Código	FPI-009
Objeto	Guía de elaboración de Informe finalde proyecto
Usuario	Director de proyecto de investigación
Autor	Secretaría de Ciencia y Tecnología de la UNLaM
Versión	5
Vigencia	03/9/2019

Departamento:
Ingeniería e Investigaciones Tecnológicas

Programa de acreditación:
PROINCE

Programa de Investigación[1]:

Código del Proyecto:
C218

Título del proyecto:
Obtención de data set a partir de imágenes capturadas del estacionamiento de UNLaM para su utilización en sistema experto de reconocimiento de imágenes

Director:
Ing. Fernando I. Szklanny

Codirector:
Lic. Carlos Eduardo Maidana

Integrantes:
Ing. Edgardo Gho
Ing. Martín Ferreyra Birón
Ing. Carlos Alberto Rodríguez

Resolución Rectoral de acreditación:
N°354/19

Fecha de inicio:
01/01/2019

Fecha de finalización:
30/04/2021

[1] Los Programas de Investigación de la UNLaM están acreditados con resolución rectoral, según lo indica la Resolución HCS Nº 014/15 sobre **Lineamientos generales para el establecimiento, desarrollo y gestión de Programas de Investigación a desarrollarse en la Universidad Nacional de La Matanza**. Consultar en el departamento académico correspondiente la inscripción del proyecto en un Programa acreditado.

A. Desarrollo del proyecto (adjuntar el protocolo)

A.1. Tal como se hizo referencia en el informe de avance el presente proyecto tuvo durante su primer año de desarrollo una serie de inconvenientes, que obligaron a los investigadores a replantear varias veces el objetivo y el enfoque del mismo.

El proyecto se inició original y teóricamente en el mes de febrero de 2019. Al comenzar a trabajar más profundamente en el tema, surgió la necesidad de algún equipamiento adicional al que posee el grupo de investigación, así como la mejora en las prestaciones de las computadoras disponibles.

Una vez adquirido el equipamiento adicional se requirió confirmación de la autorización para instalar la cámara de video en la playa de estacionamiento.

Debido a diversos inconvenientes de tipo edilicio en la zona en la que se debía instalar la mencionada cámara, se buscó, como alternativa, replantear el proyecto, cuya base era la recolección de imágenes de un conjunto conocido de patentes vehiculares para la creación de una base de datos.

Se planteó como alternativa temporaria la adquisición de datos en la vía pública por medio de una cámara tipo “Dashcam”. Esta solución requiere la toma de varias horas de imágenes en video y la posterior extracción manual de cada una de las patentes, para luego seleccionar las que fuesen lo suficientemente legibles como para ser incorporadas al acervo.

A comienzos del año 2020 se continuó trabajando con la presente investigación, pero todavía con las restricciones internas respecto a la captura de imágenes dentro de la universidad ya mencionadas. Estas restricciones , que se debieron a razones ajenas a este grupo de investigación, de alguna forma se encontraban subsanadas y , solamente, restaba esperar a instalar las cámaras de video en el estacionamiento de la universidad y a tener un flujo importante de autos en el estacionamiento , pero , por esos meses el Aislamiento Social Preventivo y Obligatorio comenzó a regir , la universidad cerró las puertas y el flujo de automóviles para capturar imágenes de automóviles y patentes fue nulo en la universidad , y en los primeros meses de la pandemia , también en gran parte del AMBA.

Esta situación, junto a las obligaciones adicionales sumadas producto del aislamiento perjudicó sustancialmente el progreso de esta investigación , sin embargo, a pesar de toda esta adversidad se analizó cuál sería la mejor manera de continuar con la misma. Nuevamente se debió cambiar el rumbo y adaptarse a las circunstancias, para subsanar ello, se comenzó a confeccionar un programa para segmentar de forma manual las patentes, para así comenzar a armar el acervo de datos. De esta forma los integrantes del grupo instalaron cámaras en los parabrisas de sus automóviles y se comenzó a grabar videos. Por lo tanto la forma en la que se continuó con la presente investigación fue en un principio tomar el programa en cuestión y mejorarlo para comenzar a crear el acervo. Se debe tener en cuenta que al usar las imágenes del estacionamiento de la universidad, se hubiera podido corroborar la chapa patente contrastando la base de datos de la universidad y el código del TAG instalado en cada vehículo, información que se posee, ya que el grupo fue el encargado del desarrollo del sistema de estacionamiento de la universidad. Al filmar autos que no existen en esa base de datos se deben corroborar las imágenes manualmente.

Una de las primeras hipótesis planteadas para el correcto reconocimiento de patentes es que, lo primero que se debe buscar en la imagen , es la posición del automóvil en la imagen, o dicho de otra manera , lo primero que se debe segmentar de la imagen es el automóvil. Esto debe ser así ya que una patente, conceptualmente, es solo un cartel, y realizar la búsqueda de este cartel en una imagen no es eficiente e incluso puede ser contraproducente debido a los falsos positivos que pueden crearse, esto aún es peor cuando las imágenes son realmente muy grandes. Por lo tanto el programa anteriormente mencionado, se adaptó no solo para segmentar patentes y caracteres sino también para crear acervos de las tres clases anteriormente mencionadas. Toda la información segmentada que conforma este acervo es guardado en una carpeta acompañado de un archivo XML que "etiqueta" cada una de las respectivas imágenes dándole así un contexto a los datos.

Es importante destacar que los distintos puntos que se buscaban alcanzar, como la creación de un acervo de datos y la detección de patentes en automóviles no se encuentra muy lejana de llevarse a cabo por completo: El acervo de datos puede crecer rápidamente debido a las herramientas desarrolladas, y si fue posible detectar automóviles con la red neuronal YOLO (You Only Look Once) , también será posible entrenar a la red neuronal para detectar patentes en los autos , tanto patentes vigentes desde 1994 a 2016 en la República Argentina como también las que respetan el formato MERCOSUR vigente desde abril de 2016. Probablemente esto implique un desafío de implementación, pero el objetivo probablemente pueda ser alcanzado. Otro punto a investigar es el comportamiento de redes neuronales similares para realizar la tarea de detección de objetos en tiempo real y poder compararla con YOLO , y por último resta investigar si la utilización de la red YOLO es útil en la detección de caracteres en tiempo real , o existe alguna alternativa para un problema que en teoría ya no es una desafío para el campo de la inteligencia artificial debido a las diversas redes neuronales que a lo largo de los años se desarrollaron para cumplir con este objetivo.

B. Principales resultados de la investigación

B.1. Publicaciones en revistas (informar cada producción por separado)

Artículo 1:	
Autores:	Martín Ferreyra Birón, Fernando I. Szklanny, Alberto Miguens
Título del artículo:	Desarrollo de un sistema de asistencia visual para un manipulador automático programable
N° de fascículo:	*1*
N° de Volumen:5	*5*
Revista:	*ReDDI*
Año:	*2020*
Institución editora de la revista	*UNLaM*
País de procedencia de institución editora	*Argentina*
Arbitraje	NO
ISSN:	*2525-1333-Redd*
URL de descarga del artículo	*https://reddi.unlam.edu.ar/index.php/ReDDi/article/view/120/222*
N° DOI	

B.2. Libros

Libro 1	
Autores	
Título del Libro	
Año	
Editorial	
Lugar de impresión	
Arbitraje	Elija un elemento.
ISBN:	
URL de descarga del libro	
N° DOI	

B.3. Capítulos de libros

Autores	
Título del Capitulo	
Título del Libro	
Año	
Editores del libro/Compiladores	
Lugar de impresión	
Arbitraje	Elija un elemento.
ISBN:	
URL de descarga del capítulo	
N° DOI	

B.4. Trabajos presentados a congresos y/o seminarios

Autores	*Ing. Martín Ferreyra Biron, Ing. Alberto Miguens, Ing. Fernando Szklanny*
Título	*Desarrollo de un sistema de asistencia visual…*
Año	*2019*
Evento	*CONAIISI 2019*
Lugar de realización	*UNLAM*
Fecha de presentación de la ponencia	*15/09/2019*
Entidad que organiza	*UNLAM*
URL de descarga del trabajo(especificar solo si es la descarga del trabajo; formatos pdf, e-pub, etc.)	*https://conaiisi2019.unlam.edu.ar/*

B.5. Otras publicaciones

Autores	
Año	
Título	
Medio de Publicación	

C. Otros resultados. Indicar aquellos resultados pasibles de ser protegidos a través de instrumentos de propiedad intelectual, como patentes, derechos de autor, de
rechos de obtentor, etc. y desarrollos que no pueden ser protegidos por instrumentos de propiedad intelectual, como las tecnologías organizacionales y otros. Complete un cuadro por cada uno de estos dos tipos de productos.

C.1. Títulos de propiedad intelectual. Indicar: Tipo (marcas, patentes, modelos y diseños, la transferencia tecnológica) de desarrollo o producto, Titular, Fecha de solicitud, Fecha de otorgamiento

Tipo	Titular	Fecha de Solicitud	Fecha de Emisión

C.2. Otros desarrollos no pasibles de ser protegidos por títulos de propiedad intelectual. Indicar: Producto y Descripción.

Producto	Descripción

D. Formación de recursos humanos. Trabajos finales de graduación, tesis de grado y posgrado. Completar un cuadro por cada uno de los trabajos generados en el marco del proyecto.

D.1. Tesis de grado

Director (apellido y nombre)	Autor (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título de la tesis

D.2 Trabajo Final de Especialización

Director (apellido y nombre)	Autor (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título del Trabajo Final

D.2. Tesis de posgrado: Maestría

Director (apellido y nombre)	Tesista (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título de la tesis

D.3. Tesis de posgrado: Doctorado

Director (apellido y nombre)	Tesista (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título de la tesis

D.4. Trabajos de Posdoctorado

Director (apellido y nombre)	Posdoctorando (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título del trabajo	Publicación

E. Otros recursos humanos en formación: estudiantes/ investigadores (grado/posgrado/ posdoctorado)

Apellido y nombre del Recurso Humano	Tipo	Institución	Período (desde/hasta)	Actividad asignada[2]

F. Vinculación[3]: Indicar conformación de redes, intercambio científico, etc. con otros grupos de investigación; con el ámbito productivo o con entidades públicas.Desarrolle en no más de dos (2) páginas.

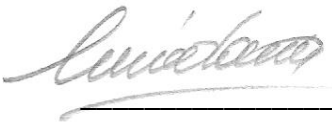
G. Otra información. Incluir toda otra información que se considere pertinente.

[2] Descripción de la/s actividad/es a cargo (máximo 30 palabras)

[3]Entendemos por acciones de "vinculación" aquellas que tienen por objetivo dar respuesta a problemas, generando la creación de productos o servicios innovadores y confeccionados "a medida" de sus contrapartes.

H. Cuerpo de anexos:

- Anexo I: Descripción detallada del proyecto.
- Anexo II: Copia de cada uno de los trabajos mencionados en los puntos B, C y D, y certificaciones cuando corresponda.[4]
- Anexo III:
    - FPI-013: Evaluación de alumnos integrantes. (si corresponde)
    - FPI-014: Comprobante de liquidación y rendición de viáticos. (si corresponde)
    - FPI-015: Rendición de gastos del proyecto de investigación acompañado de las hojas foliadas con los comprobantes de gastos.
    - FPI-035: Formulario de reasignación de fondos en Presupuesto.
- Anexo IV: Alta patrimonial de los bienes adquiridos con presupuesto del proyecto (FPI 017)
- Nota justificando baja de integrantes del equipo de investigación.



Lic. Carlos Eduardo Maidana
Codirector

Lugar y fecha: San Justo 30 de abril de 2021

[4]En caso de libros, podrá presentarse una fotocopia de la primera hoja significativa o su equivalente y el índice.

ANEXO I

Descripción detallada del proyecto

A comienzos del año 2020 se continuó trabajando con la presente investigación, pero todavía con las restricciones internas respecto a la captura de imágenes dentro de la universidad ya mencionadas en el informe anterior. Estas restricciones, que se debieron a razones ajenas a este grupo de investigación, de alguna forma se encontraban subsanadas y solamente, restaba esperar a instalar las cámaras de video en el estacionamiento de la universidad y poder tener un flujo importante de autos en el estacionamiento, pero, por esos meses el Aislamiento Social Preventivo y Obligatorio comenzó a regir, la universidad cerró las puertas y el flujo de automóviles para capturar imágenes de automóviles y patentes fue prácticamente nulo en la universidad y en los primeros meses de la pandemia en gran parte del AMBA. Esta situación junto a las obligaciones adicionales sumadas producto del aislamiento perjudicó sustancialmente el progreso de esta investigación , sin embargo, a pesar de toda esta situación se analizó cuál sería la mejor manera de continuar con la misma. A fines del año 2019, ya con la negativa de no poder obtener imágenes de automóviles en la universidad, se comenzó a confeccionar un programa para segmentar de forma manual las patentes, para así comenzar a armar el acervo de datos, como también se comenzó a grabar videos con una cámara instalada en los vehículos de los integrantes. Por lo tanto la forma en la que se continuó con la presente investigación fue en un principio tomar el programa en cuestión y mejorarlo para comenzar a crear el acervo.

Una de las primeras hipótesis planteadas para el correcto reconocimiento de patentes es que, lo primero que se debe buscar en la imagen , es la posición del automóvil en la imagen, o dicho de otra manera , lo primero que se debe segmentar de la imagen es el automóvil. Esto debe ser así ya que una patente, conceptualmente, es solo un cartel, y realizar la búsqueda de este cartel en una imagen no es eficiente e incluso puede ser contraproducente debido a los falsos positivos que pueden crearse, esto aún es peor cuando las imágenes son realmente muy grandes. Por lo tanto el programa anteriormente mencionado, se adaptó no solo para segmentar patentes y caracteres sino también para crear acervos de las tres clases anteriormente mencionadas. Toda la información segmentada que conforma este acervo es guardado en una carpeta acompañado de un archivo XML que "etiqueta" cada una de las respectivas imágenes dándole así un contexto a los datos.

A continuación se puede observar la pantalla principal del programa confeccionado el cual permite elegir el video principal del que se obtendrán las imágenes para así segmentar un auto o una patente (Fig. 1) para luego continuar con la siguiente pantalla que permite seleccionar el tipo de dato que fue segmentado para finalmente poder guardarlo como se muestra en la Fig. 2.

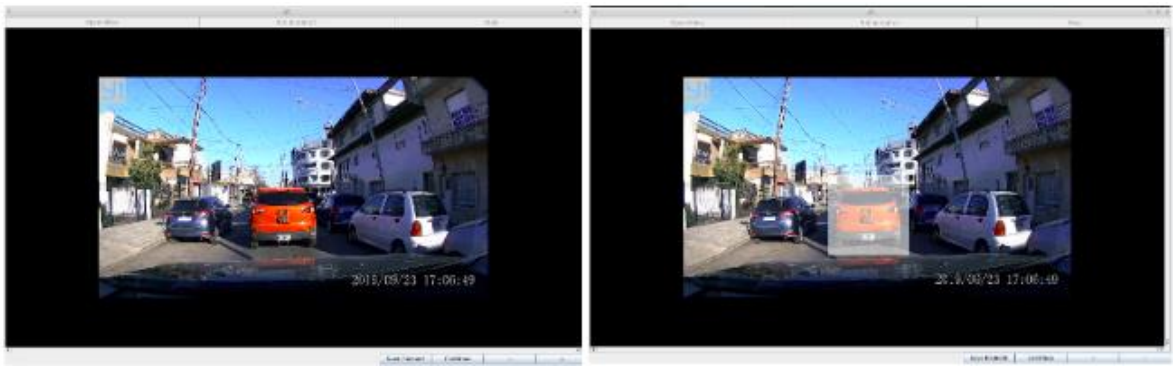
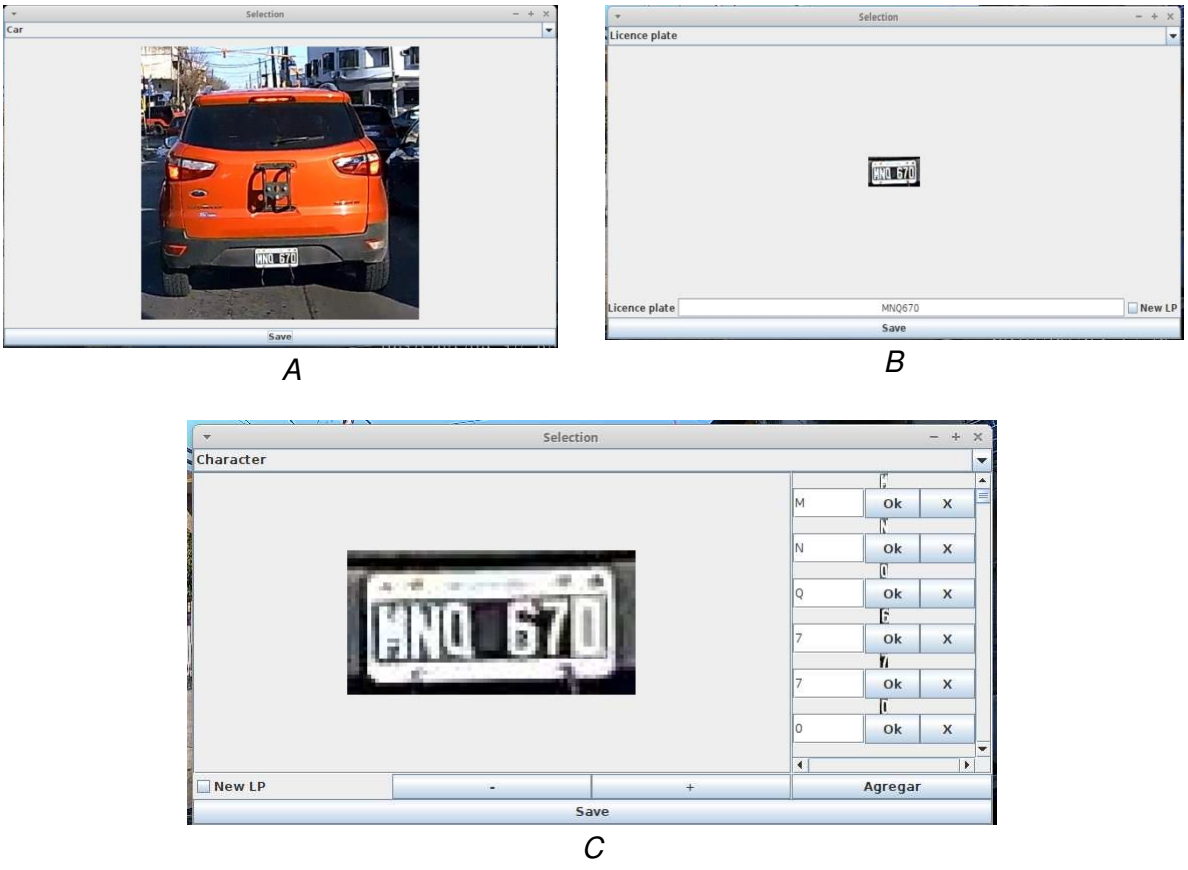


Fig. 1: Pantalla principal del programa desarrollado para segmentar autos, patentes y caracteres de patentes



A *B* *C*

Fig 2. A. Pantalla diseñada para anexar una imagen de un automóvil al acervo. B. Pantalla diseñada para anexar una patente al acervo. C. Pantalla para anexar los caracteres segmentados de las patentes.

.

Ya teniendo en nuestro poder este programa, nos enfocamos en parte, a segmentar manualmente autos , patentes y caracteres, empezando a cumplir con nuestro primer objetivo en la investigación que es

confeccionar un acervo de datos de imágenes de patentes. En este caso el acervo aumentó en clases debido a que segmentamos automóviles, patentes y caracteres de las mismas patentes. Habiéndose comenzado a lograr este objetivo también fue imperioso lograr visualizar de manera acertada este acervo , para así poder manipularlo , curar las imágenes que se obtuvieron si fuera necesario y poder administrarlo de alguna forma. Para esto se confeccionó otro programa el cual toma el archivo XML y muestra cada uno de los elementos que posee el acervo, con la posibilidad de modificar la imagen , o cualquier dato asociado al archivo XML. A continuación se puede observar la Fig 3. la cual muestra como se observan las tres clases o categorías en el programa

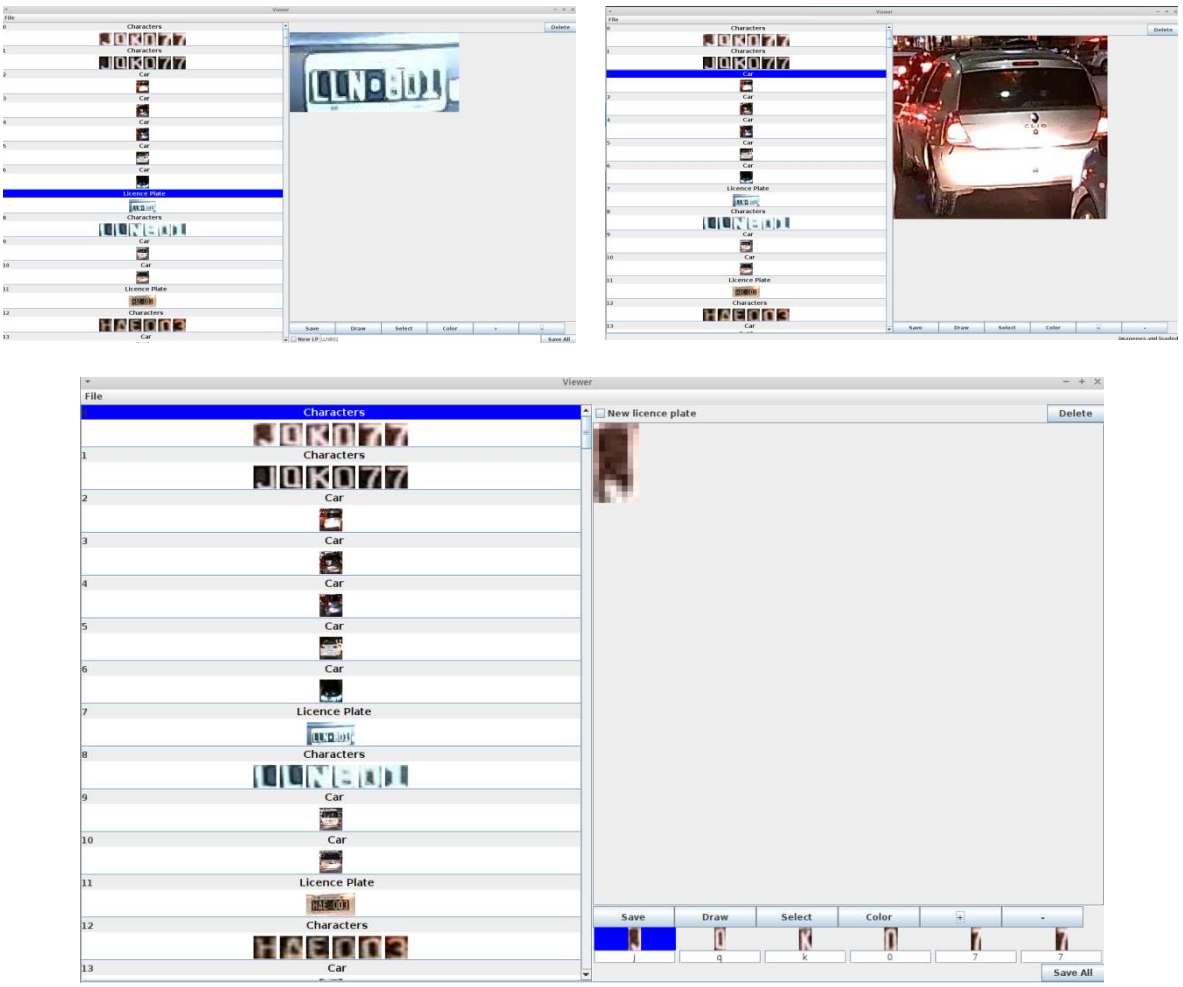


Fig. 3: Vista de la selección de un ejemplo de cada una de las clases segmentadas

Finalmente el programa permite guardar las imágenes de cada una de las clases segmentadas, (autos , caracteres , patentes o un carácter específico) y ejecutar por cada uno de los elementos guardados un comando particular , si es que se lo desea (por ejemplo si se desea redimensionar las imágenes se puede eje-

cutar por cada una de las imágenes el programa convert). La Fig. 4 muestra la ventana que permite esta acción.

Fig 4. Ventana para exportar las imágenes guardadas

Por último el programa permite observar las estadísticas de las patentes, caracteres y autos recopilados como se muestra en la Fig. 5.

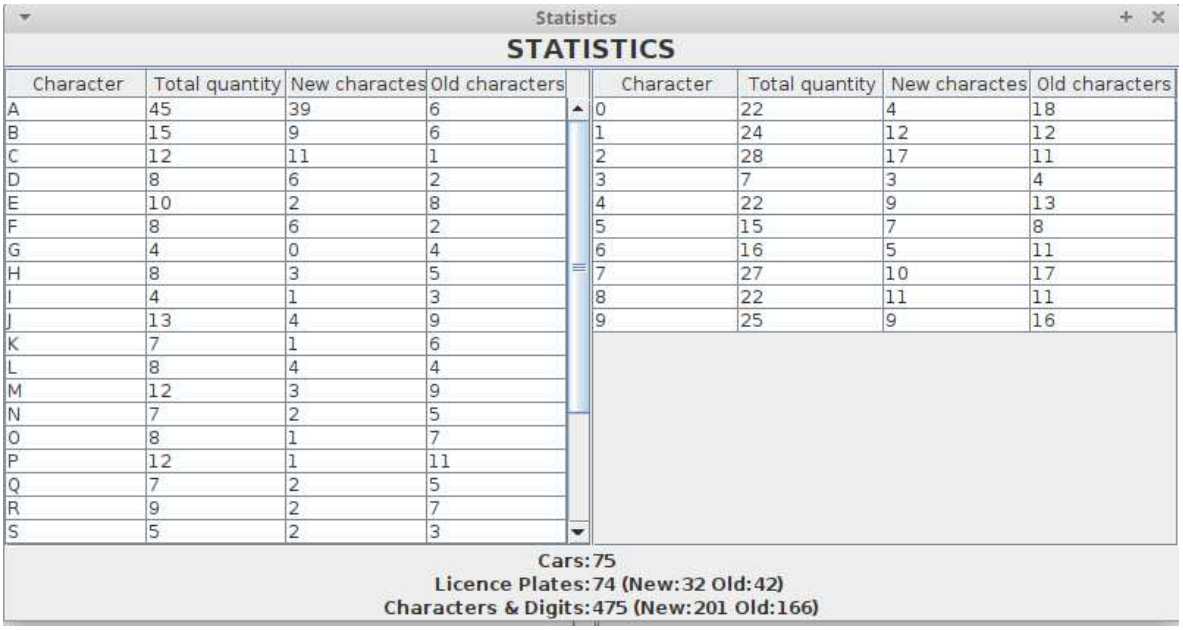


Fig. 5: En esta ventana se puede observar cuántos caracteres, dígitos, autos y patentes nuevas y viejas fueron recolectadas

La Fig. 5. muestra la cantidad de elementos que fueron segmentados al comenzar con la etapa de reconocimiento. Si bien la cantidad segmentada es magra , fue imperioso comenzar a investigar sobre re-

des neuronales que realmente puedan detectar objetos en tiempo real. De cualquier forma este acervo puede crecer muy rápidamente debido a que ya se encuentran confeccionados todos los programas pertinentes

Como se mencionó anteriormente, teniendo ya una pequeña cantidad de imágenes para poder utilizar nos adentramos en investigar qué redes neuronales son capaces de detectar correctamente en tiempo real elementos en un video. Nuestra primera aproximación es entonces validar de alguna forma las imágenes ya obtenidas entrenando una red neuronal y en caso de funcionar correctamente utilizarla como apoyatura , si fuera posible, para hacer la segmentación manual más rápida.

Como grupo de investigación ya conocíamos las redes neuronales convolucionales como el Neocognitron y las redes LeNet y en parte su principio de funcionamiento, pero desde el año 2015 una red neuronal particular comenzó a ser realmente muy práctica, utilizada y precisa para las tareas de reconocimiento de imágenes en tiempo real. Esta nueva red neuronal, convolucional , se denomina YOLO (You Only Look Once). El principio de funcionamiento es complejo ya que YOLO define una arquitectura de red neuronal convolucional particular que trata de predecir en qué lugar se encuentra el o los objetos en la imagen, para esto YOLO escala la imagen a un determinado ancho y alto (esto anchos y altos están definidos por las distintas resoluciones de redes YOLO, ya que la entrada de esta red neuronal convolucional es toda la imagen) y la divide en segmentos. YOLO busca el segmento que mayor probabilidad posee para determinada clase y devuelve un rectángulo que contiene el objeto reconocido en la imagen. Esta red neuronal fue evolucionando con el paso del tiempo , y en el año 2018 la versión 3 de YOLO fue presentada (YOLOv3) y la versión 4 de YOLO fue presentada en abril del año 2020. En teoría la red neuronal es perfecta para nuestra primera etapa de segmentación de autos y quizás también podría funcionar para detectar las patentes con la imagen del automóvil. Para comprobar que realmente esta red neuronal es realmente funcional, lo primero que realizamos fue intentar hacer funcionar un ejemplo de esta red para conocer su eficiencia.

La red YOLOv3 comúnmente se ofrece pre entrenada con la una base de datos de patrones llamada COCO (Common Objects in Context), esta red pre entrenada reconoce 80 tipos de objetos distintos como personas, perros, motos y entre éstos también automóviles, por lo tanto fue la red neuronal perfecta para probarla. Después de una difícil implementación obtuvimos resultados muy satisfactorios con en una imagen estática de 1920x1080 píxeles utilizando una red YOLO cuya entrada es de 416x416, sin embargo la detección en videos en tiempo real (de la misma resolución que la imagen estática) era demasiado lenta debido a que la inferencia se realizaba solamente en la CPU sin utilizar la GPU en lo absoluto, aunque la eficiencia en la detección era igualmente buena (Fig. 6). Cabe destacar que esta prueba se realizó sobre una computadora con microprocesador I7-7700HQ con 16 GB de memoria y una placa de video Nvidia GTX 1050.

Evidentemente el uso de una GPU disminuiría considerablemente el tiempo de detección de los objetos en YOLO por lo tanto después de instalar todas las dependencias pertinentes pudimos ejecutar en otra prueba esta red neuronal convolucional utilizando la GPU integrada en la computadora y los resultados fueron realmente muy alentadores. mientras que utilizando solo la CPU se alcanzaban valores de entre 5 a 6 fps utilizando la GPU se alcanzaron valores de entre 14 a 15 fps

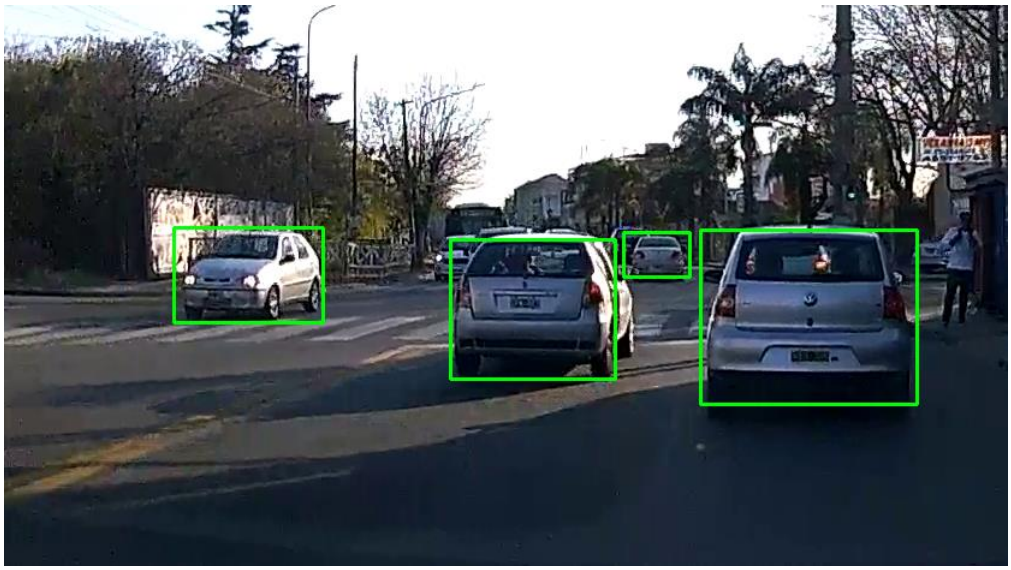


Fig. 6. Fotograma en donde una red neuronal YOLOv3 320 (pre entrenada con la base de datos de COCO) detecta correctamente 4 vehículos

También es importante recalcar que esta no es la red neuronal más precisa en la detección de objetos , pero si es realmente rápida cuando se habla de detección de objetos en tiempo real. Un ejemplo de la falta de precisión de esta red lo pudimos observar en la detección de una camioneta , la red detecta dos automóviles en vez de una sola camioneta. Fig. 7.

Fig 7. Una detección fallida de la red YOLO, a la derecha la ampliación del fallo

Este tipo de errores repercuten demasiado a la detección , ya que la camioneta se encuentra alejada, pero en ciertas ocasiones , el auto negro que se encuentra a la derecha y al frente de la imagen , algunas veces no es detectado correctamente, aunque esto puede ser completamente compensado entrenando mejor la red o agregando un poco más de lógica al detector

Si bien la red YOLOv3 es realmente buena en la detección en tiempo real de objetos parte de nuestro objetivo es implementar todo el sistema de detección de imágenes en tiempo real en un sistema embebido, como por ejemplo una Raspberry 2 / 3 , y si bien la ejecución de YOLOv3 cumple con las expectativas esperadas , la realidad es que la computadora en la que se ejecutó la prueba dista por mucho a las características de hardware que ofrece un sistema embebido como la placa anteriormente mencionada. Sin embargo para esto, ya existe un diseño de una arquitectura de YOLOv3 adaptada para sistemas embebidos que se denomina YOLO-Tiny, que como su nombre lo indica es una versión reducida de la red neuronal convolucional YOLO así que ya con un pequeño acervo de datos , nos dedicamos a entrenar una red YOLO en su versión más pequeña con el conjunto de datos obtenidos de automóviles. Después de 15 horas de entrenamiento de la red YOLO (40000 iteraciones) los resultados no fueron en lo absoluto alentadores, ya que los rectángulos que marcaban los automóviles eran muy grandes respecto a la del objeto encontrado haciendo que la precisión de la detección sea realmente muy deficiente. La figura 8 muestra esta situación. Esto se debe a la forma en la que la red neuronal aprende cuales son los tamaños de los rectángulos que deben contener al objeto al encontrarlo, cuando entrenamos la red neuronal con las imágenes obtenidas del acervo , estas imágenes contenían solamente al objeto en cuestión pero YOLO también necesita la información de la imagen completa y de la relación entre el ancho del objeto que

Fig 8. Una detección fallida después de entrenar la red YOLO por primera vez

que va a ser utilizado para entrenar y el ancho de la imagen donde se encuentra dicho objeto , el alto del objeto que va a ser utilizado para entrenar y el alto de la imagen donde se encuentra dicho objeto, la posición absoluta respecto en la imagen , etc. Al tener ya el acervo creado, en lugar de rearmar el mismo te-

niendo en cuenta los datos necesarios para entrenar la red neuronal YOLO, creamos un pequeño programa para encontrar los recortes de imágenes de los automóviles en los fotogramas de los videos que fueron utilizados para crear el acervo, de esta forma ya obtuvimos las nuevas imágenes y los datos necesarios para poder entrenar la red YOLO. Si bien ahora los rectángulos son acordes con los automóviles , y el entrenamiento es mucho más rápido , después de 20000 iteraciones la red neuronal no tiene una buena performance, aunque es atribuible al magro acervo de datos.

Conclusión:

Debido a los diversos inconvenientes que se presentaron en esta investigación, los cuales fueron motivos ajenos a este grupo, y que golpearon con fuerza a la misma, la investigación no se pudo concluir o finalizar por completo, sin embargo los distintos puntos que se buscaban alcanzar, como la creación de un acervo de datos y la detección de patentes en automóviles no se encuentra muy lejana de llevarse a cabo por completo: El acervo de datos puede crecer rápidamente debido a las herramientas desarrolladas, y si fue posible detectar automóviles con la red YOLO , también será posible entrenar a la red neuronal para detectar patentes en los autos , tanto patentes nuevas como viejas. Probablemente esto implique un desafío de implementación, pero el objetivo probablemente pueda ser alcanzado. Otro punto a investigar es el comportamiento de redes neuronales similares para realizar la tarea de detección de objetos en tiempo real y poder compararla con YOLO , y por último resta investigar si la utilización de la red YOLO es útil en la detección de caracteres en tiempo real , o existe alguna alternativa para un problema que en teoría ya no es una desafío para el campo de la inteligencia artificial debido a las diversas redes neuronales que a lo largo de los años se desarrollaron para cumplir con este objetivo.

Anexo II

Desarrollo de un sistema de asistencia visual para un Manipuladorautomáticoprogramable(robotindustrial)

*Ing. Martín Ferreyra Biron, Ing. Alberto Miguens, Ing. Fernando SzklannyDepartamentode IngenieríaeInvestigacionesTecnológicas UniversidadNacional deLa Matanza*mferreyra@unlam.edu.ar,amiguens@unlam.edu.ar,fszklanny@unlam.edu.ar

Resumen

El presente trabajo hace referencia al desarrollo de unsistema de procesamiento digital de imágenes, basado encámarasdealtavelocidad,paraincorporarenunautómata programable, capaz de moverse articuladamenteen seis ejes, y destinado a aplicaciones industriales conexigenciadeeficiencia.

El autómata, que actualmente se encuentra funcionandoprácticamenteenun100%,estáformadoportreselementos: un manipulador (o "brazo"), un controlador yunconjuntodesensores.Elcontroladorcontieneelsistema procesador y las unidades de control de energía.El brazo es accionado por un servomecanismo y es capazdemoversearticuladamenteenseisejes.Lossensoresutilizados,conlafinalidaddemejorarelcontroldelautómatay la seguridaden el entorno de trabajo,soncámarasde altavelocidad yacelerómetros.

Elfundamentodelpresentetrabajoconsisteeneldesarrollodelsistemadeprocesamientodigitaldeimágenesqueda soportealautómataencuestión.

Introducción

Laaplicacióndelossistemasdigitalesdelprocesamientodeseñalesysuaplicaciónalcontroldemotores de inducción y autómatas programables tiene suorigen en la década de 1970, con los primeros desarrollosde procesadores digitales de señales DSP por Intel, AMI yBell Labs. Las dificultades en cuanto a la capacidad deprocesamientoy velocidad para realizar cálculos,hacenquesuaplicaciónselimiteacontrolesdebajacomplejidad.Noobstante,elgranavancetecnológicoproducido desde ese momento, tanto en la miniaturizaciónde los sistemas digitales, la capacidad de cálculo en losprocesadores de la época y su velocidad de procesamientohacen que se puedan aplicar a controles con algoritmos dealta complejidad.

El autómata propuesto es un brazo mecánico capaz demoversearticuladamenteenseisejes,deltipode"articulacióncoordinada",llamadoasíporlasemejanzade susmovimientosconlosdelcuerpohumano.Laprogramación de los movimientos del autómata se realizadesde una computadora personal mediante un software adesarrollar.

Los autómatas programables, tal como se mencionó enelapartadoanterior,seutilizanampliamenteenlaindustria automotriz. Pero existen numerosos sistemas deproducción que no cuentan con autómatas que se adapten asus necesidades, por lo que el trabajo que aquí se describepermiteproponersolucionesinnovadorasquecubrandichasnecesidades.

Enparticularsehaceindispensableeldesarrollodealgoritmos para el guiado de autómatas en base a la visión,loquepermitiráutilizarlosparadetectarymanipularobjetos de diferentes tamaños, color y contraste, mejorar laprecisión al realizar una trayectoria, y evitar objetos noprevistos en una tarea, haciendo al autómata adaptable adiferentescondiciones detrabajo.

En conclusión,con elfin de mejorarla productividadde las industrias, la calidad y el costo de fabricación paraganarcompetitividad anivel globalsehaceinteresantecontarconautómatasprogramablesenlosprocesosdefabricacióncon-sistemas devisión.

Marcoteórico

El procesamiento digital de imágenes ha tomado ungranimpulsoenlaúltimadécada,derivadodelaaltacapacidad de los procesadores que posibilitan el desarrollodesistemasmásversátilesydecostoreducido,ejemplificadosenelcontroldeprocesosrealizadosporrobots industriales, la navegación en vehículos autónomos,ladetecciónde eventoseinspecciónde objetosenprocesos de manufacturas y en un sin fin de aplicacionesmás.

Enlaactualidadexisteunagrancantidaddebibliografía sobre estudios y algoritmos desarrollados parael control de motores de inducción, sistemas de controlelectromagnéticos,vehículoseléctricos,procesamientodigitaldeimágenesyautómatasprogramablesconunavastaaplicaciónenlaindustria.Sinembargo,existennumerososistemasdeproducciónquenocuentancon

autómatas que se adapten a sus necesidades de procesoautomatizados para un aumento del nivel de producción ycalidad,menos aúnque utilicenel procesamientodigitaldeimágenesyaseaporsuelevadocostooporlainexistencia en el mercado. A partir de esta situación sedecidiódesarrollarunbrazorobotquepuedaserautomatizadoytomardecisionesensumovimiento,nutriéndosedelaentrada queproporcionancámaras devideo.

Para que el brazo pueda tomar decisiones necesitaráreconocerobjetos,comosemencionóanteriormente,yestoimplicalaadecuadaelecciónde unalgoritmoquecumplaconestefin.Almomentodeiniciarlainvestigaciónsobreestosalgoritmossecomenzaronarealizarpruebascon-SURF[1]yORB[2]peronoselograba una correcta detección en objetos que no tuvierantextura.Tampocosedeseabautilizarredesneuronalesconvolucionales [3] debido al costo de procesamiento y deentrenamientomásalládesubuendesempeñoenreconocer patrones y la atención [4], junto con las redesneuronales recurrentes no se encontraban en ese momentotan difundidas en el campo del procesamiento digital deimágenes.

Detodaslasopcionesestudiadaslaquemásseacercabaanuestrosrequerimientosfueelalgoritmo DOT [5] que fue el implementado y el cual se detalla en lossiguientespárrafos.

Implementaciónde DOT

ComopartedelproyectodesarrolladoseplanteóeldesarrolloimplementacióndeltrabajodeStefanHinterstoisser y otros, en el que se presenta un método(DOT) para detectar objetos tridimensionales entiemporeal, bajo una cierta cantidad de ruido, que no requiere unaetapa de entrenamientodemasiado costosa y que puededetectarobjetossintextura(problemaque,síseobservó,demaneraempírica,al utilizarSURFcomo semencionó).

DOT

DOTesunalgoritmodedetecciódeobjetospresentado en la conferencia de visión por computadora yreconocimientos de patrones del año 2010. Este algoritmotiene la capacidad de poder detectar los objetos de unamanera muy rápida y repetitiva para lo cual se utilizandistintasimágenesdelmismoobjeto,alasquesedenominapatrones(apartirdeahoraseutilizarálaterminologíaoriginal“templates”)delobjetoabuscar.Debido a la forma en que se generan estos “templates” y almétodo con el que se busca el objeto en la escena, estealgoritmoesrobustoenladetecciódeobjetossintexturas.DOTsepuede resumirentresgrandespasos:

- Creación de los“templates”.
- Preparaciónde la escena.
- Búsquedadelobjeto.

A continuación, y rápidamente se realizará una brevedescripcióndecadaunodelospasosanteriormentemencionados

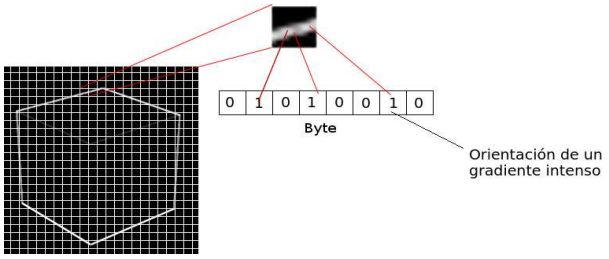


Figura 1:Ejemplodecomoserealizauntemplate

Figura 2 : Templates del envase de unadhesivovinílico

Creacióndetemplates

La primera etapa es la creación de los “templates” apartir de los objetos a buscar. La metodología propuestapor DOT es simple: obtener imágenes del objeto desdedistintas perspectivas. Mientras más imágenes distintas seobtenganparagenerarlos“templates”,máseficienteserála búsqueda, pero también más lenta. Estas imágenes seránlamateriaprimadelos“templates”ysebuscaránposteriormenteenlaescena.Unejemplodecómoserealiza untemplate sepuedeobservarenla figura1.

Posteriormente, a cada “template” se le aplica el filtroSobel [6] para así poder hallar los gradientes de la imagen.Laimagenresultante sedivide en cuadrados“pequeños”deNxNpixelyserecogerálaorientacióndelos7 primerosgradientes que tenganmás intensidad encadacuadrícula. Estos datos se guardarán en un byte siendo elMSB el bit que indique si en esa celda de la cuadrículahubo o no gradientes. Un ejemplo de templates se puedeobservarenlafigura2.

Preparacióndelaescena

Se denomina “escena” a la imagen en la que se buscaráelobjetoadetectar.Elprocesoessimilaracomoseprepara un “template”: los bordes de la escena se detectancon un filtro Sobel para así obtener los gradientes de esta,luegosedividelaimagenencuadradosdelamismamedida utilizada en los “templates” con la diferencia deque no se buscan los 7 gradientes más intensos por cadauno de los cuadrados, sino que se guarda la orientación delgradiente más intenso. En el caso de trabajar con videosesto se tiene que hacer fotograma a fotograma y la maneraen que se debe programar el algoritmo tiene que ser muyrápida para ofrecer una detección acorde a los tiempos delbrazorobot.

Búsqueda

Laetapadebúsquedaesconceptualmentemuysencilla.Setienenencuentalasorientacionesdelosgradientes más fuertes de cada cuadrado en el "template" yla orientación del gradiente más fuerte de cada cuadrículade la escena. Según la investigación DOT, se debe deslizarel"template"sobrelaescena,calculandocuántasorientaciones delmismocoincidenconlas orientacionesdelaescena.Enellugardondehubomayorescoincidencias es donde se supondría que se encuentra elobjeto adetectar.

Primeraimplementación

EnestainvestigaciónelalgoritmodeDOTfueimplementado en más de una ocasión y en diversas formasbuscandosiempreunaoptimizaciónenvelocidad.LaprimeraimplementacióndeDOTfuerealizadaenunacomputadoraconarquitecturax86sintenerencuentaningún tipo de optimización. Si bien los resultados fueronalentadores,desde elpunto de vista delcomportamiento

Figura 3: Detección de un envase de adhesivovinílico conunapequeñaoclusiónyunapiedrade desgaste.

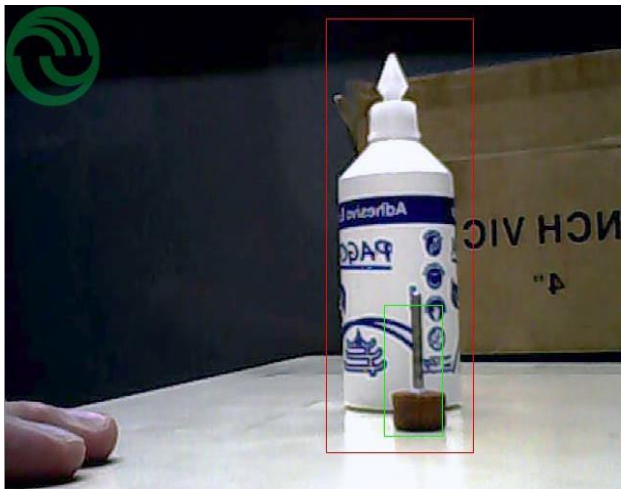


Figura 4: Dos elementos detectadosmientrasunoocluyealotro

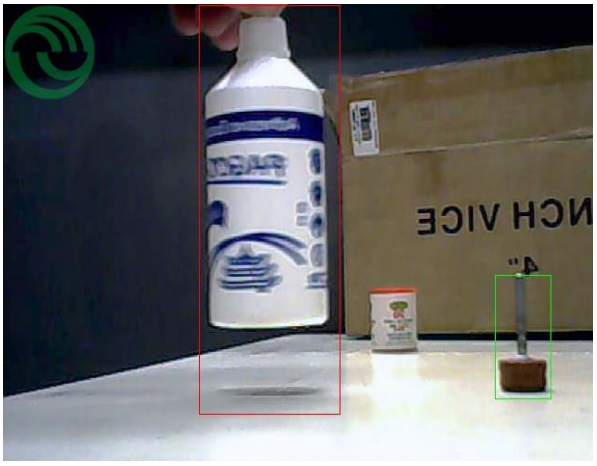


Figura5:Detecciónenmovimiento

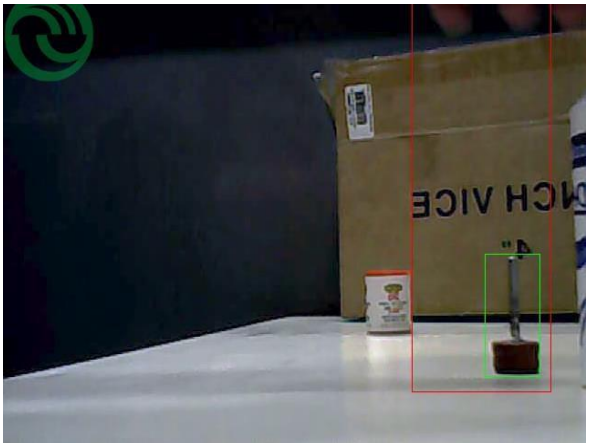


Figura6: Errorenladetección

delalgoritmo,fuenecesarioprogramarelmismonuevamente conmuchomáscuidado.

Enlasegundaimplementación,también programadaparaunaarquitecturax86,setuvoencuentalaoptimización del código y se trató de implementar, comorecomiendanlosautoresdeDOT[7],lasinstruccionesMMX. Con todos estos cambios, se obtuvo un aumento enlaperformancedelalgoritmo, pero lamismanofue lasuficienteparapoderutilizarloenunbrazorobot.Sinembargo, se pudo programar el algoritmo de una formamucho máseficientecomose detallaráacontinuación.

En la figura 3 se puede observar la detección de unenvase de adhesivo vinílico y una piedra de desgaste. En elcasodelenvasesepuededetectarconciertoniveldeoclusiónsinmayoresinconvenientes.Enelcasodelafigura4sepuedeobservarcomoelalgoritmopuededetectarunobjetoocluyendoaotro,siendoambosdetectadoscorrectamente.

En la figura 5 se puede observar cómo se detecta elenvase de adhesivo vinílico en movimiento. En la figura 6el algoritmo programado falla al no encontrar el objeto.Este esunodelos problemasaresolver.

ImplementacióndeDOT enGPU

Conelobjetivodeavanzareneldesarrollodealgoritmos para la detección y procesamiento de imágenes,se procedió a la incorporación al sistema de una placa dedesarrollo Nvidia Jetson TK1. La placa en cuestión estáconstruidasobrelabasedeunprocesadorARM®Cortex™-A15 y una unidad destinada alprocesamientográfico de NVIDIA Kepler GPU con 192 núcleos CUDA.Cabe mencionar que CUDA es una arquitectura de cálculoparalelo de NVIDIA que aprovecha la potencia de la GPU(unidad de procesamiento gráfico) para proporcionar unincremento delrendimientodelsistema.

En un primer intento se compiló el código realizadoanteriormente con mínimas modificaciones para ejecutarloen la placa de desarrollo. Las consideraciones iniciales,basadasenquegranpartedelcódigohaceusodelabiblioteca OpenCV, [8] y en que varios métodos utilizadosestánoptimizadosparalaejecuciónenunaGPU,prometían o sugerían un buen rendimiento. Pero estonofue así. La detección en un video de prueba superaba pormuchoeltiempodeprocesamientoaldesuejecuciónmediante elanálisisenunaCPU.

Visto y considerando este inconveniente se comenzó aanalizar el código con detenimiento.Se observó que lamanera en la que se estaba intentando utilizar la GPU noera la correcta, por lo que se examinó cada uno de losmétodos utilizados para identificar aquellos que producíanlamayor demora yoptimizarlos. A partir de esteanálisisse llegó a la conclusión de que el cálculo de gradientes eraelprimeroenquemásdemorabaenejecutarse.Estemétodo se debeejecutarfotogramaa fotograma.

En el cálculo de gradientes las tareas a realizar son lassiguientesycada unade ellasdependede la anterior:

1. Seconvierteaescaladegriseslaimagendelfotograma aanalizar.
2. Se calculaelfiltroSobelporXeY.
3. Secalculaelgradienteapartirdelosfiltrosanteriores.
4. Sehallaelmáximogradienteencuadradosde NxNpixelesentodala imagen.
5. Se calcula el ángulo de dicho gradiente guardandosu orientaciónenunbyte.
6. Sealmacena elresultado.

De toda la lista anteriormente mencionada, el primerlugar en donde se atacó la demora fue en el cálculo delfiltro Sobel. Se utilizó la función optimizada para CUDAde OpenCV de manera correcta y debido a que se tienenque hallar los componentes en X y en Y, estas tareas seejecutaron en paralelo. Sin embargo, si bien se obtuvo unamejorasustancial,lacomprensiónyelanálisisdelostiempos globales de demora llevó a los responsables de lainvestigaciónalaconclusióndequelamejormanerade

implementarelalgoritmoDOTnoerautilizandolosmétodos optimizados ofrecidos por OpenCV, sino a travésdeldesarrollodeunkernel.Estofueproductodedoscircunstancias, en un principio, desconocidas al momentode iniciarlaspruebas:

- Eltiempodetransmisióndedatosdesdelamemoriaprincipala la memoriade la GPU (y viceversa)noesdespreciableydemanerainherentegeneraunasobrecargaenlaejecucióndelalgoritmo(*"overhead")*.Porlotanto,utilizarunaGPUsintenerencuentaloanteriorproducedemorasylasmismasdebenserminimizadas.
- Se debe buscar en la imagen máximos locales enregionesdeinterésdeNxNpixeles.Enprimerlugar,OpenCVnoofreceunafunciónoptimizadaqueutiliceCUDA para realizar esta tarea y en el caso de ofrecerla sedebería tener en cuenta el tiempo de transferencia entre lamemoria principalylamemoriadelaGPU.

Por todo esto se procedió al diseño de un kernel quepudiera acelerary haceruso correctode la GPU y asíhallarlosgradientesdemanerarápidaycorrecta.Elenfoque utilizado para resolver el problema fue dividir ysolucionarelmismoenetapasparciales.

Lo primero que se realizó fue el desarrollo de un kernelquepermitieraaplicarelfiltroSobelaunaimagenutilizando la GPU. Este paso fue fundamental ya que en unkernel se solucionaban los puntos 2 y 3 de la lista de tareasmencionadaanteriormente.

Seguidamente se ensayó el comportamiento del kernelsobre un video y los resultados fueron muy alentadores, enun tiempo mucho menor de lo esperado la aplicación delfiltro era calculado. Además, se comprobó también que laplaca aceptara el uso de una webcam la cual se utilizó conel algoritmo y funcionó de manera correcta. Por lo tanto, lalistade tareasque hasta ahora se resumeesla siguiente:

1. Seconvierteaescaladegriseslaimagendelfotograma aanalizar.
2. SecalculanelfiltroSobelporXeYyelgradiente (GPU).
3. Sehallaelmáximogradienteencuadradosde NxNpixelesentodala imagen.
4. Se calcula el ángulo de dicho gradiente guardandosu orientaciónenunbyte
5. Sealmacena elresultado.

El paso siguiente consistió en la búsqueda del máximogradiente en cuadrados de NxN pixelesenlaGPU.

Laprimeraaproximaciónparasolucionaresteproblema fue programar en un kernel distinto la búsquedade máximos locales utilizando la GPU en bloques de NxNpixeles. De esta manera se trabajó la imagen resultante delfiltroanteriorencuadrados-NxN,pudiendosehallarparalelamentelosmáximosdecadabloque.Estofuecorrecto pero el hecho de utilizar un único hilo para buscarelmáximogradientedesperdiciabaelpoderdeprocesamientodelaGPU.Es porestoquedespués deinvestigaryanalizarelproblemasereprogramólasolución utilizando el concepto de reducciones paralelas,en estecaso,enmatrices.

La idea de las reducciones paralelas es sencilla: Parahallar un máximo se deben utilizar comparaciones. Si seutilizaun solonúcleo oun solo hilo, eseúnicohilo onúcleo debe realizar todas las comparaciones. Pero al tenerunaGPUy poder decidir cuantos hilos seejecutanporcada bloque, esta situación cambia. Como la operación decomparación es binaria, la máxima cantidad de hilos quepueden utilizarse para solucionar el problema de maneraóptima, es la mitad de la cantidad de elementos (si ésta espar).

Por ejemplo, si se trata de hallar el máximo en unamatrizde8x8elementosexistirán64elementosacomparar y si se utilizan 32 hilos se podrán comparar 32pares de elementos de una sola vez, guardando el resultadoenunadelasmitades.Siestaoperaciónsecontinúarealizando(conloselementos quesobrancomparar)laeficiencia que seobtiene esrealmenteconsiderable.

Lafigura7ejemplificadeunamaneraclaraelconceptodereducciónque seprogramó.

Aposterioridehabersehalladoelmáximo,enelmismo kernel se programó el cálculo para transformar yexpresar su orientación medida en un byte, tal como loproponen los autores del algoritmo DOT. Por lo tanto, lalistade tareasque se resume hasta ahoraeslassiguiente:

1. Seconvierteaescaladegriseslaimagendelfotograma aanalizar.
2. SecalculanelfiltroSobelporXeY,yelgradiente (GPU)
3. SehallaelmáximogradienteencuadradosdeNxN pixeles en toda la imagen, se calcula el ángulo dedicho gradiente guardando su orientación en un byte y sealmacena elresultado(GPU).

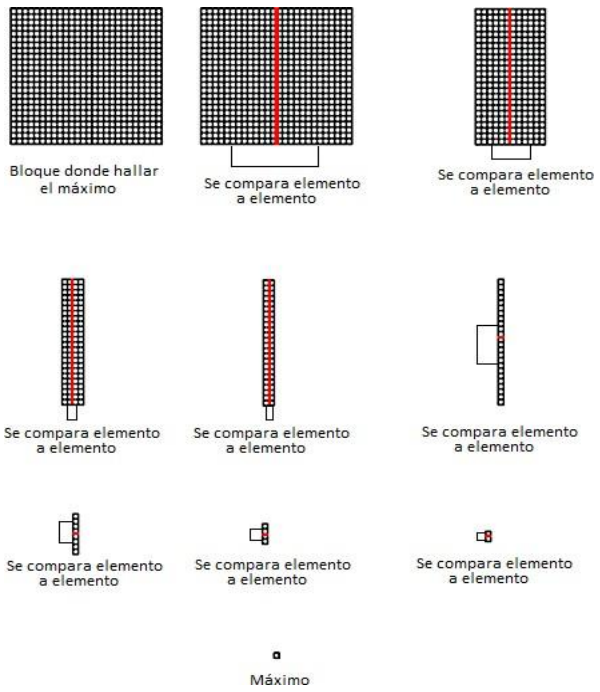


Figura7:Ejemplificacióndelareduccióndeunamatrizutilizandounagpu

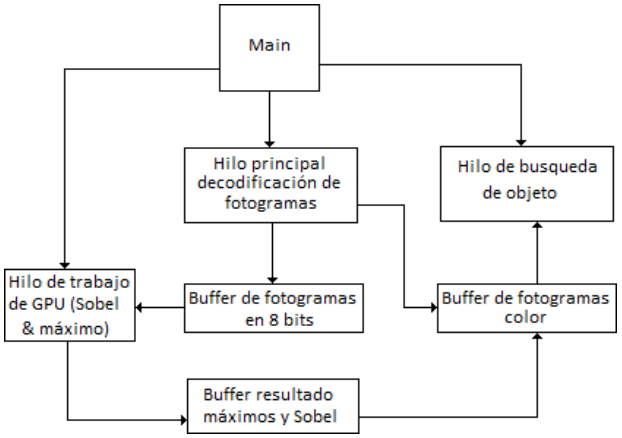


Figura 8:Esquemadeejecucióndelalgoritmo

Porúltimo,debidoaqueexistendemorasentrelaejecución de ambos kernels, estos se combinaron en unosolo.Todasestasmejorasdejanasíunaumentodevelocidadconsiderablerespectoalasmismastareasejecutadas en una CPU o utilizando en parte la GPU, atravésdeOpenCV.

Ya solucionado elproblema generado porla demoradel método que calcula los gradientes, se siguió analizandoel programa en búsqueda de otras demoras. Un punto claveque producía las mismas, fue la decodificación del métododecompresiónutilizadoenlosvideosoenlamismacámara de la que se obtienen los fotogramas. Debido a quelaúnicamaneradeaumentarlavelocidaddeestadecodificaciónesutilizarlaGPU,ylatransferenciaentrela memoria principal y la de la GPU produce demoras, sedecidió decodificarlos fotogramasenunhilo.

Sedebe tener encuenta que laplacaNvidia JetsonTK1 posee un microcontrolador ARM de cuatro núcleos yutilizar uno en particular para decodificar los fotogramasno produceningúntipodeproblemas.

Valiéndosedeunaarquitecturadeproductorconsumidor,enelqueenunhilosedecodificanlosfotogramas (productor) y en otro hilo se ejecuta el kerneldondesebuscanlosgradientesylosmáximos(consumidor),seaumentóaúnmáslaeficienciadeejecución del algoritmo DOT. El esquema de la figura 8muestracómosecomunicanyejecutanlosdiversosalgoritmos.

Solamenterestaoptimizarlabúsquedadelobjetocon laGPU denominado“Match Template”.

Optimización delabúsqueda

Enprimerlugar,seentiendequenosepuedesobrecargar más a la GPU en este punto, por lo tanto, labúsquedadelobjetosedeberealizarenlaCPU.Elalgoritmo DOT plantea que lo que se debe realizar paraencontrar un objeto es buscar la coincidencia más probableentrelaescenaylosobjetosqueseaprendieron.Parapoder hacer esto se necesita utilizar una ventana deslizante(querecorretodalaescena)ybuscarcoincidenciasdeángulos de los objetos aprendidos. El ganador o la regióncandidata a ser el objeto en cuestión es aquella en la queexistanmáscoincidenciasdeorientacionesentrelaplantilla del objeto aprendido y la escena. En una versiónsimplificada delcódigoseveríaasí:

```
Por cada plantilla de objeto aprendi-
    doPor cada pixel en Y de la esce-
    naPorcada pixel en Xde la escena
                Por cada pixel en Y de la plantilla-
                 Por cadapixelenXdelaplantilla
  Match+=Escena(x,y)&plantilla(x,y)
```

Es evidente que la complejidad computacional es alta ydealgunamanerahabríaqueminimizarla.Después deanalizarlaformaenquefuncionabaelalgoritmo,sellegóa lasiguienteconclusión:

Sepodríaaumentarlaeficienciadelalgoritmosisolamentesecomprobarala coincidenciadeángulosenloslugaresdondeenlaplantillaesténdefinidos,deesamaneraseevitaríaunaiteración"*for*"y,porende,variasoperaciones en la búsqueda. Además se notó,experimentalmente,queesmuchomásrápidohacerlabúsquedacadadospixelesenlaescenaqueunoporuno(en este caso se estaría bajando la resolución de búsqueda).Paralograrestosemodificóel método queutilizaelprogramadeaprendizajeparaanalizaryguardarlosobjetosaprendidos,ahoraenvezdeguardartodalaplantillaseguardasolamente elvalordel/losángulosylaposiciónenlamisma.Enresumen,unaversión simplificadadelcódigose veríaasí

```
Por cada plantilla del objeto aprendi-
    doPorcadapixelparenYdelaescena
        Porcada pixelparenXdelaescena
            Por cada ángulo valido en la planti-
                llaIf(Escena(x,y)esunángulodefinido)
                  Match+=Escena(x,y)&plantilla(x,y)
```

Sibienelcambionoesdrástico,esevidentequeexisten mejoras.

Cambiosdrásticosen losobjetos.

El algoritmo DOT no es perfecto y en las pruebas seobservómásdeunavezquequizásporfaltadeplantillasde objetos la detección no se realizaba en el lugar correcto.Para morigerar este comportamiento se puede suavizar dealguna forma la detección brusca de objetos, teniendo encuentaquelosmismosnopuedenaparecenenotrolugaren la escena de una forma inmediata, sino que necesitantrasladarse por la misma. Para solucionar este problema seimplementó la siguiente solución: Si se detectó un objetoenunaposiciónquesuperaumbraldedetección,yahabiéndose detectado el mismo, la ventana de detección setrasladará de manera progresiva al lugar donde se suponeque seencuentraelobjeto.

Pruebasen ambientereal

Acontinuación,sepresentanalgunasimágenesdemomentosenlosqueelalgoritmoprodujoresultadosapropiados, en videos que se tomaron en ambiente real conruido de fondo. Es importante recalcar que fueron algunosmomentosdebidoaquenoseutilizarondemasiadasplantillas para realizar la detección. Las imágenes de lasfiguras9a16songrisesdebidoaqueseconsumedemasiadamemoriasiseutilizanlostrescanalesdecolores. El exceso de consumo de memoria genera, comoconsecuencia, que el sistema operativo termine de maneraabrupta elprocesodedetección.

Fig. 9. Algoritmo reconociendo el envase depegamento

Fig. 10. Algoritmo presentando una falla en ladetección

Fig.11. Algoritmodetectandoun cubocorrectamente

Fig. 12. Algoritmo fallando por falta de plantillas. Enestevideoenparticularlatasade fallasfuemuyalta. El algoritmo malinterpretaba el ruido de fondo(rompecabezas)

Fig. 13. Tomando un video donde se reconocía el cuboen la implementación en CPU se trató de reconocerlo.En rojo la detección de esta implementación y en azulla de la anterior. Cabe destacar que se utilizarondiferentesplantillas

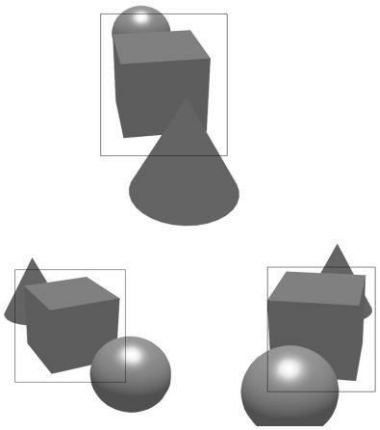


Fig. 14. Detecciones correctas en el video de pruebaqueseutilizó.Observarladetección conoclusiones

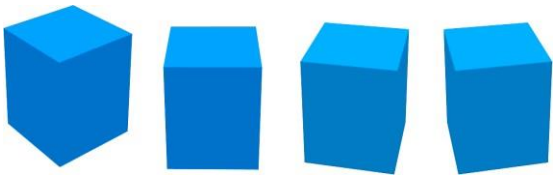


Fig.15.Plantillasqueseutilizaronparaladetecciónde la imagen anterior. El video de prueba no tuvo falsospositivos.

Detecciónde profundidad

Ladetecciónde un objeto a partir de la implementación de DOT permite encontrar el mismo en el plano, pero no en el espacio, siendo parte fundamental ladetecciónde esta dimensión debido a la naturaleza posicional del brazo robot. Es por esto que se necesita realizar la medición de la profundidad por medio de un sistema estereoscópico. La posibilidad que se evaluó es la de ubicar dos cámaras en un mismo plano *M* (con coordenadas *x* e *y*) separadas por una distancia *b* en el eje *x*, y con sus ejes focales paralelos y perpendiculares al plano *M*. De esta forma se lograrían dos imágenes disparesa nivel horizontal, y el producto de esa disparidad se utilizaría para medir la profundidad.

A nivel matemático[9], este mecanismo puede ser resuelto a través del método de triangulación. En la figura16 se puede observar un esquema básico, en el cual se muestran dos sistemas de ejes cartesianos cada uno correspondiente a cada cámara del par estereoscópico.

A una profundidad dada por la distancia focal de las cámaras *f* (que debe ser la misma en ambas) se encuentra el plano denominado epipolar que contiene los planos I_I e I_D correspondientes a las imágenes tomadas por cada cámara. En este plano es donde se miden las disparidades entre dichas imágenes.

Al encontrarse las cámaras a la misma altura (eje y), las únicas diferencias se dan a nivel horizontal ya que, como se comentó, las cámaras se ubican a una distancia de separación *b* sobre el eje *x*.

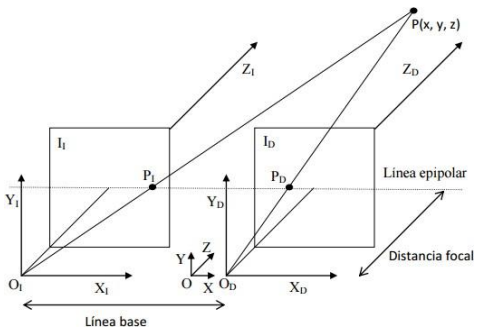


Figura 16 :Diagrama tridimensionalparael cálculodeladistanciadelpuntoP

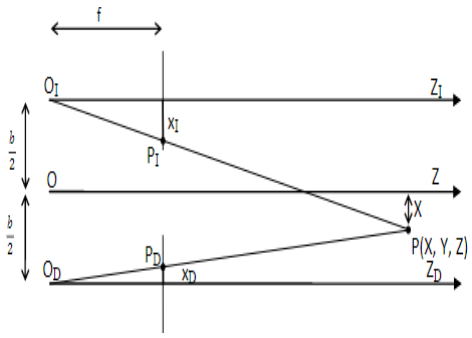


Figura 17:Diagramabidimensionalparaelcálculodela distanciadelpuntoP

En la figura 17, se pueden observar los puntos PI y PDquesonlasproyeccionesdelpuntoPsobrelalíneaepipolar.Sepuedeintuirquecuantomáslejanoseencuentreelpuntop, losvectoresdeproyecciónténderána ser paralelos al eje Z. Lo que hará PI y PD tiendan aubicarse sobreelejeZIyZD.

La disparidad del punto se calcula como XI - XD = d yen el caso de que un objeto se encuentre ubicado en elinfinitodelejeZharáquedvalga0.Asíentonceselcálculo dela distanciadelobjetosobre ejeZserá

$$ImagenIzquierda: \frac{\frac{b}{2}+x}{z} = \frac{x_I}{f} \Rightarrow x_I = \frac{(x+)\frac{b}{2}}{z}$$

$$ImagenDerecha: \frac{\frac{b}{2}-x}{z} = \frac{x_D}{f} \Rightarrow x_D = \frac{(x-)\frac{b}{2}f}{z}$$

En la ecuación*Imagen Derecha*se asigna un signonegativoaXDparacompatibilizarlossistemasdecoordenadas de los ejes OI y OD. Siendo la disparidad delpunto P: x_I-x_D= d reemplazando y despejando se llega a laecuación:

$$z = \frac{fb}{d}$$

Con la cual se puede calcular la distancia z del punto P(objeto detectado)

Porotraparte,eldesarrollodelosalgoritmosdecontroldelautómataseapoyóenlasbibliotecasdesarrolladasporPeterCorke,realizándosesimulaciónmediante elsoftwareMATLAB.

Queda pendiente la integración con las bibliotecas quese desarrollaron para la detección de objetos, debido a queseprioriz óeldesarrollodelasbibliotecasdeprocesamientográfico.

Calibracióndelascámaras

Loidealesque,enunprincipio,elajustefueserealizado a grandes rasgos lo más precisamente posible.para que luego el algoritmo de calibración tuviese mejoresresultados.

Unavezhechoesteajuste,secomenzóconelcálculode las matrices de calibración, para lo que se utilizó labiblioteca de OpenCV que contiene clases para calibracióny reconstrucción de imágenes 3D: “Camera calibration and3D reconstruction”. Para la calibración se usó la aplicación*calibrate_cameras*,queformapartedelabibliotecaStereoVision(Bajolicencia públicaGNU)

Estaaplicacióntomacomoargumentosimágenesestéreo que contienen en su escena un patrón cuadriculadosímila un tablero deajedrez, concapturas en distintasperspectivasdeeste(figura18).Eltableroserotayademás se lo traslada a lo largo de las múltiples tomas deimágenes.Matemáticamentesepuededemostrarquelacantidadmínimadetomasdependedelacantidaddeesquinasquecontieneeltableropatrón.Eltérmino“esquinas”hacereferenciaalacantidaddevértices decuadrados negros que hacen contacto entre sí (léase negroso blancos). La máxima cantidad utilizada es 50, bajo elcriteriodecuantasmásmuestrashayaloscálculosdecorrecciónderrores serán másprecisos.

Laaplicación*Calibrate_cameras*dacomoresultadounacarpetaconunconjuntodematricesquecontienelos parámetros intrínsecos (focos, centrados), extrínsecos (traslaciónyrotación)ylasmatricesderectificaciónde distorsiones. Con estas matrices se realizan lascorreccionessobrelasimágenestomadasencadacámara.

Unavezrectificadaslasimágenesepuederealizarel cálculodeleje Zcon lafórmulacitadamásarriba.

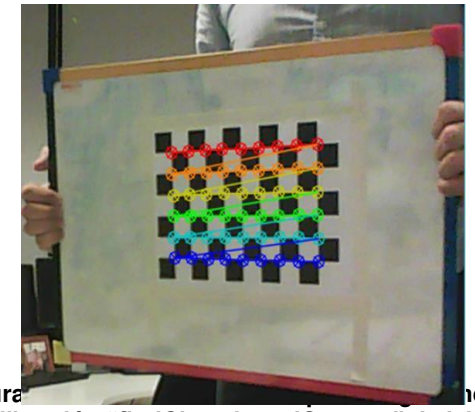


Figura [illegible]no decalibración “findChessboardCorners” de labiblioteca OpenCV

Paralograrestecálculohayqueubicarenambasimágenes del par estéreo el mismo objeto, para esto esnecesarioun algoritmo de búsqueda e identificacióndeobjetos. Esto se puede resolver con el algoritmo DOT quefue detalladoanteriormente.

Verificacióncon mapadedisparidad

Laverificaciónconmapadedisparidadserealizóutilizandolasbibliotecasde*StereoVision*,unapruebaempírica realizando un mapa de disparidad y una nube depuntos.Lanubedepuntosconsisteenconseguirlascoordenadas espaciales de cada pixel de la escena tomadaspor las cámaras. De esta forma se puede visualizar unaescena desde diferentes perspectivas rotando los ejes decoordenadas.Estotieneunlímiteyesqueesos"puntos"delaimagenseobtienendesdeunaposiciónfijadelespacio,si se deseara observarperspectivas laterales senecesitaría más información de la escena. Esto se puedesalvarhastaciertoángulo,replicandolospíxelesdelafrontera, de forma que cubran los huecos de informaciónfaltante.

Para lograr un buen mapa de disparidad se necesita quelos objetos de la escena sean detectados inequívocamenteen las imágenes del par estéreo. Se utilizó la herramienta*tune_blockmatcher* de la biblioteca *StereoVision* con el findeajustarlosparámetrosdeldetectordeobjetosyseverificóen paralelo la "calidad" delmapa de disparidadque semuestraenla figura19.

Los ajustes mencionados se deben realizar en formaiterativayempíricayaqueresultaserelmétodomásrápido y eficiente. Luego de realizar el ajuste del algoritmode búsqueda y detección de objetos, se utilizó la función*images_to_pointcloud*(tambiéndelacitadabibliotecaStereoVision), la que toma como argumentos las matricesderectificación,unpardeimágenesestéreo,losparámetros de ajuste del algoritmo de "*blockmatching*" yesto da como resultado una matriz de tres dimensiones quecontienelosvectores(tambiéndetresdimensiones)correspondientes a la información del color de cada pixel.Como resultado se puede observar la siguiente nube depuntos Figuras 20y21)

Conclusión

Quizás las modificaciones y los agregados realizadossobre el algoritmo DOT no sean suficientes para obtenervelocidadesconsiderablesenladetección,aunquepermitenmejoraradecuadamentelamisma.Ahoraesaceptablerealizarunabúsquedaenunaimagen HD,pero aun así sigue siendo relativamente lento. Como se planteaalfinaldelapartado"ImplementaciónDOTenGPU"quizáslarespuestainvolucrelaimplementacióndelproceso de "match template" en la GPU, aprovechando eltiempoociosodelamisma.

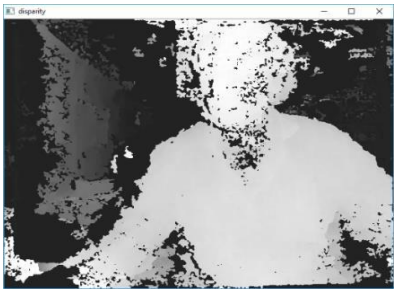


Figura 19:Mapadedisparidad

Figura20:Unpardeimágenes (imagen estéreo)

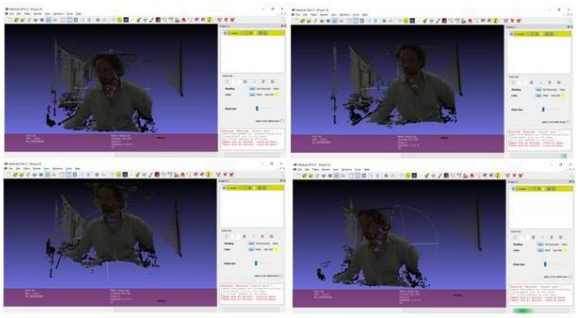


Figura 21: resultado de la nube de puntos, apartirdelaimagenestéreo.

Respecto al algoritmo, presenta fallas que quizás seanmejorables.DOTnoconsideracolores,soloconsideraformas, y queda claro que resolver esa falencia sería unamejora a esta implementación. Otro punto importante adestacar es que el tiempo de reconocimiento de un objetodepende de la cantidad de plantillas y de la dimensión delvideo. Esto no debería ser así, aunque es una característicaintrínseca delalgoritmo.

Una solución que se podría proponer para la mejora delalgoritmo es la de no utilizarla ventana deslizante,sinoqueporalgúnmediosedetectaranlosposibles candidatosalaimagendelaplantillabuscadayluegorealizarel"match template" entre las plantillas y esa región de laimagen.

Respectoalcálculoyanálisisdeprofundidad,lacalibración de las cámaras y la comprensión del algoritmoestánenunnivelmuyavanzado,alpuntodequesolamenteseríanecesarioprogramarladetecciónyagregarla a la de objetos. El único inconveniente será eltiempo de procesamiento que se sumaría al que posee ladetección DOT y, como se expresó, se debería buscar unasoluciónde compromisooanalizarlauniónde ambosalgoritmosmuchomásdetenidamente.

En etapas siguientes de este proyecto de investigación,quesiguevigenteenelámbitodelaUniversidad,seprocederá a continuar el trabajo sobe aquellas cuestionesquehoysemencionancomodificultadesoasuntospendientes.

Referencias

[1] Bay,Herbert,Tuytelaars,Tinne,VanGool,Luc.:SURF:Speeded UpRobustFeatures, EuropeanConferenceonComputerVision 2006

[2] Rublee,Rabaud,Konolige,Bradski:ORB:anefficientalternative to SIFT or SURF, IEEE International Conference onComputerVision 2011

[3] LeCun,Haffner,Bottou,Bengio,ObjectRecognitionwithGradient-BasedLearning,CPVR2004

[4] Vaswani,Shazeer,Parmar,Uszkoreit,Jones,Gomez,Kaiser,Polosukhin,Attentionisallyouneed,CornellUniversity2017

[5] HinterstoisserStefan,LepetitVincent,IlicSlobodan,FuaPascal, Navab Nassir.: Dominant Orientation Templates for Real-TimeDetectionofTexture-LessObjects,ComputerVisionPatternRecognition2010.

[6] Gonzales,Woods:DigitalImageProcessing,TerceraEdición,PrenticeHall2007

[7] Hinterstoisser Stefan, Cagniart Cedric, Ilic Slobodan, Sturm-Peter,NavabNassir,FuaPascal,LepetitVincent.:GradientResponseMapsforReal-TimeDetectionofTexture-LessObjects,IEEETransactionsonpatternanalysisandmachineintelligence2011

[8] OpenCVhttp://opencv.org/

[9] MartinMontalvo,Técnicasdevisiónestereoscópicaparadeterminarlaestructuratridimensionaldelaescena,Autor:Cursoacadémico2009/2010,MásterenInvestigaciónenInformática, Facultad de Informática, Universidad Complutense-deMadrid 2009/2010.

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

Artículo original

DESARROLLO DE UN SISTEMA DE ASISTENCIA VISUAL PARA UN MANIPULADOR AUTOMÁTICO PROGRAMABLE (ROBOT INDUSTRIAL)

VISUAL ASSISTANCE SYSTEM FOR A PROGRAMMABLE AUTOMATIC HANDLER (INDUSTRIAL ROBOT)

Ing. Martín Ferreyra Biron, Ing. Alberto Miguens, Ing. Fernando Szklanny

Departamento de Ingeniería e Investigaciones Tecnológicas, Universidad Nacional de La Matanza
mferreyra@unlam.edu.ar , amiguens@unlam.edu.ar , fszklanny@unlam.edu.ar

DESEA PUBLICAR SU E-MAIL (tachar lo que no corresponda): (NO)

Resumen:

En el presente trabajo se describe el desarrollo de un sistema de procesamiento digital de imágenes y la adaptación de un algoritmo detector de objetos, utilizando cámaras de alta velocidad con el objetivo final de incorporar dicho sistema a un autómata programable, apto para moverse con seis grados de libertad de forma articulada y con el fin de ser utilizado en aplicaciones industriales con ciertas exigencias de eficiencia.El autómata, que se encuentra funcionando casi en su totalidad, está conformado por tres elementos: un manipulador, un controlador y un conjunto de sensores.

El controlador es el encargado de procesar la información para mover el manipulador y las unidades de control de energía.Por otra parte, el brazo está comandado por un servomecanismo que permite mover al mismo articuladamente en seis ejes, y por último los sensores utilizados, con la finalidad de mejorar el control del autómata y la seguridad en el entorno de trabajo, son cámaras de alta velocidad y acelerómetros.

El objetivo principal del presenteconsiste en el desarrollo del sistema de visión que dará soporte al autómata descripto.

Abstract:

This document describes the development of a digital image processing system and an object detector adaptationusing high-speed cameras, with the final goal of being included into a programmable automat, suitable to move in six degrees of freedom, and oriented to industrial applications with specific efficiency requirements. This robot, currently working almost entirely, includes three elements: a manipulating system (the arm), a controller and a quantity of sensors.

The controller is the device that process the movement information and the energy control units. The arm is moved by a servomechanism and is capable of moving in six axis and, finally, the sensors required, in order to get a better control of the robot and to ensure safety in the work environment, are high-speed cameras and accelerometers.

In other terms, the main objective of the current work consists in the development of a visual system that will support the abovementioned robot.

Palabras Clave:*Visión, robot industrial, velocidad, seguridad.*

Key Words:*Vision, Industrial robot, speed, safety.*

I. INTRODUCCIÓN

La aplicación de los sistemas digitales de procesamiento de señales y su aplicación al control de motores de inducción y autómatas programables tiene su origen en la década de 1970, con los primeros desarrollos de procesadores digitales de señales DSP por Intel, AMI y Bell Labs. Las dificultades en cuanto a la capacidad de procesamiento y velocidad hicieron que su aplicación se limite a controles de baja complejidad. Sin embargo,el avance tecnológico producido desde ese momento hasta hoy,hace factible aplicar estos sistemas digitales de procesamiento a controles con algoritmos de alta complejidad.

El autómata propuesto es un brazo mecánico capaz de moverse articuladamente en seis ejes, del tipo de "articulación coordinada", llamado así por la semejanza de sus movimientos con los del cuerpo humano. La programación de los movimientos del autómata se realiza desde una computadora personal mediante un software a desarrollar. Los autómatas programables se utilizan ampliamente en la industria automotriz, aunque existen numerosos sistemas de producción que no cuentan con autómatas que se adapten a sus necesidades, por lo que el trabajo que aquí se describe permite proponer soluciones innovadoras que cubran dichas necesidades. En particular se hace indispensable el desarrollo de algoritmos para el guiado de autómatas en base a la visión, lo que permitirá utilizarlos para detectar y manipular objetosde diversas índoles, mejorar la precisión al realizar una trayectoria, y evitar objetos no previstos en una tarea, haciendo al autómata adaptable a diferentes condiciones de trabajo.

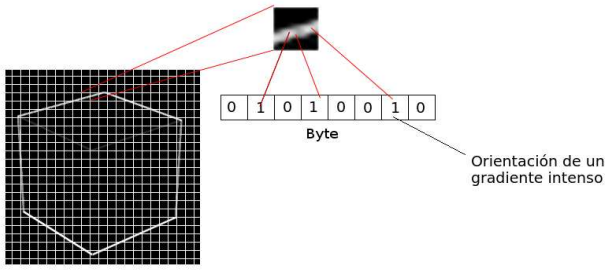


Fig.1. Ejemplo de cómo se realiza un “template”

En conclusión, con el fin de mejorar la productividad de las industrias, la calidad y el costo de

fabricación para ganar competitividad a nivel global, se hace interesante contar con autómatas programables en los procesos de fabricación con sistemas de visión.

II. MARCO TEÓRICO

Elprocesamiento digital de imágenes ha tomado un gran impulso en la última década y si bien en la actualidad existe una gran cantidad de bibliografía sobre estudios y algoritmos desarrollados para el control de motores de inducción, sistemas de control electromagnéticos, vehículos eléctricos, procesamiento digital de imágenes, entre otros, no abundan sistemas de producción que cuenten con autómatas que se adapten a sus necesidades respecto a procesos automatizadosorientado a un aumento del nivel de producción y calidad ymenos aún que utilicen el procesamiento digital de imágenes.A partir de esta situación se decidió desarrollar un brazo robot que pueda ser automatizado y tomar decisiones en sus movi-

mientos, valiéndose de la entrada que proporcionan cámaras de video. Para que el brazo pueda tomar decisiones necesitará reconocer objetos y esto implica la adecuada elección de un algoritmo que cumpla con tal fin. Al inicio de la investigación sobre estos algoritmos se hizo un estudio empírico sobre SURF[1] y ORB[2] los cuales forman parte de conocidos detectoresaunque no se lograba una correcta detección en objetos que no tuvieran textura. Tampoco se deseaba utilizar redes neuronales convolucionales [3] debido al costo de procesamiento y de entrenamiento más allá de su buen desempeño en reconocer patrones y la atención [4], junto con las redes neuronales recurrentes no se encontraban en ese momento tan difundidas en el campo del procesamiento digital de imágenes, por lo tanto de todas las opciones estudiadas la que más se acercaba a nuestros requerimientos fue el algoritmo DOT [5] que fue el implementado y el cual se detalla en los siguientes párrafos.

III.IMPLEMENTACIÓN DE DOT

Como se comentó en párrafos anteriores,partedelproyecto implica el desarrollo e implementación de DOT, algoritmo del cual daremos una breve reseña.

A. DOT

DOT es un algoritmo de detección de objetos presentado en la CVPR 2010el cual tiene la capacidad de poder detectar objetos con o sin textura de una manera rápida y repetitiva valiéndose de distintas imágenes procesadas delobjeto a buscar, a las que se denomina patrones (a partir de ahora se utilizará la terminología original "*templates*"). DOT se puede resumir en tres grandes pasos:

- Creación de los "templates".
- Preparación de la escena.
- Búsqueda del objeto.

B. CREACIÓN DE TEMPLATES

La primera etapa es la creación de los "templates" a partir de los objetos a buscar, siendo la metodología propuesta por DOT simple: obtener imágenes del objeto desde distintas perspectivas para luego procesarlas y obtener finalmente los "templates". Mientras más "templates" distintos se obtengan, más eficiente será la búsqueda, pero también más lenta. Estas imágenes serán la materia prima de los "templates" y se buscarán posteriormente en la escena. Luego, a cada imagen del objeto obtenida se le aplica el filtro Sobel [6] para así poder hallar los gradientes de la imagen. La imagen resultante se divide en "pequeños" cuadrados de N x N pixel y se recogerá la orientación de los primeros 7gradientes más intensos en cada cuadrícula. Estos datos se guardarán en un byte siendo el MSB el bit que indique si en esa celda de la cuadrícula hubo o no gradientes.

C. PREPARACIÓN DE ESCENA

Se denomina "escena" a la imagen en la que se buscará el objeto a detectar. El proceso es similar a como se prepara un "template": los bordes de la escena son detectados con el filtro Sobel para así obtener los gradientes, luego se divide la imagen en cuadrados de la misma medida utilizada en los "templates" con la diferencia de que no se buscarán los 7 gradientes más intensospor cada cuadricula, sino que se reservará únicamente la orientación del gradiente más intenso.

D. BÚSQUEDA

Finalmente, la etapa de búsqueda, conceptualmente sencilla, consta en "deslizar" sobre la escena cada "template" calculando cuántas orientaciones de los gradientes más intensos de la escena coinciden con las orientaciones de los "templates". En el lugar donde haya mayores coincidencias es donde se supondría que se encuentra el objeto a detectar.

IV. IMPLEMENTACIÓN DE DOT EN GPU

Si bien la implementación propuesta en el trabajo de Hinterstoisserfue realizada en una computadora con arquitectura x86 sin GPU, la performance obtenida no fue la esperadapara poder aplicar esta técnica a un sistema de visión de un brazo robot, por lo tanto, con el objetivo de avanzar en el desarrollo del algoritmo, se procedió a la incorporación al sistema de una placa de desarrollo NvidiaJetson TK1 la cual posee una GPU.

En un primer intento se compiló el código realizado anteriormente con mínimas modificaciones para adaptarlo a la placamencionada. Las consideraciones iniciales, sumado a que gran parte del código hace uso de la biblioteca OpenCV, [8] y que varios métodos utilizados están optimizados para la ejecución en una GPU, sugerían un buen rendimiento,pero esto no fue así. La detección en un video de prueba superaba por mucho el tiempo de procesamiento al de su ejecución mediante el análisis en una CPU. Observando este inconveniente se comenzó a analizar el código con detenimiento. Se observó que la manera en la que se estaba intentando utilizar la GPU no era la adecuada, por lo tanto, se examinó cada uno de los métodos utilizados para identificar aquellos que producían la mayor demora y optimizarlos. A partir de este análisis se llegó a la conclusión de que el cálculo de gradientes en la escena era el primero que más demoraba en ejecutarse. Recordemos que elmétodo se debe ejecutar fotograma a fotograma ya que se trata de un video.En el cálculo de gradientes las tareas a realizar son las siguientes y cada una de ellas depende de la anterior:

1. Se convierte a escala de grises la imagen del fotograma a analizar.
2. Se calcula el filtro Sobel por X e Y.
3. Se calcula el gradiente a partir de los filtros anteriores.
4. Se halla el máximo gradiente en cuadrados de NxN pixeles en toda la imagen.
5. Se calcula el ángulo de dicho gradiente guardando su orientación en un byte.
6. Se almacena el resultado.

De toda la lista anteriormente mencionada, el primer lugar en donde se atacó la demora fue en el cálculo del filtro Sobel. Se intentó utilizar, nuevamente, pero de otra forma, el filtro Sobel optimizado para CUDA ofrecido en OpenCV y ejecutado en paralelo (para obtener los componentes X e Y) pero los tiempos globales de detección nos hizo pensar que la mejor forma de ganar rendimiento al implementar el algoritmo DOT no era utilizando los métodos optimizados ofrecidos por OpenCV, sino a través del desarrollo de un kernel. Esto fue producto de dos circunstancias, en un principio desconocidas, al momento de iniciar las pruebas:

- El tiempo de transmisión de datos desde la memoria principal a la memoria de la GPU (y viceversa) no es despreciable y de manera inherente genera

una sobrecarga de tiempo en la ejecución del algoritmo. Estos tiempos deben ser minimizados.

- Se debe buscar en la imagen máximos locales en regiones de interés de NxN pixeles ya queOpenCV no ofrece una función optimizada que utilice CUDA para realizar esta tarea.

Por todo lo expuesto se procedió al diseño de un kernel que pudiera acelerar y hacer uso correcto de la GPU. y así hallar los gradientes eficientemente.El enfoque utilizado para resolver el problema fue dividiry solucionar el mismo en etapas parciales.

El primer paso fue el desarrollo de un kernel que permitiera aplicar el filtro Sobel a una imagen utilizando la GPU. Estofue fundamental ya que en un kernel se solucionaban los puntos 2 y 3 de la lista de tareas mencionada.A continuación,se ensayó el comportamiento del kernel sobre un video y los resultados fueronalentadores: en un tiempo mucho menor de lo esperado, el filtro era aplicado. Por lo tanto, con estos cambios la lista de tareas quedaría de la siguiente forma:

1. Se convierte a escala de grises la imagen del fotograma a analizar.
2. Se calculan el filtro Sobel por X e Y y el gradiente (GPU).
3. Se halla el máximo gradiente en cuadrados de NxN pixeles en toda la imagen.
4. Se calcula el ángulo de dicho gradiente guardando su orientación en un byte
5. Se almacena el resultado.

El paso siguiente consistió en la búsqueda del máximo gradiente en losbloques de NxN pixeles en la GPU.La primera aproximación para solucionar este problema fue programar en un kernel distinto la búsqueda de máximos locales utilizando la GPU en bloques de NxN pixeles. De esta formase trabajó la imagen resultante del filtro anterior en cuadradosNxN, pudiéndose hallar paralelamente los máximos de cada bloque. Esto fue correcto, pero el hecho de utilizar

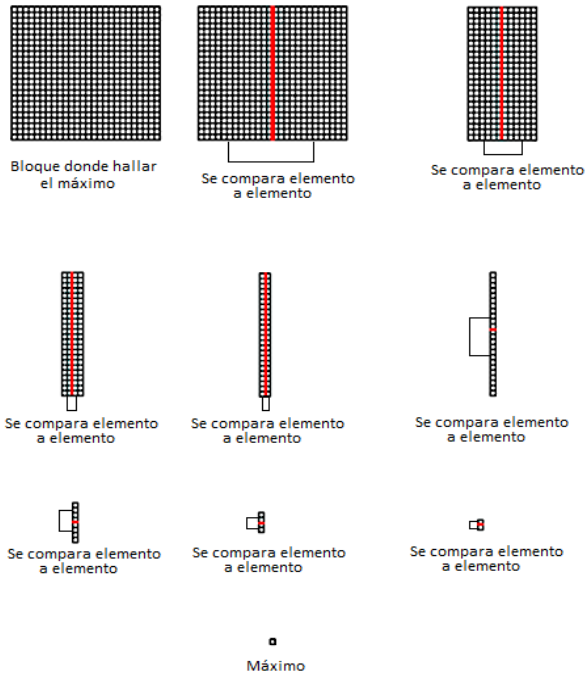


Fig.2. Ejemplificación de la reducción de una matriz utilizando una GPU

un único hilo para buscar el máximo gradiente desperdiciaba el poder de procesamiento de la GPU. Es por esto por lo que después de analizar el problema se reprogramó la solución utilizando el concepto de reducciones paralelas, en este caso, en matrices.

La idea de las reducciones paralelas es sencilla (la figura 2 ejemplificade una manera clara el concepto de reducción que se programó): para hallar un máximo se deben utilizar comparaciones. Si se utiliza un solo núcleo o hilo, ese único hilo o núcleo debe realizar todas las comparaciones. Pero

al tener una GPU y poder decidir cuantos hilos se ejecutan por cada bloque, esta situación cambia: como la operación de comparación es binaria, la máxima cantidad de hilos que pueden utilizarse para solucionar el problema de manera óptima es la mitad de la cantidad de elementos (si esta es par). Por ejemplo, si se trata de hallar el máximo en una matriz de 8x8 elementos existirán 64 elementos a comparar y si se utilizan 32 hilos se podrán comparar 32 pares de elementos de una sola vez, guardando el resultado en una de las mitades. Si esta operación se continúa realizando (con los elementos que sobran comparar) la eficiencia que se obtiene es realmente considerable.

A posteriori de haberse hallado el máximo, en el mismo kernel se programó el cálculo para transformar y expresar su orientación medida en un byte, tal como lo proponen los autores del algoritmo DOT. Por lo tanto, la lista de tareas con los cambios realizados se resume en:

1. Se convierte a escala de grises la imagen del fotograma a analizar.
2. Se calculan el filtro Sobel por X e Y, y el gradiente (GPU)
3. Se halla el máximo gradiente en cuadrados de NxN pixeles en toda la imagen, se calcula el ángulo de dicho gradiente guardando su orientación en un byte y se almacena el resultado (GPU)

Las optimizaciones descriptasproducen un considerable aumento de velocidad respecto a las mismas tareas ejecutadas en una CPU o utilizando en parte la GPU, a través de OpenCV.

Ya solucionado en gran parte la demora producida por el cáculo de los gradientes, secontinuó analizando el programa en búsqueda de otras demoras. Un punto clave que producía las mismas, fue la decodificación del método de compresión utilizado en los videos o en la misma cámara de la que se obtienen los fotogramas. Debido a que la única manera de aumentar la velocidad de esta decodificación es utilizar la GPU, y la transferencia entre la memoria principal y la de la GPU produce demoras, se decidió decodificar los fotogramas en un hilo común en la CPU.

Se debe tener en cuenta que la placa NvidiaJetson TK1 posee un microcontrolador ARM de cuatro núcleos y utilizar uno en particular para decodificar los fotogramas no produce ningún tipo de problemas. Valiéndose de una arquitectura de productor consumidor, en el que en un hilo se decodifican los fotogramas (productor) y en otro hilo se ejecuta el kernel donde se buscan los gradientes y los máximos (consumidor), se aumentó aún más la eficiencia de ejecución del algoritmo DOT.

A. OPTIMIZACIÓN DE LA BÚSQUEDA

En este puntose entiende que la GPU ya no puede ser sobrecargada, por lo tanto, la búsqueda del objeto se debe realizar en la CPU. El algoritmo DOT proponequepara encontrar un objeto es necesario buscar la coincidencia más probable entre la escena y los objetos que se aprendieron. Para lograr esto se necesita utilizar una ventana deslizante (que recorre toda la escena) y buscar coincidencias de ángulos de los objetos aprendidos. El “ganador” o la región candidata a ser el objeto en cuestión es aquella en la que existan más coincidencias de orientaciones entre la plantilla del objeto aprendido y la escena. En una versión simplificada del código se vería así:

Por cada plantilla de objeto aprendido

Por cada pixel en Y de la escena

Por cada pixel en X de la escena

Por cada pixel en Y de la plantilla

Por cada pixel en X de la plantilla

Match+=Escena(x,y)&plantilla(x,y)

La alta complejidad computacional salta a la vista y de alguna manera habría que minimizarla. Luego de analizar el algoritmo se llegó a las siguientes conclusiones:se podría aumentar la eficiencia si solamente se comprobara la coincidencia de ángulos en los lugares dondeestén definidos en la plantilla, de esa manera se evitaría una iteración "*for*" y, por ende, varias operaciones en la búsqueda. Además, se notó, experimentalmente, que es mucho más rápido hacer la búsqueda cada dos pixeles en la escena que uno por uno (sacrificando resolución en la búsqueda por velocidad).

Para lograr esto se modificó el método que utiliza el programa de aprendizaje para analizar y guardar los objetos aprendidos, ahora en vez de guardar toda la plantilla se guarda solamente el valor del/los ángulos y la posición en la misma. En resumen, una versión simplificada del código se vería así

Por cada plantilla del objeto aprendido

Por cada pixel par en Y de la escena

Por cada pixel par en X de la escena

Por cada ángulo válido en la plantilla

If(Escena(x,y) es un ángulo definido)

Match+=Escena(x,y)&plantilla (x,y)

Si bien el cambio no es drástico, es evidente que existen mejoras.

B. CAMBIOS DRÁSTICOS EN EL OBJETO

El algoritmo DOT no es perfecto y en las pruebas se observó quequizás por falta de plantillas la detección no se realizaba en el lugar correcto. Para morigerar este comportamiento se intentósuavizar la detección brusca de objetos, teniendo en cuenta que estosno pueden aparecen en otro lugar en la escena de instantáneamente.Esta situación se trató de solucionar de la siguiente manera: si se detectó un objeto en una posición que supera un umbral de detección, la ventana de detecciónse trasladará de manera progresiva al lugar donde se supone que se encuentra el objeto.

C. PRUEBAS

En la figura 3se presentan algunas imáge-

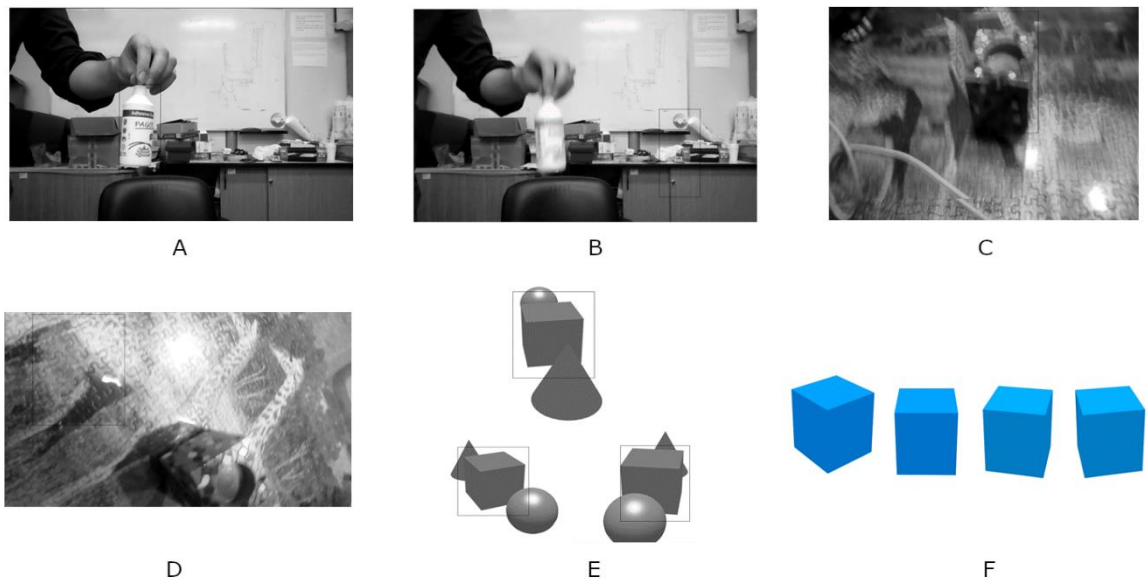


Fig.3. Imágenes resultantes del detector.A) Algoritmo reconociendo el envase de pegamento. B) Algoritmo presentando una falla en la detección C) Algoritmo detectando un cubo correctamente D) Algoritmo fallando por falta de plantillas. E) Detecciones correctas en el video de prueba que se utilizó. Observar la detección con oclusiones. F) Plantillas que se utilizaron para la detección de la imagen E.

nes de momentos en los que el algoritmo produjo resultados apropiados. Es importante recalcar que fueron algunos momentos, debido a que no se utilizaron demasiadas plantillas para realizar la detección. Las imágenes son grises debido a que se consume demasiada memoria si se utilizan los tres canales de colores. El exceso de consumo de memoria genera, como consecuencia, que el sistema operativo termine de manera abruptael proceso de detección.

V.DETECCIÓN DE PROFUNDIDAD

La detección de un objeto a partir de la implementación de DOT permite encontrar el mismo en el plano, pero no en el espacio, siendo parte fundamental la detección de esta dimensión debido a la naturaleza posicional del brazo robot. Es por esto por lo que se necesita realizar la medición de la profundidad por medio de un sistema estereoscópico. La posibilidad que se evaluó es la de ubicar dos cámaras en un mismo plano *M* (con coordenadas *x* e *y*) separadas por una distancia *b* en el eje *x*, y con sus ejes focales paralelos y perpendiculares al plano *M*. De esta forma se lograrían dos imágenes dispares a nivel horizontal, y el producto de esa disparidad se utilizaría para medir la profundidad.

A nivel matemático [9], este mecanismo puede ser resuelto a través del método de triangulación. En la Figura 4se puede observar un esquema básico, en el cual se muestran dos sistemas de ejes cartesianos cada uno correspondiente a cada cámara del par estereoscópico.

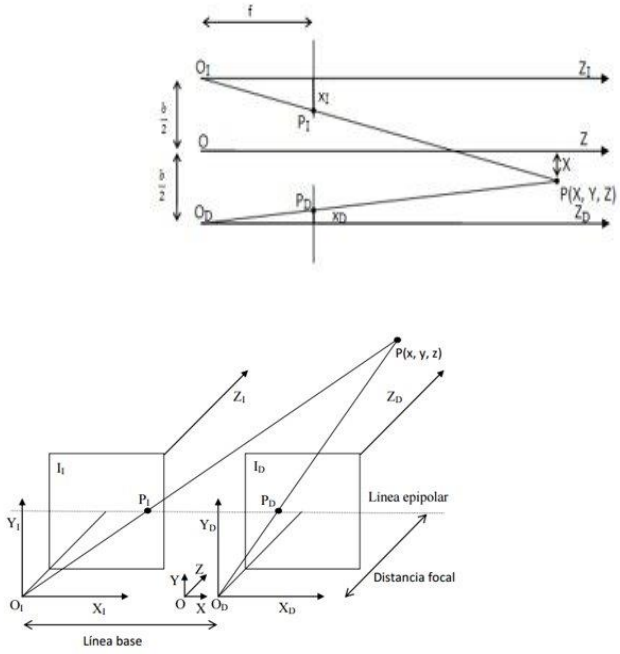


Fig.4.Diagrama tridimensional para el cálculo de la distancia del punto P

A una profundidad dada por la distancia focal de las cámaras *f* (que debe ser la misma en ambas) se encuentra el plano denominado epipolar que contiene los planos I_I e I_D correspondientes a las imágenes tomadas por cada cámara. En este plano es donde se miden las disparidades entre dichas imágenes.

Al encontrarse las cámaras a la misma altura (eje y), las únicas diferencias se dan a nivel horizontal ya que, como se comentó, las cámaras se ubican a una distancia de separación *b* sobre el eje *x.*

En la Figura 5, se pueden observar los puntos P_I y P_D que son las proyecciones del punto P sobre la línea epipolar. Se puede intuir que cuanto más lejano se encuentre el punto P, los vectores de proyección tenderán a ser paralelos al eje Z lo que hará queP_I y $P_D$tiendan a ubicarse sobre el eje Z_I y Z_D. La disparidad del punto se calcula como X_I - X_D = d y en el caso de que un objeto se encuentre ubicado en el infinito del eje Z hará que d valga 0.Así entonces el cálculo de la distancia del objeto sobre el eje Z será:

$$ImagenIzquierda: \frac{\frac{b}{2}+x}{z} = \frac{x_I}{f} \Rightarrow x_I = \frac{\left(x+\frac{b}{2}\right)f}{z}$$

$$ImagenDerecha: \frac{\frac{b}{2}-x}{z} = \frac{x_D}{f} \Rightarrow x_D = \frac{\left(x-\frac{b}{2}\right)f}{z}$$

Fig. 5.Diagrama bidimensional para el cálculo de la distancia del punto P

En la ecuación *ImagenDerecha* se asigna un signo negativo a X_D para compatibilizar los sistemas de coordenadas de los ejes O_I y O_D. Siendo la disparidad del punto P: x_I-x_D= d reemplazando y despejando se llega a la ecuación:

$$z = \frac{fb}{d}$$

con la cual se puede calcular la distancia z del punto P (objeto detectado).

Por otra parte, el desarrollo de los algoritmos de control del autómata se apoyó en las bibliotecas desarrolladas por Peter Corke, realizándose simulación mediante el software MATLAB. Queda pendiente la integración con las bibliotecas que se desarrollaron para la detección de objetos, debido a que se priorizó el desarrollo de las de procesamiento gráfico.

A. CALIBRACIÓN DE LAS CÁMARAS

Lo aconsejablefue, en un principio, que el ajuste se realizaraen forma mecánica logrando la mejor alineación posible de las cámaras (ejes horizontal y vertical) para que luego el algoritmo de calibración tuviese mejores resultados.Una vez realizadoeste ajuste, se comenzó con el cálculo de las matrices de calibración, para lo que se utilizó la biblioteca de OpenCV que contiene clases para calibración y reconstrucción de imágenes 3D: “Camera calibration and 3D reconstruction”. Para la calibración se usó la aplicación *calibrate_cameras*, que forma parte de la biblioteca StereoVision[10] (Bajo licencia pública GNU).

Esta aplicación toma como argumentos imágenes estéreo que contienen en su escena un patrón cuadriculado símil a un tablero de ajedrez, con capturas en distintas perspectivas de éste (Figura

Fig.6. Vértices detectados por el algoritmo de calibración

6). El tablero se rota y,además, se lo traslada a lo largo de las múltiples tomas de imágenes. Matemáticamente se puede demostrar que la cantidad mínima de tomas depende de la cantidad de esquinas que contiene el tablero patrón. El término “esquinas” hace referencia a la cantidad de vértices de cuadrados negros que hacen contacto entre sí (léase negros o blancos). La máxima cantidad utilizada es 50, bajo el criterio de que cuantas más muestras haya los cálculos de corrección de errores serán más precisos.

La aplicación *Calibrate_cameras* da como resultado una carpeta con un conjunto de matrices que contiene los parámetros intrínsecos (focos, centrados), extrínsecos (traslación y rotación) y las matrices de rectificación de distorsiones. Con estas matrices se realizan las correcciones sobre las imágenes tomadas en cada cámara. Una vez rectificadas las imágenes se puede realizar el cálculo del eje Z con la fórmula citada más arriba.Para lograr este cálculo hay que ubicar en ambas imágenes del par estéreo el mismo objeto, para esto es necesario un algoritmo de búsqueda e identifica-

ción de objetos. Esto se puede resolver con el algoritmo DOT que fuedetallado anteriormente.

B. *VERIFICACIÓN CON MAPA DE DISPARIDAD*

La verificación con mapa de disparidad se realizó utilizando las bibliotecas de StereoVision, una prueba empírica realizando un mapa de disparidad y una nube de puntos. La nube de puntos consiste en conseguir las coordenadas espaciales de cada pixel de la escena tomadas por las cámaras.

De esta forma se puede visualizar una escena desde diferentes perspectivas rotando los ejes de coordenadas. Esto tiene un límite y es que esos "puntos" de la imagen se obtienen desde una posición fija del espacio, si se deseara observar perspectivas laterales se necesitaría más información de la escena. Esto se puede salvar hasta cierto ángulo, replicando los pixeles de la frontera, de forma que cubran los huecos de información faltante.

Para lograr un buen mapa de disparidad se necesita que los objetos de la escena sean detectados inequívocamente en las imágenes del par estéreo. Se utilizó la herramienta *tune_blockmatcher* de la biblioteca *StereoVision* con el fin de ajustar los parámetros del detector de objetos y se verificó en paralelo la "calidad" del mapa de disparidad que se muestra en la Figura 7.

Los ajustes mencionados se deben realizar en forma iterativa y empírica ya que resulta ser el método más rápido y eficiente. Luego de realizar el ajuste del algoritmo de búsqueda y detección de objetos, se utilizó la función *images_to_pointcloud* (también de la citada biblioteca StereoVision), la que toma como argumentos las matrices de rectificación, un par de imágenes estéreo, los parámetros de ajuste del algoritmo de "*blockmatching*", esto da como resultado una matriz de tres dimensiones que contiene los vectores (también de tres dimensiones) correspondientes a la información del
color de cada pixel. Se puede observar la nube de puntos en la Figura 8)

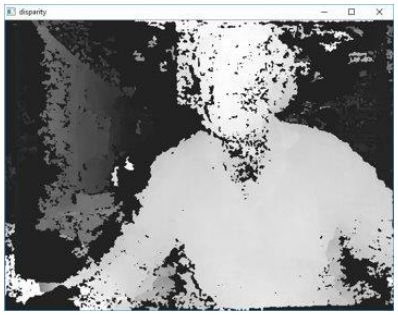


Fig7. Mapa de disparidad


Fig. 8. Resultado de la nube de puntos a partir de la imagen estéreo

V. CONCLUSIONES

Quizás las modificaciones y los agregados realizados sobre el algoritmo DOT no sean suficientes para obtener velocidades considerables en la detección, aunque permiten mejorar adecuadamente la misma. Ahora es aceptable realizar una búsqueda en una imagen HD, pero aun así sigue siendo relativamente lento. Como se plantea al final del apartado "Implementación de DOT en GPU" quizás la respuesta involucre la implementación del proceso de "match template" en la GPU, aprovechando el tiempo ocioso de la misma.

Respecto al algoritmo, presenta fallas que quizás sean mejorables. DOT no considera colores, solo considera formas, y queda claro que resolver esa falencia sería una mejora a esta implementación. Otro punto importante para destacar es que el tiempo de reconocimiento de un objeto depende de la cantidad de plantillas y de las dimensiones del video. Esto no debería ser así, aunque es una característica intrínseca del algoritmo.Una solución que se podría proponer para la mejora del algoritmo es la de no utilizar la ventana deslizante, sino que por algún medio se detectarán los posibles candidatos a la imagen de la plantilla buscada y luego realizar el "match template" entre las plantillas y esa región de la imagen.

Respecto al cálculo y análisis de profundidad, la calibración de las cámaras y la comprensión del algoritmo están en un nivel muy avanzado, al punto de que solamente sería necesario programar la detección y agregarla a la de objetos. El único inconveniente será el tiempo de procesamiento que se sumaría al que posee la detección DOT y, como se expresó, se debería buscar una solución de compromiso o analizar la unión de ambos algoritmos mucho más detenidamente.

En etapas siguientes de este proyecto de investigación, que sigue vigente en el ámbito de la Universidad, se procederá a continuar el trabajo sobre aquellas cuestiones que hoy se mencionan como dificultades o asuntos pendientes.

VII. REFERENCIAS Y BIBLIOGRAFÍA

A. Referencias bibliográficas:

[1] Bay, Herbert,Tuytelaars, Tinne, Van Gool, Luc.: SURF:Speeded Up Robust Features, European Conference on Computer Vision 2006

[2] Rublee,Rabaud,Konolige,Bradski: ORB: an efficient alternative to SIFT or SURF, IEEE International Conference on Computer Vision 2011

[3] LeCun,Haffner,Bottou,Bengio, Object Recognition with Gradient-Based Learning, CPVR 2004

[4] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez , Kaiser, Polosukhin, Attention is all you need, Cornell University 2017

[5] Hinterstoisser Stefan, Lepetit Vincent, Ilic Slobodan, Fua Pascal, Navab Nassir.: Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects, Computer Vision Pattern Recognition 2010.

[6] Gonzales, Woods: Digital Image Processing ,TerceraEdición, Prentice Hall 2007

[7] Hinterstoisser Stefan, Cagniart Cedric, Ilic Slobodan, Sturm Peter, Navab Nassir, Fua Pascal, Lepetit Vincent.: Gradient Response Maps for Real-Time Detection of Texture-Less Objects, IEEE Transactions on pattern analysis and machine intelligence 2011

[8] OpenCV 9/2/2005[en línea]Fecha de consulta: 27/7/2020http://opencv.org/

[9] Martin Montalvo, Técnicas de visión estereoscópica para determinar la estructura tridimensional de la escena, Autor: Curso académico 2009/2010, Máster en Investigación en Informática, Facultad de Informática, Universidad Complutense de Madrid 2009/2010.

[10] StereoVision15/04/2017 [en línea] Fecha de consulta 27/7/2020

https://pypi.org/project/StereoVision/

Recibido:A completar por el Editor. Formato: AAAA-MM-DD
Aprobado:A completar por el Editor. Formato: AAAA-MM-DD
Hipervínculo Permanente: A completar por el Editor
Datos de edición: Vol. [A completar por el Editor]-Nro. [A completar por el Editor]-Art. [A completar por el Editor]
Fecha de edición: Formato: AAAA-MM-DD

CONAIISI

VII Congreso Nacional de Ingeniería Informática - Sistemas de Información

2019

San Justo, 5 de diciembre de 2019

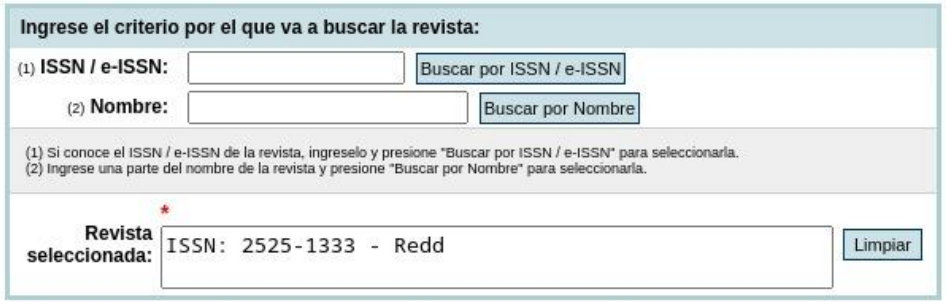
Se certifica que **Martin Ferreyra Biron, Alberto Raul Miguens, Fernando Ignacio Szklanny** han participado como Autores del artículo 342 *"Desarrollo de un sistema de asistencia visual para un manipulador automático programable (robot industrial)"*, aceptado en el VII Congreso Nacional de Ingeniería Informática – Sistemas de Información, CONAIISI 2019, realizado los días 14 y 15 de noviembre en la Universidad Nacional de La Matanza.

Ing. Claudio D'Amico
Coord. Gral. CONAIISI

Dr. Carlos Neil
Coordinador RIISIC

Mg. Jorge Etcrovio
Decano DIIT

UNLaM



Editorial: * UNLAM

País de edición: * Argentina **Ciudad de la editorial:**

Título del artículo: * Desarrollo de un sistema de asistencia visual para un manipulador automático programable

Idioma: * Español

Volumen: 5 **Tomo:** **Número:** 1

Página inicial: **Página final:**

(3) **Fecha de publicación:** * 10/2020

(4) **URL:** https://reddi.unlam.edu.ar/index.php/ReDDi/article/view/120/222

(4) **DOI:**

Referato: * (●) Sin referato ( ) Con referato

(3) **Estado de publicación:** * (●) Publicado ( ) En prensa