UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería


# Tecnologías de Computación para Monitoreo y Análisis Óptico a bajo costo computacional

Trabajo de graduación presentado por Juan Pablo Ortiz Portillo para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2017

# Tecnologías de Computación para Monitoreo y Análisis Óptico a bajo costo computacional

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería


# Tecnologías de Computación para Monitoreo y Análisis Óptico a bajo costo computacional

Trabajo de graduación presentado por Juan Pablo Ortiz Portillo para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2017

**Vo. Bo**

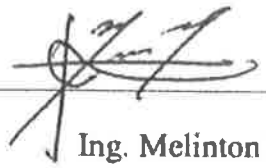
(f) 

Ing. Melinton Navas

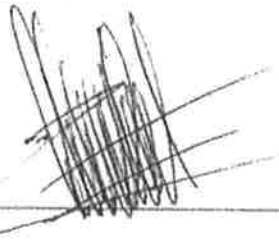
**Tribunal Examinador**

f) 

MSc. Douglas Leonel Barrios González

f) 

Ing. Melinton Navas

f) 

Ing. Christian Medina Armas

Fecha de aprobación del examen de graduación:

(Guatemala, 15 de junio de 2017)

# ÍNDICE

Página

# LISTADO DE CUADROS, GRÁFICAS & DIAGRAMAS

## Listado de cuadros



## Listado de diagramas



## Listado de gráficas

# LISTADO DE FIGURAS

# RESUMEN

La capacidad de la máquina de intentar comprender su entorno es una técnica de aprendizaje proveniente de “Inteligencia artificial” y la evolución de “Machine Learning”. Esta técnica de aprendizaje le otorga la capacidad a la máquina de analizar, comprender, categorizar, reaccionar, interactuar y auto-aprender aplicando diferentes técnicas basadas en algoritmos de “Inteligencia artificial”.

Análisis complejos, como análisis de imágenes o videos para comprender el entorno, la capacidad de vehículos de auto pilotearse o reaccionar de forma autónoma a diferentes eventos por medio de diferentes sensores, estas son técnicas de análisis por visión por computadora. Varias de estas tecnologías están conformados en varios programas o “frameworks” que analizan por medio de diferentes sensores entradas la información del exterior y aplican técnicas diferentes algoritmos de inteligencia artificial para analizar distintos tipos de patrones para generar una respuesta o acción con la mayor exactitud posible. (News18 AFP Relaxnews, 2017)

Este trabajo se basa en el desarrollo de una plataforma de reconocimiento visual de placas vehiculares utilizando hardware accesible, y por medio de algoritmos de detección de contornos y otros métodos de “Machine Learning”. Para efectos de prueba y desarrollo de este trabajo se implementará un sistema de captura de video e imagen, detección y aislamiento del objeto de interés (placa vehiculares) y por último el análisis de patrones e identificación de caracteres utilizando un algoritmo de clasificación conocido como “vecino cercano”,  en abreviatura conocido como k-NN (k-nearest neighbors algorithm).

El algoritmo k-NN es parte de las variedad de fórmulas y métodos aplicables en “Machine Learning” y es importante para los resultados de este proyecto. Los resultados obtenidos durante el desarrollo del proyecto fueron variados, alta y baja precisión, factor a la gran cantidad de tiempo y nuestros necesarios a realizar para entrenar este algoritmo de “Machine Learning”, pero permitió obtener suficientes resultados para conocer la capacidad de análisis de este algoritmo. En el caso de la detección de patrones o contornos, los resultados fueron bastante satisfactorios, donde la efectividad radica en conocer el patrón de interés y las características que tienen al momento de programar el sistema.

# I. INTRODUCCIÓN

Con el paso del tiempo la relación entre el potencial computacional sobre el costo de adquisición de la tecnología ha mejorado en beneficio para los consumidores pequeños. Es viable la adquisición de hardware de alto rendimiento para realizar estudios, pruebas de nuevo software o ejecutar algoritmos de alta complejidad, sea por investigación, negocio o incluso recreación sin necesidad de incurrir en un procesador de gama alta. Algoritmos en el ámbito de la "Inteligencia Artificial" son un ejemplo de cómo la evolución del hardware permite el desarrollo de nuevas tecnologías. El poder computacional anteriormente necesario para procesar grandes cantidades de datos, como los que requiere estos algoritmos de "Inteligencia Artificial", ahora son posibles de realizar en un sistema pequeño y portable a bajo costo. Un ejemplo es la compra de hardware de uso comercial, ahora pueden equiparar la potencia computacional de un procesador para servidores de no hace muchos años. Un GPU de videojuegos puede servir para programación paralela y ser utilizado para análisis de datos masivos. Un celular moderno puede realizar tareas de captura de video y procesamiento de imagen y transmisión masiva de datos.

Aun así, si el poder computacional de uso comercial existente no es suficiente para realizar este tipo de pruebas existe otro tipo de soluciones fáciles de adquirir, como lo son los servicios de computación en la nube. Los servicios de la nube, conocidos como SaaS y PaaS (*Software as a Service y Platform as a Service*, respectivamente por sus siglas en inglés). Estos servicios cuentan con grandes ventajas en cuestión de escalabilidad con base en demanda. Esto significa que, si el hardware actual no es lo suficiente para soportar una plataforma de análisis de datos, puede utilizarse estos servicios en la nube para encontrar el punto ideal de poder computacional a un bajo costo. Algunos ejemplos son los servicios de "Google Cloud Services" y "GPU Cloud Computing" de NVidia. (Galen Gruman, 2007) Este incremento en el potencial computacional y nuevas plataformas de servicios tecnológicos que han surgido son el producto de la evolución de años de constante mejora en el hardware de los ordenadores. Siendo así fáciles de adquirir para realización de investigación de nuevas tecnologías, soluciones o ideas innovadoras.

Estudios en el ámbito de "Inteligencia Artificial" han tomado mayor auge con la evolución del hardware. Algoritmos que le permiten a un sistema abstracto como un software el comprender un entorno, a partir de diferentes tipos de sensores entrada. Tomando como ejemplo al humano, sus sentidos son la forma de percepción del entorno, similar sucede con un sistema computacional, una cámara se convierte en una forma de capturar el exterior, al igual que un sensor de movimiento, un receptor de sonido, sensor térmico o infrarrojo. Cosas que incluso el humano no puede percibir, pueden ser analizadas por una computadora y programarla a modo que reaccione o interactúe con el ambiente.

# II. OBJETIVOS

## 1. OBJETIVO GENERAL

- Presentar una plataforma de monitoreo y análisis óptico por medio de computadora para demostrar el potencial de las herramientas en la recolección de datos y generación de resultados, para así proponer todo un conjunto de herramientas comercializables o ideas innovadoras.

## 2. OBJETIVOS ESPECÍFICOS

- Profundizar en los conceptos involucrados en el desarrollo de una herramienta de monitoreo y análisis óptico por computadora, a modo de indagar y profundizar en los conceptos de otras ramas de la computación como ¨Inteligencia Artificial¨.
- Analizar la infraestructura necesaria para darle soporte a la plataforma que se desarrollará con el fin de proponer formas de implementación del sistema a un costo moderado y óptimo sea cual sea el cliente objetivo.
- Analizar el costo con base en la escalabilidad según la demanda de procesamiento computacional que pueda requerir la plataforma.

# III. JUSTIFICACIÓN

El término emprendedurismo ha tomado mayor apogeo, nuevos emprendedores surgen con capital limitado, otros como pequeñas empresas buscan plataformas tecnológicas económicamente más accesibles, mientras grandes empresas buscan ahorrar costos en sus procesos o sistemas tecnológicos. Todas las siguientes afirmaciones tienen algo en común, buscar precios bajos que generen ahorro, pero de calidad y que resuelvan un problema. Todo esto teniendo en cuenta un balance entre costo y utilidad en sistemas tecnológicos.

La ley de Moore es una buena predicción acerca de la evolución de la tecnología y el cómo el potencial computacional ha incrementado a modo de ser más accesible durante el paso de los años. Ya no es necesario implementar un "Data Center" con un sistema de refrigeración y con redundancia energética compleja cuando una laptop puede realizar las tareas de procesamiento para "hosting" de un sistema web simple. También se puede optar por la contratación de la infraestructura de un sistema tecnológico en una modalidad "PaaS", en donde se tiene la ventaja de fácil escalabilidad, un requisito indispensable cuando se trata de implementar herramientas de análisis complejas como algoritmos de "Inteligencia Artificial", sea un ejemplo aquellos de computación óptica.

Es indispensable este avance en el potencial computacional debido a la evolución e innovación tecnología que requiere de algoritmos de gran complejidad, sea un ejemplo aquellas herramientas de análisis óptico por computadora, las cuales requieren de gran cantidad de procesamiento dada la gran cantidad de información que pueden llegar a procesar en un instante. Algunas tecnologías innovadoras pueden llegar a requerir de resultados en tiempo real, sea por ejemplo los auto autónomos.

Un motivante para el desarrollo de una herramienta de análisis óptico es por el auge que ha tenido en los últimos años este tipo de tecnologías y el potencial que pueden mostrar. Con el avance tecnológico, es más accesible y fácil el desarrollo este tipo de herramientas a bajo costo y con un alto potencial utilizando una computadora común o comercial. Empresas pioneras como NVIDIA o Automotrices innovan en sus plataformas y productos introduciendo el análisis óptico por medio de cámaras y sensores . En varios de los casos, basados en los conceptos de "Inteligencia Artificial", "Deep Learning" o "Machine Learning" para analizar en tiempo real lo que sucede, es capturado y genera un resultado rápido.(Robohub, Brad Templeton, 2017)

# IV. ALCANCES Y LÍMITES

El alcance para este trabajo es la implementación de toda una cadena de procesos que conforma un sistema de detección óptica por computadora utilizando un algoritmo de “Machine Learning”. Cada proceso es una sub-herramienta desarrollada para cumplir una tarea independiente y específica, pero necesaria dentro de toda la cadena de análisis. El propósito de modularizar cada proceso en sub-herramientas independientes es estimar las diferentes complicaciones y costos que podría incurrir el desarrollo de cada módulo para realizar un estimado de sobre el sistema ya unificado.

La herramienta a implementar será un reconocimiento de placas por medio de cámara aplicando la técnicas de “Machine Learning” utilizando el algoritmo k-NN para el análisis de datos. El plan de desarrollo será realizar las diferentes fases del sistema por separado teniendo una entrada y una salida, siendo la entrada inicial la captura de la imagen en tiempo real y la salida final, o resultado, el dato ya procesado por todas las sub-herramientas.

Siendo una captura en tiempo real por medio de una cámara, el alcance que se propone es la capacidad de generar un análisis aproximado a partir de una muestra tomada por la cámara, teniendo como resultado los dígitos que componen las placas de los automóviles muestra. Se utilizarán como base patrones para el aprendizaje introducidos en el módulo que aplica el algoritmo de “Machine Learning” k-NN. Al momento a ser reconocidos, el sistema finalizará guardando la captura en una base de datos y generando el resultado final. Una vez completo las diferentes sub-herramienta, generar proyecciones de costos que puede incurrir la implementación de todo el sistema completo y por sub-herramientas midiendo los tiempos de respuestas en base a la máquina de referencia utilizada.

Limitantes presentes en el proyecto serán el mismo hardware a utilizar, dado que no se tiene un punto de referencia comparativo del nivel de procesamiento de la maquina a utilizar, se deberá experimentar modificando el algoritmo hasta mejorar los tiempos lo mejor posible. Además, el propósito del trabajo es validar los costos de implementación proyectando a crear un sistema con recursos limitados para generar el menor costo posible. (véase sección VIII. Software y Hardware)

El entregable final será un prototipo de varias sub-herramienta y su capacidad de cómputo estará basado en una computadora de uso personal. El prototipo contará con: Sistema de Captura y Procesamiento en Tiempo Real, Sistema de Análisis y Sistema de Almacenamiento.

# V. MARCO TEÓRICO

La programación utilizando algoritmos de "Inteligencia Artificial" permite a los sistemas computacionales analizar casos complejos. Entre la variedad de algoritmos en este ámbito se encuentra "Machine Learning", conocido también como aprendizaje de máquina. Como refiere su nombre, "Machine Learning" permite crear un sistema con la capacidad de aprender progresivamente a partir de alimentar el sistema con muestreos. El resultado de esto permite clasificar patrones, identificarlo con mayor precisión y devolver un resultado aproximado. Entre mejor sea el entrenamiento mayor será la capacidad de estos algoritmos de obtener una solución óptima. Algunas de las áreas que estudia estas técnica se encuentra en la computación visual, reconocimiento automático de audio y reconocimiento de patrones o señales, comportamientos de usuario para realizar sugerencias.

Algunos algoritmos de procesamiento visual de imágenes utilizados con la metodología de "Machine Learning" se asemejan a los OCR. El realizar programas de este tipo requieren de algoritmos eficaces para generar una respuesta optima en un tiempo adecuado. Algunas formas para aumentar la capacidad de respuestas es la implementación de programación paralela, con el fin de mejorar el rendimiento en la velocidad de cálculo tomando ventaja de la reducción de tareas similares. La aplicación del paralelismo aprovechando el hardware permite la utilización de multiprocesamiento o procesamiento paralelo, resolviendo zonas críticas, cuellos de botella o fragmentos de código repetitivo. (Mohamed Cherier, 2007). El algoritmo que se realizara para el reconocimiento de placas aplica la misma metodología, capturando la placa por medio de algoritmos de detección de contornos que luego aislaran los caracteres dentro de la placa para luego procesarlos uno a uno aplicando un algoritmo de "Machine Learning". (Sumanta Subhadhira, 2014)

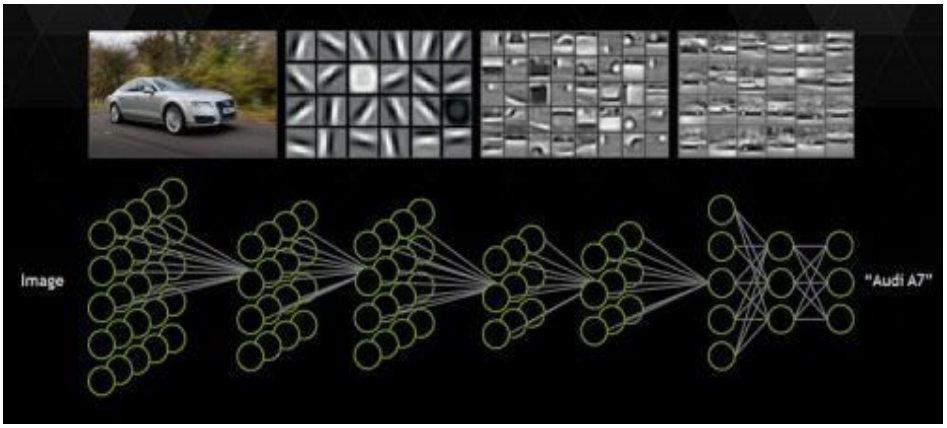



Figura 1. Representación de patrones detectados en una imagen

Para poner en marcha todo un sistema de reconocimiento óptico es necesario analizar los diferentes procesos que lo conforman y separarlos para desarrollo por individual. Iniciando por lo más básico, debe definirse cuál será el objetivo o patrón de interés a analizar, el contenido que tendrá como letras, números u otros caracteres que deberá reconocer, el área de donde se extraerán, factores externos que puede generar varianzas, etc. Al completar esta definición puede procederse a armar una base de conocimiento y analizar la técnica más apropiada para solventar el problema.

La base del conocimiento contiene todas las representaciones de las figuras o patrones que se proyectan identificar en el resultado final, sean este caso caracteres. Esta base de conocimiento representa la memoria de la "Inteligencia Artificial" que tendrá el sistema. Esta base de conocimiento ayudará a identificar por medio de aproximaciones las letras o dígitos que analizara el sistema de reconocimiento óptico. La técnica de reconocimiento a utilizar puede variar en algoritmos, tipo de análisis y estructura, pero basa sus decisiones a partir de la base de conocimiento con que se alimentara el sistema.

Previo a poner en acción esta sección de identificación del sistema completo, debe analizarse e identificarse los caracteres dentro de la placa. Para ello se requiere de un subsistema de captura y procese las imagen o video (entrada de datos). Este subsistema limpiará y normaliza el archivo de entrada con base en las reglas o especificaciones del patrón de búsqueda de interés que se espera que reconozca. Parte importante en este subsistema es definir el hardware de entrada y la infraestructura de computación que será necesario para sopórtalo. (Helen Rodríguez, 2012)

Componente clave para el desarrollo de un sistema de detección y análisis óptico por computadora es la librería OpenCV (*Open Source Computer Vision Library* por sus siglas en inglés). OpenCV es una librería de código abierto con licencia BSD desde el año 2000, capaz de soportar múltiples leguajes de programación por medio de interfaces. Soportar múltiples sistemas operativos utilizando lenguajes como: MATLAB, C, C++, Python y Java. Es una librería creada para sacar provecho del poder computacional de un ordenador para ser eficiente en el procesamiento real de imágenes o video. La librería incorpora una gran variedad de funciones para filtrar, ecualizar, normalizar y detectar patrones dentro de una imágen. También incorpora funciones para crear un sistema de "Machine Learning" para clasificación e identificación de patrones que será necesario para generar el resultado final de la detección.

Entre los lenguajes y "frameworks" soportados por la librería de OpenCV esta Netbeans. Netbeans es un paquete con múltiples herramientas para facilitar desarrollo, es capaz de soportar múltiples lenguajes de programación, pero muy utilizado cuando se trata de utilizar Java. El lenguaje de programación Java se orienta en la programación por objetos. Utiliza un compilador llamado JVM (Java virtual machine) para la compilación de su código creando un archivo clase como el ejecutable final del código. (Robert, 2009)

Otro lenguaje aplicable dentro de la librería de OpenCV es Python. Python es un lenguaje de programación interpretado, muy utilizado para realizar programas enfocados en estadísticas y grandes cálculos matemáticos por la gran variedad de librerías, complementos y simplicidad que puede integrase a este lenguaje. Su forma de ejecución varia a la de un compilador por funcionamiento de ejecutar el código línea por línea. Su integración con OpenCV requiere únicamente de la descarga de paquetes pre compilados, siendo así el más simple de los lenguajes de programación para integrarse con la librería de OpenCV. (Neil Anuskiewicz, 2016)

Otros "frameworks" o lenguajes revisado para este proyecto fue Visual Studio. Visual Studio es toda una plataforma para desarrollo creado por Windows y enfocado para el mismo. Integra diferentes tipos de lenguaje de programación como C, C++, C#, Visual Basic y soporta otros lenguajes también. Para su utilización con OpenCV se pueden codificar en los lenguajes de programación C o C++. Su integración requiere de mayor trabajo dado que requiere de la interacción con herramienta terceras. Esta herramientas terceras permite compilar el código fuente de la librería OpenCV a modo de poder personalizar el paquete final que luego se incorporará al proyecto de Visual Studio, es considerado uno de los más personalizables pero a su vez requiere de mayor cantidad de pasos, bastante detallados, para su configuración.

El análisis de datos para este tipo de sistemas es muy recurrente, tareas similares que deben aplicarse muchas veces. La aplicación de paralelismo es una opción muy recomendable para la reducción de tiempos y análisis de resultados. Agregando, el tiempo entrenamiento del se ve impactada favorablemente aplicando paralelismo, ya que requiere de mucha tarea repetitiva. Para estos casos el uso de tarjetas gráficas se vuelve indispensable para proyectos de mayor magnitud.

Durante la fase de pruebas a realizar, un factor importante que se identifico fue la redundancia de cálculos similares. En software, múltiples tareas similares puede mejorarse implementando paralelismo. Además, OpenCV cuenta con una sección dedicada a realizar paralelismo nativamente con CPU, pero además cuenta con la capacidad de utilizar paralelismo utilizando ciertas tarjetas de video.

Las tarjetas de video o aceleradores gráficos (GPU) fueron introducidos cercanos a los años 90. Su primer uso fue para nivel industrial bajo el propósito de acelerar el cálculo computacional en renderización de objetos 3D. NVidia, marca y empresa dedicada al desarrollo de hardware para aceleración de gráficos cuenta con gran variedad de productos. NVidia es reconocida por ser pionera en la innovación de diferentes tecnologías, para proveer al usuario final de un hardware con capacidades de ejecutar operaciones y algoritmos sofisticados. Noticias o tendencias actuales como vehículos autónomos, detección de peatones, análisis de entornos, en varios casos llevan atrás la imagen de esta compañía por el uso de sus productos, siendo una pieza de hardware clave para entrenamiento de "Machine Learning" o "Red Neuronal" procesamiento y filtrado de imágenes. (Luebke, 2007)

Internamente los aceleradores gráficos de NVidia cuentan con gran cantidad de núcleos de procesamiento denominados “CUDA Cores”. (NVidia, 2017) Es común relacionar la programación paralela con estos múltiples núcleos donde la librería OpenCV cuenta con la capacidad de mejorar el rendimiento de diferentes funciones utilizándolos, véase Figura 2. (OpenCV, 2014) NVidia cuenta también con servicios en línea (nube) para la adquisición de infraestructura como servicio (Iass) poniendo a venta el uso de “Cluster” de GPUs para investigación o uso empresarial.

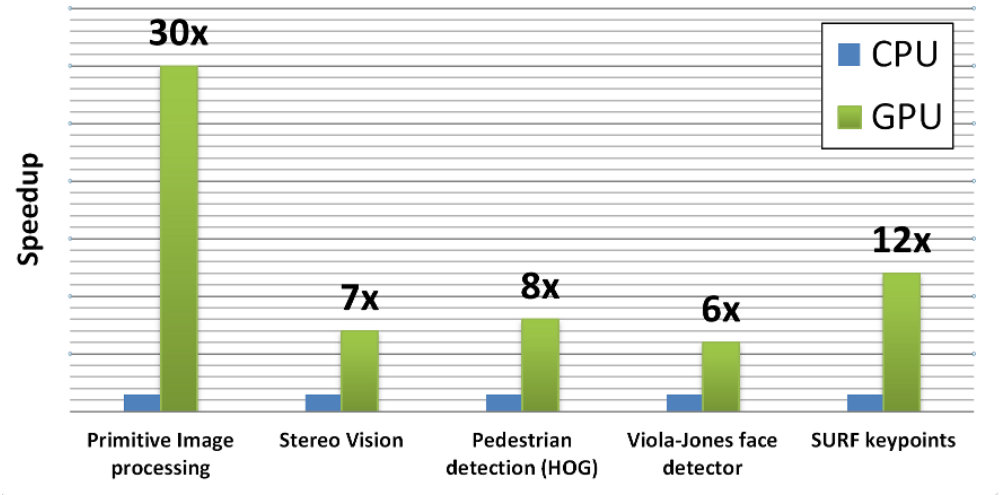



Figura 2. Comparativo de velocidad de procesamiento CPU vs GPU con OpenCV

Parte del enfoque principal del proyecto es la captura de imágenes por computadora previo a procesarlas, por lo que una pieza de hardware clave es la cámara. Esta pieza es la que permitirá conectar el entorno exterior con la máquina, similar a los ojos del ser humano. Conocer el tipo de cámara y las características que posee es clave para la parametrización del desarrollo previo a la adquisición. Varias de las características a considerar serán la calidad de pixeles que brinda, la velocidad de enfoque y capacidad de luz que soporta el obturador. Se Debe considerarse además opciones de HDR o niveladores de contraste para balancear los picos de luz presentes en la imagen y obtener una imagen de mayor fidelidad.

Parte del proceso es la búsqueda y detección de con ciertas características o formas. Esta función es propia de la librería de OpenCV, pero previo a utilizarla es necesario realizar un proceso de normalización utilizado filtrado y ecualización sobre la imagen capturada para sobresaltar dichos contornos. Durante este proceso de detección, la imagen capturada requerirá pasar por un proceso de filtrado para nivelar ruido, luz, contraste. En esta etapa la imagen es filtrada por diferentes funciones proporcionada por la librería  OpenCV.

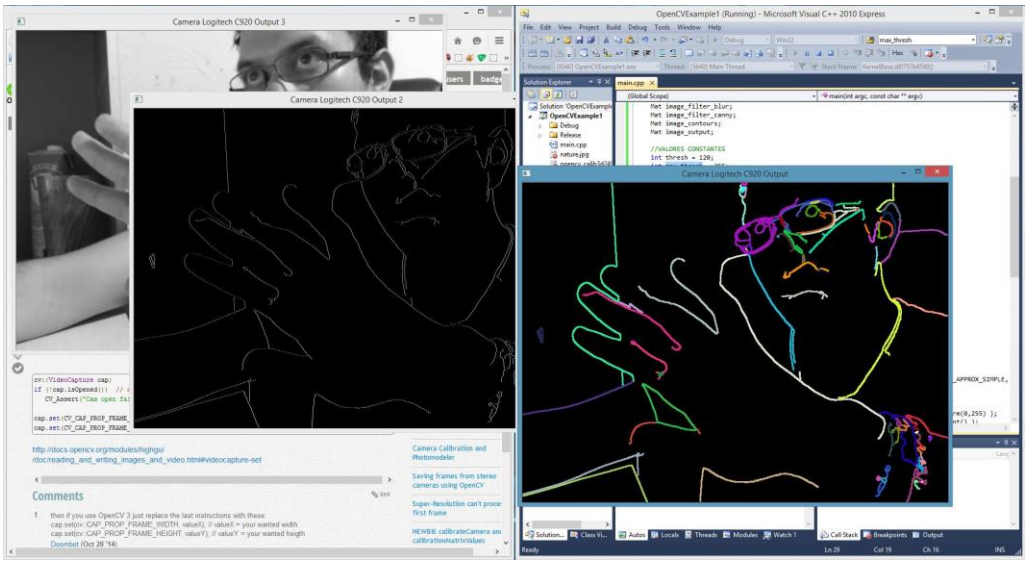



Figura 3. Proceso de resaltar contornos en una imagen utilizando OpenCV

Parte del proceso requiere la nivelación de iluminación presente en la imagen. La iluminación puede variar debido a factores como sombras, posición del sol, luz artificial, etc. Estos factores puede afectar la detección del contorno, causando punto atípicos de iluminación dentro del patrón a buscar, dificultado la identificación correctamente del contorno o incluso distorsionarlo. Para reducir estos factores en la fase de detección, deben aplicarse funciones de ecualización de contraste. La librería de OpenCV cuenta con su propio ecualizador de contraste llamada CLAHE. Las siglas de CLAHE significan "Contrast Limited Adaptive Histogram Equalization", este algoritmo de ecualización para el contraste es realizada por pequeñas regiones o bloques de toda la imagen, igualando la luminosidad por segmentos. La ecualización por bloques permite mantener restringido el generar picos altos de contraste ya que el algoritmo utilizado basa su ecualización en el anterior bloque generado. Otras funciones de contraste ecualizan la imagen abarcando todo el tamaño de la imagen mientras que la ecualización por CLAHE es distribuida por segmentos. (OpenCV, 2014)

El siguiente proceso de filtrado en la imagen es aplicar un algoritmo de "BLUR" o desenfoque. Este filtro reduce el ruido presente en la imagen y mejora la calidad del filtrado anteriormente. El principal objetivo de este algoritmo es alisar segmentos de la imagen eliminado residuos que puedan distorsionar la forma del contorno. OpenCV cuenta con gran variedad de filtros "BLUR", siendo para este caso el tipo BilateralFilter. Este tipo de algoritmo "BLUR" tiene como ventaja realizar el desenfoque de la imagen por segmentos respetando áreas límites como orillas con picos de alta intensidad. (OpenCV, 2014) Al respetar estos puntos de cambio permite suavizar el ruido e intensificar la orilla para la detección del contorno.

Parte de los procesos finales de filtrado están la aplicación de filtro morfológico. Este proceso requiere pasar previamente la imagen a modo binario (solo pixeles negros y blancos). El filtro morfológico comprende dos técnicas sobre la imagen: erosión y dilatación. (Alexander Mordvintsev, 2013) Partiendo de la imagen binaria, aplicar estos filtros aumentan la cantidad de pixeles blancos (dilatación) o pixeles negros (erosión) con base en las orillas donde se encuentre un cambio de color de píxel de blanco al negro o viceversa. Esta función es similar a la técnica "BLUR", en aplicar el filtro por bloques de pixel.

Posterior a detectar el contorno y aislarlo lo siguiente es la detección e identificación de los caracteres dentro del mismo. En este punto es donde debe implementarse un metodología similar a un OCR, "Optical Character Recognition". Para la implementación del sistema de identificación pueden utilizarse diferentes algoritmos de reconocimiento para ejecutar el proceso de estimación e identificación del carácter, para el caso de este proyecto se utilizó el algoritmo de k-Nearest Neighbors (k-NN), una técnica de "Machine Learning" en la cual se alimenta con gran variedad de viñetas de cada carácter a identificar. Este algoritmo otorga pesos a las viñetas ingresadas asignándoles a la vez un valor de equivalencia el cual sería la identificación de la letra o número. El fuerte de este algoritmo recaen en la cantidad de viñetas ingresada, por lo que entre más viñetas se introduzcan será más preciso, permitiendo mejores aproximaciones de los caracteres a reconocer. (OpenCV, 2014)

# VI. MARCO METODOLÓGICO

## A. Plataforma y sistemas

A continuación se describe las diferentes sub-herramientas que conforman la plataforma de análisis óptico por computadora:

i. Sistema de captura y procesamiento
ii. Sistema de análisis
iii. Sistema de almacenamiento

El primer sistema es la captura y procesamiento en tiempo real. Este contempla la comunicación hardware entre la cámara y la computadora, el cual luego es procesado por el software de procesamiento de imagen. El software de procesamiento de imagen realiza un análisis en búsqueda del objeto de interés (en este caso la placa vehicular) aplicando gran variedad de filtros a la entrada de video.

El objetivo de aplicar diferentes filtros en cierto orden es para la generación de contornos. Los contornos detectados en la imagen juegan un papel crucial dado que son los que permiten detectar patrones del objeto de interés que se busca dentro del "frame" del video o imagen. En esta fase se busca analizar patrones previamente programados de un objeto de interés sobre toda una imagen (análisis óptico).

La cantidad de contornos que se quiera generar debe pasar por una fase de calibrado. Esta fase de calibrado es crucial, dado que en ella se modifica diferentes características y parámetros de los filtros aplicados al "frame" del video o imagen. Muchas de las variantes presentes en la parametrización de esta fase puede ser el ambiente exterior, luces, clima, horario, densidad de movimiento, etc. El calibrado permite fijar una cantidad aceptable de contornos generados y la precisión del mismo para la fase siguiente de análisis, tomando en cuenta también la capacidad de cómputo del hardware que aloja el sistema.

Una vez adquirida la extracción de la placa se procede a generar una imagen aislada de la misma, para luego ser procesada por el sistema de análisis. En esta fase se procede a trabajar con conceptos de "Machine Learning" en el cual se implementa un algoritmo para asemejar un "Optical Character Recognition" o OCR. Dicho algoritmo utilizado es el k-NN, el cual se entrena por medio de muestras diferentes caracteres, entre más muestras se ingresan al motor de reconocimiento este se vuelve más preciso. Esta fase va de la mano con el sistema de análisis, el cual fragmenta la imagen de la placa detectando posibles contornos carácter. Cada contorno carácter es procesado por el motor de reconocimiento, el cual, al completar su análisis, regresa una cadena de texto indicando la identificación de la imagen de la placa aislada.

Para finalizar, se tiene un sistema de almacenamiento que registra todos los análisis de las placas identificadas y procesadas para persistencia de los resultados en un base de datos. Esta almacenará información como hora y lugar de registro de la placa entre otros datos relevantes. El propósito es contar con un sistema de persistencia de registros que luego puede ser utilizado para generación de reportes o para inteligencia de negocios.

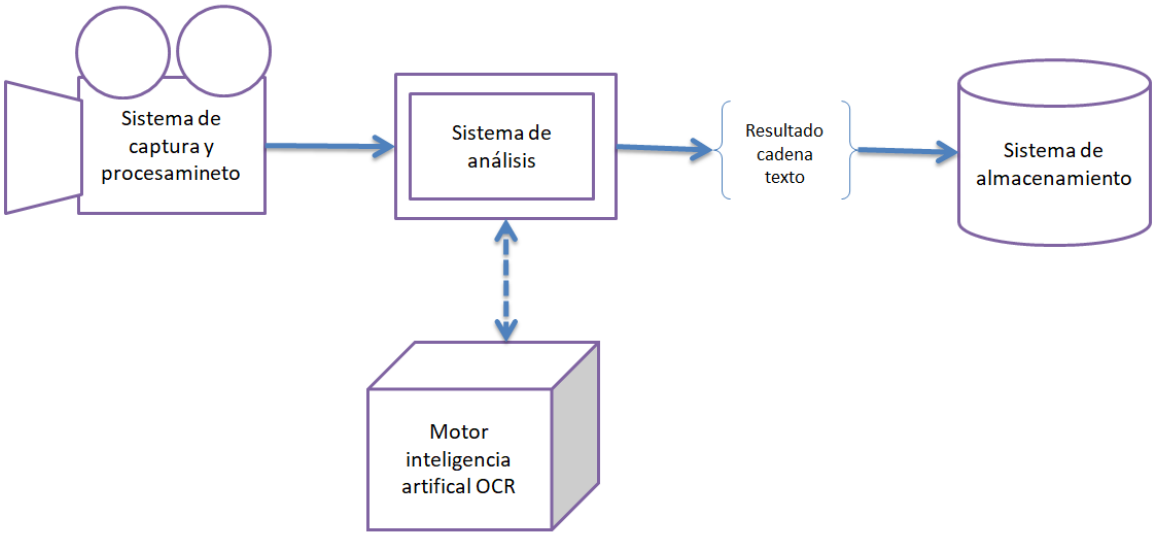



Diagrama 1. Diagramas de sistemas unificados

## B. Especificaciones software

Softwares utilizados en el proyecto

- Python 2.7.8
- C++ Visual Studio 2010
- Librería OpenCV 2.4.9
- MySQL 5.6.17
- WAMP
- Sistema Operativo Windows 8.1

Otros softwares propuestos o descartados

- Java junto Netbeans 8.0
- C++ Visual Studio 2012
- CUDA ToolKit
- Sistema Operativo Mac OSX Mavericks

Para el desarrollo del proyecto se utilizaron diferentes lenguajes de programación bajo propósitos de investigación y pruebas para determinar el lenguaje de programación a utilizar en el desarrollo final. El primer intento se realizó con Netbeans utilizando Java (descartado), seguido con Python (sistema de detección por imagen y sistema de identificación) y por ultimo Visual Studio utilizando C++ (sistema de detección por video tiempo real).

Durante esta fase inicial se estudiaron compatibilidad entre diferentes frameworks, sistemas operativo y lenguajes de programación para validar cuál sería el software final a utilizar. Debido a diferentes motivos de integración con el framework OpenCV, los lenguajes Java en NetBeans y C++ en Visual Studio 2012 quedaron descartados. En el caso de Visual Studio, se realizó un "downgrade" a la versión del 2012 a la 2010, el cual permitió la compatibilidad con el framework OpenCV.

En el intento de utilizar Java se descartó por la instalación de OpenCV no era simple, requiriendo ejecución de instrucciones por línea de comando, fallando en múltiples ocasiones por diferentes motivos, por lo que se decidió descartar por motivos de tiempo para avanzar con el proyecto.

Durante la fase de desarrollo, se optó por utilizar C++ con el sistema de captura/detección y Python en sistema de análisis y motor de reconocimiento e identificación, tomando en cuenta el tiempo, documentación y facilidad de integración con el framework de OpenCV. Ambos lenguajes implementados se utilizaron en sistema operativo Windows 8.

La librería OpenCV juega un papel importante en las sub-herramientas desarrolladas. Esta librería permite realizar funciones de captura, cargar, filtrado, ecualización, detección de contornos e identificación de caracteres. Originalmente, OpenCV está diseñado para trabajar C++, bajo propósitos de realizar análisis óptico por medio de computadora. Su integración con Python es por medio de "wrappers" que permite aplicar gran parte de las funcionalidades del framework de C++.

Se intentó instalar OpenCV en una Macbook Air con OS Mavericks para tener el desarrollo de forma portátil, pero no se logró conseguir la instalación del framework. Para Mac OS se requiere de otra complejidad para la instalación de OpenCV, al punto de realizar una compilación de los códigos fuente por medio de una herramienta llamada CMake. Durante los diferentes intentos por instalar, un requerimiento era la actualización de librerías del framework que por motivos desconocidos, posiblemente por versiones y compatibilidad, no se logró actualizar y por ende su utilización en este sistema operativo queda descartado.

Agregando, el framework de CUDA Toolkit se propuso para realizar pruebas de paralelismo y aumentar el desempeño del programa, pero surgió la dificultad que no existe el "wrapper" para el módulo de CUDA en Python y además la alta complejidad y conocimiento que requiere la programación orientada a procesos paralelos. A pesar de ello, su integración sigue siendo una propuesta atractiva para la aceleración y mejora de los procesos computacionales para la detección y análisis.

Por último, WAMP se utilizó para gestionar la base de datos para el registro y persistencia de los datos capturados y analizados. El tipo de base de datos implementada fue MySQL utilizando los conectores de las librerías propias de Python para realizar él envió y guardado de los datos. WAMP es el servicio de "webserver" que permite levantar la base de datos para la conexión del sistema de análisis.

## C. Especificaciones hardware

A continuación se especifica las características de hardware de la computadora utilizada para este proyecto:

- CPU: Intel i5 3570K a 3.4GHz
- RAM: 8GB 1333mHz
- Webcam Logitech C920 PRO de 15 Megapixeles
- Cámara de celular para fotos de placas vehiculares.

Se utilizó cámaras de celulares para tomar fotos de muestras en diferentes lugares, tiempos y condiciones climáticas. El único requerimiento necesario era cámaras con más de 2 Megapíxeles de resolución fotográfica para tener fotos de buena calidad y definición.

## D. Codificación del sistema

La codificación del sistema de análisis óptico por computadora se realizó en tres fases. Cada fase dedicada a una sub-herramientas: captura, análisis y almacenamiento. En el proceso de análisis se contempla el entrenamiento de motor de reconocimiento de caracteres.

La primera fase se realizó utilizando Visual Studio con C++ y Python. Esta fase su propósito era utilizar una cámara como entrada de imagen o video y analizar en tiempo real patrones para identificar y aislar la imagen de la placa. Para ello, se realizó un tratamiento en la imagen para filtrar, ecualizar y aislar los contornos de que cumplan con el patrón buscado. Durante una fase preliminar se realizó experimentación en el proceso de filtrado utilizando imágenes estáticas utilizando lenguaje Python. La metodología y procesos de filtrados aplicados en esta fase con Python luego se migraron a Visual Studio C++ obteniendo captura y aislamiento de la placa en tiempo real por entrada de video.

La segunda fase es el análisis de la imagen de la placa aislada en la fase anterior. Esta fase vuelve a realizar nuevamente un proceso de tratamiento en la imagen y búsqueda de contornos. Para este caso, el aislamiento de contornos se enfoca en posibles caracteres. Una vez detectados los contornos de los posibles caracteres el motor de reconocimiento se encarga de detectar cada posible letra a un carácter previamente introducido al motor de reconocimiento de caracteres. Para este caso de estudio, el analizar una placa vehicular debe detectar al menos siete contornos de cierta dimensión, de ser menos contornos identificados le indicará al programa que no es una placa o la placa está incompleta.

En la fase de análisis se contempla también el entrenamiento del motor de reconocimiento de caracteres. El tipo de algoritmo que aplica este motor de reconocimiento es de “Machine Learning” denominado k-NN de “k-nearest neighbors”. Este entrenamiento se basa en introducir ciertos patrones que representan un valor. Una vez entrenado el motor recibe una entrada y realizar una búsqueda de reconocimiento por patrones similares.

Una última sub-herramienta es desarrollada para dar fin al proceso de detección. Esta última fase no requiere de openCV, sino es la encargada de recibir el resultado de la detección y guárdalos en base de datos. Esta base de datos es administrada por un “localhost” por medio de WAMP y es procesada por medio librerías propias de Python para obtener la conexión y envío de datos desde código Python a MySQL.

## E. Flujo de detección y análisis

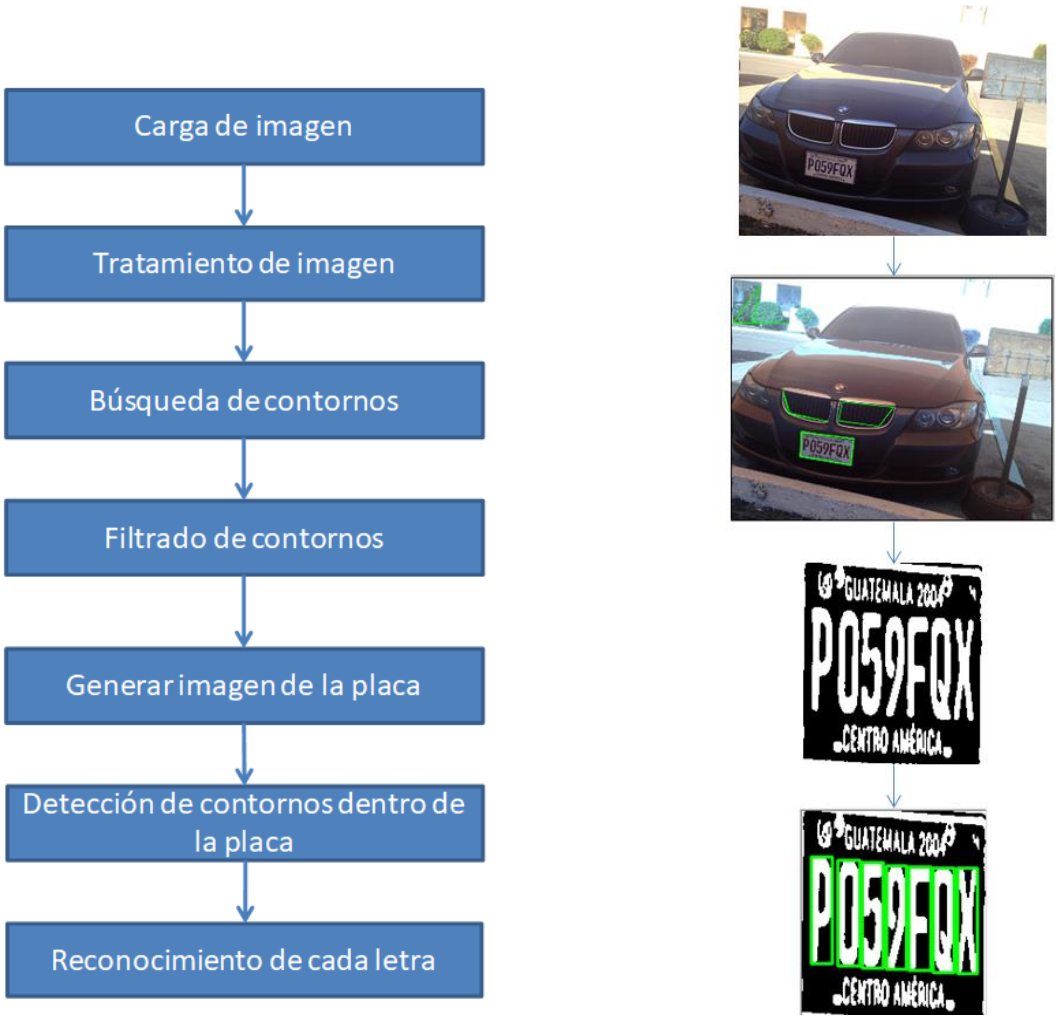



Diagrama 2. Flujo de detección de la placa

# VII. TRATAMIENTO DE IMAGEN

Durante la fase de detección es necesario aplicar varios procesos de filtración a los que en conjunto denominaremos como “tratamiento de imagen”. En este tratamiento de imagen el objetivo es ecualizar y depurar la imagen lo mejor posible. Dichos procesos de filtración puede variar según factores de propios del contorno patrón objetivo a detectar. También existen factores internos como calidad de captura de imagen o factores externos como clima o luz donde se espera realizar la captura.

A continuación se muestra el flujo de filtros y ecualización a realizar durante el proceso de tratamiento de la imagen o entrada de video:

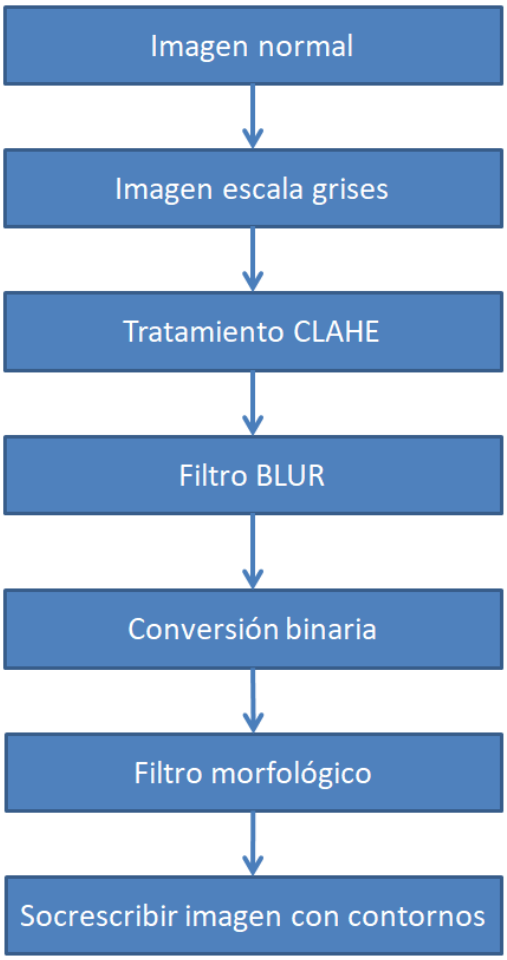



Diagrama 3. Flujo de filtros y ecualización de la imagen

## A. Escala de grises

El proceso de tratamiento de imagen empieza reduciendo la complejidad de la imagen. Esta es convertida a escala de grises eliminando los canales RGB. Esto reduce la complejidad de procesamiento para los siguientes filtros dado que la imagen solo cuenta con una canal de color. Dicho proceso puede realizarse por medio de la misma instrucción de carga de imagen de la librería OpenCV indicándose en los parámetros de la función que la captura de la imagen o video sea por canal de grises.

Figura 4. Imagen convertida en escala de grises

## B. CLAHE

El siguiente paso es la nivelación de contraste en la imagen. El propósito de esta etapa es aumentar la claridad del contorno a detectar, reduciendo el ruido por la luz de ambiente o puntos de luminosidad muy altos o bajos. La configuración de bloques utilizada para la detección de los sistemas fue de 16x16 píxeles para la detección de la placa y de 2x2 pixeles para la detección de los caracteres en la placa. Analizando la imagen de escala de grises se aprecia como la luz debajo del automóvil, pavimento y la reflexiones aumenta la claridad y definición.

Figura 5. Imagen ecualizada por método CLAHE.

## C. Filtrado BLUR

El siguiente filtro a aplicar es desenfoque o “BLUR”. Este filtro permite eliminar ruido y difuminar la imagen. El filtro aplicado en este caso es propio de la libraría OpenCV y permite difuminar la imagen por bloques respetando picos de cambio de color. Enfocándonos en la placa, el aplicar el desenfoque respetara el cambio de color que existe entre la placa y el bumper del vehículo evitando mezclar colores entre ambos y resaltar la placa vehicular.

Figura 6. Imagen filtrada por método BLUR “BilateralFilter”.

## D. Imagen binaria

Una filtro indispensable para la detección de contornos es pasar la imagen a binaria. Esta función pasa el canal grises a negro/blanco absoluto según criterios de conversión. Estos criterios de conversión a binario se indican a partir de escalas para determinar si un punto es considerado un pixel negro (0) o pixel blanco (1), siendo el rango de la escala de grises [0...124] para pixel negro y de [125...255] para un pixel blanco. Esto acelerar drásticamente la búsqueda de contornos dado que ya únicamente existen dos colores predominantes en la imagen.

Figura 7. Imagen convertida a modo binario.

## E. Imagen morfológica

En la imagen del paso anterior “Figura 7” se aprecia cómo el recuadro que comprende la placa es clara y limpia, aun así, puede suceder ocasiones en donde el filtro de desenfoque “BLUR” no es suficiente para eliminar el ruido provocado por los caracteres o factores externos sobre la placa, provocando distorsión al momento de detectar el recuadro del contorno. Una manera de solventar este problema es aplicando un filtro morfológico.

Al aplicar filtros morfológicos se puede manipular la cantidad de pixeles blancos y negros presentes en la imagen, permitiendo intensificar el área de la placa vehicular. Para efectos de este desarrollo se aplicaron dos niveles en las dos modalidades siendo 3x3 píxeles en nivel 1 y 5x5 píxeles para nivel 2 para realizar variaciones en la detección.

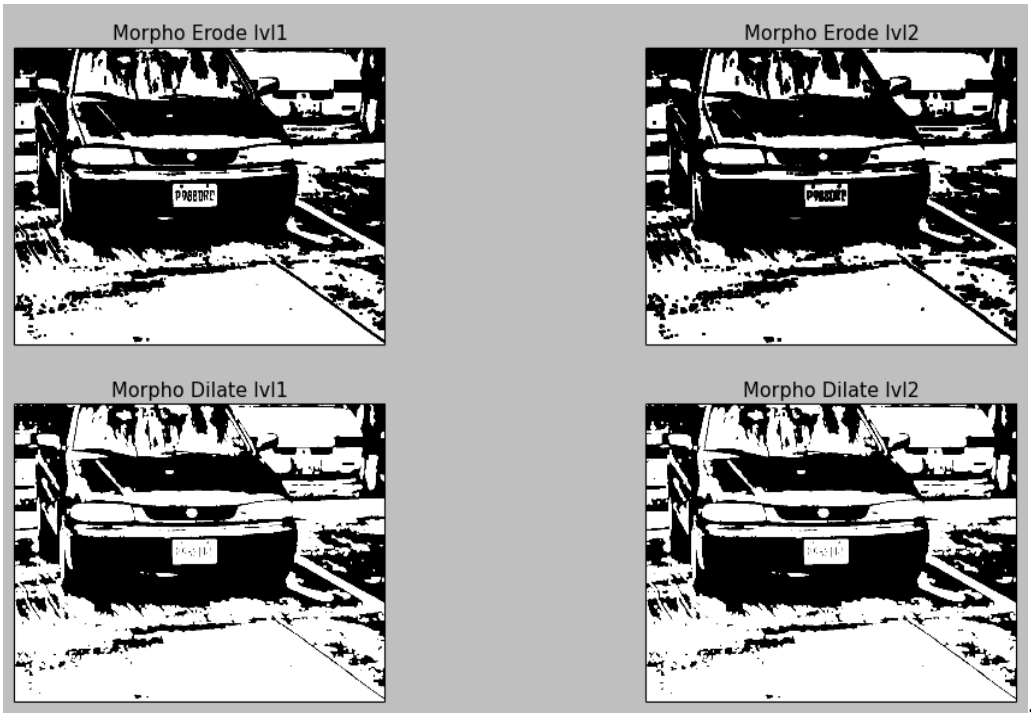



Figura 8. Conversión de imagen en modo erosión y dilatación.

## F. Detección de contornos

Completado el proceso de “tratamiento de imagen” se procede a la detección de contornos. Para la detección esta se realiza por medio de una función propia de OpenCV llamada “findContours”, dicha función genera todo los contornos posibles basándose en polígonos blancos presentes en la imagen. Luego, estos son guardados en un arreglo línea donde se filtra por paramentos de dimensión. Por ejemplo, todo aquel contorno que no tenga cuatro lados o cumplan cierta proporción de ancho y alto es descartado.

Una vez aplicado la función de detección contornos se manda a sobrescribir la imagen o entrada de video con las posibles soluciones ya filtradas. Para efectos de este desarrollo, se intensificó con coloración verde el contorno detectado que contiene el posible patrón de búsqueda.


Figura 9. Sobre escritura de la imagen original con contornos detectados

Durante este proceso se delimita el tamaño mínimo y máximo que de los contornos detectados sobre toda la imagen, permitido valores mínimos de 1/25 del tamaño de la imagen y ¼ el tamaño máximo respecto a la imagen. Variando estos parámetros se establece la distancia mínima y máxima en que una placa será detectada siendo por diferentes pruebas realizadas una distancia aproximada entre 2 a 4 metros del punto de la cámara hacia la placa.

## G. Detección de placa

Todo aquel contorno detectado es aislado y guardado como imagen en formato PNG, sin importar si la detección contiene la placa o un segmento falsamente detectado. La  imágenes son guardadas con el formato PNG.

Se crea una carpeta identificada con el ID de la imagen procesada en donde quedan guardadas todas las imágenes de contornos posibles. Dependiendo de la cantidad de contornos detectados estas carpetas pueden tener un preso aproximado de 1MB a 1.5MB por imagen procesada.


Figura 10. Imagen del contorno de la placa detectada

Otros elementos que son detectados como posibles contornos quedan guardados como contornos falsos positivos. Estos contornos se producen por ruido en la imagen que son detectados por el sistema como un posible contorno que cumple con las características del patrón de interés. A continuación se presentan algunos casos:

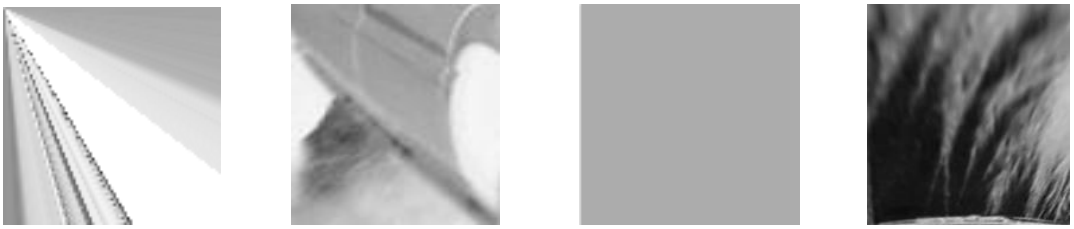


Figura 11. Contornos falso positivos detectados como posibles contornos de la placa.

# VIII. PRUEBAS DE DETECCIÓN

El sistema de captura y el sistema de análisis puede ser parametrizados de modo que puede variarse la distancia y proporción del contorno a detectar. Una vez realizado la fase de tratamiento de imagen se busca encontrar contornos de 4 lados que puedan representar la placa o los caracteres dentro de ella. *A priori*, sabemos que el ancho de la placa es mayor que el alto, por lo que se parametriza la detección para encontrar contornos con esas características. En el caso de los caracteres, se parametriza para encontrar contornos con mayor alto que ancho, agregando que, la proporción del ancho no debe sobrepasar 1/7 del ancho de la placa.

Nótese que todas esta configuraciones se realizan dado que el patrón de búsqueda que se busca detectar son placas. Si en caso fuera otro objeto, debe analizarse este tipo de características para luego configurar los sistemas de captura y análisis a modo que busquen dicho patrón.

Para estimar las distancia de captura se procedió a tomar una muestra (un vehículo) en las distancia de 1, 2, 3, 4 & 5 metros de distancia. Para realizar la medición de ángulo se procedió a poner el foco de la cámara en puntos aproximados con ángulos de +/- 30 grados, +/- 45 grados y +/- 60 grados desde una distancia de 3 metros. Las respectivas pruebas fueron realizadas en el horario de 2:00PM, día soleado y despejado.

NOTA: Siendo únicamente una muestra la utilizada en esta prueba, la detección de la placa pueden variar según otros factores, léase la sección de "*Factores de distorsión*"

# A. Prueba detección por distancia


Distancia: 1 Metro (Frontal)

Resultado: Placa no detectada

Debido a la distancia, los parámetros del sistema de captura están configurados para que no se detecte un contorno que abarque más de ¼ del ancho o alto de la imagen, siendo en este caso el tamaño de la placa aproximadamente 1/2 el ancho respecto al tamaño de la imagen.


Distancia: 2 Metros (Frontal)

Resultado: Placa detectada

El último carácter quedo entrecortado en la detección de la placa.



Distancia: 3 Metros (Frontal)

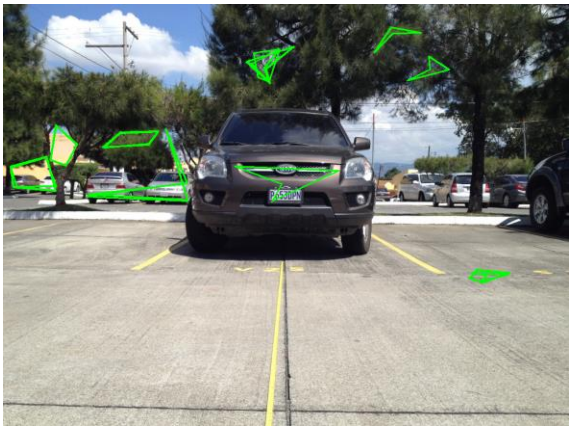
Resultado: Placa detectada

Se produjo una distorsión en la proporción del alto lado izquierdo.

Distancia: 4 Metros (Frontal)

Resultado: Placa detectada

Se produjo una mayor distorsión en la proporción del alto lado izquierdo.



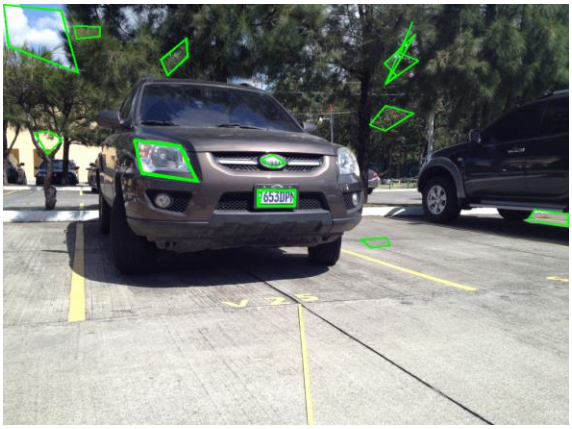
Distancia: 5 Metros (Frontal)

Resultado: Placa no detectada

Debido a la distancia, los parámetros del sistema de captura están configurados para que no se detecte un contorno que abarque menos de 1/25 del ancho o alto de la imagen. Además, a una mayor distancia el tratamiento de la imagen no es suficiente para eliminar ruido, lo que provoca mayor dificultad para detectar contornos muy pequeños.

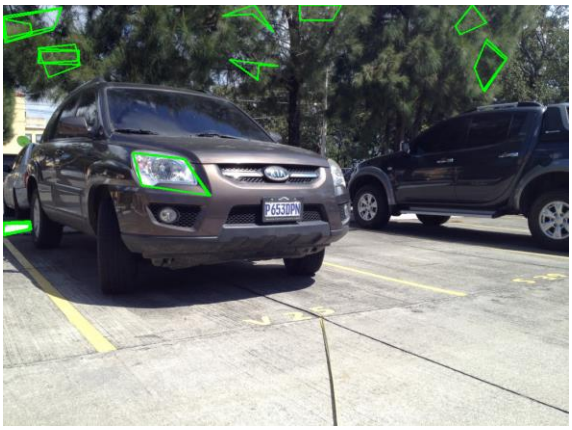
Cuadros 1. Resultados de pruebas de detección por distancias.

## B. Prueba detección por vista angular



Distancia: 3 Metros (ángulo 30 grados negativo)

Resultado: Placa detectada

Distancia: 3 Metros (ángulo 45 grados negativo)

Resultado: Placa no detectada

La luminosidad y reflexión del bumper frontal provoco que en el tratamiento de imagen la placa no quedara aislada correctamente

Distancia: 3 Metros (ángulo 60 grados negativo)

Resultado: Placa detectada

En esta detección la placa fue capturada sin la letra "P".

| | |
|---|---|
|  | Distancia: 3 Metros (ángulo 30 grados positivos)<br><br>Resultado: Placa detectada<br><br>Aunque la placa fue detectada no se logró capturar la letra “N”.<br><br>  |
|  | Distancia: 3 Metros (ángulo 45 grados positivos)<br><br>Resultado: Placa detectada<br><br>Similar a la configuración anterior, la placa fue detectada sin capturar la letra “N”.<br><br>  |
|  | Distancia: 3 Metros (ángulo 60 grados positivos)<br><br>Resultado: Placa detectada<br><br>Nuevamente el último carácter no fue detectado y ya se produce una mayor distorsión en las proporciones de alto del contorno.<br><br>  |

Cuadro 2. Resultados de pruebas de detección por ángulo

# IX. FACTORES DE DISTORSIÓN

Existen factores de distorsión pertenecientes al ambiente o propio de los vehículos causan confusión al momento que se produce la detección de los contornos y caracteres. Durante el proceso de pruebas se detectaron diferentes escenarios que provocan distorsión en la detección, a continuación se enlista varios de ellos:

- Ambiente climático (lluvia o niebla)
- Horario y posición de luz de día
- Luz por faros delanteros o traseros
- Autos con pintura muy clara (blanco, dorado, plateado)
- Detalles cromáticos en los autos
- Diseño en la carrocería que provoca obstrucción
- Placas con el diseño antiguo

## A. Ambiente climático

A partir de las diferentes pruebas realizadas, los dos ambientes climáticos que presentaron mayor dificultad en la detección de la placa fueron cuando el ambiente se encontraba nublado y/o lluvioso. En el caso de ambiente nublado este provoca que la intensidad de luz en el ambiente varié mucho, provocando que las fotos tenga una sensación "fría". (Véase Figura 12) Las fotos fueron tomadas en diferentes días alrededor de las 2:00PM a 3:00PM en ambiente nublado y ambiente soleado.

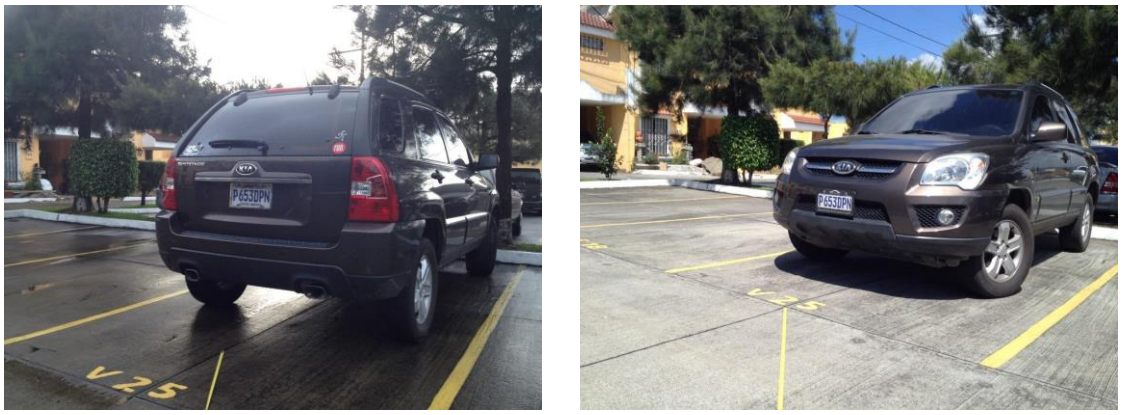


Figura 12. Comparativa de luz en el ambiente en ambiente nublado (izquierda) y soleado (derecha)

Esta varianza de luz en el ambiente provoca variación en la entrada de luz del obturador de la cámara, causando altos cambios de luminosidad de blancos/negros y disminución en la tonalidad de colores. Esto

introduce ruido al sistema de detección, obligando a realizar ajustes personalizados al tratamiento de la imagen a partir de las condiciones climáticas para evitar la distorsión que causa la luz de ambiente.

Una manera de mejorar la luminosidad en estas situaciones es aumentar el área de ecualización de contraste, CLAHE, con el fin de ecualizar bloques de mayor área para nivelar la luminosidad de la imagen de una manera más equitativa.

Otro problema que se presentó durante las pruebas es en ambiente lluvioso, o posterior a ella. La lluvia provoca que el ambiente aumente su nivel de reflexión en el asfalto contra luz, provocando que el detector de contornos identifique segmentos en el asfalto como contornos dada reflexión de la luz. Agregando, según la intensidad de caída de lluvia (gotas), provocará el equivalente a tener ruido sobre la imagen al momento de realizar la captura y pasar por los filtros de tratamiento de imagen, los cuales pueden ser eliminados con el filtro de desenfoque (BLUR).


Figura 13. Reflexión en el asfalto de la imagen original (izquierda) provoca detección de contorno (derecha)

## B. Horario y posicionamiento de luz de día

El horario influye directamente en la cantidad de luz que captura obturador de la cámara. Picos muy altos de oscuridad o luminosidad solo provocan una captura poco visible de la imagen total, causando dificultad en la detección de la placa.

En el caso de poca luz la situación puede variar para bien o mal. En la detección de placas, horarios nocturnos o lugares oscuros, el lente se ve beneficiado si realiza la captura desde la parte trasera del automóvil. Esto debido a que la mayoría de autos cuenta con luz en el porta placas por lo que la claridad de la placa en la imagen sería muy buena. Caso peculiar serían vehículos con mal funcionamiento en las luces de porta placas que no permitiría capturar la imagen claramente en dichas condiciones dado que estaría muy oscuro.

Caso inverso, una gran cantidad de luz puede provocar el efecto de contraluz, similar a la dificultad de ver en los atardeceres. El efecto contraluz sucede al momento de que el lente recibe demasiada luz o tiene regiones con picos elevados de luminosidad. Esto provoca que el lente se cierre para nivelar la cantidad de entrada de luz según su foco, pero a la vez, oscurece el resto de la imagen provocando una imagen poco visible.

Otro caso es la luz del día en posición del horizonte, no directamente en la cámara sino en la reflexión de la carrocería de vehículo. El ángulo de rebote de luz en superficies muy claras provoca que la pintura de los carros sea casi blanca, dificultando el tratamiento de imagen y pareciendo un contorno más en la detección del patrón. Véase los siguientes ejemplos:

Figura 14. Foto frontal del vehículo, ángulo de rebote de luz en el bumper (izquierda) provoca que la pintura en el área sea detectada como blanca (derecha).

Figura 15. Foto trasera del auto, ángulo de rebote de luz en porta equipaje (izquierda) provoca que la pintura en el área sea detectada como blanca (derecha).

## C. Luces de faros delanteros y traseros

En ambientes con poca luz la captura de la imagen para la detección de la placa se complica en ciertas situaciones. Una captura frontal del vehículo con los faros prendidos es una de las situaciones más complicadas, esto debido a la intensidad de la luz de los faros. La complejidad en esta situación se debe a que el obturador de la cámara se cierra contra luz dando como resultado el oscureciendo del resto de la imagen. Al capturar una imagen muy oscura se complica detectar el contorno de la placa dado que se complica el detectar contornos claramente. Una forma de mejorar este tipo de capturas es aplicando un filtro de erosión a la imagen de las funciones morfológicas, permitiendo incrementar el contorno de la placa, aunque también debe mejorarse la iluminación contra el vehículo y la posición de la cámara a modo de obtener una mayor reflexión del contorno de la placa y menor oscurecimiento por el cierre del obturador.

Figura 16. Foto frontal del auto, la intensidad de la luz de los faros (izquierda) provoca áreas blancas en la conversión a binario de la imagen (derecha).

## D. Autos con pintura muy clara

Autos con pintura clara, nótese dorada, plateada, blanca, amarilla, roja, pueden causar dificultad al detector de contornos si la iluminación ambiente de la imagen es alta. Al momento de realizar la ecualización de contraste el exceso de luz ambiental en la imagen y una pintura muy clara o reflectiva causa que sea difícil disminuir la luminosidad de la imagen total. El resultado empeora al momento de pasar la imagen a filtro binario, donde la alta cantidad de pixeles claros causan que la cantidad de pixeles blancos sea excesiva.


Figura 17. Foto frontal del auto, el color de la pintura del auto puede ser muy clara (izquierda) al momento de convertir en imagen binaria (derecha).

## E. Detalles cromáticos en los autos

Detalles cromáticos cerca de la placa o el mismo porta placas son motivo de reflexión o confusión para el detector de contornos. La alta luminosidad que provocan contra cámara puede causar distorsión en donde el detector espera encontrar el contorno de la placa. Mucha de la distorsión se produce al momento que la imagen pasa por el filtro binario, en donde  la mayoría de casos los detalles cromáticos serán convertidos a blanco.

Figura 18. Foto trasera del auto, detalle cromático cerca de la placa (izquierda) puede provocar confusión en la detección de la placa (derecha)

## F. Diseño de la carrocería provoca obstrucción

Dependiendo del diseño del auto, la placa puede verse obstruida por sombras internas o el mismo diseño de la carrocería. Esto dificulta la detección del contorno de la placa dado que la puede detectar entre cortada. Para este tipo de problema la mejor solución es identificar el ángulo óptimo para poner en punto de detección de la cámara.

Figura 19. Foto trasera del auto, diseño de la cajuela provoca sombra interna (izquierda) y no permite visualizar correctamente toda la placa (derecha).

Otro tipo de problema que surge en vehículos tipo camionetas o pick-ups son las defensas metálicas. Las compañías de automóviles diseñan sus vehículos para dar visibilidad a la placa como parte de sus estándares, pero estas defensas suele ser una extensión ajena y no contemplada en el diseño del mismo vehículo, por lo que son propensas a obstruir la visualización clara de la placa.

Figura 20. Foto frontal del auto, la defensa no permite visualizar la placa del automóvil.

## G. Placas con el diseño antiguo

Placas vehiculares con el diseño anterior causan distorsión en su detección a que el fondo de la plan no es blanco totalmente. Además, la pintura de los caracteres en este tipo de placas es menos brillante y reflectora como la pintura utilizada en las placas actuales. Esto causa que la captura de la placa sea de baja definición provocando problemas al pasar la imagen por el filtro binario, el cual genera ruido en los caracteres de la placa.

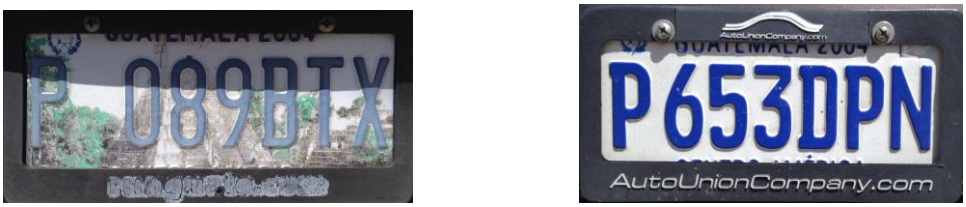
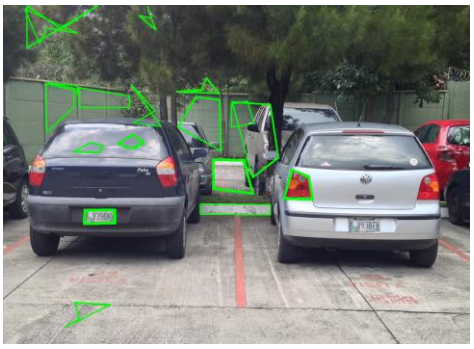



Figura 21. Comparativa entre la placa con antiguo diseño (izquierda) y la placa actualmente utilizada (derecha).

El diseño de las placas actuales está hecho para resaltar los caracteres de la placa sobre fondo blanco. El aplicar el tratamiento de imagen permite resaltar las letras a modo de facilitar el aislamiento de cada carácter. Debido a que las placa antiguas tiene en el fondo un diseño artístico, los filtros y ecualización en el tratamiento de imagen causan ruido dentro de la placa. Esto provoca distorsión al momento de realizar la etapa de aislamiento de los caracteres de la placa. El motivo principal de la falla es que los caracteres tienden a fusionarse con elementos del diseño de fondo haciéndolas irreconocibles para la fase de detección de caracteres.

| | |
|---|---|
|  | Dos carros con placas con antiguo diseño. Vehículo derecha no detecto la placa. El vehículo izquierdo logro detectar el contorno de la placa beneficiándose del contraste de color del bumper trasero.<br><br>Resultado: Placa detectada pero imposible de identificar los contornos de los caracteres. |

Cuadro 3. Resultados de pruebas de detección de placas con antiguo diseño.

Por lo tanto, placas vehiculares con valores “P ### DG#” o inferior, las cuales tienen el diseño antiguo, difícilmente podrán ser identificadas con este sistema. Tendría que mejorarse el tratamiento de imagen o aplicar otras técnicas para aumentar la definición de estas placas.

# X. PRUEBAS DE DETECCIÓN DE CONTORNOS

## A. Pruebas realizadas en ambientes con malas condiciones

**Descripción prueba:** Captura trasera del vehículo en horario nocturno, reflexión de luz de faros y luz en la placa en contra.

**Resultado:** Captura de placa realizada pero con desenfoque por poca luz en el obturador.



**Descripción prueba:** Captura trasera del vehículo en horario nocturno, reflexión de luz de faros y luz en la placa en contra

**Resultado:** Captura de placa realizada pero con desfase. Posiblemente legible para detección de caracteres.



**Descripción prueba:** Captura angular con poca luz, reflexión de luz de faros y luz en la placa en contra.

**Resultado:** Captura de placa realizada pero con desfase. Posiblemente legible para detección de caracteres.



Cuadro 4. Resultados de pruebas realizadas en malas condiciones

## B. Pruebas exitosas en condiciones óptimas.

**Descripción prueba:** Captura frontal del vehículo en vista angular. Condición climática nublada.

**Resultado:** Captura completa y legible para detección de caracteres.




**Descripción prueba:** Captura trasera en vista angular pronunciada. Condición de luz ambiental regular. Auto bajo techo lo que provoca mayores sombras internas.

**Resultado:** Captura completa y legible para detección de caracteres.



**Descripción prueba:** Captura frontal en vista angular poco pronunciada. Condición climática despejado y soleado con iluminación ideal.

**Resultado:** Captura completa y legible para detección de caracteres.




**Descripción prueba:** Captura trasera del vehículo en vista angular poco pronunciada. Condición climática nublada y con poca luz ambiental.

**Resultado:** Captura completa y legible para detección de caracteres.



**Descripción prueba:** Captura trasera del vehículo. Condición climática nublada con luz ambiental regular.

**Resultado:** Captura completa pero entrecortada en algunos caracteres, posiblemente no se detecte todos los caracteres.



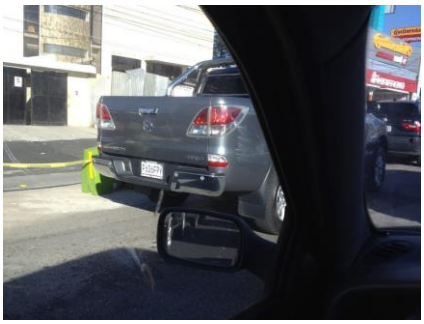
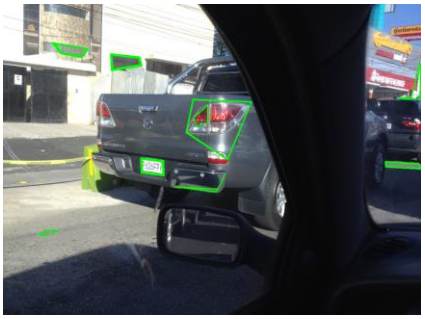
**Descripción prueba:** Captura trasera del vehículo. Condición climática nublada con luz ambiental regular.

**Resultado:** Captura entrecortada en algunos caracteres, posiblemente no se detecte todos los caracteres.



**Descripción prueba:** Captura trasera en vista angular pronunciada. Condición climática despejado y soleado. Altas reflexiones presentes por detalles cromáticos.

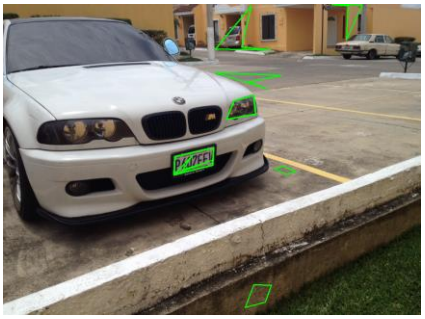
**Resultado:** Captura completa, caracteres borrosos pero con posibilidades de detección si se aplican filtros extra morfológicos de dilatación.



**Descripción prueba:** Captura frontal en vista angular poco pronunciada. Condición climática despejado y soleado.

**Resultado:** Captura completa y legible para detección de caracteres.

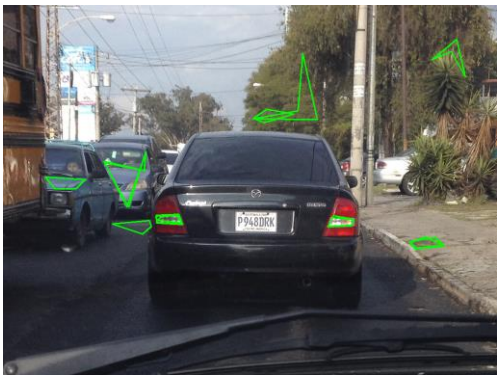


Cuadro 5. Resultados de pruebas exitosas realizadas en condiciones variadas

## C. Pruebas fallidas en condiciones variadas

**Descripción prueba:** Captura trasera del vehículo. Condición climática despejado y soleado

**Motivo fallo:** El contorno de la placa no se detectó por ruido en los alrededores provocados por la reflexión de la luz contra la carrocería. Esto causó que el sistema no detectará un patrón rectangular.

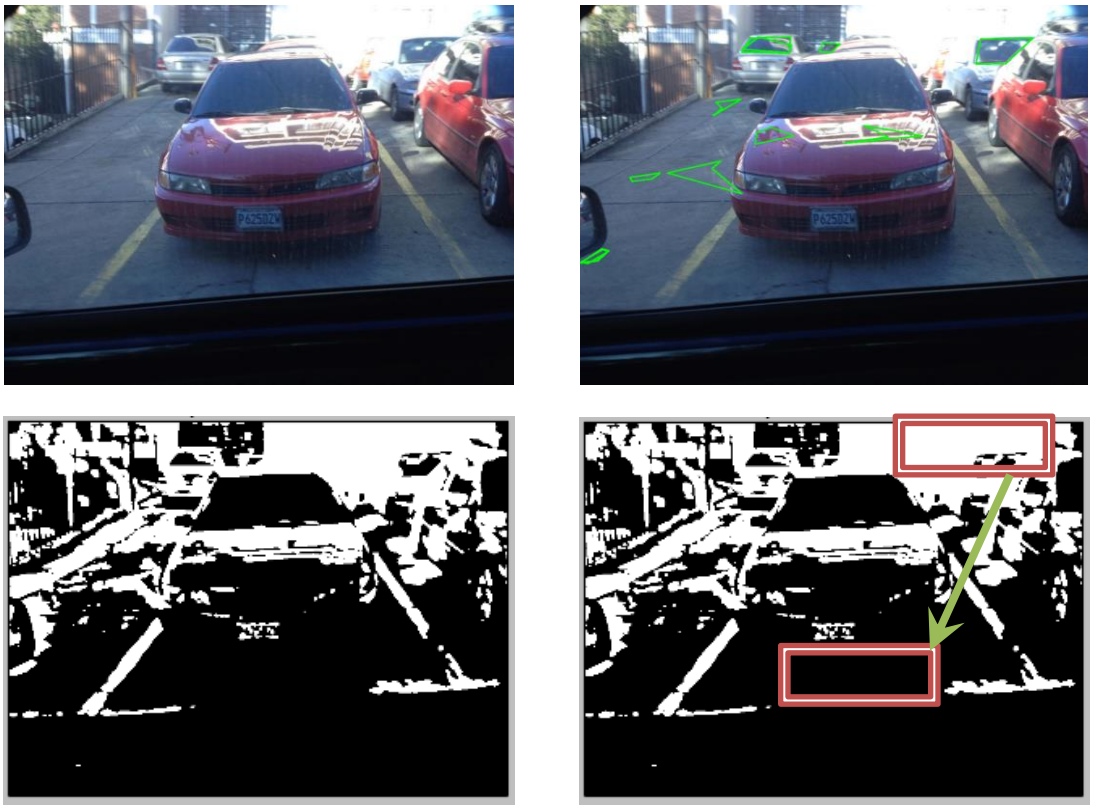


**Descripción prueba:** Captura trasera del vehículo con vista angular. Condición climática nublada y luz ambiental regular.

**Motivo fallo:** Hay presente una cámara en el marco de la placa, causando ruido al tratar de detectar el patrón rectangular. Agregando, la misma cámara provoca una sombra interna distorsionando la forma rectangular del patrón.

**Descripción prueba:** Captura frontal del vehículo. Condición climática despejado y soleado pero bajo sombra.

**Motivo fallo:** El cambio de luz que hay entre el fondo y el punto de interés donde está situada la placa tiene un cambio de contraste drástico. Esto causa que el obturador de la cámara se cierre para nivelar la luz del foco. A pesar de ello, no es lo suficiente para aumentar la claridad de la imagen lo que provoca que no aclare suficiente para el filtro binario.



**Descripción prueba:** Captura frontal del vehículo. Condición climática despejado y soleado con buenas condiciones de iluminación.

**Motivo fallo:** Alta claridad presente en la imagen dado el color de la pintura del vehículo. Esto puede provocar que el filtro CLAHE no logre nivelar correctamente la imagen y no resalta el contorno de la placa.

Cuadro 6. Resultados de pruebas fallidas realizadas en condiciones variadas

# XI. ENTRENAMIENTO DEL SISTEMA DE IDENTIFICACIÓN

Para el entrenamiento del sistema de identificación se debe conocer los elementos que deberá identificar, en este caso caracteres. También es necesario identificar los escenarios comunes y las diferentes posibilidades que se pueden presentar en el proceso de identificación. Por ejemplo, el patrón de los caracteres de una placa vehicular particular está formado por la letra "P" empezando por el lado izquierdo seguido de tres dígitos, del 0 al 9, otros tres caracteres del abecedario proceden luego de la numeración. De los caracteres abecedario son descartados vocales. Dado el supuesto anterior, no hay necesidad de entrenar vocales, dado que no son utilizados en placas vehiculares particulares.

Otra mejora para el módulo de identificación es separar por segmentos, nótese, luego del carácter "P", los siguientes tres caracteres deben ser números y seguido de un segmento de caracteres del abecedario. Conociendo esto se puede mejorar el algoritmo para que comprenda que estimar durante la identificación del objeto.

Para entrenar este módulo se extrajo diferentes caracteres de un grupo de fotografías de placas muestra recolectadas durante el proceso de pruebas de detección de contornos y se procedió a aislar los caracteres a modo que cada letra quedase en una única imagen.

Figura 21. Diferentes viñetas utilizadas para el entrenamiento del sistema identificación

Para aumentar la variedad y mejorar la aproximación del reconocimiento de caracteres se introdujo tres viñetas de cada carácter. Entre más viñetas se introduzca la precisión en el entrenamiento aumenta la capacidad de aproximación de identificación, para ello se emplea el algoritmo k-NN para determinar cuál es la representación es la más próxima. Todas las viñetas se re-escalaron a una altura de 80 pixeles mientras que el ancho varía según el aspecto original de la imagen.

Figura 22. Variación de viñetas según cada letra o dígito.

A partir de todas la viñetas generadas se unificaron en una sola imagen para cárgalo en el entrenador del reconocedor de caracteres. Empleado el programa “Gdx-TexturePacker.jar” perteneciente al framework Libgdx utilizado para programación de dispositivos móviles, se consiguió crear una imagen unificada de todas las viñetas.

Figura 23. Imagen de viñetas unificadas por el framework Gdx-TexturePacker.

El programa utilizado para entrenar el sistema de identificación se denominó CameraTrain.py, software desarrollado en lenguaje Python. En este programa se carga la imagen unificada de todas las viñetas y luego por medio de reconocimiento de sectores el programa detecta una por una cada letra y se ingresa la equivalencia de cada carácter. En ocasiones el detector de sectores reconoce un sector interno de un carácter, como el centro de los caracteres “0”, “Q”, “8”, “B”, “9”, “P”, de reconocer este tipo de sector se procede a descartar la entrada para que el motor de detección no lo incluya.

El resultado de este proceso genera dos archivos clase propios del motor de reconocimiento. El primero archivo, denominado “general-responses.data”, guarda las equivalencias de ingreso de cada carácter al momento de identificar los sectores. El segundo archivo, denominado “general-samples.data” guarda las viñetas en un archivo codificado el cual contiene valores de punto flotante que son la representación de cada carácter.

Al momento de realizar la detección de caracteres con el programa “CameraDetection.py”, se cargar el modelo de reconocimiento con el algoritmo k-NN con los dos archivos anteriores. Luego, por medio de funciones propias de OpenCV se manda a solicitar una búsqueda de cada carácter identificado en la placa y por medio la función “Find_Nearest” se manda a solicitar la representación que mejor se aproxime al carácter aislado enviado al motor de reconocimiento.

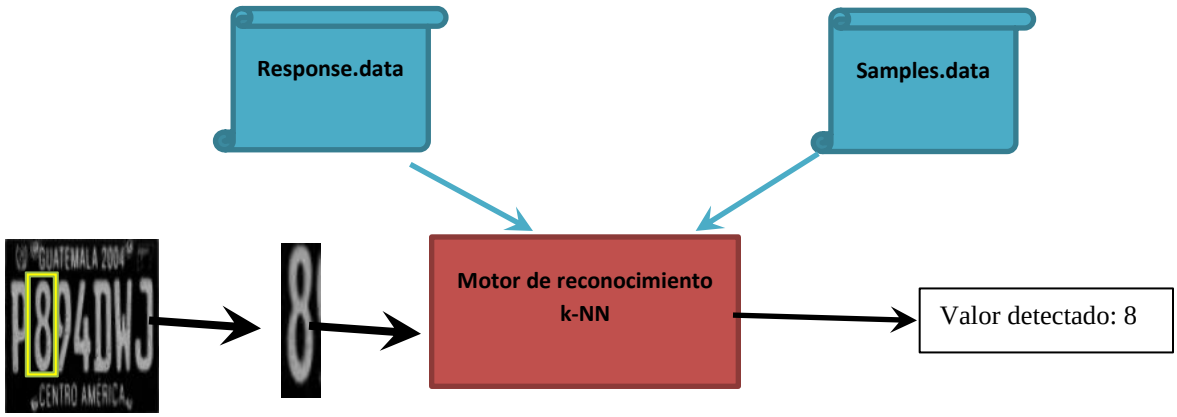



Diagrama 4. Flujo de detección del motor de reconocimiento

# XII. DETECCIÓN DE CARACTERES

El programa desarrollado para la detección de caracteres en la placa se desarrolló en lenguaje Python. Debido que, los archivos creados por entrenamiento para el motor de detección k-NN están codificadas para ejecutarse en lenguaje Python. A este subsistema se denominó como “Sistema de Análisis”. Al igual que sub-sistema que captura que aísla la placa de la entrada de video o imagen, el sistema de análisis utiliza la misma metodología, realizado un barrido en búsqueda de patrones que se asemejen a caracteres y los aísla.

Al momento de realizar el barrido sobre la imagen de la placa, el sistema de detección busca posibles contornos carácter y luego el programa a base de criterios verifica y aisla. El programa está configurado para descartar cualquier contorno detectado el cual su ancho (W) sea mayor a su alto (H), dado que, todo carácter en las placas vehiculares debe ser más alto que ancho. Este criterio está con base en las diferentes pruebas realizadas con diferentes placas vehiculares.

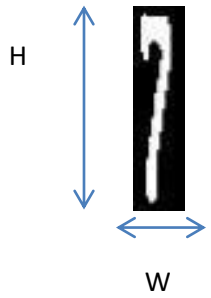


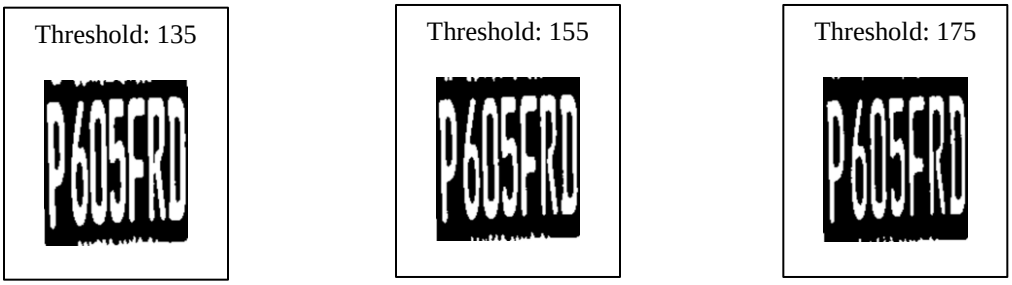

Figura 25. Muestra de las proporciones de la viñeta siendo (H) el alto y (W) el ancho.

A partir de este criterio, se procede a descartar posibles contornos detectados como falsos positivos. En algunos casos, decoraciones en el marco de la placa vehicular o ruido que se aisló junto a la imagen durante el proceso de captura pueden ser tomados como posibles contornos carácter falsos positivos.


Figura 26. Ruido en la conversión binaria provocada por el diseño de marco de la placa

Una vez identificados los posibles caracteres presentes en la imagen se procede a comparar uno por uno utilizando las funciones de reconocimiento del motor de detección k-NN, esté previamente ya entrenado.

Para realizar pruebas de variación durante la fase de detección de cada carácter, se procedió a variar el parámetro de la función de OpenCV “Threshold” durante la conversión binaria. Esta función delimita en que rango de coloración un píxel cuando es convertido a negro o blanco, permitiendo variar la densidad de pixeles blancos de cada carácter. Variar este parámetro permite aumentar o disminuir el aislamiento sobre la detección, dado que, a una menor densidad de blancos por carácter el aislamiento es mejor pero la identificación del carácter disminuye, lo mismo aplica de forma inversa.



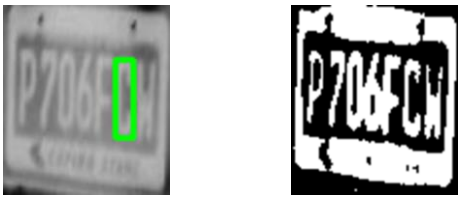

Cuadro 7. Variación del parámetro disparador “Threshold” para la conversión binaria.

# XIII. PRUEBAS DE DETECCIÓN E IDENTIFICACIÓN DE CARACTERES

Las imágenes de placas vehiculares utilizadas para estas prueba fueron aquellas que estuviesen en su gran mayoría completas. Los resultados muestran la cantidad de caracteres aislados de la placa y el porcentaje de acierto en la identificación de equivalencia de carácter. El nivel de acierto se basa en la cantidad de caracteres aislados sobre la cantidad que lograron ser identificados con su letra o número correspondiente. En los resultados se mostrará el valor "#", representando un carácter no identificado y se introduce para mantener el orden de la secuencia de los caracteres de resultado de la placa vehicular.

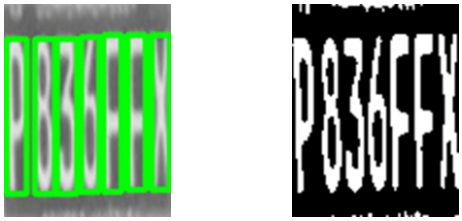
Aislados: 1/7 Acierto: 100%

Resultado: "# ### #C#"



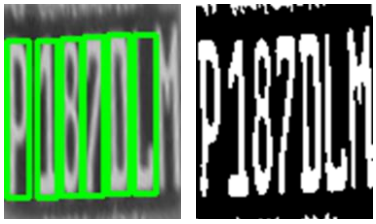
Aislados: 7/7 Acierto: 86%

Resultado: "P B36 FFX"



Aislados: 6/7 Acierto: 50%

Resultado: "P 1B7 81#"



Aislados: 6/7 Acierto: 50%

Resultado: "9 27# WZM"

Aislados: 7/7 Acierto: 86%

Resultado: "P 745 DSM

Aislados: 6/7 Acierto: 83%

Resultado: "P #N5 FFX

Aislados: 4/7 Acierto: 100%

Resultado: “# 349 #R#”

Aislados: 7/7 Acierto: 71%

Resultado: “P 988 8R8”

Aislados: 6/7 Acierto: 33%

Resultado: “P 533 M#X”

Aislados: 4/7 Acierto: 33%

Resultado: “# 18Q ##J”

Aislados: 5/7 Acierto: 80%

Resultado: “P #51 D#H”

Aislados: 7/7 Acierto: 71%

Resultado: “P 353 PPN”

Aislados: 7/7 Acierto: 71%

Resultado: “P Q94 PWJ”

Aislados: 7/7 Acierto: 86%

Resultado: “P 647 FFG”

Cuadro 8. Resultados del sistema de identificación y detección de caracteres.

# XIV. ALMACENAMIENTO EN LA BASE DE DATOS

La fase de almacenamiento es la etapa final en la cadena de procesos de los subsistemas anteriores. Su propósito es generar un registro histórico de las placas capturadas y analizadas. Una vez realizado la fase de detección e identificación se genera una cadena de caracteres (String) con el valor resultado, esta cadena de caracteres es enviada por medio de un comando de Python a un WebServices el cual crea un "Query" a la base de datos MySQL para el registro del mismo.

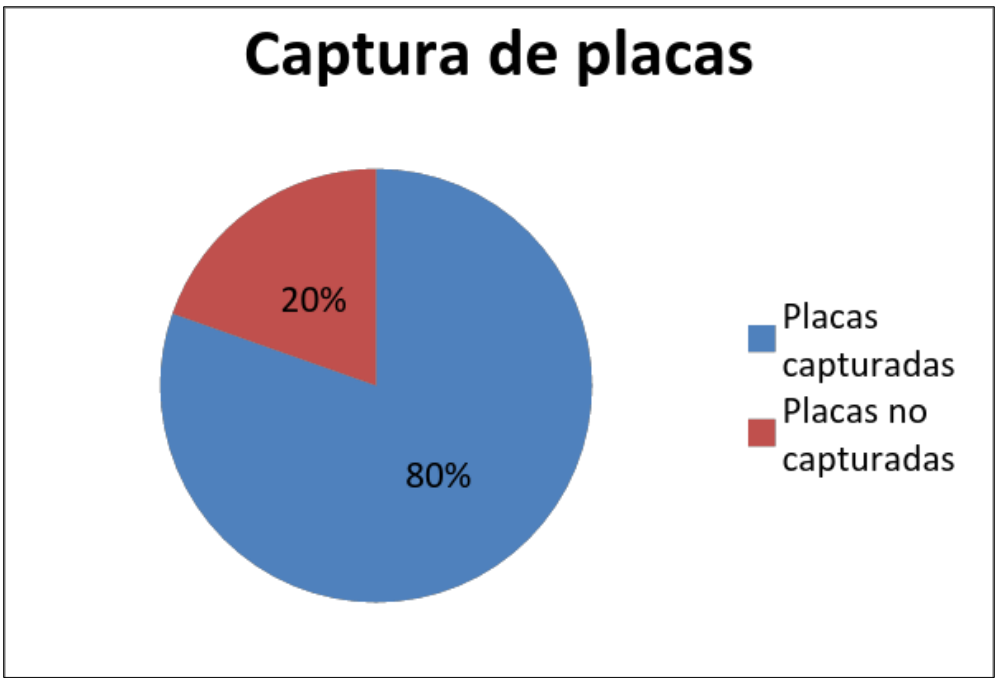
El soporte de la base de datos se realizó por medio del servicio WAMP, el cual se ejecuta en paralelo en el sistema operativo. El servicio al estar siempre presente, recibe solicitudes de nuevas entradas de registro y por medio de una interfaz instalada en el sub-sistema de detección en Python se ejecuta un comando al momento que se concluye con el proceso de identificación de caracteres enviando el resultado a la base de datos.


Figura 27. Captura de pantalla muestra la recolección de las placas detectadas en el software WAMP.

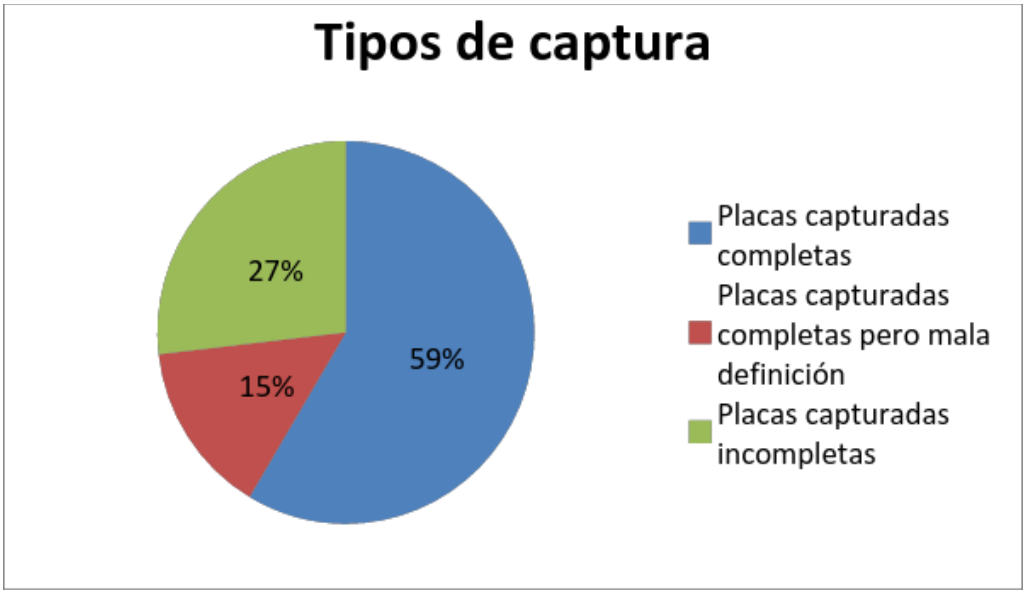
Los campos para la base de datos para este proyecto se limitaron a un ID único e incrementable automático con el que se identifica la entrada del registro a la tabla y el resultado de la placa vehicular detectada.

Otros campos puede agregarse para aumentar la cantidad de información, de modo que la base de datos se enriquezca de información valiosa que permitan a futuro realizar estadísticas o investigación de comportamiento. Algunos campos extra serian: horario de registro, el ID de la cámara, parámetros propios de la cámara y de los subsistemas.

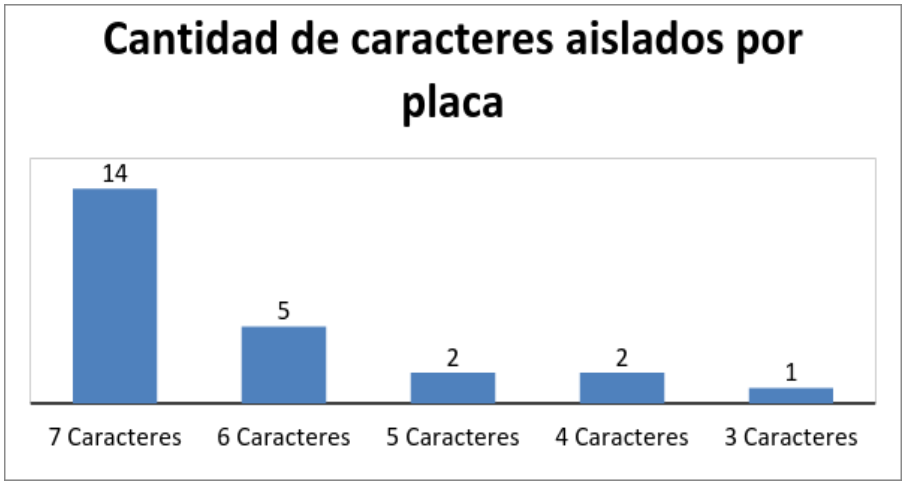
# XV. RESULTADOS



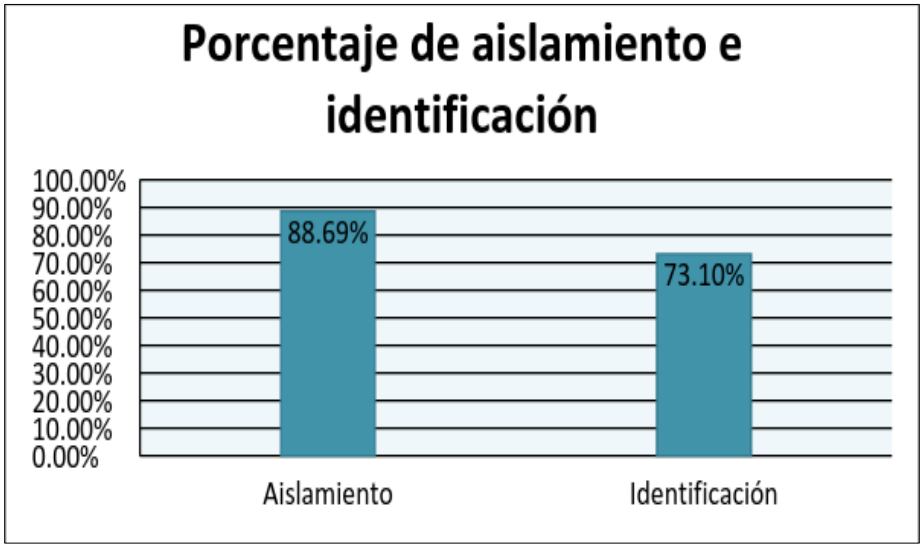

Gráfica 1. Porcentajes de éxito en detección de contorno objetivo.




Gráfica 2. Porcentajes de las condiciones de las placas detectadas.




Gráfica 3. Cantidad de caracteres identificados en las pruebas.




Gráfica 4. Porcentajes de éxito en el aislamiento e identificación de los caracteres.

# XVI. ANÁLISIS DE RESULTADOS

## A. Resultados sistema de captura, aislamiento e identificación.

En total se utilizaron 51 fotos para detección de placas vehiculares y 11 para realizar las pruebas de distancia y ángulos. Las 62 fotos pasaron por el proceso de captura y aislamiento, posteriormente fueron procesadas por el sistema de detección de caracteres.

A continuación se presenta los resultados de captura y aislamiento utilizando las 51 fotos para dicha pruebas. No se tomaron las 11 para realizar las pruebas de distancia y ángulo en estos resultado, esto debido que, no contribuyen en un resultado verídico por el motivo que fue utilizado el mismo auto en condición climática controlada, disminuyendo la variabilidad en los resultados.

En los resultados de captura y aislamiento de la placa vehicular *(véase Grafica 1)* se detectaron 41 placas vehiculares exitosamente, siendo esto un 80% de éxito en localizar el contorno objetivo en una imagen estática.

De las 41 fotos de placas capturadas y aisladas, se dividieron en tres diferentes tipos *(véase Grafica 2):* **placas completas (59%)**, las cuales son aquellas en donde los siete caracteres de la placa están contenidos en la imagen y muestra alta definición. **placas completas pero en mala definición (15%)**, son aquellas imagen en donde la placa encierra los siete caracteres de la placa, pero por bajas condiciones de luz ambiental u otros factores no son claras o están desenfocadas. Y la última categoría son **placas incompletas (27%)**, en donde la placa fue capturada pero uno o dos caracteres están cortados o fuera de la imagen.

Con base en las 24 **placas capturadas completas (59%),** se procedió a pasar las imágenes aisladas de las placas vehiculares por el sistema de identificación de caracteres, resultados véase *Grafica 3*. Teniendo 7 caracteres por placa vehicular y siendo 24 las placas vehiculares procesadas se tiene un total de 168 caracteres a aislar e identificar. El promedio de aislamiento de contornos carácter fue de 6.21 caracteres por placa (149 caracteres identificados), lo que da un porcentaje de 88.69% de identificación de caracteres en la placa.

Una vez aislado los contornos carácter se procede a reconocer su identificación. Sobre el 6.21 de aislamiento de contorno carácter en la placa vehicular, el promedio de identificación fue de 4.5 caracteres por placa vehicular. Dando como resultado un porcentaje promedio de acierto de 73.095%. Siendo la letra "P" la que más veces fue detectada (tres veces no identificada y tres veces reconocida como otro carácter), el valor "4" tuvo 100% de reconocimiento en todas las placas en donde estuvo presente, una placa con

100% de acierto pero con únicamente cuatro caracteres identificados y seis placas con un reconocimiento de seis caracteres sobre siete identificados. Los resultados generales pueden observarse en la *Grafica 4*.

Los caracteres que provocaron mayor conflicto durante el proceso de reconocimiento de las 24 placas debido a factores fueron:

- "8" que fue reconocido como "B" en varias placas
- "8" que fue reconocido como "Q" en una placa
- "D" que fue reconocido como "8" en varias placas
- "D" que fue reconocido como "P" en una placa
- "P" que fue reconocido como "Q" en una placa.

De las 24 placas utilizadas en las pruebas de identificación y reconocimiento, ninguna identificó en su totalidad la placa vehicular. En su mayoría, se tuvo una alta aproximación faltándoles un carácter por aislar o por identificar. Tomando los el porcentaje de detección de la placa (80%), el porcentaje de placas que fueron capturadas completas (58%), el porcentaje de identificación de caracteres (88.69%) y el porcentaje de reconocimiento de un carácter (73.10%) , el porcentaje de éxito de capturar una placa e identificarla se aproxima a un 30%.

Factores que afectan directamente a este bajo porcentaje de identificación son por la falta de entrenamiento de motor de detección k-NN. Entre mayor sea la cantidad de viñetas ingresadas, mayor es la posibilidad de éxito de identificar el carácter.

## B. Tiempos de ejecución del software

En los sistema desarrollados se tiene tiempos de ejecución diferentes. Se analizaron los tiempos de tres tipos de sistema, dos de tipo captura y procesamiento y uno de análisis. Los dos sistema de captura y procesamiento están en modalidad imagen estática y video en tiempo real mientras que el sistema de análisis sólo está en modalidad imagen estática. Nótese, las versión de imagen estática están desarrollada en lenguaje Python mientras que la de tiempo real es C++.

Los sistema de captura y aislamiento de la placa vehicular comprende de la fase de cargar la entrada de datos, aplicar filtros y ecualización, detección de contornos y respuesta de imagen original con el contorno de la placa vehicular desplegada. En el caso de la modalidad imagen estática estos pasos tomaron un tiempo mínimo de 200ms hasta un máximo de 400ms, bajo las especificaciones de hardware mencionadas anteriormente. En el caso de modalidad video tiempo real el aislamiento y despliegue de la imagen se logró realizar en tiempo real sin problemas de lentitud bajo una velocidad de 12 frames por segundo (FPS),

siendo un aproximado de 83ms el tiempo de captura y procesamiento por frame.

El sistema de análisis de caracteres de la placa vehicular aislada comprende de las fases de cargar la imagen de la placa aislada, filtrar y ecualizar la imagen, detectar contornos carácter, realizar un barrido de los contornos carácter y generar la búsqueda e identificación aproximada, por último ordenar el resultado y guardarlo en la base de datos. Todos estos pasos en un tiempo aproximado entre 120ms a 170ms, bajo las especificaciones de hardware mencionadas anteriormente.

El tiempo total de ejecución en los dos sistema modalidad imagen estática en el peor de los casos de 570ms y en el mejor de los casos de 320ms. Debe tenerse en cuenta en el sistema de captura y procesamiento en modalidad imagen estática debe realizarse un filtrado o mejorarse la detección de falsos positivos previo a pasarlo al sistema de análisis a modo de solo pasar una sola imagen con la placa vehicular aislada.

En cuestión del rendimiento según el lenguaje de programación se notó mucha mayor velocidad utilizando C++. La librería de OpenCV está originalmente desarrollada en C, mientras que la versión para Python es un “wrapper” extraído de la original de C. Dado esto, se puede notar que la versión de Python es mucho más lenta para análisis, más aún si requiere análisis en tiempo real. Por ello es recomendable realizar el desarrollo en C++ si se desea resultados rápidos.

# XVII. ESCALABILIDAD Y TIPOS DE SERVICIO

Con base en los resultados anteriores, se estima que el sistema de detección de contornos y aislamiento de la placa no requiere de demasiada demanda de procesamiento CPU. Un núcleo del procesador es suficiente para realizar la labor de captura, filtrado y aislamiento para una cámara o incluso más. Bajo esta estimación, se puede proponer el desarrollar plataformas de detección pequeñas y portables a bajo costo.

Para el sistema de análisis e identificación de caracteres, este se apoya directamente del "Machine Learning" k-NN. El entrenamiento de este motor de identificación de caracteres fue pequeño y simple, ideal para un ambiente controlado y con poca variación en los datos de entrada. Entornos con gran variación en los datos de entrada pueden causar problemas, donde situaciones como clima, luz de día y obstáculos naturales puede generar gran variación en la entrada de datos; esto obliga a realizar un entrenamiento más amplio en el motor de identificación de caracteres. Un entrenamiento con mayor cantidad de muestras de entrada implica un mayor tiempo de entrenamiento, alta demanda computacional, y conseguir grandes cantidades de muestras. Este incremento impacta directamente al tiempo necesario para finalizar el entrenamiento del "Machine Learning" y es aquí donde el uso de GPU se vuelve indispensable dentro del ordenador.

Dependiendo de la necesidad del proyecto donde se desea emplear la detección e identificación de objetos la potencia computacional será diferente. Estimando que un ordenador portátil será capaz para soportar una sola cámara, se puede escalar en las especificaciones de hardware a partir de esto y escalar según la demanda de entradas y complejidad de análisis que se tenga. También aplicar modelos PaaS o SaaS dentro de los sistemas a modo de permitir la escalabilidad dinámica en escenarios muy demandantes o como alternativa de servicio a proveer.

Siendo así, cada uno de los subsistemas desarrollados pueden adaptarse a la demanda de cada modelo de negocio y modularizar para seguir trabajando en conjunto o separado según la demanda del cliente. A continuación se analiza los diferentes tipos de plataformas hardware posibles y su implicaciones en fases clave del proyecto, siendo estos

- Ordenador portátil: Rapsberry Pi o Netbook costo aproximado de $150 - $300
- Ordenador simple: Computador personal costo aproximado de $800 - $2000
- Ordenador avanzado: Servidor o uso de PaaS y Saas, usando AWS con Tier t2.xlarge a un m4.xLarge con costo de $0.4 a $0.8 dólares la hora bajo demanda ($3500 - $7000 anual)

# A. Sistema de captura y procesamiento

El sistema de captura y procesamiento puede configurarse en modalidades, tiempo real por cámaras de video o imagen estática por medio de fotos, cada una tiene ventajas y desventajas en rendimiento y configuración donde será necesario evaluar la inversión o la necesidad del cliente .

En un ordenador portable es importante evaluar la cantidad de frames por segundo si se desea un sistema de análisis en tiempo real. Con buenas optimizaciones en el software, el ordenador portátil sería capaz de realizar análisis y procesar imágenes en ambientes controlados con una inversión muy baja. Es muy probable que por la capacidad computacional, la cantidad de cámaras posibles a conectar seria de una o dos. El uso de webcams es una alternativa para no incurrir en altos gastos obteniendo una calidad de video intermedia. Para este tipo de esquema, lo ideal es entregar el paquete de software y hardware "customizado" para la necesidad del cliente.

En caso de un ordenador simple, este es ideal para centro de monitoreo fijo, con el fin de soportar una mayor cantidad de cámaras con mejor capacidad de captura y definición en la imagen. Puede utilizarse cámaras webcam de mayor definición o incluso el uso de cámaras tipo IP. Utilizando un router de vigilancia se puede alcanzar la capacidad de 16 o incluso 32 cámaras por ordenador utilizando un router especializado, en la mayoría de casos 2 a 8 son las que se comercializan en kits. A una mayor cantidad de cámaras es de considerarse el uso de GPU.

Para aplicar el sistema en una red de cámaras considerablemente alta es de plantear el uso de un ordenador ya avanzado, como un servidor. Con una gran cantidad de cámaras es necesario analizar la arquitectura del sistema para su escalabilidad.

Para los dos anteriores, ordenador simple y ordenador avanzado, se contaría con dos esquema de interacción con el usuario final. El primer esquema, uno donde se le brinde el software de captura y procesamiento para ser instalado en ordenadores propios. El segundo esquema el usuario genera a su forma la captura de la imagen y envía la entrada de imagen o video a un sistema de procesamiento en la nube por medio de un API donde se procese y este devuelve un resultado.

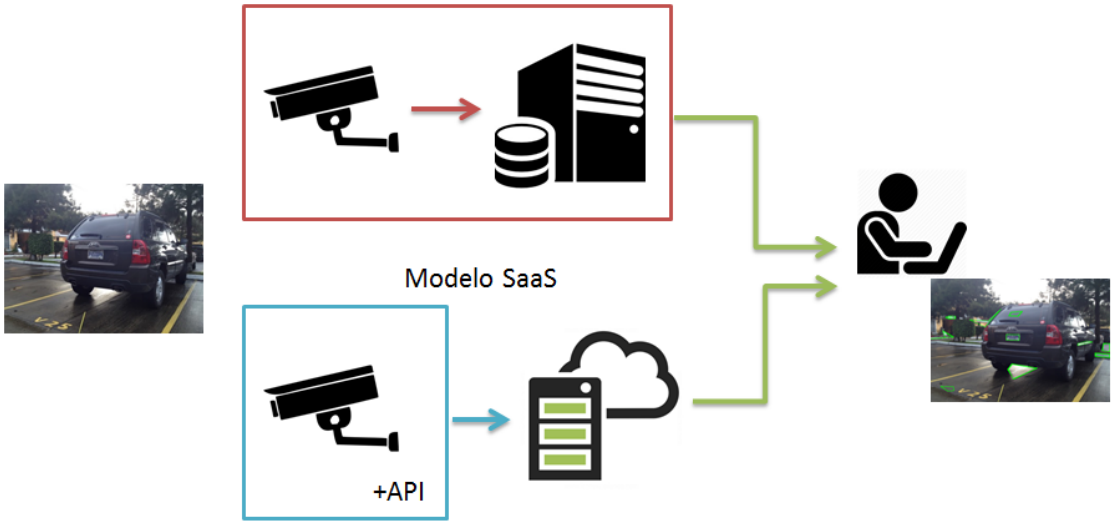



Diagrama 5. Plataforma nube para sistema de captura y procesamiento

Abarcando más en el modelo SaaS, este puede manejarse por medio de un modelo de cobro de demanda (solicitudes/día), similar a como es trabajado por el motor de Google Directions. Siguiendo este ejemplo, se brindaría como un servicio gratuito de "N" solicitudes por día y luego de sobrepasar la cantidad límite gratuita de solicitudes de procesamiento se empezaría a cobrar por siguientes solicitudes. Este tipo de modelo permite atraer clientes con interés en realizar pruebas simples.

En el caso de la modalidad video de alta definición se aconseja brindar el servicio exclusivamente como instalable. El principal motivo es por el alto flujo de información que las cámaras de video generarían y requerirían pasar al software de captura y procesamiento en la nube, lo cual obliga al cliente tener un ancho de banda de conexión de datos muy alta solo para una cámara.

## B. Sistema de análisis

Este sistema de análisis requiere de una alta alimentación de muestras para el motor de reconocimiento de caracteres, el costo que incurre y forma de comercializar dependerá del tipo de ordenador foco a utilizar y la escalabilidad que requiera.

En el caso de ordenadores portátiles, estos podrían estar limitados por conexión a internet, por ello, será necesario incluir una versión compilada del motor de análisis. Conectándolo a internet, el sistema portátil actualizará el motor de k-NN de una forma similar como lo son los firmwares de celulares.

Para ordenadores simples o avanzados, los cuales seguro contarán con conexión a internet, el sistema de análisis puede operar en una modalidad en línea con un sistema de nube para análisis e identificación del objeto de interés. Para sacar provecho del enlace que existirá con el cliente y el servicio de análisis se aprovecharía sus resultado para mejora continua por retroalimentación el motor de identificación. Por este motivo es recomendable comercializarlo es sistema en una modalidad PaaS, a modo de absorber los resultado de los análisis de los usuarios finales.

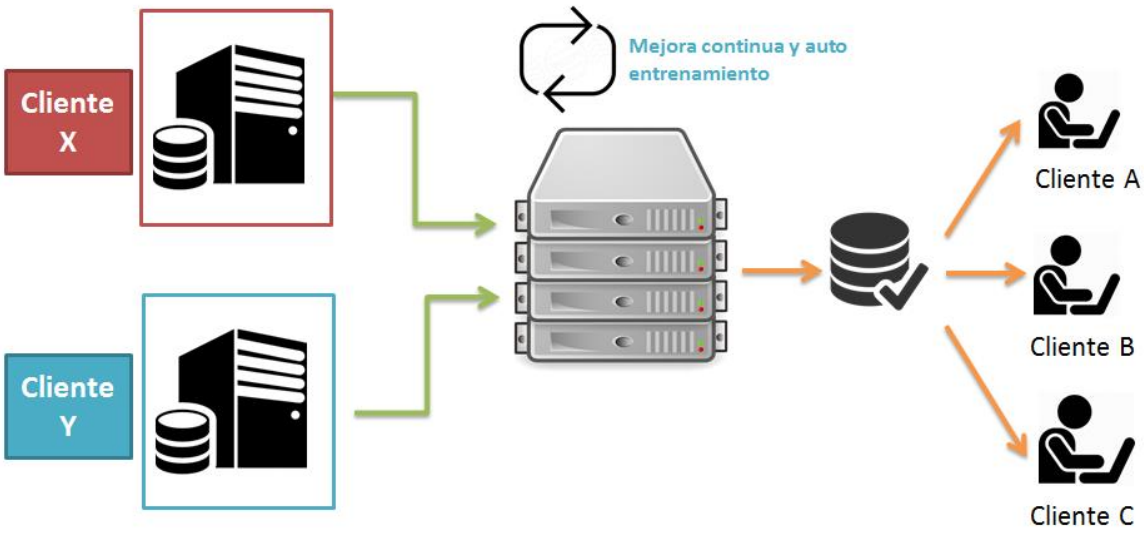



Diagrama 6. Esquema de utilización de plataforma en sistema de identificación

El servicio del sistema de análisis se venderá bajo demanda, entre más demandante sean las tareas del cliente será necesario que asigne mayor cantidad de recurso hardware para procesar sus datos (por ello un servicio tipo PaaS). Si el usuario final cuenta con el sistema de captura como instalable, este se conectará al

sistema de análisis como servicio PaaS. De contar con el API, el usuario mandara sus datos a la nube y se le devolverá el resultado.

Un elemento clave en este esquema de servicio tipo PaaS es aprovechar los resultados de todos los clientes para mejorar continuamente del motor de identificación k-NN. Entre mayor sea la cantidad de usuarios, mayor cantidad de información se generará, acelerando el entrenamiento del motor de identificación con gran diversidad de datos de entrada.

## C. Portal de servicio web

Para un cliente que no cuente con hardware básico o que por experimentación, estudios o incluso ocio se brindaría un portal de servicios en el cual crearía una sesión de usuario y subiría sus imágenes para análisis y brindarles respuestas, almacenamiento y estadísticas. Los costos para este tipo de usuarios serían los mismos por demanda, permitiendo siempre un rango de solicitudes gratuitas.

Todo procesamiento realizados por estos usuario utilizando el portal de servicios pueden ser utilizados para retroalimentación del motor de reconocimiento de caracteres, convirtiendo a estos usuarios también en parte de la comunidad.

# XVIII. CONCLUSIONES

El hardware requerido para este tipo de análisis dependerá de la complejidad y cantidad de entradas que se tenga. El rendimiento del software también está ligado a la velocidad de respuesta que requiera el cliente, donde el de mayor impacto sería el análisis en tiempo real. Brindando el servicio como un modelo tipo PaaS, facilita al cliente la inversión inicial, debido a que no debe preocuparse por compra de equipo, mantenimiento y costos de operación que la maquinaria hardware requiere. Además este tipo de modelo permite la escalabilidad fácil de los sistemas en caso de requerir mayor demanda.

El modularizar el sistema completo permite comercializar de forma separada diferentes servicios. De este modo, clientes pueden optar por gran variedad de configuraciones o personalizaciones enfocándose en su rol de negocio. El modularizar también permite analizar el impacto de cada subsistema y estimar los costos de operación por separado para la venta al cliente.

Conectar todos los sistemas de captura y procesamiento de diferentes clientes a un sistema unificado de análisis permitiría la mejora continua del motor de reconocimiento de caracteres. Esto se convertiría en un extra para los clientes, ya que no habría que actualizar sus sistemas por separado sino que todos funcionan dentro el mismo esquema. La privacidad de la información de cada cliente continuaría ya que lo único que se comparte son los patrones de respuesta para el motor de reconocimiento de caracteres.

El sistema de filtrado de la entrada de imagen o video para identificar una placa vehicular genera demasiados posibles contornos, provocando que el tiempo de procesamiento para generar el resultado de un solo “frame” del video sea muy tardado. Es necesario mejorar la parametrización de la búsqueda de contornos para acelerar este proceso. Se tendría que analizar y comprender que otras técnicas ofrece la librería de OpenCV que permitan reducir la cantidad de contornos generados reduciendo el tiempo de procesamiento.

Emplear técnicas de paralelismo en el modelo de programación será indispensable para solucionar problemas de procesar gran cantidad de entradas de video y poder proveer de un servicio rápido a todos los clientes que se tenga. Considerando el modelo de negocio tipo PaaS el implementar paralelismo por CPU o GPU sería la solución más viable en términos económicos y de programación para solucionar el problema del alto tiempo de ejecución del programa.

El porcentaje de éxito para identificar la placa es muy bajo. Es equivalente a decir que dos de tres carros que pasen por el proceso de detección no logran ser detectados. A pesar de ello, este porcentaje de identificación está basado en pruebas de imagen estática y no en tiempo real por video. En una entrada video, un mismo vehículo puede ser analizado continuamente, lo que permite tener una mayor cantidad de

muestreos y perspectivas para realizar la identificación de la placa.

El porcentaje de aislamiento de los caracteres es bastante alta, de aplicarse otras técnicas o variar los parámetros junto otras funciones de la librería de OpenCV puede mejorarse el porcentaje de éxito para el aislamiento de los siete caracteres de la placa. El porcentaje de éxito de identificación de caracteres es intermedio y debe mejorarse ya que es clave del servicio a proveer.

El porcentaje de identificación fue directamente influenciado por la falta de viñetas para alimentar el sistema de reconocimiento. Dando un ejemplo, en este proyecto se utilizó tres viñetas diferentes por número o letra, en el ejemplo incluido por OpenCV de su documentación, se emplea 250 diferentes viñetas para cada carácter.

# XIX. RECOMENDACIONES

## A. Hardware

Dependiendo del propósito de utilización de la plataforma, puede ser necesario la adquisición de cámaras para intemperie para prolongar su vida útil del equipo, la utilización de webcams puede ser de utilidad en ambientes controlados. Cámaras para intemperie recomendables son tipo IP para exterior de marca BOSCH con costo aproximado de $700 a $1000 por cámara, dichas cámaras además resaltan el brillo de las placas vehiculares permitiendo una mayor definición.

Respecto a la calidad de video, entre mayor sea la resolución de la cámara será mejor la detección del patrón de interés (en este caso la placa vehicular), pero debe considerarse que influirá directamente al tiempo de procesamiento computacional.

La posición de la cámara influirá drásticamente en la detección de la placa. Sin importar el uso de la plataforma, se recomienda realizar pruebas de luminosidad en diferentes horarios y analizar posibles obstrucciones que dificulten la detección y captura del patrón.

En términos de conectividad, al usar cámaras tipo IP recomienda la utilización del respectivo router que las administra en una conexión canal cerrado para satisfacer la demanda de transferencia de datos de video.

En el caso del sistema de captura instalable, el potencial de cómputo necesario para procesar las cámaras será proporcional a la cantidad de las mismas. Considerar la integración de GPU para casos donde la demanda será muy alta.

Para estimar el tipo de arquitectura de hardware que se tendrá que implementar se recomienda empezar con unas cuantas cámaras y un ordenador simple para analizar la capacidad de cómputo. A partir de ello, analizar el tipo de hardware que requerirá la implementación a mayor escala.

## B. Software

Para el software realizado para la captura y procesamiento de la placa vehicular se recomienda analizar otros tipos de filtros que puedan permitir aislar de mejor manera la placa vehicular. Esto permitirá tener una captura más clara y completa de la placa vehicular.

El cambio del modelo de programación lineal a paralelo es necesario para reducir los tiempos de identificación en el subsistema de análisis con el motor de reconocimiento de caracteres, "Machine Learning", utilizado con el algoritmo de k-NN. Además, por medio de paralelismo se puede realizar otras combinaciones de filtrados y ecualización de la imagen original **para generar otras variantes que permitirán la identificación del patrón de interés.**

Alimentar con mayor cantidad de muestras o buscar otros algoritmos para el reconocimiento visual de los caracteres para mejorar el porcentaje de éxito obtenido en la fase de análisis de caracteres o incluso reducir tiempo de procesamiento.

## C. General proyecto

La implementación del sistema con paralelismo requiere tarjeta NVidia, para efectos de prueba simple no se requiere una tarjeta muy avanzada. Durante la fase de investigación se realizaron pruebas muy básicas de paralelismo, operaciones matemáticas, pero se descartó su uso junto la librería de OpenCV debido no solo a su falta de documentación para utilizar con Python, sino también por falta de conocimiento y alta curva de aprendizaje que requiere cambiar el paradigma de programación de lineal a paralelismo, es recomendable que antes de involucrarse en este tipo de programación se realice los cursos y practicas necesarios y básicos para tener un mejor conocimiento de este esquema de programación.

Para mejorar la capacidad de identificación se podría crear toda una red de autoaprendizaje de diferentes usuarios finales que permitan la retroalimentación utilizando todas las cámaras y datos de entrada que envíen al sistema de análisis, a modo de formar un núcleo de análisis conglomerado que permita incrementar la capacidad de identificación del motor de reconocimiento de caracteres progresivamente.

Agregar otro tipo de entradas como sensores de movimiento, sensores de luz, sonido, etc. permitirán al computador conocer otros tipos de variantes que suceden en el exterior. Al final, es asemejar la capacidad de los sentidos humanos a una máquina o incluso mejorarlos.

# XX. BIBLIOGRAFÍA


Alexander Mordvintsev & Abid K, 2013. OpenCV-Python Tutorials. "*Morphological Transformations*". Última actualización Octubre 2014. Recurso en línea en: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html?highlight=erosion

Galen Gruman, CIO. "The Truth About Software as a Service (SaaS)" . Mayo 2007. Recurso en línea en : http://www.cio.com/article/2439004/enterprise-software/the-truth-about-software-as-a-service--saas-.html

Helen Rodriguez, Robert Vera, Boris Wintimilla. *"Detección y Extracción de Placas de Vehículos en Señales de Videos"*. Revista Tecnológica ESPOL. Octubre 2012. Recurso en http://www.rte.espol.edu.ec/index.php/tecnologica/article/viewFile/95/63

Luebke David, Humphreys Greg. *"How GPUs Work"*. 2007. Recurso en http://www.cs.virginia.edu/~gfx/pubs/Luebke_2007_HGW/luebke2007.pdf

Mohamed Cherier. Nawwaf Kharma. "*Character Recognition Systems. A Guide for Students and Practitioners*". 2007. ISBN 978-0-471-41570-1. Recurso en línea en http://people.mokk.bme.hu/~kornai/OCR/Irodalom/Cheriet_Character_Recognition_Systems_A_Guide_for_Students_and_Practitioners.pdf

Neil Anuskiewicz. Python Oficial Site. "Python – Beginners Guide Overview" Julio 2016. Recurso en línea https://wiki.python.org/moin/BeginnersGuide/Overview

News18 AFP Relaxnews. "CES 2017: Audi and Nvidia to make AI-Powered Cars by 2020." Enero 2017. Recurso en línea http://www.news18.com/news/auto/ces-2017-audi-and-nvidia-to-make-ai-powered-cars-by-2020-1332115.html

Nvidia Oficial Site, "CUDA Parrallel Computing Platform. Última actualización Mayo 2017. Recurso en línea: http://www.nvidia.com/object/cuda_home_new.html

OpenCV Oficial Site. "*CUDA".* Última actualización Mayo 2017. Recurso en línea en: http://opencv.org/platforms/cuda.html

OpenCV-Python Tutorials. "*Histograms – 2: Histogram Equalization*". Última actualización Abril 2017. Recurso en línea en: http://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html

OpenCV 2.4.9.0 documentation. "*Image Filtering*". Última actualización Abril 2014. Recurso en línea en: http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html

OpenCV-Python Tutorials. "*Understanding k-Nearest Neighbour*". Última actualización Noviembre 2014. Recurso en línea en: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html

Robert Sedgewick and Kewin Wayne 2009. “Introduction to Programming in Java”. Pag vi. Última actualización Mayo 2017. Recurso en línea en: http://introcs.cs.princeton.edu/java/home/chapter1.pdf

Robohub, Brad Templeton. “CES 2017 pre-news. CES 2017 pre-news: Ford Fusion, Nvidia, MobilEye, HERE, BMW, Intel and other partnerships” Enero 2017. Recurso en linea: http://robohub.org/ces-2017-pre-news-ford-fusion-nvidia-mobileye-here-bmw-intel-and-other-partnerships/

Sumanta Subhadhira, Usarat Juithonglang. “*License Plate Recognition Application using Extreme Learning Machines*”. 2014. ICT-ISPC2014. Recurso en línea [http://www.ispc.ict.mahidol.ac.th/documents/ICT-ISPC2014%20Proceeding/data/ISPC2014_30.pdf]

# XXI. GLOSARIO

Blur: Método de filtrado en imágenes para desenfoque.

CLAHE: Contrast Limited Adaptive Histogram Equalization

CPU: Pieza de hardware conocida por su traducción al inglés como “central processing unit”

C++: Lenguaje de programación compilado y básico

Deep Learning: Método de inteligencia artificial para aprendizaje complejo y profundo

FPS: Frames por segundo. Para referirse a la cantidad de frames renderizados por segundo.

Frame: Ciclo único en imágenes o renderización gráfico.

Framework: Paquete de funciones y software compilado para funcionar como librería.

GPU: Pieza de hardware conocida por su traducción al inglés como “graphics processing unit”.

Hosting: Servicio web que permite la accesibilidad páginas web

IDE: Integrate Development Environment

Java: Lenguaje de programación compilado y orientado a objetos

JPG: Formato de imagen de alta compresión y bajo peso.

k-NN: Técnica de Machine Learning conocida como k-Nearest Neighbors

LibGDX: Framework con funciones graficas utilizado para compresión, renderizado y diseño 3D.

Machine Learning: Método de inteligencia artificial de aprendizaje de maquina

MySQL: Base de datos básica

Nvidia: Marca de manufactura de tarjetas de video y servicios de software.

OCR: Reconocimiento óptico de caracteres (en inglés Optical Character Recognition)

OpenCV: Librería de código abierto para análisis óptico por computadora.

PaaS: Platform as a Service

PNG: Formato de imagen de alta calidad con soporte de transparencia.

Python: Lenguaje de programación interpretado

Query: Instrucción estructurada para ejecutar comando en base de datos.

RAM: Pieza de hardware conocida por su traducción al inglés como “Random access memory”

Red Neuronal: Método de inteligencia artificial para mapear complejos cálculos de decisiones por pesos.

RGB: Termino de computación para referirse a la escala de colores rojo, verde y azul

Router: Equipo hardware para re direccionar conectividad y transmisión de datos entre varios puntos.

SaaS: Software as a Service

Streaming: Termino utilizado para referirse a la transmisión de datos continua de video.

Threshold: Termino para referirse a disparador o punto de arranque en inglés.

Visual Studio: Framework de Windows con gran variedad de lenguajes de programación.

WAMP: Web Services para Windows

Webcam: Cámara compacta con conectividad USB para uso personal en computadoras.

Wrapper: Metodología de programación para funcionar como intermediario entre librerías y lenguaje de programación.