UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN – TACNA

Facultad de Ingeniería

Escuela Profesional de Ingeniería en Informática y Sistemas

OPTIMIZACIÓN DE LA VELOCIDAD DE PROCESAMIENTO DEL ALGORITMO DE LOCALIZACIÓN DE PLACAS DE AUTOS USANDO COMPUTACIÓN PARALELA

TESIS

Presentada por:

Bach. Marcos Elías Catunta Cachi

Para optar el Título Profesional de:

INGENIERO EN INFORMÁTICA Y SISTEMAS

TACNA – PERÚ

2018

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN – TACNA

Facultad de Ingeniería

ESCUELA PROFESIONAL DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS

"OPTIMIZACIÓN DE LA VELOCIDAD DE PROCESAMIENTO DEL ALGORITMO DE LOCALIZACIÓN DE PLACAS DE AUTOS USANDO COMPUTACIÓN PARALELA"

TESIS PRESENTADA A LA COMISIÓN REVISADORA Y APROBADA POR EL JURADO CALIFICADOR, INTEGRADO POR:

Presidente: _____
Dr. Edwin Antonio Hinojosa Ramos

Secretario: _____
Dr. Erbert Francisco Osco Mamani

Vocal: _____
Mgtr. Gianfranco Alexey Málaga Tejada

Asesor: _____
Msc. Edgar Aurelio Taya Acosta

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN – TACNA

Facultad de Ingeniería

JURADO CALIFICADOR Y CALIFICACIÓN DE LA SUSTENTACIÓN DE TESIS

TESIS N°..... **TÍTULO PROFESIONAL DE:**

Ingeniero en Informática y Sistemas

La secretaría académica de la facultad de ingeniería, por resolución de Facultad N°03662-2016-FAIN/UNJBG, designó Jurado para la sustentación oral de la Tesis titulada: "OPTIMIZACIÓN DE LA VELOCIDAD DE PROCESAMIENTO DEL ALGORITMO DE LOCALIZACIÓN DE PLACAS DE AUTOS USANDO COMPUTACIÓN PARALELA".

El mismo que está conformado por:
Presidente: Dr. Edwin Antonio Hinojosa Ramos
Secretario: Dr. Erbert Francisco Osco Mamani
Vocal: Mgtr. Gianfranco Alexey Málaga Tejada

Para calificar la sustentación de la Tesis en acto público el día 28 de Diciembre de 2017. Presentado por el Bachiller Marcos Elías Catunta Cachi, de la Escuela Profesional de Ingeniería en Informática y Sistemas.
El Jurado calificador en forma secreta e individual emitió su opinión sobre el tema de la tesis expuesta y procedió a obtener el promedio que arrojó el calificativo de aprobado con la nota de quince (15).
Para ratificar lo detallado firman:

.....
Dr. Edwin Antonio Hinojosa Ramos
Presidente

.....
Dr. Erbert Francisco Osco Mamani
Secretario

.....
Mgtr. Gianfranco Alexey Málaga Tejada
Vocal

Agradecimiento:

A mis padres que siempre están pendientes de mi bienestar, otorgándome su cariño, apoyo y comprensión en todo momento.

CONTENIDO

ÍNDICE DE FIGURAS v

ÍNDICE DE TABLAS vi

RESUMEN vii

ABSTRACT ix

INTRODUCCIÓN 1

CAPÍTULO I: PLANTEAMIENTO DE LA INVESTIGACIÓN 4

1.1 Descripción del problema 4

1.1.1 Antecedentes del problema 4

1.1.2 Problemática de la investigación 8

1.2 Formulación del problema 12

1.3 Justificación 12

1.4 Alcances y limitaciones 13

1.5 Objetivos 14

1.5.1 Objetivo General 14

1.5.2 Objetivos Específicos 14

1.6 Hipótesis 15

1.6.1 Hipótesis General 15

1.6.2 Hipótesis Específicas 15

1.7 Variables 16

1.7.1 Identificación de variables 16

1.7.2 Definición de las variables 16

1.7.3 Operacionalización de variables 17

1.7.4 Clasificación de las variables 17

1.8 Diseño de la investigación 18

1.8.1 Diseño experimental o no experimental 18

1.8.2 Población y muestra 19

1.8.3 Técnicas e instrumentos para recolección de datos 20

1.8.4 Análisis de los datos 20

1.8.5 Selección de pruebas estadísticas 20

CAPÍTULO II: MARCO TEÓRICO 21

2.1 Marco Referencial 21

2.1.1 Procesamiento de imágenes 21

2.1.2 Ruido en imágenes 25

2.2 Bases teóricas respecto al problema 26

2.2.1 Detector de Borde Canny 26

2.2.2 Máquina de Vectores de Soporte 33

2.2.3 Programación Secuencial 45

2.2.4 Programación Paralela 46

2.2.5 La arquitectura CUDA 47

CAPÍTULO III: MARCO METODOLÓGICO 49

3.1 Análisis y recolección de información 49

3.2 Formulación del algoritmo de localización de placas de autos 50

3.2.1 Secuencial 50

3.2.2 Paralelo 51

3.3 Implementación del algoritmo de localización de placas de autos 51

3.3.1 Secuencial 52

3.3.2 Paralelo 53

CAPÍTULO IV: ANÁLISIS DE RESULTADOS Y DISCUSIONES 56

CONCLUSIONES 66

RECOMENDACIONES 67

REFERENCIAS BIBLIOGRÁFICAS 68

ANEXOS 75

ÍNDICE DE FIGURAS

Figura 1. Máscaras de convolución para obtener el filtro gaussiano. 28

Figura 2. Algoritmo: Obtención de Gradiente 29

Figura 3. Algoritmo: Supresión no máxima. 31

Figura 4. Algoritmo: Histéresis de umbral a la supresión no máxima. 33

Figura 5. Hiperplanos de separación en un espacio bidimensional. 36

Figura 6. La distancia de cualquier ejemplo xi. 39

Figura 7. Imagen Inicial. 56

Figura 8. Resultado de convertir la imagen inicial en escala de grises. 57

Figura 9. Resultado de aplicar el detector de bordes Canny. 57

Figura 10. Resultado de aplicar SVM sin filtrar. 58

Figura 11. Resultado de aplicar SVM filtrado. 58

Figura 12. Imagen final con la placa detectada. 59

ÍNDICE DE TABLAS

Tabla 1. Tiempo de procesamiento de las imágenes en segundos 59-61

RESUMEN

La presente investigación nació con el fin de impulsar el desarrollo de nuevas tecnologías en la región de Tacna, tomando en consideración la gran cantidad de cámaras de seguridad que se han instalado en toda la ciudad, aprovechando así este recurso tan valioso que en la actualidad no se está usando de manera óptima.

Se ha elegido el sector transporte por ser uno de los sectores con tendencia a crear muchos problemas en la sociedad. El poder localizar placas de autos, es un primer paso para desarrollar sistemas complejos de control vehicular, los cuales serán de mucha utilidad para la sociedad.

La investigación se desarrolló en el cercado de Tacna, para así obtener resultados acordes a la realidad de la región, con proyección de poder implementarse en toda la región, adecuando al hardware con que se cuente en las diferentes zonas de la región de Tacna.

Para realizar la investigación, primero se hizo una recolección de información, en la cual se armó una base de datos de imágenes de vehículos. Luego se procedió a editarlas, extrayendo los segmentos de

imágenes que interesaban, que para este caso fueron los segmentos donde se encontraban ubicadas las placas.

Se formuló e implementó el algoritmo de localización de placas de autos de manera secuencial, y con base en ello, se procedió a formular e implementar el algoritmo de localización de placas de autos de forma paralela.

Se hicieron pruebas a los algoritmos, tanto al secuencial como al paralelo, los tiempos de procesamientos obtenidos tuvieron una media de 25,376 y 0,466 segundos respectivamente. Con estos resultados se determinó que el algoritmo de localización de placas de autos en forma paralela es el que tiene un menor tiempo de procesamiento.

Palabras clave: *Optimización de la velocidad, procesamiento del algoritmo, localización de placas*

ABSTRACT

The present investigation was born in order to promote the development of new technologies in the Tacna region, taking into account the large number of security cameras that have been installed throughout the city, taking advantage of this valuable resource that currently does not it is being used optimally.

The transport sector has been chosen because it is one of the sectors with a tendency to create many problems in society. Being able to locate automobile license plates is a first step to develop complex systems of vehicle control, which would be very useful for society.

The research focused on the field of Tacna, to obtain results according to the reality of the region, with the projection of being able to be implemented throughout the region, adapting the hardware that is available in the different areas of the Tacna region.

To carry out the investigation, first a collection of information was made, in which a database of vehicle images was assembled. Then we proceeded to edit them, extracting the segments of interesting images, which for this case were the segments where the plates were located.

We formulated and implemented the algorithm for locating license plates sequentially, and based on this, we proceeded to formulate and implement the algorithm for locating license plates in parallel.

The algorithms were tested, as well as the parallel, the processing times had an average of 25,376 and 0,466 seconds respectively. With these results, it is determined that the algorithm for locating car plates in parallel form is the one with the shortest processing time.

Keywords: *speed optimization, algorithm processing, plate localization*

INTRODUCCIÓN

Hoy en día la tecnología va evolucionando a pasos agigantados, estas mejoras son utilizadas por todo el mundo para solucionar problemas que aquejan a la sociedad en los sectores de: seguridad, militar, espacial, industria, transporte, entre otros; cada nueva solución trae consigo nuevas necesidades, ya que para la mayoría de personas el solucionar el problema no es suficiente y es necesario mejorar dicha solución, optimizando su velocidad, mejorar la calidad o simplemente su estética.

Uno de los avances más significativos que se ha logrado es referente al procesamiento de imágenes, en la actualidad podemos tomar una imagen y obtener una gran cantidad de información de la misma, un ejemplo muy claro y muy utilizado en el mundo es la localización de un determinado objeto en una imagen cualquiera, para lo cual se han desarrollado diferentes algoritmos.

La presente tesis se desarrolla para dar solución a uno de los principales problemas del procesamiento de imágenes, específicamente en la localización de placas de autos, y es referente a su elevado tiempo de procesamiento, debido a que las imágenes procesadas son de alta resolución, para ello se utiliza la computación paralela con la cual se podrá

reducir dicho tiempo de procesamiento y así poder generar aplicaciones las cuales podrán trabajar no sólo con imágenes estáticas, sino que también con videos, obteniendo respuestas en tiempo real.

En el capítulo I se encuentra el objetivo de la investigación el cual es optimizar el algoritmo de localización de placas de autos, definiendo que el alcance de la investigación se enfoca en la ciudad de Tacna y las limitantes que se encontraron, que son básicamente referentes al hardware utilizado.

En el capítulo II se encuentra la teoría en la cual se ha basado la presente investigación. Definiendo conceptos importantes respecto al procesamiento de imágenes, los cuales nos ayudarán a comprender los algoritmos que vamos a utilizar, como ser el Algoritmo de Detección de Bordes Canny el cual se va a usar para pre procesar nuestras imágenes y así pasarlas al siguiente conjunto de algoritmos denominado como Máquina de Vectores de Soporte (*Support Vector Machine* - SVM), el cual nos ayudará a alcanzar nuestro objetivo, el cual es detectar la placa de un auto de una imagen.

En el capítulo III encontramos las imágenes que utilizamos en la investigación y el pre procesamiento que se les aplica. Luego se procede con la formulación del algoritmo de detección de placas de autos y su posterior implementación. Utilizando esto de base se procede a formular e implementar el algoritmo de localización de placas de autos utilizando el

enfoque de programación paralela. Se realizan pruebas para ambos algoritmos y se realiza una comparación entre ellos.

Finalmente se encuentran las conclusiones y recomendaciones obtenidas después de realizar la investigación, las cuales reflejan los resultados obtenidos, así como los problemas encontrados, propuestas para mejorar la investigación e ideas de futuras investigaciones.

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1 Descripción del problema

1.1.1 Antecedentes del problema

Hoy en día existen varios algoritmos para la localización de placas, los cuales se diferencian en el enfoque empleado, su tiempo de procesamiento y su porcentaje de acierto, aunque a pesar de estas diferencias no existe un algoritmo que sea identificado como el mejor de todos ellos. (Hadi y Asadollah, 2011).

Algunos de dichos algoritmos se muestras a continuación:

Localización de Placa Tailandesa usando SVM. (Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C., 2007). Consiste en aplicar el algoritmo de detección de bordes de Canny y luego enviar la imagen resultante al SVM, el cual se encarga de detectar si en la imagen se encuentra alguna placa de auto.

Detección de Placa de Auto Basado en la Técnica Convencional de Detección de Bordes. (Mokayed, H., kim, L., Hock, H., & Hooi N., 2014).

En este artículo se enfoca en la localización de placas, mediante la detección de bordes de Sobel, y luego se aplican procesos morfológicos (dilatación) para expandir las regiones blancas y facilitar el proceso de extracción.

Sistema de Reconocimiento de Placa de Auto Iraní Basado en las Características de Color. (Hossein, A., Jan, M., & Fathy, M., 2014). En este artículo se basa en las características de colores de cada pixel para determinar qué zonas de la imagen corresponde a la placa de auto.

Detección Basada en la Morfología de la Placa de Escenas Complejas (Yu, S., Hsieh, J., & Chen, Y., 2002). Este artículo se utiliza muchas operaciones morfológicas para encontrar el área de algo contraste con importantes características para detectar las placas. Se extraen todos los posibles bordes verticales y se procede a conectarlos. Luego se etiquetan y se ejecuta el proceso de extracción de segmentos de placa.

Detección y Reconocimiento de Placa de Auto Inarí con un Nuevo Método Basado en Morfología. (Hamidreza, S., Mohammadreza, S., & Alireza, S., 2010). La investigación comienza con la conversión de la imagen a escala de grises. Luego se realizan operaciones morfológicas para detectar líneas horizontales y verticales; y así formar el rectángulo de la placa de auto. Aunque esta información no es suficiente para

discriminar a una palca de auto, por ello toma en cuenta otros factores como el tamaño del rectángulo, su inclinación y su radio.

Detección de Múltiples Placas de Autos para Fondos Complejos. (Ching-Tang, H., Yu-Shan, J., & Kuo-Ming, H., 2005). Este método presenta un algoritmo para la extracción de la placa y el proceso consiste en 3 etapas fundamentales: primero, se aplica la transformada de Wavelet para obtener las propiedades de la imagen como el contraste de color fundamental para la siguiente etapa, una vez que se tiene las características previas se forma una línea de referencia, la cual es vital para separar las regiones que no son importantes en el procesamiento. Finalmente, se extrae la región que cumple con las características geométricas de la placa.

Detección de Placa de Auto Basado en el Aprendizaje de Características locales y Globales. (Huaifeng, Z., Wenjing, J., Xiangjian, H., & Qiang, W., 2006). En este método primero se procede con la etapa de entrenamiento para la cual se necesita imágenes donde se encuentren placas y donde no se encuentren. Luego se calcula el gradiente de densidad y se obtiene el umbral que nos ayuda a determinar cuáles imágenes son placas de autos y cuáles no.

Detección de Placa de Auto Usando el Aprendizaje de Árbol en Cascada Basado en Características Híbridas del Objeto. (Qiang, W., Huaifeng, Z., Wenjing, J., Xiangjian, H., Jie, Y., & Tom, H., 2007). Este artículo se basa en dos conjuntos de característica, uno de ellos son los valores de las gradientes para lo cual se utiliza el detector de bordes Sobel y el otro son las características Harr. El cual se calcula en base a la intensidad de cada píxel. El aprendizaje de árbol en cascada utiliza estas características obtenidas para ubicar a la placa de auto.

Método de Extracción de Placas de Auto Híbrido Basado en Estadística de Bordes y Morfología. (Bai, H. & Liu, C., 2004). En este artículo se muestran las dos primeras etapas para la detección de la placa vehicular usando algoritmos para la obtención de los bordes y así determinar rectángulos, los cuales pueden ser posibles candidatos de placas; sin embargo, algunas veces las imágenes pueden estar dañadas o ser de mala calidad, ello hace que se detecten falsos candidatos y en ocasiones no esté incluido la placa original. Luego se hace un filtrado en base a las escalas de umbralización y por último se procede a dilatar la imagen y así discriminar qué candidato cumple las características de una placa de auto.

Detección de Placa de Auto Usando el Descriptor de Covarianza en una Plataforma de Red Neuronal. (Fatih, P & Tekin, K., 2007). En este artículo primero se calcula el descriptor de covarianza basado en la información de cada pixel, como el color, ubicación y gradiente. Entonces entrena una red neuronal para discriminar que región de imagen contiene una placa de auto.

Uno de los aspectos más relevantes a tomar en cuenta en la elección del algoritmo a utilizar es el tiempo de procesamiento, muchos algoritmos han sido descartados por este motivo, debido a que, a pesar de tener un buen porcentaje de acierto, el tiempo que les toma el procesamiento es demasiado elevado.

En Tacna se están instalando cada día más las cámaras de seguridad, pero aún no son muy bien aprovechadas debido a que no contamos con la tecnología necesaria para mejorar el sistema de seguridad de la ciudad, por ello el control de dichas cámaras se sigue realizando de forma manual realizando únicamente la tarea de monitoreo.

1.1.2 Problemática de la investigación

En el procesamiento de imágenes, uno de los principales problemas que afronta es el tiempo de procesamiento, debido a que deben de procesar imágenes de gran resolución y en tiempo real, lo cual implica

la necesidad de utilizar hardware con gran capacidad de procesamiento. Debido a ello los costos para su implementación son muy elevados por ende la idea de implementar esta tecnología de manera masiva se dificulta.

Aprovechando las mejoras que surgen con la computación paralela, el procesamiento de imágenes encuentra una gran opción para resolver sus problemas de velocidad de procesamiento. Aunque tiene que afrontar nuevos problemas, como el hecho de rehacer los algoritmos ya creados tomando el enfoque de computación paralela y reemplazar el hardware utilizado por uno nuevo que cumpla con las nuevas necesidades que han surgido.

Hoy en día en los países desarrollados del mundo cuentan con miles de cámaras de seguridad vigilando sus calles y muchos de ellos ya se encuentran utilizando dichas cámaras para diversos propósitos, entre los cuales están el control del tráfico, seguridad, parqueo de vehículos. Gracias al gran número de cámaras de vigilancia con las que cuentan, pueden obtener una gran cantidad de información de todo su país, lo cual les da la posibilidad de poder utilizar dicha información para diferentes propósitos. Aunque ello también implica que, para manejar un gran tamaño de información, es necesario contar con hardware que

soporte dicha tarea, así como también de algoritmos que puedan trabajar de una manera eficiente, siendo capaces de procesar las imágenes en tiempo real. Como se puede ver el incremento de la cantidad de cámaras de vigilancia origina nuevas necesidades que deben de ser satisfechas para que dicho incremento no sea desperdiciado.

En el Perú la situación del tránsito vehicular es cada vez más alarmante, debido principalmente al incremento de vehículos. Esto con lleva a que la inseguridad ciudadana crezca día con día. Aunque ya se cuenta con muchas cámaras de vigilancia instaladas, esto por sí sólo no basta, ya que es necesario el uso de sistemas automáticos que puedan ayudar en el control del tránsito vehicular y así como también de la seguridad ciudadana. En Lima ya se están empezando a utilizar estos sistemas, aunque en un bajo nivel, esto se refiere al uso de fotopapeletas y su control sobre los excesos de velocidad en la Costa Verde, en los distritos de San Miguel y Magdalena según el Diario Correo (2016). Esto es un inicio de la solución, pero es necesario expandirla y profundizarla aún más, ya que como sabemos la policía no cuenta con suficientes efectivos para llevar a cabo un mejor control del tránsito.

En Tacna la computación paralela aún no se ha desarrollado, lo cual complica la aplicación de esta tecnología al procesamiento de imágenes.

Además del hecho de que es necesario un hardware especial para este propósito, cuyo costo dificulta aún más su desarrollo. Aunque poco a poco se están dando avances con la instalación de cámaras de vigilancia en diferentes partes de la ciudad.

La presente investigación se basa en el algoritmo de localización de placas de autos propuesta por Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2014), el cual tiene un 80% de éxito en la localización de placas en una imagen bajo diferentes circunstancias, como ser la luz, distancia, color de placas y ángulo de la cámara. A pesar de todas estas ventajas del algoritmo, se presenta un gran problema, el cual es el tiempo de procesamiento. La investigación se propone solucionar este problema a través del uso de la programación paralela. Para poder implementar el algoritmo en su totalidad, en la etapa de clasificación será necesario implementar diferentes escenarios, debido a que la placa de un auto puede tener diferentes medidas en una imagen, pero para propósito de nuestra investigación, sólo se trabajará con un escenario, por lo que las imágenes utilizadas para poner a prueba el algoritmo deberán de tener características que correspondan al escenario utilizado.

1.2 Formulación del problema

General:

¿Cuál es la influencia de la computación paralela en la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2014) en la localización de placas de autos?

Específica

a) ¿Cómo es la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2014) en la localización de placas de autos utilizando computación secuencial?
b) ¿Cómo es la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2014) en la localización de placas de autos utilizando computación paralela?

1.3 Justificación

El desarrollo de esta nueva tecnología será el componente fundamental para la creación de nuevos y mejores sistemas de control de tránsito, pudiendo aprovechar de mejor manera las cámaras de seguridad instaladas en la ciudad.

El potencial de esta tecnología no se limita al sector de tránsito, se podría implementar en otros sectores adaptando el algoritmo acorde al objeto de estudio y los resultados buscados.

1.4 Alcances y limitaciones

Alcances:

La investigación se realizará enfocada a los vehículos que ingresen al estacionamiento “COINVIT” ubicado frente al Arco Parabólico de la ciudad de Tacna. El color de la placa, el tipo y la nacionalidad de los vehículos no es relevante para el algoritmo utilizado. El alcance de la investigación dependerá de las imágenes recolectadas para la base de datos y del hardware con que se cuente.

Limitaciones:

Los resultados de la investigación serán influenciados en gran manera por el hardware utilizado (tarjeta de video), el costo del hardware es una gran limitante para la investigación. Además, la escasa información acerca del algoritmo de localización de placas de autos, dificulta su comprensión, implementación y que se pueda adaptar para su posterior uso con programación paralela.

Para superar la limitación del hardware se utilizará una tarjeta de video de un costo moderado y con una capacidad de procesamiento aceptable (tarjeta NVIDIA GeForce GTX 750), que permitirá obtener un buen resultado en la investigación. Respecto a la limitación de la escasa información del algoritmo de localización de placas de autos, si bien no se encuentra algoritmos detallados, se puede encontrar investigaciones donde indican a grandes rasgos como trabaja dicho algoritmo, lo cual se utiliza como base para la investigación.

1.5 Objetivos

1.5.1 Objetivo General

Determinar el nivel de influencia de la computación paralela en la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2014) en la localización de placas de autos.

1.5.2 Objetivos Específicos

a) Determinar la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2014) en la localización de placas de autos utilizando computación secuencial.

b) Determinar la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2014) en la localización de placas de autos utilizando computación paralela.

1.6 Hipótesis

1.6.1 Hipótesis General

Existe una influencia significativa de la computación paralela en la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos.

1.6.2 Hipótesis Específicas

a) Es posible medir la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación secuencial.
b) Es posible medir la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación paralela.

1.7 Variables

1.7.1 Identificación de variables

Variable 1:

Metodología de Programación.

Variable 2:

Velocidad de procesamiento del algoritmo de localización de placas de autos.

1.7.2 Definición de las variables

Metodología de Programación:

Es la metodología utilizada para implementar el algoritmo utilizado para detectar placas de autos en una imagen.

Velocidad de procesamiento del algoritmo de localización de placas de autos:

Es el tiempo de respuesta que tiene el algoritmo desde el inicio de su ejecución hasta la presentación del resultado final.

1.7.3 Operacionalización de variables

Metodología de Programación:

- Tipo de variable : Cualitativa nominal
- Indicador : Placas detectadas (unidades)
- Unidad de medida : u

Velocidad de procesamiento del algoritmo de localización de placas de autos:

- Tipo de variable : Cuantitativa discreta
- Indicador : Segundos
- Unidad de medida : s

1.7.4 Clasificación de las variables

Algoritmo de localización de placas de autos:

- Por su naturaleza: Cualitativa nominal
- Por su dominio: Independiente
- Por su amplitud: Individuales

Velocidad de procesamiento del algoritmo de localización de placas de autos:

- Por su naturaleza: Cuantitativa discreta

- Por su dominio: Dependiente
- Por su amplitud: Individuales

1.8 Diseño de la investigación

1.8.1 Diseño experimental o no experimental

Esta investigación cuenta con un diseño experimental, de tipo experimental pura, porque se manipulará la variable independiente, es decir el algoritmo de localización de placas de autos con un enfoque secuencial es alterado para que pueda trabajar de manera paralela.

“Los experimentos puros son aquellos que reúnen los dos requisitos para lograr el control y la validez interna: grupos de comparación y manipulación de la variable independiente o de varias independientes y equivalencia de los grupos. Estos diseños llegan a incluir una o más variables independientes y una o más dependientes. Asimismo, pueden utilizar pre pruebas y post pruebas para analizar la evolución de los grupos antes y después del tratamiento experimental” (Hernández, Fernández & Baptista, 2006).

1.8.2 Población y muestra

La población: Se tomará en cuenta la imagen de todos los vehículos que ingresen en el estacionamiento “COINVIT” sin discriminar el tipo de vehículo, nacionalidad o color de placa.

La muestra: Como se explicó en la problemática de la investigación, el algoritmo trabaja con varios escenarios en la etapa de clasificación, y para propósito de la investigación, sólo utilizaremos un escenario. Por ello la muestra a utilizar debe de cumplir las condiciones del escenario elegido, la cual será que en la imagen se encuentre la placa de auto con un tamaño de 90x50 píxeles. Para determinar el número de la muestra se toma como referencia la información de entrada de vehículos al estacionamiento en Abril y Mayo del 2017 encontrados en el anexo 5, aplicando la prueba de Normalidad de Anderson Darling se encontraron los siguientes resultados:

Media	:	141,3
Desviación Estándar	:	15,75
N	:	61
AD	:	0,483
Valor p	:	0,222

Con un nivel de significancia de 0,05, se tiene que el valor de p es mayor, por lo que se demuestra que los datos obtenidos tienen una

distribución normal, por lo, tanto la cantidad de la muestra es la media obtenida que es 141. Debido a que no hay diferencia significativa para el algoritmo por los diferentes tipos de vehículos que entran al estacionamiento y por la dificultad de recolectar las imágenes se reduce la muestra a 70.

1.8.3 Técnicas e instrumentos para recolección de datos

Para la recolección de las imágenes se utiliza una cámara Samsung PL120 14.4 Mega pixeles. Para la medición del tiempo de ejecución del algoritmo tanto secuencial como paralelo se utilizará la clase Clock de Visual Studio 2012 Express. Se utiliza la técnica de observación para determinar si en una imagen se encuentra alguna placa de auto.

1.8.4 Análisis de los datos

Se hará uso de las medidas de centralización para analizar los tiempos de respuesta de los algoritmos secuencial y paralelo que se obtengan cuando se utilice la muestra para probar ambos algoritmos, específicamente la media y la varianza.

1.8.5 Selección de pruebas estadísticas

La prueba estadística a utilizar será t-student para diferencia de medias, en el cual se compararán los datos obtenidos por el algoritmo

de localización de placas de autos secuencial y el implementado con computación paralela.

“La prueba t se utiliza para comparar los resultados de una pre-prueba con los resultados de una post prueba en un contexto experimental” (Hernández, Fernández & Baptista, 2006).

CAPÍTULO II

MARCO TEÓRICO

2.1 Marco Referencial

2.1.1 Procesamiento de imágenes

A diferencia del estudio de los mecanismos de la visión humana, el procesamiento de imágenes digitales nace en el momento en que se dispone de recursos tecnológicos para captar y manipular grandes cantidades de información espacial en forma de matrices de valores. Esta distinción sitúa al procesamiento de imágenes digitales como una tecnología asociada a las ciencias de la computación y, por tanto, cabe pensar en ella como una proyección del término visión artificial, dentro del ámbito de la inteligencia artificial (Esqueda & Palafox, 2005).

Algunos conceptos básicos relacionados con el procesamiento de imágenes se definen a continuación:

a. Visión por computadora:

Consiste en la adquisición, procesamiento, clasificación y reconocimiento de imágenes digitales.

b. Píxel:

Elemento básico de una imagen.

c. Imagen:

Arreglo bidimensional de píxeles con diferente intensidad luminosa. Si la intensidad luminosa de cada píxel se representa por n bits, entonces existirán 2^n escalas de gris diferentes. Matemáticamente una imagen se representa por r = f(x,y), donde r es la intensidad luminosa del píxel, cuyas coordenadas son (x,y). También matemáticamente, un sistema para procesar imágenes se representa como g(x,y) = T[f(x,y)].

d. Color

El color se forma mediante la combinación de los tres colores básicos: rojo, verde y azul (en inglés corresponde a las siglas RGB) y puede expresarse mediante una tripleta de valores de 0 a 1 (R, G, B) donde R, G y B representan las intensidades de cada uno de los tres colores básicos.

e. Brillo

Indica si un área está más o menos iluminada.

f. Tono

Indica si un área parece similar al rojo, amarillo, verde o azul, o a una proporción de ellos.

g. Luminosidad

Es el brillo de una zona respecto a otra zona blanca en la imagen.

h. Croma

Indica la colaboración de un área respecto al brillo de un blanco de referencia. Para obtener una imagen a color debe transformarse primero los parámetros cromáticos en eléctricos y representar los colores, lo cual puede realizarse de diferentes maneras, dando lugar a diferentes espacios de colores o mapas de color.

i. Espacio RGB

Se basa en la combinación de tres señales de luminancia cromática distinta: rojo, verde, azul (*red, green, blue*). La forma más sencilla de obtener un color específico es determinando la cantidad de color rojo, verde y azul que se requiere combinar para obtener el color deseado, para lo cual se realiza una suma aritmética de los

componentes: X = R + G + B, gráficamente representada por un cubo.

En la recta que une el origen con el valor máximo se encuentran ubicados los grises (escala de gris) debido a que sus tres componentes son iguales. Cuando una cámara adquiere una imagen a color, para cada píxel en color se tienen en realidad tres componentes, una para cada uno de los colores básicos (rojo, verde y azul); la ganancia máxima para cada componente corresponde a la longitud de onda de los tres colores básicos.

j. Mapa de color

El mapa de color es una matriz de n x 3, donde cada renglón es una tripleta de colores. El primer renglón corresponde al valor mínimo del eje de color y el último renglón al máximo. Al definir diferentes distribuciones de intensidad de los tres colores básicos, se crean diferentes mapas de color.

k. Histograma de una imagen

El histograma de una imagen es una representación del número de píxeles de cierto nivel de gris en función de los niveles de grises.

2.1.2 Ruido en imágenes

"Todas las imágenes tienen cierta cantidad de ruido, la cual se puede deber a la cámara o al medio de transmisión de la señal. Generalmente el ruido se manifiesta como píxeles aislados que toman un nivel de gris diferente al de sus vecinos. El ruido puede clasificarse en los siguientes tipos" (Esqueda & Palafox, 2005):

a. Gaussiano

Produce pequeñas variaciones en la imagen; generalmente se debe a diferentes ganancias en la cámara, ruido en los digitalizadores, perturbaciones en la transmisión, etcétera. Se considera que el valor final del píxel sería el ideal más una cantidad correspondiente al error que puede describirse como una variable aleatoria gaussiana.

b. Impulsional

El valor que toma el píxel no tiene relación con el valor inicial, sino con el valor del ruido que toma valores muy altos o bajos (puntos altos y/o negros) causados por una saturación del sensor o por un valor mínimo captado, si se ha perdido la señal en ese punto. Se encuentran también al trabajar con objetos a altas

temperaturas, ya que las cámaras tienen una ganancia en el infrarrojo que no es detectable por el ojo humano; por ello las partes más calientes de un objeto pueden llegar a saturar en un píxel.

c. Multiplicativo

La imagen obtenida es el resultado de la multiplicación de dos señales.

2.2 Bases teóricas respecto al problema

2.2.1 Detector de Borde Canny

"En el área de procesamiento de imágenes, la detección de los bordes de una imagen es de suma importancia y utilidad, pues facilita muchas tareas, entre ellas, el reconocimiento de objetos, la segmentación de regiones, entre otras" (Valverde, 2007).

Se han desarrollado variedad de algoritmos que ayudan a solucionar este inconveniente. El algoritmo de Canny es usado para detectar todos los bordes existentes en una imagen. Este algoritmo está considerado como uno de los mejores métodos de detección de contornos mediante el empleo de máscaras de convolución y basado en la primera derivada. Los puntos de contorno son como zonas de píxels en las que existe un cambio brusco de nivel de gris. En el tratamiento de imágenes, se trabaja

con píxels, y en un ambiente discreto, es así que en el algoritmo de Canny se utiliza máscaras, las cuales representan aproximaciones en diferencias finitas.

El algoritmo de Canny consiste en tres grandes pasos:

a) Obtención del gradiente

> “Para la obtención del gradiente, lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original con el objetivo de suavizar la imagen y tratar de eliminar el posible ruido existente. Sin embargo, se debe de tener cuidado de no realizar un suavizado excesivo, pues se podrían perder detalles de la imagen y provocar un pésimo resultado final. Este suavizado se obtiene promediando los valores de intensidad de los píxels en el entorno de vecindad con una máscara de convolución de media cero y desviación estándar σ. En la figura 1 se muestran dos ejemplos de máscaras que se pueden usar para realizar el filtrado gaussiano” (Valverde, 2007).

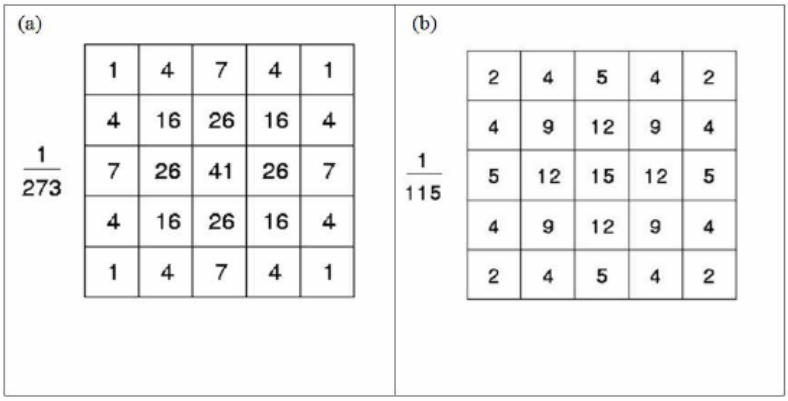


Figura 1. Máscaras de convolución para obtener el filtro gaussiano.

Fuente: Detección de bordes mediante el algoritmo de Canny. Valverde, J. (2007)

Una vez que se suaviza la imagen, para cada píxel se obtiene la magnitud y módulo (orientación) del gradiente, obteniendo así dos imágenes. El algoritmo para este primer paso se describe a continuación.

Algoritmo: Obtención de Gradiente

Entrada: imagen I
máscara de convolución H, con media cero y desviación estándar σ.
Salida: imagen E_m de la magnitud del gradiente
imagen E_o de la orientación del gradiente

1. Suavizar la imagen I con H mediante un filtro gaussiano y obtener J como imagen de salida.
2. Para cada píxel (i, j) en J, obtener la magnitud y orientación del gradiente basándose en las siguientes expresiones:
   El gradiente de una imagen $f(x,y)$ en un punto (x,y) se define como un vector bidimensional dado por la ecuación:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$$

   siendo un vector perpendicular al borde, donde el vector G apunta en la dirección de variación máxima de f en el punto (x,y) por unidad de distancia, con la magnitud y dirección dadas por:

$$|G| = \sqrt{G_x^2 + G_y^2} = |G_x| + |G_y| ,$$

$$\phi(x,y) = \tan^{-1} \frac{G_y}{G_x}$$

3. Obtener E_m a partir de la magnitud de gradiente y E_o a partir de la orientación, de acuerdo a las expresiones anteriores.

Figura 2. Algoritmo: Obtención de Gradiente

Fuente: Detección de bordes mediante el algoritmo de Canny. Valverde, J. (2007).

b) Supresión no máxima al resultado del gradiente

"Las dos imágenes generadas en el paso anterior sirven de entrada para generar una imagen con los bordes adelgazados. El procedimiento es el siguiente: se consideran cuatro direcciones identificadas por las orientaciones de 0º, 45º, 90º y 135º con respecto al eje horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente" (Valverde, 2007).

Posteriormente se observa si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior. De ser así se asigna el valor 0 a dicho píxel, en caso contrario se asigna el valor que tenga la magnitud del gradiente.

La salida de este segundo paso es la imagen In con los bordes adelgazados, es decir, $E_m(i,j)$, después de la supresión no máxima de puntos de borde.

c) Histéresis de umbral a la supresión no máxima

"La imagen obtenida en el paso anterior suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral. El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo" (Valverde, 2007).

Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto seguir las cadenas de máximos locales

conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayores al primer umbral. Así se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. Es así como en este paso se logra eliminar las uniones en forma de Y de los segmentos que confluyan en un punto.

Algoritmo: Supresión no máxima

Entrada: imagen E_m de la magnitud del gradiente
imagen E_o de la orientación del gradiente
Salida: imagen I_n
Considerar: cuatro direcciones d_1, d_2, d_3, d_4 identificadas por las direcciones de 0°, 45°, 90° y 135° con respecto al eje horizontal

1. Para cada píxel *(i, j)* :
   1.1. Encontrar la dirección d_k que mejor se aproxima a la dirección $E_o(i, j)$, que viene a ser la perpendicular al borde.
   1.2. Si $E_m(i, j)$ es más pequeño que al menos uno de sus dos vecinos en la dirección d_k, al píxel *(i, j)* de I_n se le asigna el valor 0, $I_n(i, j) = 0$ (supresión), de otro modo $I_n(i, j) = E_m(i, j)$.
2. Devolver I_n

Figura 3. Algoritmo: Supresión no máxima.

Fuente: Detección de bordes mediante el algoritmo de Canny. Valverde, J. (2007).

d) Un cuarto paso

"Frecuentemente, es común que un cuarto y último paso se realice en el algoritmo de Canny, este paso consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de

ruido. Un método muy utilizado es el algoritmo de Deriche y Cocquerez. Este algoritmo utiliza como entrada una imagen binarizada de contornos de un píxel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con otro extremo abierto" (Valverde, 2007).

El procedimiento consiste en buscar para cada píxel uno de los ocho patrones posibles que delimitan la continuación del contorno en tres direcciones posibles. Esto se logra con la convolución de cada píxel con una máscara específica. Cuando alguno de los tres puntos es ya un píxel de borde se entiende que el borde se ha cerrado, de lo contrario se elige el píxel con el valor máximo de gradiente y se marca como nuevo píxel de borde y se aplica nuevamente la convolución. Estos pasos se repiten para todo extremo abierto hasta encontrar su cierre o hasta llegar a cierto número de iteraciones determinado.

Algoritmo: Histéresis de umbral a la supresión no máxima

Entrada: imagen I_n obtenida del paso anterior
imagen E_o de la orientación del gradiente
umbral t_1
umbral t_2, donde $t_1 < t_2$
Salida: imagen G con los bordes conectados de contornos

1. Para todos los puntos de I_n y explorando I_n en orden fijo:
   1.1. Localizar el siguiente punto de borde no explorado previamente, $I_n(i,j)$, tal que $I_n(i,j) > t_2$
   1.2. Comenzar a partir de $I_n(i,j)$, seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal de borde, siempre que $I_n > t_1$.
   1.3. Marcar todos los puntos explorados y, salvar la lista de todos los puntos en el entorno conectado encontrado.
2. Devolver G formada por el conjunto de bordes conectados de contornos de la imagen, así como la magnitud y orientación, describiendo las propiedades de los puntos de borde.

Figura 4. Algoritmo: Histéresis de umbral a la supresión no máxima.

Fuente: Detección de bordes mediante el algoritmo de Canny. Valverde, J. (2007).

2.2.2 Máquina de Vectores de Soporte

"Las máquinas de vectores soporte (SVM, del inglés *Support Vector Machines*) tienen su origen en los trabajos sobre la teoría del aprendizaje estadístico y fueron introducidas en los años 90 por Vapnik y sus colaboradores (Boser et al., 1992, Cortes & Vapnik, 1995). Aunque originariamente las SVMs fueron pensadas para resolver problemas de clasificación binaria, actualmente se utilizan para resolver otros tipos de problemas (regresión, agrupamiento, multiclasificación). También son diversos los campos en los que han sido utilizadas con éxito, tales como

visión artificial, reconocimiento de caracteres, categorización de texto e hipertexto, clasificación de proteínas, procesamiento de lenguaje natural, análisis de series temporales. De hecho, desde su introducción, han ido ganando un merecido reconocimiento gracias a sus sólidos fundamentos teóricos" (Carmona, E., 2014).

Dentro de la tarea de clasificación, las SVM pertenecen a la categoría de los clasificadores lineales, puesto que inducen separadores lineales o hiperplanos, ya sea en el espacio original de los ejemplos de entrada, si éstos son separables o cuasi-separables (ruido), o en un espacio transformado (espacio de características), si los ejemplos no son separables linealmente en el espacio original. Como se verá más adelante, la búsqueda del hiperplano de separación en estos espacios transformados, normalmente de muy alta dimensión, se hará de forma implícita utilizando las denominadas funciones kernel.

Mientras la mayoría de los métodos de aprendizaje se centran en minimizar los errores cometidos por el modelo generado a partir de los ejemplos de entrenamiento (error empírico), el sesgo inductivo asociado a las SVM radica en la minimización del denominado riesgo estructural. La idea es seleccionar un hiperplano de separación que equidista de los ejemplos más cercanos de cada clase para, de esta forma, conseguir lo

que se denomina un margen máximo a cada lado del hiperplano. Además, a la hora de definir el hiperplano, sólo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de vectores soporte. Desde un punto de vista práctico, el hiperplano separador de margen máximo ha demostrado tener una buena capacidad de generalización, evitando en gran medida el problema del sobreajuste a los ejemplos de entrenamiento.

Dado un conjunto separable de ejemplos S = f{(x_1:y_1);…;(x_n:y_n)}, donde x_i ϵ R_d e y_i ϵ f{+1;-1} se puede definir un hiperplano de separación como una función lineal que es capaz de separar dicho conjunto sin error:

$$D(x) = (w_1 x_1 + \cdots + w_d x_d) + b =< w, x > +b \quad [1]$$

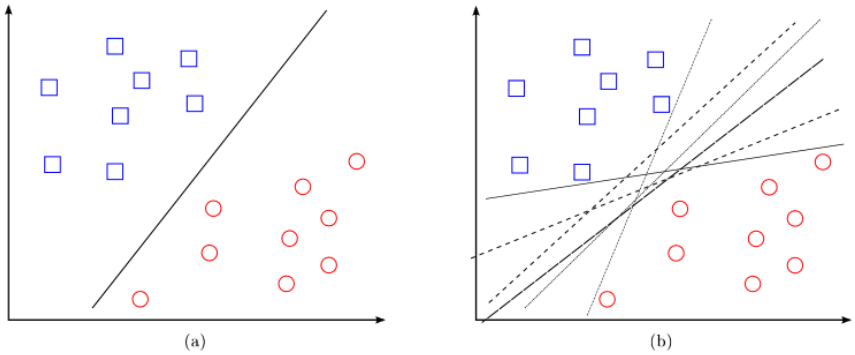


Figura 5. Hiperplanos de separación en un espacio bidimensional.

Fuente: Tutorial sobre Máquinas de Vectores Soporte (SVM). Carmona, E. (2014).

Donde w y b son coeficientes reales. El hiperplano de separación cumplirá las siguientes restricciones para toda x_i del conjunto de ejemplos:

$$< w, x_i > + b \geq 0 \quad si \quad y_i = +1$$

$$< w, x_i > + b \geq 0 \quad si \quad y_i = -1, i = 1, \dots, n \qquad [2]$$

O también:

$$y_i(< w, x_i > + b) \geq 0, \quad i = 1, \dots, n \qquad [3]$$

O de forma más compacta:

$$y_i D(x_i) \geq 0, \quad i = 1, \dots, n \qquad [4]$$

Tal y como se puede deducir fácilmente de la figura 5.b, el hiperplano que permite separar los ejemplos no es único, es decir, existen infinitos hiperplanos separables, representados por todos aquellos hiperplanos que son capaces de cumplir las restricciones impuestas por cualquiera de las expresiones equivalentes [3-4]. Surge entonces la pregunta sobre si es posible establecer algún criterio adicional que permita definir un hiperplano de separación óptimo. Para ello, primero, se define el concepto de margen de un hiperplano de separación, denotado por T, como la mínima distancia entre dicho hiperplano y el ejemplo más cercano de cualquiera de las dos clases (figura 5.a). A partir de esta definición, un hiperplano de separación se denominará óptimo si su margen es de tamaño máximo (figura 5.b).

Una propiedad inmediata de la definición de hiperplano de separación óptimo es que éste equidista del ejemplo más cercano de cada clase. La demostración de esta propiedad se puede hacer fácilmente por reducción al absurdo. Supongamos que la distancia del hiperplano óptimo al ejemplo más cercano de la clase +1 fuese menor que la correspondiente al ejemplo más cercano de la clase -1. Esto significaría que se puede alejar el hiperplano del ejemplo de la clase +1 una distancia tal que la distancia del hiperplano a dicho ejemplo sea mayor que antes y, a su vez, siga siendo menor que la distancia al ejemplo más

cercano de la clase -1. Se llega así al absurdo de p o der aumentar el tamaño del margen cuando, de partida, habíamos supuesto que éste era máximo (hiperplano óptimo). Se aplica un razonamiento similar en el caso de suponer que la distancia del hiperplano óptimo al ejemplo más cercano de la clase -1 fuese menor que la correspondiente al ejemplo más cercano de la clase +1.

Por geometría, se sabe que la distancia entre un hiperplano de separación D(x) y un ejemplo x´ viene dada por

$$\frac{|D(x')|}{\|w\|} \qquad [5]$$

Siendo |.| el operador valor absoluto, ||.|| el operador norma de un vector y w el vector que, junto con el parámetro b, define el hiperplano D(x) y que, además, tiene la propiedad de ser perpendicular al hiperplano considerado. Haciendo uso de las expresiones [4] y [5], todos los ejemplos de entrenamiento cumplirán que:

$$\frac{y_i D(x_i)}{\|w\|} \geq \tau, \quad i = 1, \dots, n \qquad [6]$$

De la expresión anterior, se deduce que encontrar el hiperplano óptimo es equivalente a encontrar el valor de w que maximiza el margen. Sin embargo, existen infinitas soluciones que dieren solo en la escala de w. Así, por ejemplo, todas las funciones lineales λ (<w;x>+b), con λ ϵ R, representan el mismo hiperplano. Para limitar, por tanto, el número de

soluciones a una sola, y teniendo en cuenta que [6] se puede expresar también como

$$y_i D(x_i) \geq \tau \|w\|, \quad i = 1, \ldots, n \qquad [7]$$

La escala del producto de T y la norma de w se fija, de forma arbitraria, a la unidad, es decir

$$\tau \|w\| = 1 \qquad [8]$$

Llegando a la conclusión final de que aumentar el margen es equivalente a disminuir la norma de w, ya que la expresión anterior se puede expresar como

$$\tau = \frac{1}{\|w\|} \qquad [9]$$

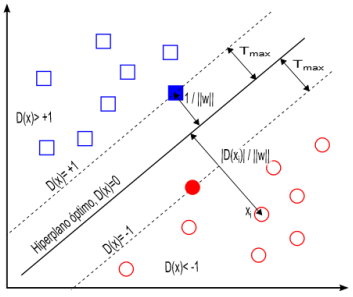


Figura 6. La distancia de cualquier ejemplo xi.

Fuente: Tutorial sobre Máquinas de Vectores Soporte (SVM). Carmona, E. (2014)

Por tanto, de acuerdo a su definición, un hiperplano de separación óptimo como en la imagen anterior será aquel que posee un margen

máximo y, por tanto, un valor mínimo de ||w|| y, además, está sujeto a la restricción dada por [7], junto con el criterio expresado por [8], es decir,

$$y_i D(x_i) \geq 1, \;\; i = 1, \ldots, n \qquad [10]$$

O lo que es lo mismo

$$y_i(< w, x_i > + b) \geq 1, \;\; i = 1, \ldots, n \qquad [11]$$

El concepto de margen máximo está relacionado directamente con la capacidad de generalización del hiperplano de separación, de tal forma que, a mayor margen, mayor distancia de separación existirá entre las dos clases. Los ejemplos que están situados a ambos lados del hiperplano óptimo y que definen el margen o, lo que es lo mismo, aquellos para los que la restricción [11] es una igualdad, reciben el nombre de vectores soporte. Puesto que estos ejemplos son los más cercanos al hiperplano de separación, serán los más difíciles de clasificar y, por tanto, deberían ser los únicos ejemplos a considerar a la hora de construir dicho hiperplano. De hecho, se demostrará más adelante, en esta misma sección, que el hiperplano de separación óptimo se define sólo a partir de estos vectores.

La búsqueda del hiperplano óptimo para el caso separable puede ser formalizado como el problema de encontrar el valor de w y b que minimiza el funcional f(w) = ||w|| sujeto a las restricciones [11], o de forma equivalente

$$mín\ f(w) = \frac{1}{2}||w||^2 = \frac{1}{2} < w, w >$$

$$s.a. \quad y_i(< w, x_i > +b) - 1 \geq 0, \ \ i = 1, \dots, n \qquad [12]$$

Este problema de optimización con restricciones corresponde a un problema de programación cuadrático y es abordable mediante la teoría de la optimización. Dicha teoría establece que un problema de optimización, denominado primal, tiene una forma dual si la función a optimizar y las restricciones son funciones estrictamente convexas. En estas circunstancias, resolver el problema dual permite obtener la solución del problema primal.

Así, puede demostrarse que el problema de optimización dado por [12] satisface el criterio de convexidad y, por tanto, tiene un dual. En estas condiciones, y aplicando los resultados descritos en el anexo, al final de este tutorial, se pueden enumerar los siguientes pasos encaminados a resolver el problema primal. En el primer paso, se construye un problema de optimización sin restricciones utilizando la función Lagrangiana.

$$L(w, b, \alpha) = \frac{1}{2} < w, w > - \sum_{i=1}^{n} \alpha_i [y_i(< w, x_i > +b) - 1] \qquad [13]$$

Donde los αi≥0 son los denomiandos multiplicadores de Lagrange.

El segundo paso consiste en aplicar las condiciones de Karush-Kuhn-Tucker, también conocidas como condiciones KKT.

$$\frac{\partial L(w^*,b^*,\alpha)}{\partial w} \equiv w^* - \sum_{i=1}^{n} \alpha_i y_i x_i = 0, \;\; i = 1, \dots, n \qquad [14]$$

$$\frac{\partial L(w^*,b^*,\alpha)}{\partial w} \equiv \sum_{i=1}^{n} \alpha_i y_i = 0, \;\; i = 1, \dots, n \qquad [15]$$

$$\alpha_i [1 - y_i (< w^*, x_i > + b^*)] = 0, \;\; i = 1, \dots, n \qquad [16]$$

Las restricciones representadas por [14-15] corresponden al resultado de aplicar la primera condición KKT, y las expresadas en [16], al resultado de aplicar la denominada condición complementaria (segunda condición KKT). Las primeras permiten expresar los parámetros de w y b en términos de α_i:

$$w^* = \sum_{i=1}^{n} \alpha_i y_i x_i, \;\; i = 1, \dots, n \qquad [17]$$

Además, establecen restricciones adicionales para los coeficientes α_i:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \;\; i = 1, \dots, n \qquad [18]$$

Con las nuevas relaciones obtenidas, se construirá el problema dual. Así, bastara usar [17] para expresar la función Lagrangiana sólo en función de αi. Antes de ello, se puede reescribir [13] como

$$L(w,b,\alpha) = \frac{1}{2} < w, w > - \sum_{i=1}^{n} \alpha_i y_i < w, x_i > - b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

Teniendo en cuenta que, según la condición [18], el tercer término de la expresión anterior es nulo, la substitución de [17] en dicha expresión resulta ser

$$L(\alpha) = \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)\left(\sum_{j=1}^{n} \alpha_j y_j x_j\right) - \left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)\left(\sum_{j=1}^{n} \alpha_j y_j x_j\right) + \sum_{i=1}^{n} \alpha_i$$

$$L(\alpha) = -\frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)\left(\sum_{j=1}^{n} \alpha_j y_j x_j\right) + \sum_{i=1}^{n} \alpha_i$$

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j < x_i, x_j > \qquad [19]$$

Es decir, hemos transformado el problema de minimización primal [12], en el problema dual, consistente en maximizar [19] sujeto a las restricciones [18], junto a las aso ciadas originalmente a los multiplicadores de Lagrange

$$máx\ L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < x_i, x_j > \qquad [20]$$

$$s.a. \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \alpha_i \geq 0, i = 1, \dots, n$$

Al igual que el problema primal, este problema es abordable mediante técnicas estándar de programación cuadrática. Sin embargo, como se puede comprobar, el tamaño del problema de optimización dual escala con el número de muestras n, mientras que el problema primal lo hace con la dimensionalidad, d. Por tanto, aquí radica la ventaja del problema dual, es

decir, el coste computacional aso ciado a su resolución es factible incluso para problemas con un número muy alto de dimensiones.

La solución del problema dual α*, nos permitirá obtener la solución del problema primal. Para ello, bastará sustituir dicha solución en la expresión [17] y, finalmente, sustituir el resultado así obtenido en [1], es decir:

$$D(x) = \sum_{i=1}^{n} \alpha_i^* y_i < x, x_i > + b^* \qquad [21]$$

Volviendo a las restricciones [16], resultantes de aplicar la segunda condición KKT, se puede afirmar que si αi>0 entonces el segundo factor de la parte izquierda de dicha expresión tendrá que ser cero y, por tanto

$$Y_i(< w^*, x_i > + b^*) = 1 \qquad [22]$$

Es decir, el correspondiente ejemplo, (x_i;y_i), satisface la correspondiente restricción del problema primal [12], pero considerando el caso “igual que”. Por definición, los ejemplos que satisfacen las restricciones expresadas en [12], considerando el caso “igual que”, son los vectores soporte y, por consiguiente, se puede afirmar que sólo los ejemplos que tengan aso ciado un αi >0 serán vectores soporte. De este resultado, también puede afirmarse que el hiperplano de separación [21] se construirá como una combinación lineal de sólo los vectores soporte del conjunto de ejemplos, ya que el resto de ejemplos tendrán asociado un αj =0.

Para que la definición del hiperplano [21] sea completa, es preciso determinar el valor del parámetro b*. Su valor se calcula despejando b de [22]:

$$b^* = Y_{vs} - < w^*, x_{vs} > \qquad [23]$$

Donde (x_{vs};y_{vs}) representa la tupla de cualquier vector soporte, junto con su valor de clase, es decir, la tupla de cualquier ejemplo que tenga asociado un αi distinto de cero. En la práctica, es más robusto obtener el valor de b* promediando a partir de todos los vectores soporte N_{vs}. Así, la expresión [23] se transforma en:

$$b^* = \frac{1}{N_{vs}} \sum_{i=1}^{N_{vs}} (Y_{vs} - < w^*, x_{vs} >) \qquad [24]$$

Finalmente, haciendo uso de [17] en [23], o en [24], permitirá también calcular el valor de b* en función de la solución del problema dual.

Obsérvese que tanto la optimización de [20] como la evaluación de [21] dependen del producto escalar de los vectores ejemplos.

2.2.3 Programación Secuencial

"La programación secuencial permite representar aquellas instrucciones que se ejecutan una tras otra, en secuencia; o sea, instrucciones en que la salida de una es la entrada de la próxima

instrucción. También se puede decir que son una o más instrucciones seguidas, donde ninguna de ellas hace que se bifurque el control de ejecución del algoritmo, o que se repita la ejecución de una o más instrucciones. Mediante esta estructura se pueden representar instrucciones de asignación, entrada y salida de datos e invocación de subprogramas. Para diferenciar una de otra, se añaden características intrínsecas de cada instrucción" (Oviedo, E., 2004).

2.2.4 Programación Paralela

"En el procesamiento en paralelo, se puede ejecutar múltiples hilos (threads), en la actualidad del orden de entre 8 y 12. Sin embargo, lo que realmente están haciendo, es fragmentar las aplicaciones secuenciales en pequeños fragmentos de código y asignando su ejecución a los diferentes cores del procesador. Por ejemplo, si en una aplicación que se ejecuta secuencialmente sobre un procesador con múltiples cores, y el desarrollador decidió que algún fragmento se distribuya entre diferentes cores, la aplicación se ejecutará por partes secuenciales en cores diferentes. Hasta que alguna parte termine su trabajo, la siguiente no iniciará. Esto significa que el paralelismo que consigue, en general, un procesador multicore, representa un paralelismo que se ejecuta sobre múltiples aplicaciones en paralelo, pero que normalmente, no ejecuta

múltiples secciones de la aplicación en paralelo" (Rivera & Vargas, 2012).

Para conseguir mejorar este objetivo, las Unidades Gráficas de Procesamiento (o sus siglas en inglés GPU) integran lógicas de control más simples y que pretenden ejecutar el máximo de *threads* en paralelo. Integran memoria cache de menor tamaño, compartidas por diferentes unidades de proceso para evitar tener que acceder a la memoria principal, pero siguen manteniendo una mayor superficie dedicada a las unidades de procesado. Así mismo, integran buses de comunicación de mayor ancho de banda que las Unidades Centrales de Procesamiento (con sus siglas en inglés CPU) del orden de 10 veces más rápidos que las CPU.

2.2.5 La arquitectura CUDA

"A diferencia de las previas generaciones que segmentaban los recursos computacionales entre los sombreados (*shader*), en la computación gráfica, es un programa de computadora utilizado primordialmente para calcular efectos en el despliegue de imágenes en hardware de gráficos con cierto grado de flexibilidad de vértice y de pixeles. La arquitectura CUDA incluyó la tubería (hilo de ejecución) unificada de sombreadores, permitiendo que todas y cada una de las

unidades aritméticas lógicas (ALU) en el chip, a ser comandadas por el programa que intenta realizar cálculos de propósito general. Debido a que NVIDIA pretende que esta nueva familia de procesadores gráficos sea usada para la computación general, estas ALU fueron creadas con los estándares de la IEEE para aritmética de punto flotante de precisión simple y diseñadas para usar un conjunto de instrucciones, adaptados para la computación general y no específicamente para gráficos. Además, las unidades de ejecución en la GPU, permiten la lectura aleatoria y el acceso a escritura en la memoria como un buen acceso a un software administrado en caché, conocido como memoria compartida (*shared memory*). Todas estas características de la arquitectura CUDA, fueron agregadas en medida para crear un GPU, que se destacara en cálculos, además de un rendimiento adecuado en tareas gráficas adicionalmente" (Rivera & Vargas, 2012).

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Análisis y recolección de información

Como lugar de estudio para la investigación se ha elegido el estacionamiento “COINVIT” ubicado frente al Arco Parabólico de Tacna, debido a que el contexto que presenta puede ser bien controlado.

Para tomar las fotos se utiliza una cámara Samsung PL120 14.4 mega píxels, se procedieron a tomar 30 fotos de autos para la base de datos y 70 fotos para realizar pruebas, las fotos tienen una resolución de 1920x1080, debido a que no se cuentan con los recursos para procesar una imagen de ese tamaño, se recorta la imagen a un tamaño de 800x400 píxeles, algunas de las fotos tomadas se encuentran en el anexo 02.

De las 30 fotos tomadas para la base de datos, se extraen 30 imágenes donde se encuentra la placa del auto y 30 donde no se encuentran, estas imágenes tienen una resolución de 90x50 píxeles, algunas de las fotos tomadas donde aparecen las placas se encuentran en el anexo 03 y en donde no aparecen las placas en el anexo 04. Para realizar esta tarea se utiliza el programa Microsoft Picture Manager.

3.2 Formulación del algoritmo de localización de placas de autos

Para la investigación, la formulación del algoritmo de localización de placas de autos se divide en dos partes:

3.2.1 Secuencial

A continuación, se describe el algoritmo que se utiliza para la localización de placas de autos de manera secuencial:

- Cargar la imagen a procesar.
- Convertir la imagen a escala de grises.
- A la imagen anterior aplicar el detector de bordes Canny.
- Dividir la imagen en bloques de búsqueda.
- Aplicar la Maquina de Vectores de Soporte a los bloques de búsqueda y obtener un listado de bloques candidatos a placas de autos.
- Realizar un filtrado para descartar bloques.
- Obtener la imagen inicial con las placas detectadas.

3.2.2 Paralelo

A continuación, se describe el algoritmo que se utiliza para la localización de placas de autos de manera paralela:

- Cargar la imagen a procesar en memoria del CPU.
- Pasar la imagen en memoria del CPU al GPU
- Convertir la imagen a escala de grises en el GPU.
- A la imagen anterior aplicar el detector de bordes Canny en el GPU.
- Aplicar la Maquina de Vectores de Soporte a los bloques de búsqueda y obtener un listado de bloques candidatos a placas de autos.
- Realizar un filtrado para descartar bloques.
- Obtener la imagen inicial con las placas detectadas.

3.3 Implementación del algoritmo de localización de placas de autos

Para la investigación, la implementación del algoritmo de localización de placas de autos se divide en dos partes:

3.3.1 Secuencial

Para realizar la implementación vamos a utilizar Microsoft Visual Studio 2012 Express en conjunto de la librería OpenCV, la cual nos provee de una gran variedad de funciones con las cuales vamos a poder trabajar con las imágenes.

El programa desarrollado se divide en 4 funciones las cuales se describen a continuación:

- La primera función es la general y recibe como parámetro de entrada la ubicación de la imagen a procesar en el disco duro, esta función se encarga de llamar a las demás funciones y envía como respuesta un binario que representa si se encontró alguna placa de auto.
- La segunda función recibe una matriz de píxeles en formato RGB y se encarga de convertirla en una matriz de píxeles en formato de escala de grises.
- La tercera función recibe la matriz de píxeles en escala de grises y le aplica un filtro gaussiano para eliminar el ruido, luego de ello se le aplica el detector de bordes Canny y así obtiene una matriz de píxeles resultante con los bordes detectados.

- La cuarta función recibe la matriz de píxeles con bordes detectados y la secciona en pequeños bloques de tamaño 90x50 píxeles, dichos bloques son pasados a la función SVM que viene incluida en la librería OpenCV y se determina que bloques son aptos para ser identificados como placas de autos. La función retorna una matriz de píxeles marcando las zonas que son aptas para ser placas de autos.

Adicionalmente se implementa un programa para pre-procesar las 60 imágenes recortadas de la base de datos. Se realiza una iteración por cada imagen, aplicándole la primera y segunda función mencionadas anteriormente, estas imágenes son utilizadas por el SVM.

3.3.2 Paralelo

Para realizar la implementación vamos a utilizar Microsoft Visual Studio 2012 Express en conjunto de la librería OpenCV la cual nos provee de una gran variedad de funciones con las cuales vamos a poder trabajar con las imágenes y la librería de CUDA que nos permite poder trabajar con los núcleos del procesador de la tarjeta gráfica NVIDIA GeForce GTX 750.

El programa desarrollado se divide en 3 funciones las cuales se describen a continuación:

- La primera función es la general y recibe como parámetro de entrada la ubicación de la imagen a procesar en el disco duro, esta función se encarga de llamar a las demás funciones y envía como respuesta un binario que representa si se encontró alguna placa de auto.
- La segunda función recibe una matriz de píxeles en formato RGB, luego pasa la imagen del CPU al GPU y se encarga de convertirla en una matriz de píxeles en formato de escala de grises. Después se le aplica un filtro gaussiano para eliminar el ruido, luego de ello el detector de bordes Canny y así obtiene una matriz de píxeles resultante con los bordes detectados, la cual se encuentra almacenada en memoria del GPU.
- La tercera función recibe la matriz de píxeles con bordes detectados en el GPU, primero se entrena al SVM que viene incluida en la librería OpenCV con las imágenes de la base de datos en el CPU, el vector obtenido luego del entrenamiento del SVM es pasado a memoria del GPU y la matriz de píxeles recibida son procesados en una sub función en CUDA. La función retorna una matriz de píxeles marcando las zonas que son aptas para ser

placas de autos. Se realiza un filtrado final para encontrar la placa de auto.

Adicionalmente se implementa un programa para pre-procesar las 60 imágenes recortadas de la base de datos. Se realiza una iteración por cada imagen, aplicándole la primera y segunda función mencionadas anteriormente. Estas operaciones se realizan en memoria del GPU.

CAPÍTULO IV

ANÁLISIS DE RESULTADOS Y DISCUSIONES

Las 70 imágenes que se obtuvieron en la etapa de análisis y recolección de información son utilizadas para realizar las pruebas de los algoritmos de localización de placas de autos, tanto del secuencial, como del paralelo.

A continuación, se muestra una secuencia de imágenes que resultan del procesamiento de la imagen inicial en cada etapa:

Figura 7. Imagen Inicial.

Fuente: Propia

Figura 8. Resultado de convertir la imagen inicial en escala de grises.

Fuente: Propia

Figura 9. Resultado de aplicar el detector de bordes Canny.

Fuente: Propia

Figura 10. Resultado de aplicar SVM sin filtrar.

Fuente: Propia

Figura 11. Resultado de aplicar SVM filtrado.

Fuente: Propia

Figura 12. Imagen final con la placa detectada.

Fuente: Propia

Los resultados de los tiempos de las pruebas se muestran a continuación:

Tabla 1

Tiempo de procesamiento de las imágenes en segundos

	TRATAMIENTO 1	TRATAMIENTO 2
REPETICION 1	27,819	0,422
REPETICION 2	28,457	0,422
REPETICION 3	30,276	0,421
REPETICION 4	27,058	0,898
REPETICION 5	26,758	0,391
REPETICION 6	27,647	0,438
REPETICION 7	28,450	0,422
REPETICION 8	27,928	0,391
REPETICION 9	28,387	0,406
REPETICION 10	26,741	0,406

REPETICION 11	27,386	0,422
REPETICION 12	25,022	0,406
REPETICION 13	29,729	0,406
REPETICION 14	27,648	0,406
REPETICION 15	25,925	0,406
REPETICION 16	28,387	0,742
REPETICION 17	30,336	0,422
REPETICION 18	29,911	0,406
REPETICION 19	28,729	0,743
REPETICION 20	29,588	0,407
REPETICION 21	28,675	0,406
REPETICION 22	26,932	0,422
REPETICION 23	26,803	0,406
REPETICION 24	22,434	0,421
REPETICION 25	27,850	0,422
REPETICION 26	28,115	0,407
REPETICION 27	29,102	0,406
REPETICION 28	27,432	0,406
REPETICION 29	25,191	0,406
REPETICION 30	23,897	0,407
REPETICION 31	25,850	0,422
REPETICION 32	23,491	0,422
REPETICION 33	22,712	0,406
REPETICION 34	22,071	0,406
REPETICION 35	21,152	0,390
REPETICION 36	23,979	0,390
REPETICION 37	21,618	0,730
REPETICION 38	22,696	0,409
REPETICION 39	27,355	0,422
REPETICION 40	26,429	0,730
REPETICION 41	24,854	0,407
REPETICION 42	21,974	0,730
REPETICION 43	25,678	0,422
REPETICION 44	22,087	0,741
REPETICION 45	22,226	0,406
REPETICION 46	22,009	0,730
REPETICION 47	24,116	0,406
REPETICION 48	22,728	0,422

REPETICION 49	22,210	0,406
REPETICION 50	24,818	0,746
REPETICION 51	25,682	0,407
REPETICION 52	22,712	0,422
REPETICION 53	25,678	0,407
REPETICION 54	22,087	0,742
REPETICION 55	23,897	0,422
REPETICION 56	26,758	0,422
REPETICION 57	22,009	0,406
REPETICION 58	22,712	0,422
REPETICION 59	22,087	0,421
REPETICION 60	22,009	0,422
REPETICION 61	24,116	0,421
REPETICION 62	23,897	0,407
REPETICION 63	26,758	0,742
REPETICION 64	22,009	0,422
REPETICION 65	25,678	0,422
REPETICION 66	23,897	0,407
REPETICION 67	22,087	0,422
REPETICION 68	22,712	0,407
REPETICION 69	26,758	0,406
REPETICION 70	24,116	0,421

El diseño estadístico que se utilizará para la investigación es “t-student para diferencia de medias”.

$$t_0 = \frac{\bar{y}_1 - \bar{y}_2}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

$$Sp^2 = \frac{(n_{1-1})s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

Teniendo la siguiente hipótesis:

$$H_0 \equiv \mu_1 = \mu_2$$

$$H_1 \equiv \mu_1 \neq \mu_2$$

Siendo: H0: No existe diferencia significativa entre medias

H1: Existe diferencia significativa entre medias

Se trabaja con un nivel de confianza del 95 % (α = 0,05).

Con la tabla de datos obtenida en las pruebas se obtienen la siguiente información:

	SECUENCIAL	PARALELO
Y	25,376	0,466
S^2	15,086	0,039
S	2,639	0,126
n	70	70

Reemplazamos la información obtenida en las fórmulas, tenemos:

$$Sp^2 = \frac{(n_{1-1})s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

$S_p = 4,92$

$$t_0 = \frac{\bar{y}_1 - \bar{y}_2}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

$t_0 = 29{,}94$

Dado que tenemos:

$t_{0{,}025;138} = 1{,}98$

Y como $|t_0| > t_{0{,}025;138}$

No se acepta H_0, esto es, que existe diferencia significativa entre las dos medias al nivel de confianza del 95 %.

Con el desarrollo de la investigación se han logrado obtener dos aplicativos, capaces de detectar la ubicación de una placa de auto a partir de una imagen, uno programado con un enfoque secuencial y el otro con un enfoque de programación paralela.

Comparando los tiempos de procesamiento obtenidos (ver Tabla 1), se puede determinar la gran diferencia que existe entre ambos enfoques, con una media de 25,376 en el secuencial y 0,466 en el paralelo, lo cual nos da a entender cuan beneficioso resulta aplicar el enfoque de programación paralela a este tipo de algoritmos que trabajan con procesamiento de imágenes, siendo posible realizar la localización de placas de autos en tiempo real.

DISCUSIONES

Debido a que la programación paralela hace uso del procesamiento de varios núcleos para cumplir una tarea determinada, provoca el incremento de la velocidad de procesamiento de dicha tarea, reduciendo así el tiempo de espera del usuario de 25,376 a 0,466 segundos, este enfoque sólo es posible si el algoritmo es capaz de ser procesado paralelamente.

Los núcleos de la tarjeta gráfica no tienen la misma capacidad de procesamiento como los núcleos del microprocesador, por ello son ideales para procesar operaciones simples. Si son utilizados para realizar operaciones complejas, el resultado será que el tiempo de procesamiento será mayor (pero siempre menor que si se procesara secuencialmente en un núcleo) y además el resultado final de dicha operación compleja podría verse afectado (redondeo de decimales).

El tamaño de las imágenes a procesar con el CPU es limitado y su tiempo de procesamiento muy largo. Sin embargo, en el GPU el tiempo de procesamiento se reduce drásticamente, en la investigación se demuestra una reducción de 25,376 a 0,466 segundos; y el tamaño de las imágenes a procesar aumenta considerablemente, pero aún

presenta un límite que será determinado de acuerdo a la capacidad de la tarjeta gráfica utilizada.

La confiabilidad de acierto del algoritmo de detección de placas de autos depende del tipo y la cantidad de imágenes que se almacenes en la base de datos que utilizará el SVM.

CONCLUSIONES

1. El nivel de influencia de la computación paralela en el algoritmo de localización de placas de autos es elevado, mostrando grandes resultados respecto a la reducción del tiempo de procesamiento de 25,376 a 0,466 segundos.
2. El tiempo promedio de procesamiento del algoritmo de localización de placas de autos con una metodología secuencial es 25,376 segundos, siendo este muy elevado para ejecutarse en tiempo real.
3. El tiempo promedio del procesamiento del algoritmo de localización de placas de autos con una metodología paralela es de 0,466 segundos, siendo este resultado muy cercano para ejecutarlo en tiempo real.

RECOMENDACIONES

1. Es recomendable reforzar el presente algoritmo con el de “Extracción de caracteres” para mejorar el porcentaje de acierto del mismo y aumentar la información obtenida.
2. Para mejorar el algoritmo se recomienda incrementar el número de imágenes utilizadas por el SVM, dichas imágenes deben de ser escogidas con un buen criterio para que sean de utilidad.
3. De llegar a implementarse el algoritmo es recomendable realizar una buena evaluación de los componentes a utilizar, como ser la cámara (de foto o video), la tarjeta gráfica y la computadora.

REFERENCIAS BIBLIOGRÁFICAS

Abe, S. (2010). Support Vector Machines for Pattern Classification. Recuperado de https://books.google.com.pe/books?id=Uy1pqRzABDgC&printsec=frontcover&dq=support+vector+machine&hl=es&sa=X&sqi=2&redir_esc=y#v=onepage&q=support%20vector%20machine&f=false

Bai, H. & Liu, C. (2004). A hybrid License Plate Extraction Method Based On Edge Statistics and Morphology. Recuperado de http://s3.amazonaws.com/academia.edu.documents/9146742/v23_6_12.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1473127539&Signature=XHwv68XADmT2e7EttTo%2FP6VHFNI%3D&response-content-disposition=inline%3B%20filename%3DA_hybrid_license_plate_extraction_method.pdf

Bradski, G. & Kaehler, A. (2008). Learning OpenVC. Recuperado de https://books.google.com.pe/books?id=seAgiOfu2EIC&pg=PA151&dq=canny+algorithm&hl=es&sa=X&redir_esc=y#v=onepage&q=canny%20algorithm&f=false

Carmona Suárez, Enrique J. (2014). Tutorial sobre Máquinas de Vectores Soporte (SVM). Universidad Nacional de Educación a distancia (UNED*)*. Recuperado de http://www.researchgate.net/profile/Enrique_Carmona/publication/263817587_Tutorial_sobre_Mquinas_de_Vectores_Soporte_%28SVM%29/links/0046353bfa8c621114000000.pdf

Ching-Tang, H., Yu-Shan, J., & Kuo-Ming, H. (2005). Multiple License Plate Detection for Complex Background. Recuperado de http://tkuir. lib.tku.edu.tw/dspace/retrieve/80786/0769522491_2p389-392.pdf

Esqueda, J. & Palafox, L. (2005). Fundamentos de procesamiento de imágenes. Recuperado de https://books.google.com.pe/books?id=h4Gj8GuwPVkC&pg=PA9&dq=procesamiento+de+imagenes&hl=es&sa=X&redir_esc=y#v=onepage&q=procesamiento%20de%20imagenes&f=false

Ekstrom, M. (2012). Digital Image Processing Techniques. Academic Press. Recuperado de https://books.google.com.pe/books?hl=es&lr=&id=h9qN1KlX388C&oi=fnd&pg=PP1&dq=digital+image+processing+technique&ots=xnyuykEInX&sig=ViOEdEjjG3-ltPb-nNTnxwh-

_4g&redir_esc=y#v=onepage&q=digital%20image%20processing%20technique&f=false

Fatih, P & Tekin, K. (2007). A hybrid License Plate Extraction Method Based On Edge Statistics and Morphology. Recuperado de http://www.merl.com/publications/ docs/TR2006-100.pdf

Hamidreza, S., Mohammadreza, S., & Alireza, S. (2010). New Morphology-Based Method for Robust Iranian Car Plate Detection and Recognition. Recuperado de http://ijcte.org/papers/150-G667.pdf

Hernández, R., Fernández, C. & Baptista, P.(2006). Metodología de la Investigación. Recuperado de http://s3.amazonaws.com/academia.edu.documents/38758233/sampieri-et-al-metodologia-de-la-investigacion-4ta-edicion-sampieri-006_ocr.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1473188301&Signature=ysVgiHd26VbGlHqc7LxP4amJ0Bs%3D&response-content-disposition=inline%3B%20filename%3DSampieri-et-al-metodologia-de-la-investi.pdf

Hossein, A., Jan, M., & Fathy, M., (2014). An Iranian License Plate Recognition System Based on Color Features. Recuperado de https://www.researchgate.net/profile/Md_Jan_Nordin/publication/262

098444_An_Iranian_License_Plate_Recognition_System_Based_on_Color_Features/links/0deec536a07a5e6b02000000.pdf

Huaifeng, Z., Wenjing, J., Xiangjian, H., & Qiang, W. (2006). Learning-Based License Plate Detection Using Global and Local Features. Recuperado de https://staff.fnwi.uva.nl/r.vandenboomgaard/UvAwiki/uploads/HVisser/Learning_Based_License_Plate_Detection_Using_Global_and_Local_Features_Zhang_Jia_He_Wu.pdf

Jayaraman, S., Esakkirajam, S. & Veerakumar T. (2009). Digital Image Processing. Recuperado de https://books.google.com.pe/books?id=JeDGn6Wmf1kC&printsec=frontcover&dq=image+processing&hl=es&sa=X&redir_esc=y#v=onepage&q=image%20processing&f=false

Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. (2007). A Thai License Plate Localization using SVM. Mahidol University, Thailand. Recuperado de http://mucc.mahidol.ac.th/~worapan.kun/index_files/2014ICSEC_car_plate_accepted.pdf

Lipo, E. (2005). Support Vector Machine: Theory and Applications. Recuperado de

https://books.google.com.pe/books?id=uTzMPJjVjsMC&printsec=frontcover&dq=support+vector+machine&hl=es&sa=X&sqi=2&redir_esc=y#v=onepage&q=support%20vector%20machine&f=false

Maiti, M. (1991). Metodología de la Investigación. Recuperado de http://eprints.uanl. mx/195/1/1020070614.PDF

Martínez, H. (2012). Metodología de la Investigación. Recuperado de http://datateca.unad.edu.co/contenidos/210101/metodologiade_la_investigacion_clave.pdf

Martínez, P. (2006). Estrategia metodológica de la investigación científica. Recuperado de http://rcientificas.uninorte.edu.co/index.php/pensamiento/article/viewFi le/3576/2301

Mokayed, H., kim, L., Hock, H., & Hooi N., (2014), Car Plate Deetection Engine Based on Conventional Edge Detection Technique. Recuperado de sdiwc. us/digitlib/request.php?article=afbd5e19da98a1899e21df6f94dc9972

Oviedo, E. (2004). Lógica de Programación. Recuperado de https://books.google.com.pe/books?id=Z_n5lbyJfrQC&printsec=front

cover&dq=programacion+secuencial&hl=es&sa=X&redir_esc=y#v=onepage&q=secuencial&f=false

Qiang, W., Huaifeng, Z., Wenjing, J., Xiangjian, H., Jie, Y., & Tom, H. (2007). Car Plate Detection Using Cascaded Tree-Style Learner Based on Hybrid Object Features. Recuperado de https://staff.fnwi.uva.nl/r.vandenboom gaard/UvAwiki/uploads/Fibbe/car_plate_detection_using_cascaded_tree-style_learner_based _on_hybrid_object_features.pdf

Rivera, I., & Vargas, M. (2012). Principios y Campos de Aplicación en CUDA Programación paralela y sus potencialidades. Recuperado de http://dialnet.unir ioja.es/servlet/articulo?codigo=5006285

Sanders, J. & Kandrot, E. (2011). CUDA BY EXAMPLE: An Introduction to General Purpose GPU Programming. Recuperado de https://books.google.com.pe/books?id=49OmnOmTEtQC&printsec=frontcover&dq=cuda&hl=es&sa=X&redir_esc=y#v=onepage&q=cuda&f=false

Valverde, J. (2007). Detección de bordes mediante el algoritmo de Canny. Universidad Nacional de Trujillo. Recuperado de ttitulo1.googlecode.com/svn /trunk/informe/img/Detecci%C3%B3ndeBordes-Canny.pdf

Yu, S., Hsieh, J., & Chen, Y. (2002). Morphology-based License Plate Detection from Complex Scenes. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.507.6824&rep=rep1&type=pdf

ANEXOS

ANEXO 01

MATRIZ DE CONSISTENCIA

Título: Optimización de la velocidad de procesamiento del algoritmo de localización de placas de autos usando computación paralela.

FORMULACIÓN DEL PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES	METODOLOGÍA
PROBLEMA GENERAL	OBJETIVO GENERAL	HIPÓTESIS GENERAL	- Variable 1 Metodología de programación. • Placas detectadas. - Variable 2 Velocidad de procesamiento del algoritmo de localización de placas de autos. • Tiempo de ejecución	**Diseño de la investigación:** Experimental pura **Tipo de investigación:** Cuantitativa discreta **Técnicas de recolección de datos:** Técnica de la Observación. **Instrumentos de Medición:** Cámara Samsung PL120 y clase Clock de Visual Studio 2012 Express **Prueba estadística:** T-student diferencia de medias **Población:** Las imágenes obtenidas por la cámara en el estacionamiento "COINVIT". **Muestra:** 70 imágenes de autos.
¿Cuál es la influencia de la computación paralela en la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos?	Determinar el nivel de influencia de la computación paralela en la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos.	Existe una influencia significativa de la computación paralela en la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos.		
PROBLEMA ESPECÍFICO 1	OBJETIVO ESPECÍFICO 1	HIPÓTESIS ESPECÍFICA 1		
¿Cómo es la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación secuencial?	Determinar la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación secuencial.	Es posible medir la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación secuencial.		
PROBLEMA ESPECÍFICO 2	OBJETIVO ESPECÍFICO 2	HIPÓTESIS ESPECÍFICA 2		
¿Cómo es la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación paralela?	Determinar la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación paralela.	Es posible medir la velocidad de procesamiento del algoritmo de Kusakunniran, W., Ngamaschariyakul, K., & Chantaraviwat, C. en la localización de placas de autos utilizando computación paralela.		

ANEXO 02

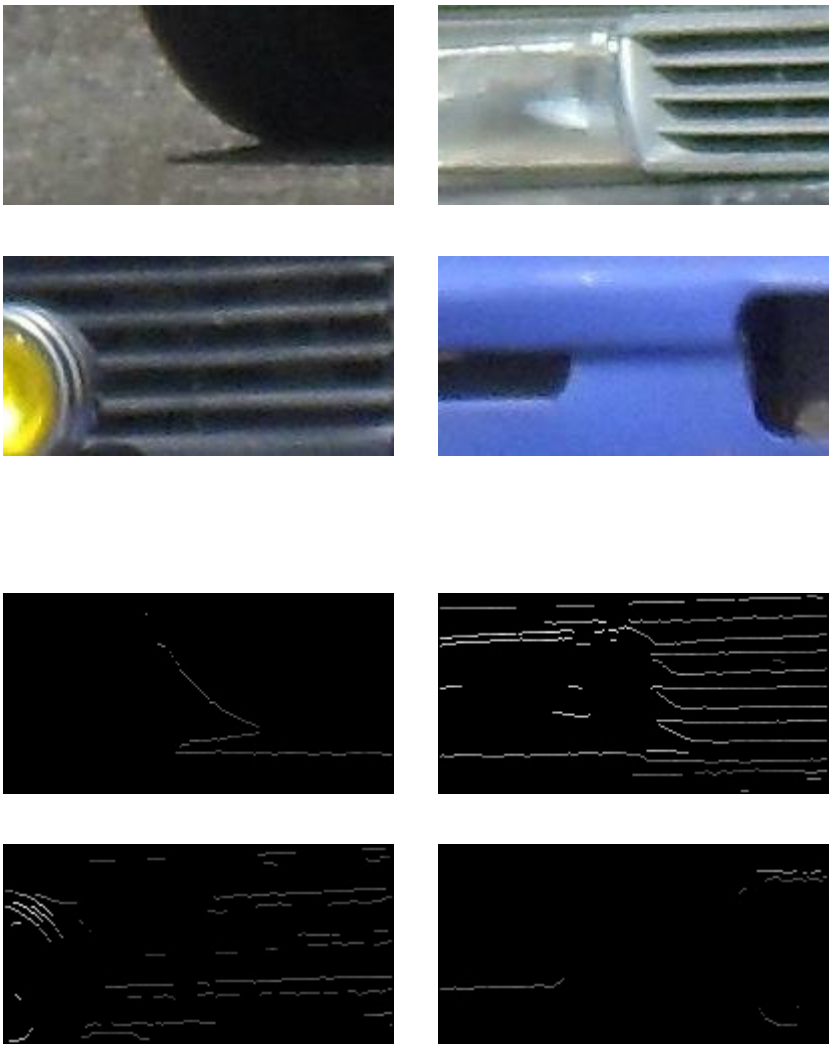
IMÁGENES DE LA BASE DE DATOS (800x400)

ANEXO 03

IMÁGENES DE LA BASE DE DATOS RECORTADAS CON PLACA (90x50)

ANEXO 04

IMÁGENES DE LA BASE DE DATOS RECORTADAS SIN PLACA (90x50)



ANEXO 05

INGRESO DE VEHICULOS AL ESTACIONAMIENTO “COINVIT”

FECHA	AUTOS
1/04/2017	146
2/04/2017	173
3/04/2017	163
4/04/2017	148
5/04/2017	157
6/04/2017	120
7/04/2017	135
8/04/2017	145
9/04/2017	128
10/04/2017	134
11/04/2017	168
12/04/2017	135
13/04/2017	151
14/04/2017	148
15/04/2017	144
16/04/2017	138
17/04/2017	142
18/04/2017	119
19/04/2017	143
20/04/2017	139
21/04/2017	152
22/04/2017	148
23/04/2017	140
24/04/2017	121
25/04/2017	166
26/04/2017	112
27/04/2017	154
28/04/2017	141
29/04/2017	170
30/04/2017	126

FECHA	AUTOS
1/05/2017	138
2/05/2017	112
3/05/2017	168
4/05/2017	133
5/05/2017	139
6/05/2017	165
7/05/2017	118
8/05/2017	121
9/05/2017	132
10/05/2017	136
11/05/2017	118
12/05/2017	122
13/05/2017	173
14/05/2017	148
15/05/2017	137
16/05/2017	136
17/05/2017	142
18/05/2017	158
19/05/2017	143
20/05/2017	128
21/05/2017	142
22/05/2017	125
23/05/2017	139
24/05/2017	151
25/05/2017	142
26/05/2017	163
27/05/2017	138
28/05/2017	160
29/05/2017	120
30/05/2017	141
31/05/2017	125