre de 2015, de http://OpenCV.jp/OpenCV-2svn_org/c/highgui_reading_and_writing_images_and_ video.html

OpenCV.org. (s.f.). *OpenCV Documentation*. Recuperado el 13 de Noviembre de 2015, de http://docs.OpenCV.org/2.4.9/modules/core/doc/basic_ structures.html#Mat

OpenCV.org. (s.f.). *OpenCV Documentation*. Recuperado el 12 de Noviembre de 2015, de http://OpenCV.jp/OpenCV-2svn_org/c/highgui_reading_and_writing_images_and_ video.html

OpenCV.org. (s.f.). *OpenCV Documnetatión*. Recuperado el 15 de Noviembre de 2015, de http://docs.OpenCV.org/2.4.9/modules/refman.html

Oscar B. Gracia, L. V. (2011). *Estudio comparativo de descriptores para detección de escenas cuasi-duplicadas.* Madrid.

Pederson J. (2011). *cs,au.dk.* Recuperado el 10 de Noviembre de 2015, de http://cs.au.dk/~jtp/SURF/report.pdf

Prof. Dr. Nicolás L. (s.f.). *Visión Artificial*. Recuperado el 5 de Noviembre de 2015, de Visión Artificial: http://www.uco.es/users/ma1fegan/2011-2012/vision/Temas/Vision-artificial.pdf

Seijas M. (2011). Recuperado el 12 de Noviembre de 2015, de http://digital.bl.fcen.uba.ar/Download/Tesis/Tesis_4997_Seijas.pdf

Stackoverflow. (s.f.). *Stack overflow*. Recuperado el 26 de Novimbre de 2015, de http://stackoverflow.com/questions/17613723/whats-the-meaning-of-minhessian-surffeaturedetector

Vasquez. Arthur. (11 de Septiembre de 2014). *Prezi*. Recuperado el 4 de Noviembre de 2015, de Prezi: https://prezi.com/gxwqnwczos-r/senales-de-transito/

Visión Artificial. *Visión Artificial*. Recuperado el 5 de Noviembre de 2015, de Conceptos Generales: http://www.etitudela.com/celula/downloads/visionartificial.pdf

ANEXOS

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el señor: LUIS ERNESTO CAIZA ANDRANGO

En la ciudad de Latacunga, a los 10 días del mes de marzo del 2016

Ing. Eddie Galarza
DIRECTOR DEL PROYECTO

Aprobado por:

Ing. Franklin Silva
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO