Universidad Nacional de La Matanza

Unidad Ejecutora: Departamento de Ingeniería e Investigaciones Tecnológicas

Título del proyecto de investigación:
Manipulador automático programable asistido por visión

Código del proyecto: C – 197

Programa de acreditación: PROINCE

Director del proyecto: Ing. Fernando I. Szklanny

Co-Director del proyecto: Ing. Nicolás Molina Vuistaz

Integrantes del equipo:
Ing. Gustavo Sagarna
Ing. Alejandro Martínez
Ing. Nahuel Nieva
Ing. Mario Juan Krajnik
Ing Hugo R. Tantignone
Ing. Alberto R. Miguens
Sr. Martín Ferreyra Birón
Ing. Alejandro Pérez Aráuz (1.9.2016 – 31.7.2017)

Fecha de inicio: 01.01.2016

Fecha de finalización: 31.12.2017

Informe final

Sumario:

Contenido

1.- Resumen del Proyecto

El objetivo del presente proyecto es el de completar la investigación y desarrollo de los algoritmos destinados al control de un autómata programable, capaz de moverse articuladamente en seis ejes, asistido por el procesamiento digital de imágenes captadas por cámaras de alta velocidad, y destinado a aplicaciones industriales que requieren una mayor eficiencia en el trabajo.

Se propone investigar y mejorar los algoritmos de control basados en el procesamiento de imágenes que se adapte a las necesidades de producción de las empresas de la región y del país, apto para trabajar en ambientes industriales, con el objeto de permitir la automatización de procesos industriales que permita la sustitución de importaciones de tipo tecnológico, lo que permitirá reducir el alto costo que implica para las pequeñas y la pérdida de competitividad que afrontan debido el atraso tecnológico.

El autómata está formado por tres elementos: un manipulador (o “brazo”), un controlador y un conjunto de sensores. El controlador contiene el sistema procesador y las unidades de control de energía. El brazo es accionado por un servomecanismo y es capaz de moverse articuladamente en seis ejes. Los sensores utilizados, con la finalidad de mejorar el control de autómata y la seguridad en el entorno de trabajo, son cámaras de alta velocidad y acelerómetros.

El proyecto planteado tiene por objetivo fundamental permitir el reemplazo de los controladores, que hoy deben importarse, por productos de fabricación nacional. Si bien en esta etapa no se puede considerar la fabricación nacional de motores, ni de semiconductores y cámaras de alta velocidad que permitan el reemplazo total de los componentes importados, puede pensarse en su reducción, dado que se pretende reemplazar los controladores, incorporar software desarrollado localmente y, fundamentalmente, tener la propiedad intelectual del diseño, que es una importante componente del costo de este tipo de productos.

El autómata al que se hace referencia se basa en los resultados obtenidos en proyectos anteriores por este grupo de investigación, en el marco del mismo programa PROINCE, durante el período 2012 – 2015. Las dificultades presupuestarias surgidas a lo largo de los años mencionados, así como la falta de concreción de un subsidio externo, proveniente del programa PICT-O, impidieron la concreción total del proyecto original en los plazos oportunamente establecidos. Por consiguiente, la posibilidad de una nueva ampliación en los plazos y una posibilidad de nuevos recursos financieros permitirán, finalmente, el logro de los objetivos planteados. Los resultados obtenidos en los proyectos anteriores servirán como una base muy importante para permitir completar el desarrollo que se propone en el presente proyecto.

Mediante este proyecto se pretende establecer una base de conocimientos para facilitar futuros desarrollos, que continúen aportando posibilidades de mejoramiento tecnológico a las empresas locales, y consolidar un grupo de trabajo en un área en la cual hay posibilidades de agregar valor.

Pal**abras clave: Robótica industrial, visión artificial, inteligencia artificial, redes neuronales.**

Tipo de investigación: Desarrollo Experimental

Área de conocimiento: 1800 - Ingeniería de Comunicaciones, electrónica y control

Disciplina de conocimiento: 1805 – Control

Campo de aplicación: 0888 – Equipos e instrumentos científicos de medición y control

2.- Estado actual del conocimiento

La industria de la robótica evoluciona en forma exponencial y a lo largo de los últimos 50 años presenta grandes avances. Si se consideran los hechos fundamentales de estas últimas décadas, pueden verse, entre otros, los siguientes:

- En 1959 la empresa Unimation desarrolla el primer robot programable para aplicaciones industriales.
- En 1969 la empresa GM incorpora robots industriales en una línea de ensamblaje.
- En 1973 la empresa KUKA desarrolla el primer robot electromecánico de 6 ejes.
- En 1974 la empresa Cincinnati Milacron Corportation desarrolla la primera minicomputadora para controlar un robot.
- En 1985 la empresa KUKA introduce un brazo robot totalmente flexible con tres movimientos de traslación y tres movimientos de rotación.
- En 1998 ABB desarrolla el primer robot delta más rápido del mundo con tecnología de visión.
- En 2004 la empresa Motorman introduce un sistema de control capaz de controlar 4 robots y hasta 38 ejes en forma simultánea.
- En el año 2006 KUKA en colaboración con la empresa DLR y El Instituto de Robótica y Mecatrónica de Alemania se presenta el primer Robot liviano.
- En el año 2010 Fanuc laza el primer sistema de auto aprendizaje para el control de un robot a partir de las vibraciones producidas por las altas aceleraciones y velocidades.

En la actualidad los autómatas programables permiten reducir los costos de mano de obra, consumo de energía y entrenamiento de personal. Sus altos niveles de precisión y repetitividad permiten lograr productos de gran calidad, sin riesgo de cansancio o distracción, aumentan la productividad, pueden operar las 24 horas del día con mínima supervisión, logrando altas velocidades y tiempos de ciclo reducidos, permiten trabajar con múltiples formatos de productos simultáneamente y se pueden reprogramar rápidamente para realizar nuevas tareas, realizan trabajos que son peligrosos para el ser humano, actuando cerca de fuentes de riesgo o en ambientes tóxicos o contaminados, pueden manejar cargas pesadas y peligrosas, cuidando la salud ocupacional de las personas.

Los robots industriales ocupan un lugar importante dentro de la automatización de la producción, lo que ha permitido a los países desarrollados alcanzar niveles muy altos de producción y de calidad. El mercado de robots ha mantenido un crecimiento constante, en especial en Europa y Estados Unidos. Por su parte, Asía y particularmente Japón, sigue estando a la cabeza a nivel mundial. En América Latina, aunque en menor medida, la evolución también ha sido creciente; los líderes en automatización son Brasil, México, Argentina y Chile.

Según indica la IFR (International Federation of Robotics) en su último reporte ejecutivo, existen actualmente en el mundo aproximadamente 1.373 millones de robots utilizados en la industria. Los países que cuentan con mayor número de robots industriales operativos son: Japón (309.400), Estados Unidos (216.650), Alemania (165.800) y República de Corea (155.300). Si se observa el panorama a nivel mundial por continente en términos de porcentaje, Asia y Oceanía poseen el 53%, Europa el 28%, América el 17% y África el 2%. En América del Sur, Brasil cuenta con 9.170 robots, mientras que el resto cuenta con 1.730 en total. Adicionalmente, en el último año se vendieron 2.300 robots en América del sur, de los cuales 2.000 corresponden a Brasil.

En contraste, en Latinoamérica no existe ningún fabricante de robots industriales y si bien existen iniciativas en entidades públicas y privadas, están lejos de ofrecer una solución que se pueda implementar en el sector industrial de manera masiva y con la calidad necesaria, principalmente por la falta de recursos en investigación y desarrollo y una escasa visión de largo plazo.

En resumen, estos planteos permiten pensar en la relevancia del proyecto que se propone, justificada en las siguientes observaciones:

- El avance tecnológico a nivel global requiere empresas locales más eficientes con mayores índices de productividad y calidad que hacen indispensable el uso de autómatas programables.
- Las inversiones necesarias para la utilización de autómatas están fuera del alcance de muchas empresas e industrias, (en especial pymes) o bien no se encuentran desarrolladas para los volúmenes de producción local.
- No existe ninguna empresa nacional que desarrolle autómatas para la industria, lo que crea una oportunidad para su desarrollo a precios más accesibles para las empresas pymes, con la posibilidad de sustituir importaciones de alto valor agregado y generar exportaciones a los países de la región.

En cuanto al procesamiento digital de imágenes, ha tomado un gran impulso en la última década, impulso derivado de la alta capacidad de los procesadores que posibilitan el desarrollo de sistemas más versátiles y de costo reducido, ejemplificados en el control de procesos realizados por robots industriales, la navegación en vehículos autónomos, la detección de eventos e inspección de objetos en procesos de manufacturas y en un sin fin de aplicaciones más.

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 y autómatas programables con una vasta aplicación en la industria. Sin embargo, existen numerosos sistemas de producción que no cuentan con autómatas que se adapten a sus necesidades de proceso automatizados para un aumento del nivel de producción y calidad, menos aún que utilicen el procesamiento digital de imágenes ya sea por su elevado costo o por la inexistencia en el mercado. Aquí es donde intervienen las propuestas de soluciones innovadoras que cubran dichas necesidades.

3.- Problemática a investigar

La aplicación de los sistemas digitales del 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 para realizar cálculos, hacen que su aplicación se limite a controles de baja complejidad. No obstante, el gran avance tecnológico producido desde ese momento, tanto en la miniaturización de los sistemas digitales, la capacidad de cálculo y su velocidad de procesamiento hacen que se puedan aplicar a controles con algoritmos de alta complejidad.

Se propone en este proyecto completar la investigación y el desarrollo de métodos de corrección de control de sistemas autónomos basados en el procesamiento de imágenes y variables de entorno, aplicado principalmente al campo de la Robótica, control de maquinaria y vehículos eléctricos, mediante la utilización de cámaras de alta velocidad, acelerómetros y giróscopos. El desarrollo de esta tecnología es fundamental para mejorar la productividad y calidad de producción que hoy requieren las industrias locales para no perder competitividad a nivel global.

El proyecto implica conocimientos de procesadores digitales de señales, estudios de algoritmos avanzados y desarrollo de nuevos algoritmos para incorporar procesamiento digital de imágenes a los algoritmos ya desarrollados. En una etapa previa se estudiaron las tecnologías de procesadores digitales de señales y algoritmos existentes para el control de motores y autómatas, se desarrolló un autómata programable de 6 grados de libertad y sus respectivos algoritmos de posición y velocidad. El proyecto asimismo requiere conocimientos de control, procesamiento de imágenes y robótica aplicada, que los integrantes del grupo de investigación poseen por ser Ingenieros Electrónicos con orientación en robótica.

El autómata propuesto es un brazo mecánico capaz de moverse articuladamente en 6 ejes. Es del tipo "articulación coordinada", llamado así por la semejanza de sus movimientos con los del cuerpo humano. El movimiento del primer eje corresponde con el de la cintura. El segundo eje se corresponde con el movimiento vertical del hombro. El tercer eje corresponde con el giro del codo también en sentido vertical. La combinación del cuarto, quinto y sexto eje produce los movimientos de giro e inclinación de la muñeca. Cada eje posee un servomotor permitiendo movimientos precisos. En el extremo del brazo se coloca un electroimán para poder tomar y desplazar objetos metálicos. La programación de los movimientos del autómata se realizará desde una computadora personal mediante un software de libre uso.

Los autómatas programables, tal como se mencionó en el apartado anterior, son ampliamente usados en la industria automotriz. Pero existen numerosos sistemas de producción que no cuentan con autómatas que se adapten a sus necesidades, por lo que se pueden proponer soluciones innovadoras que cubran dichas necesidades. A nivel nacional no existe ningún fabricante de autómatas programables o dispositivos que utilicen procesamiento digital de señales aplicado a la automatización de procesos produciendo un atraso tecnológico con respecto a otros países.

En particular el desarrollo de algoritmos para el guiado de autómatas en base a la visión, es indispensable para detectar y manipular objetos de diferentes tamaños, color y contraste, mejorar la precesión al realizar una trayectoria, y evitar objetos no previstos en un tarea, haciendo al autómata adaptable a diferentes condiciones de trabajo.

Con el fin de mejorar la productividad de las industrias, la calidad y el costo de fabricación para ganar competitividad a nivel global; se requiere contar con autómatas programables en los procesos de fabricación con sistemas de visión. Es un objetivo del presente proyecto sentar las bases para el desarrollo de futuros autómatas y la automatización de procesos industriales.

4.- Objetivos

- Mejorar los algoritmos de procesamiento digital de imágenes y variables de entorno, basados en los nuevos avances en el campo, destinado al control de autómatas y procesos industriales.

- Mejorar la electrónica de los módulos de control y en particular la potencia de cálculo de la unidad de control principal en base a nuevos procesadores destinados al procesamiento digital de imagen.

- Completar el desarrollo y la puesta en funcionamiento sobre el modelo de autómata los algoritmos destinados al control de la dinámica del autómata programable asistido en forma visual, sobre la versión de laboratorio.

- Desarrollar y poner en funcionamiento los algoritmos destinados a la estrategia de ejecución y corrección de una tarea a realizar en base al análisis de imágenes del autómata y su entorno.

- Implementar la interfaz gráfica para la programación de tareas a realizar por el autómata en base a bibliotecas libres en el mercado.

- Mejorar el diseño y construcción de las diferentes partes mecánicas que componen la versión de laboratorio del autómata programable destinado al ensayo de los algoritmos, con materiales producidos localmente.

- Aportar posibilidades de mejoramiento tecnológico para las empresas locales.

- Lograr disminuir la brecha tecnológica existente en el campo de la robótica con relación a otros países.

- Permitir la continuidad de los desarrollos llevados a cabo en el área en los dos últimos años, consolidando una base de conocimientos y de recursos humanos para facilitar futuros desarrollos.

- Establecer una base de conocimientos para facilitar futuros desarrollos que en lo posible permitan generar patentes para la Universidad.

5.- Marco teórico

La aplicación de los sistemas digitales del 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 para realizar cálculos, hacen que su aplicación se limite a controles de baja complejidad. No obstante, el gran avance tecnológico producido desde ese momento, tanto en la miniaturización de los sistemas digitales, la capacidad de cálculo y su velocidad de procesamiento hacen que se puedan aplicar a controles con algoritmos de alta complejidad.

En cuanto al procesamiento digital de imágenes, ha tomado un gran impulso en la última década, impulso derivado de la alta capacidad de los procesadores que posibilitan el desarrollo de sistemas más versátiles y de costo reducido, ejemplificados en el control de procesos realizados por robots industriales, la navegación en vehículos autónomos, la detección de eventos e inspección de objetos en procesos de manufacturas y en un sin fin de aplicaciones más.

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 y autómatas programables con una vasta aplicación en la industria. Sin embargo, existen numerosos sistemas de producción que no cuentan con autómatas que se adapten a sus necesidades de proceso automatizados para un aumento del nivel de producción y calidad, menos aún que utilicen el procesamiento digital de imágenes ya sea por su elevado costo o por la inexistencia en el mercado. Aquí es donde intervienen las propuestas de soluciones innovadoras que cubran dichas necesidades.

6.- Hipótesis

- La tecnología actual de los procesadores digitales de señales permite grandes capacidades de cálculo a altas velocidades, lo que hace posible la ejecución de algoritmos complejos necesarios para el control de los autómatas programables.

- Existen programas especializados para el diseño y simulación de los algoritmos de control de autómatas programables.

- Existen bibliotecas avanzadas realizadas en lenguajes universales, que facilitan el desarrollo de algoritmo destinados al procesamiento digital de señales, cálculos avanzados sobre la dinámica y el control del autómata.

- Los resultados obtenidos en investigaciones previas y la relación con grupos consolidados de sensores, de programación y desarrollo hardware de la Universidad posibilitarán el desarrollo exitoso de los algoritmos de procesamiento de imágenes destinado a un autómata programable y procesos industriales que se propone en el presente proyecto, apto para trabajar en ambientes industriales.

7.- Metodología

El conocimiento de los integrantes en el campo de la robótica y los resultados obtenidos en investigaciones previas permiten el desarrollo de un autómata programable destinado a las aplicaciones industriales que requieren mayor eficiencia en la producción.

En particular en el campo de procesamiento digital de imágenes se parte de a la investigación, desarrollo y pruebas en laboratorio realizadas por los integrantes de investigación en proyectos previos, que, si bien se lograron importantes avances, se requiere mejorarlos y adaptarlos a los nuevos paradigmas y tecnologías en el campo.

El autómata está formado por tres elementos: el manipulador, el controlador y los sensores. Por consiguiente, para permitir su avance y finalización exitosos, el proyecto requiere conocimientos avanzados de mecánica, electrónica, análisis de señales, control avanzando aplicado a la robótica, procesamiento digital de imágenes y programación.

El dominio de programas para el diseño mecánico avanzados permite desarrollar y analizar las diferentes partes mecánicas que componen el manipulador. Además, es posible lograr un manipulador apto para la industria mediante la utilización de reductores de precisión y de bajo juego y servomotores de alto torque y bajo consumo.

Para la unidad de control del manipulador, se requieren procesadores con gran capacidad de cálculo capaz de ejecutar algoritmos avanzados para el control del autómata, actualmente disponibles en el mercado a un bajo costo. La gran cantidad de bibliografía referida al tema y las bibliotecas de libre uso disponibles facilitan el desarrollo de los algoritmos de control y comunicación necesarios para el control del autómata. Para simplificar y mejorar la confiabilidad de todo el sistema y disminuir la complejidad del control de cada eje, se propone usar controladores comerciales, comandados por la unidad de control.

Mediante la utilización de programas que permiten realizar ensayos y simulaciones de los diferentes algoritmos destinados a la cinemática, dinámica y control del autómata se puede evaluar el desempeño de los mismos. Adicionalmente, estos programas permiten evaluar los algoritmos de procesamiento de imágenes y su utilización eficiente en el control del autómata.

Es posible implementar un programa para planificar las tareas que debe realizar el autómata y enviarlas a la unidad de control del mismo mediante la utilización de bibliotecas de libre uso de programación de alto nivel.

En base a lo planteado, se propone una metodología de trabajo, para los dos años de duración del proyecto, que permita completar las tareas que han quedado inconclusas por las razones que ya han sido expuestas en apartados anteriores. Esas tareas se distribuyen en módulos asociados con el tipo de ingeniería involucrada en cada grupo de tareas, ya sea la ingeniería electrónica (hardware), la ingeniería mecánica y eléctrica o los desarrollos de software requeridos para completar el desarrollo del autómata.

Modulo I: Estudios de nuevos algoritmos y mejora del sistema mecánico - eléctrico ya existente.

Esta etapa se caracteriza en adaptar y mejorar el diseño mecánico del autómata que se dispone para los ensayos de algoritmos, mejorar la electrónica de la cálculo de la unidad central de computo destinado al procesamiento digital de imagen y el estudio de nuevos técnicas de procesamiento de imágenes, tomando como base los resultados en proyectos anteriores.

Tareas:

- Rediseño de partes mecánicas y reductores de la versión laboratorio del autómata.
- Mejorar y desarrollar la unidad central destinada al procesamiento de imágenes mediante la utilización de procesadores con mayor potencia de cálculo.
- Desarrollo y montaje de la electrónica adicional para la integración de toda la electrónica del control del autómata, los diversos sensores a utilizar y las cámaras de alta velocidad con la unidad central de calculo
- Investigación y desarrollo de nuevos paradigmas de procesamiento de imágenes, en cuanto a la detección de objetos, cálculo de distancia y captura de movimientos.
- Pruebas experimentales con las cámaras de alta velocidad.
- Desarrollo y ensayo de algoritmos en el laboratorio.
- Documentación.

Módulo II: Investigación y desarrollo de los algoritmos de control a implementar sobre el autómata programable basados en la asistencia visual.

En esta segunda etapa se continúa con la investigación y desarrollo de los algoritmos de procesamiento digital de imágenes ya aplicado al control del autómata, elaboración de modelos y simulación utilizando un software comercial para tal fin y finalmente adaptar los algoritmos existentes de posicionamiento, velocidad, fuerza y trayectoria a ser utilizados para la manipulación del autómata programable.

Tareas:

- Estudios de algoritmos y métodos existentes del procesamiento digital de imágenes aplicados a autómatas programables.
- Desarrollo y simulación de algoritmos avanzados de detección de objetos y cálculo de distancia.
- Desarrollo y simulación de algoritmos basados la captura de movimientos del autómata.
- Ensayos y correcciones de los algoritmos desarrollados.
- Investigación y desarrollo de algoritmos para incorporar los algoritmos al control del autómata.
- Ensayos sobre el autómata de laboratorio y correcciones de los algoritmos desarrollados.
- Documentación.

8.- Bibliografía propuesta.

8.1.- Libros:

[1] Robotics, Vision and Control: Fundamental Algorithms in MATLAB (Springer Tracts in Advanced Robotics). Peter I. Corke (Nov 3, 2011) ISBN-13: 978-3642201431, Springer.

[2] Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series). Howie Choset, Kevin M. Lynch, Seth Hutchinson and George A. Kantor (May 20, 2005) ISBN-13: 978-0262033275, Mit Pr.

[3] Introduction to Robotics: Mechanics and Control (3rd Edition). John J. Craig (Aug 6, 2004) ISBN-13: 978-0201543612, Prentice Hall.

[4] Robot Modeling and Control. Mark W. Spong (Nov 18, 2005) ISBN-13: 978-0471649908, John Wiley & Sons.

[5] Professional Microsoft Robotics Developer Studio (Wrox Programmer to Programmer). Kyle Johns (May 19, 2008) ISBN-13: 978-0470141076, Wrox.

[6] Springer Handbook of Robotics. Bruno Siciliano and Oussama Khatib (Jun 27, 2008) ISBN-13: 978-3540239574, Springer.

[7] Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.(tercera edición), Thomas Bräunl (Oct 24, 2008) ISBN-13: 978-3540705338, Springer.

[8] Experimental Robotics: The 10th International Symposium on Experimental Robotics (Springer Tracts in Advanced Robotics).Oussama Khatib, Vijay Kumar and Daniela Rus (Dec 1, 2010) ISBN-13: 978-3642096105, Springer.

[9] Handbook of Industrial Robotics, 2nd Edition. Shimon Y. Nof (Feb 16, 1999) ISBN-13: 978-0471177838, John Wiley & Sons.

[10] Industrial Robotics: How to Implement the Right System for Your Plant. Andrew Glaser (Aug 1, 2008.) ISBN-13: 978-0831133580, Industrial Pr Inc.

[11] IEEE/RSI International Conference on Intelligent Robots and Systems Proceedings -IEEE Robotics & Automation Society (Feb 2001) ISBN-13: 978-0780363496, I.E.E.E.Press (28 de febrero de 2001).

[12] 2006 IEEE Conference on Robotics, Automation and Mechatronics (Nov 14, 2008) ISBN-13: 978-1424400256, Institute of Electrical & Electronics Engineers (IEEE)

[13] Digital Signal Processing (4th Edition). John G. Proakis, Dimitris K Manolakis (Author) (Apr 7, 2006) ISBN-13: 978-0131873742, Prentice Hall.

[14] Understanding Digital Signal Processing. Richard G. Lyons (Nov 6, 1996) ISBN-13: 978-0201634679, Addison Wesley.

[15] Discrete-Time Signal Processing (3rd Edition). Alan V. Oppenheim (Aug 28, 2009). ISBN-13: 978-0131988422, Addison Wesley Pub Co Inc.

[16] Signal Processing and Linear Systems.B. P. Lathi (Feb 24, 2000) ISBN-13: 978-0195219173, OUP USA.

[17] The DSP Handbook: Algorithms, Applications and Design Techniques. Andrew Bateman (Oct 26, 2002) ISBN-13: 978-0201398519, Prentice Hall.

[18] Real-Time Digital Signal Processing from MATLAB® to C with the TMS320C6x DSPs, Second Edition. Thad B. Welch (Dec 22, 2011) ISBN-13: 978-1439883037, CRC Press,

[19] DSP Filter Cookbook (Electronics Cookbooks). John Lane (Dec 1, 2000) ISBN-13: 978-0790612041, Premier Pr.

[20] Introduction to Mechatronic Design. J. Edward Carryer (Dec 31, 2010) ISBN-13: 978-0131433564, Prentice Hall.

[21] System Dynamics: Modeling and Simulation of Mechatronic Systems. Dean C. Karnopp, Donald L. Margolis and Ronald C. Rosenberg (Jan 3, 2006) ISBN-13: 978-0471709657, John Wiley & Sons, cuarta edición.

[22] Mechatronics & Machine Tools. Hindustan Machine Tools Limited and Hmt Limited (Dec 31, 1998) ISBN-13: 978-0071346344, McGraw Hill Higher Education.

9.- Programación de actividades (de acuerdo con la presentación inicial)

El desarrollo del proyecto se prevé en dos etapas de un año de duración cada una. A continuación se describe la programación de actividades prevista en la presentación inicial del proyecto, de acuerdo a la metodología expuesta en el apartado anterior. En los apartados siguientes se detallan las actividades realizadas durante los dos años de vigencia del proyecto, justificando aquellos casos en los que, por distintas razones, se produjeron desvíos con respecto a la programación original

9.1.- Etapa I – Primer año

1.1. Rediseño de partes mecánicas y reductores de la versión laboratorio del autómata.

1.2. Mejorar y desarrollar la unidad central destinada al procesamiento de imágenes mediante la utilización de procesadores con mayor potencia de cálculo.

1.3. Desarrollo y montaje de la electrónica adicional para la integración de toda la electrónica del control del autómata, los diversos sensores a utilizar y las cámaras de alta velocidad con la unidad central de calculo

1.4. Investigación y desarrollo de nuevos paradigmas de procesamiento de imágenes, en cuanto a la detección de objetos, cálculo de distancia y captura de movimientos.

1.5. Pruebas experimentales con las cámaras de alta velocidad.

1.6. Desarrollo y ensayo de algoritmos en el laboratorio.

1.7. Documentación.

	Responsables	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
1.1	GS, NON, AM, MJK	X	X	X									
1.2	NMV,ARM	X	X	X	X	X	X	X	X	X	X		
1.3	GS, NON, AM, MJK								X	X	X	X	
1.4	GS, MFB,ARM	X	X	X	X	X	X	X	X	X	X	X	
1.5	GS, NON, AM, MJK				X	X	X	X					
1.6	GS, MFB						X	X	X	X	X	X	
1.7	FIS, HRT, NMV			X				X				X	X

9.2.- Etapa II – Segundo año

2.1 Estudios de algoritmos y métodos existentes del procesamiento digital de imágenes aplicados a autómatas programables.

2.2 Desarrollo y simulación de algoritmos avanzados de detección de objetos y cálculo de distancia.

2.3 Desarrollo y simulación de algoritmos basados en la captura de movimientos del autómata.

2.4 Ensayos y correcciones de los algoritmos desarrollados.

2.5 Investigación y desarrollo de algoritmos para incorporar los algoritmos al control del autómata.

2.6 Ensayos sobre el autómata de laboratorio y correcciones de los algoritmos desarrollados.

2.7 Documentación.

	Responsables	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
2.1	GS, MFB	X	X	X	X	X	X						
2.2	GS, MFB						X	X	X	X	X	X	
2.3	NMV, NON, AM, MJK	X	X	X	X	X	X						
2.4	NMV, NON, AM, MJK					X	X	X	X				
2.5	NMV, NON, AM, MJK								X	X	X	X	
2.6	Todos					X	X	X	X	X	X	X	
2.7	FIS, HRT, NMV							X	X			X	X

Responsables:

FIS Ing. Fernando I. Szklanny
NMV Ing. Nicolás Molina Vuistaz
GS Ing. Gustavo Sagarna
HRT Ing Hugo R. Tantignone
MJK Ing. Mario J. Krajnik
NON Ing. Nahuel O. Nieva
AM Ing. Alejandro Martínez
MFB Sr. Martín Ferreyra Birón
ARM Ing. Alberto R. Miguens

10.- Resultados esperados en cuanto a la producción de conocimiento:

El desarrollo de un autómata programable permitirá la ampliación de los conocimientos adquiridos de los integrantes del grupo de investigación en lo que hace a la tecnología del procesamiento digital de señales aplicado al control de sistemas electromagnéticos y el procesamiento digital de imágenes. Permitirá sentar las bases para futuros desarrollos en el área.

11.- Resultados esperados en cuanto a la formación de recursos humanos:

Se prevé la capacitación de alumnos de la materia de robótica de la Universidad Nacional de La Matanza con el fin de obtener resultados académicos óptimos entre la teoría y la práctica, incorporando posibles propuestas de mejoras del sistema.

12.- Resultados esperados en cuanto a la difusión de resultados:

Se prevé presentar el proyecto en desarrollo a distintas entidades empresariales que puedan estar interesadas en la evaluación y el apoyo a la tecnología desarrollada. Se espera poder lograr el compromiso de empresas, cámaras empresariales, etc, con el objeto de permitir continuar con desarrollos de tecnología que se ajuste a sus necesidades, mejorando la productividad y disminuyendo los costos de producción. Se prevé continuar con la publicación de trabajos en Congresos afines a la especialidad relacionados con los temas estudiados. A la fecha no se ha resuelto aún, dado que se trata de un proyecto nuevo, que continúa con los desarrollos realizados en proyectos anteriores, los eventos en los que se prevé participar. En función de los avances a lograr, podría intentarse una participación en congresos auspiciados por AADECA o cámaras empresariales afines, o en congresos académicos como los de Microelectrónica Aplicada, SASE, etc.

13.- Resultados esperados en cuanto a transferencia hacia las actividades de docencia y extensión:

El avance del proyecto permitirá la ampliación de conocimiento sobre procesamiento digital de señales aplicado al control de sistemas electromagnéticos, procesamiento digital de imágenes y conocimientos sobre autómatas programables. Estos conocimientos podrán ser incorporados por los docentes que estén vinculados a cátedras afines de las carreras de ingeniería actualmente existentes o por crearse. Obviamente, la incorporación de conocimientos por parte de los docentes tendrá fundamental importancia en lo que implica la trasferencia de dichos conocimientos a los alumnos de grado y posgrado en carreras afines.

Asimismo, y en cuanto se relaciona con actividades de extensión, se podrán organizar cursos de capacitación o de divulgación sobre temas asociados con la robótica industrial, el procesamiento de señales y la aplicación de los resultados obtenidos, orientando dichos cursos hacia diversas áreas de la comunidad universitaria.

14.- Resultados esperados en cuanto a la transferencia de resultados a organismos externos a la UNLaM:

Los resultados que se esperan en este caso tienen que ver con los destinatarios posibles del proyecto:

- empresas industriales que puedan analizar la incorporación de autómatas a sus procesos de fabricación,
- escuelas técnicas y entes académicos que se dediquen a la enseñanza de temas asociados con la robótica industrial.
- Fabricantes nacionales que puedan promover la producción de servomotores y reductores de alta precisión para el campo de la Robótica, logrando así una importante sustitución de importaciones en áreas de alto valor agregado.

15.- Vinculación del proyecto con otros grupos de investigación del país y del extranjero:

Se propone seguir en contacto con aquellos grupos de investigación del país que forman parte de la red Vitec y que desarrollan proyectos de investigación relacionados con los objetivos del presente. En caso de presentarse la posibilidad, se establecerán contactos con otros grupos de investigación o investigadores del extranjero.

16.- Memoria técnica

16.1.- Año 2016

Al igual que ocurriera en otros proyectos en los que participa el grupo de investigación en lógica Programable, hubo, durante 2016, una importante limitación en las actividades originalmente dedicadas al proyecto, lo que hizo que muchas de las tareas originalmente planeadas para este año se vieran postergadas para 2017.

Entre las razones que motivaron la mencionada restricción de tareas, se pueden plantear:

- Si bien se aliviaron las limitaciones en las posibilidades de importación de algunos elementos necesarios para el desarrollo del sistema, la disponibilidad local de los mismos siguió siendo insuficiente, por cuanto, hasta bastante entrado el año 2016, no se habían dispuesto modificaciones a los regímenes de importación vigentes. En consecuencia, esta limitación requirió la necesidad de compra a proveedores extranjeros. En este caso, uno de los inconvenientes surgidos fue la decisión de muchos proveedores del exterior de limitar sus envíos a la Argentina. En otros casos, en los que no existió esa limitación, aparecían los inconvenientes de incontrolables demoras en el despacho a plaza de los mismos, a pesar de la puesta en vigencia del servicio puerta a puerta. Esto hizo que varias de las compras previstas para el 2016 se postergaran para el próximo año, cuando la situación fuese más manejable.

- Desde el punto de vista del desarrollo general del proyecto, se produjeron importantes demoras en el llamado a presentación de los mismos, lo que derivó en una cadena de eventos que implicó sucesivamente la demora en las evaluaciones de los mismos, la demora de la aprobación de dichas evaluaciones, y como consecuencia de ambos, la demora en la acreditación de los fondos correspondientes al año 2016. Debe hacerse notar que los mismos, como consecuencia de todo lo mencionado, fueron acreditados prácticamente sobre el final del mes de noviembre. Esto hizo que, con la inminente finalización del período académico, y la limitación típica de la época de las festividades, se complicaran las posibilidades de compra de elementos con plazos de entrega razonables por parte de los proveedores.

- En esa situación, y con el objeto de independizar en alguna manera las necesidades del grupo de investigación respecto de proveedores externos, tanto en lo que tiene que ver con este proyecto, como con otros en los cuales se hallan involucrados los integrantes del mismo, se optó por una inversión que permitiera al grupo desarrollar partes mecánicas sin depender de dichos proveedores externos, lo que llevó a la adquisición, por parte del grupo de investigación, de una impresora 3D para el desarrollo y producción de partes y elementos mecánicos.

- Asimismo, y en función de las necesidades de capacitación permanente de los integrantes del grupo de investigación se priorizó la participación del Ing. Gustavo Sagarna en la Escuela de Sistemas Embebidos llevada a cabo en Tucumán durante el pasado mes de diciembre, por lo que

una parte del subsidio recibido se utilizó para financiar la asistencia al mismo del mencionado profesional. El informe del Ing. Sagarna se acompaña como anexo.

- Para completar esta serie de situaciones, durante gran parte del año 2016 (al igual que ocurriera en años anteriores) varios de los integrantes del grupo de investigación debieron repartir sus tiempos de actividad profesional dentro de la Universidad entre el proyecto de que se trata en este informe y otros proyectos, requeridos con urgencia por las autoridades del Departamento de Ingeniería e Investigaciones Tecnológicas y del Rectorado de la Universidad. Esto hizo que los tiempos disponibles para la actividad de investigación se redujeran con respecto a los tiempos estimados en la propuesta original.

- A pesar de lo planteado, se pudieron llevar adelante un conjunto de tareas que no requirieron de un aporte monetario, y que se fueron realizando aun cuando no se había definido todavía la continuidad de los proyectos de investigación.

- Como conclusión, se considera que se logró un avance satisfactorio en temas asociados con los algoritmos de detección de objetos y cálculo de distancia, dado que se pudo desarrollar un autómata apto para las pruebas de dichos algoritmos. Si bien se usaron cámaras de buena resolución y una computadora personal para correr los algoritmos desarrollados, queda pendiente para el año 2017 la compra de las cámaras de alta velocidad y de los componentes para el desarrollo de la unidad central de procesamiento de imágenes, a fin de mejorar la performance de los algoritmos.

Entre las tareas desarrolladas durante el año 2016 se pueden detallar:

a.- El avance, por parte del Sr. Martín Ferreyra Birón y del Ing. Alberto Miguens, en el estudio de algoritmos relacionados con las técnicas de reconocimiento de objetos. Los informes individuales de los mencionados integrantes, correspondientes al año 2016 se incluyen dentro del presente informe final.

b.- La construcción de un nuevo esquema de brazo, para lo cual aportó su experiencia en la parte mecánica el Ing. Alejandro Pérez Arauz, recientemente incorporado al grupo de investigación. A través de elementos donados por el mencionado profesional, que permitieron un avance sin necesidad de gastos para la compra de materiales, se pudo lograr un nuevo modelo que perfecciona los planteos anteriores en lo que hace a la mecánica del mismo. Por razones que se detallan más adelante, dicha propuesta no pudo ser utilizada finalmente en el avance del proyecto

c.- Ante la falta de certezas sobre la continuidad de los proyectos de investigación, los Ing. Nahuel Nieva y Alejandro Martínez dedicaron la primera parte del año 2016 a una investigación sobre controladores lógicos programables (PLC). El resultado de dicha investigación culminó con el desarrollo de un controlador lógico programable, de bajo costo, que, entre otras cosas, podrá ser utilizado para el control del brazo mecánico. La actividad de los mencionados profesionales derivó, una vez habilitado el llamado a presentación de proyectos, en una propuesta de proyecto de investigación CYTMA, oportunamente aprobada por la Universidad, por lo que su actividad

orientada al proyecto al que se refiere este informe fue menor que la planteada originalmente. El proyecto CYTMA 034, al que se hace referencia, está encaminado y en construcción del prototipo de producción.

16.2.- Actividades de investigación referidas a la detección de objetos

Durante el año 2016 se planteó el desarrollo e implementación del trabajo de Stefan Hinterstoisser y otros [29], en el que se presenta un método (DOT) para detectar objetos tridimensionales en tiempo real, bajo una cierta cantidad de ruido, que no requiere una etapa de entrenamiento demasiado costosa y que puede detectar objetos sin textura (problema que si se observó, de manera empírica, al utilizar SURF).

En base al método mencionado en el párrafo anterior se desarrolló un algoritmo, con el que se obtuvo un resultado satisfactorio en un principio, con el inconveniente en que la forma en la que fue escrito no contemplaba casos de optimización y compatibilidad. El código programado era de difícil lectura, y no era posible a corto plazo implementar el uso de una GPU o utilizar, como proponen los autores, las instrucciones y registros que otorga la tecnología MMX. Debido a esto se comenzó nuevamente la programación del algoritmo, esta vez en forma mucho más cuidadosa, teniendo en cuenta todo lo aprendido en la primera implementación.

En esta nueva versión en la que se trabajó se pudo implementar el uso de la tecnología MMX como propone la investigación mencionada anteriormente. Sin embargo, la implementación de MMX no es compatible en todas las arquitecturas (solo en la x86), siendo en parte importante portabilidad del algoritmo, además de no otorgar la velocidad de detección esperada. Es por este motivo que para poder obtener mayores velocidades en el reconocimiento de los objetos se comenzó con el estudio e investigación de la biblioteca multiplataforma OpenCL, la que surge como resultado de un esfuerzo conjunto de varias empresas importantes como Apple, Nvidia, Amd, Xilinx, etc para poder paralelizar procesos en CPU's y GPU's indistintamente.

Cabe destacar que para comprobar la eficiencia del algoritmo se han realizado algunas pruebas en ambientes naturales dando resultados en principio aceptables pero no con el grado de aserción esperado. Esto significa que si bien el algoritmo funciona y ya es conocido, después de la etapa de implementación de OpenCL se deberá hacer un esfuerzo para mejorarlo y así satisfacer nuestras necesidades de detección.

Capturas de imágenes de los resultados del algoritmo

En las siguientes imágenes se puede observar el comportamiento de la implementación del algoritmo propuesto por Hinterstoisser y otros. Como ya se expresó anteriormente la implementación realizada es pasible de mejoras para tener mejores resultados.

En la figura 1 se puede observar la detección de una envase de adhesivo vinílico y una piedra de desgaste. En el caso del envase de adhesivo vinílico se puede detectar con cierto nivel de oclusión sin mayores inconvenientes.

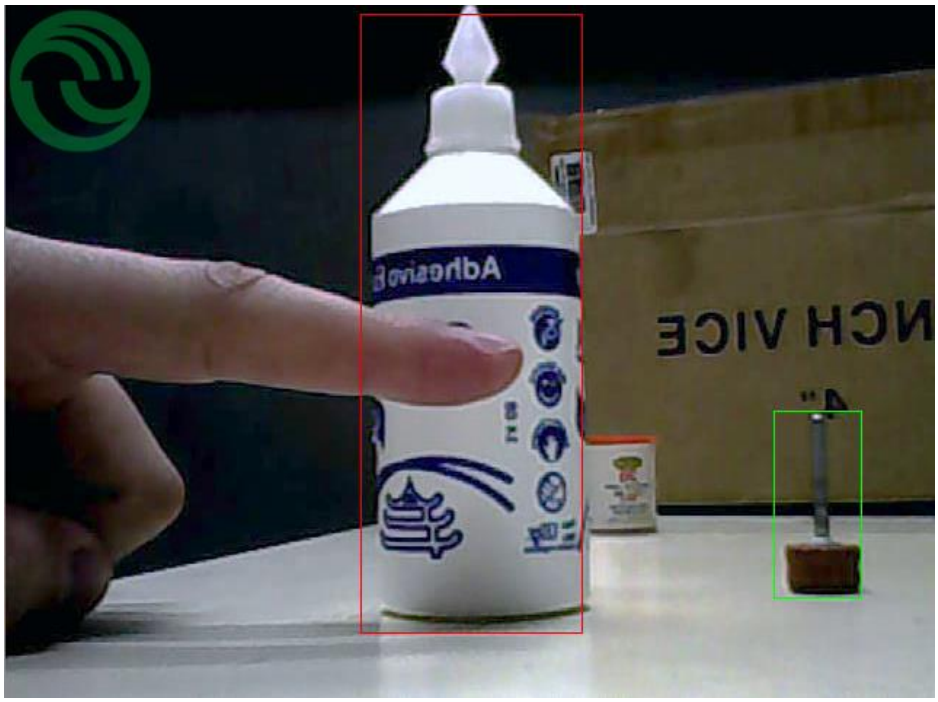


Figura 1 Detección de dos objetos con ruido de fondo

A modo de ejemplo en el caso del envase del adhesivo vinílico se utilizaron estas plantillas (“templates”)

Figura 2 Plantillas (“templates”) del envase del adhesivo vinílico

En el caso de la figura 3 se puede observar como el algoritmo puede detectar un objeto ocluyendo a otro objeto, siendo ambos detectados correctactamente.

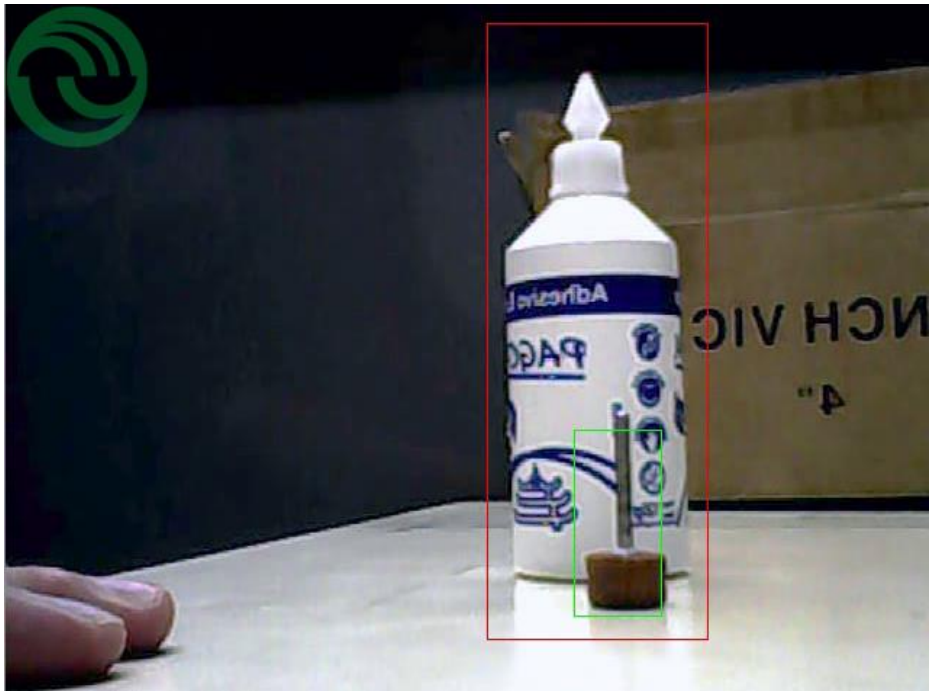


Figura 3 Detección de dos objetos superpuestos

En la figura 4 se puede observar cómo se detecta el envase de adhesivo vinílico en movimiento.

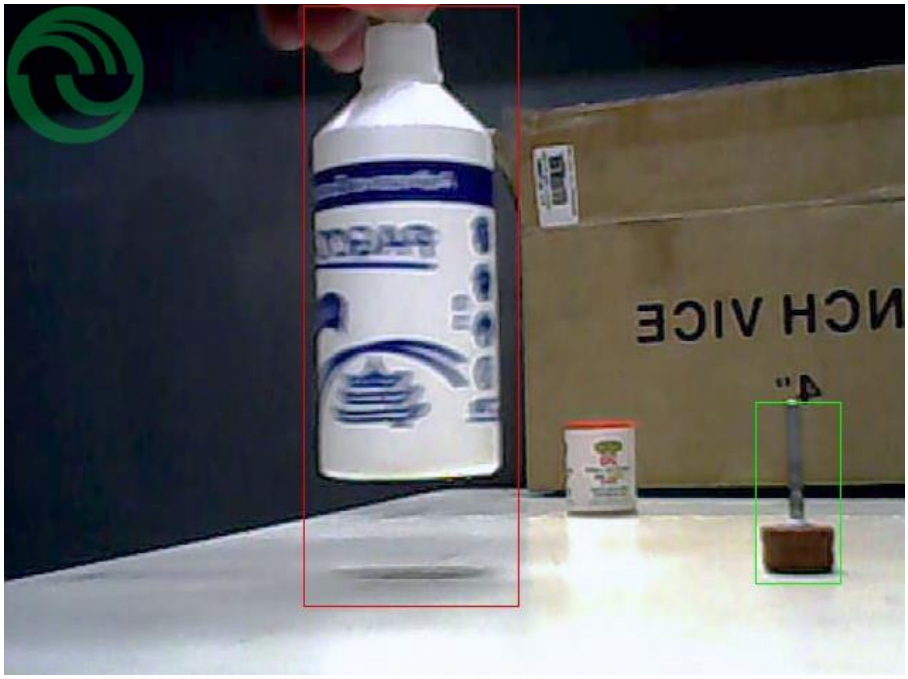


Figura 4 Detección de los objetos variando su posición para evaluar la respuesta del algoritmo en función del movimiento

En la figura 5 se puede observar como el algoritmo programado falla al no encontrar el objeto. Este es uno de los problemas a resolver.

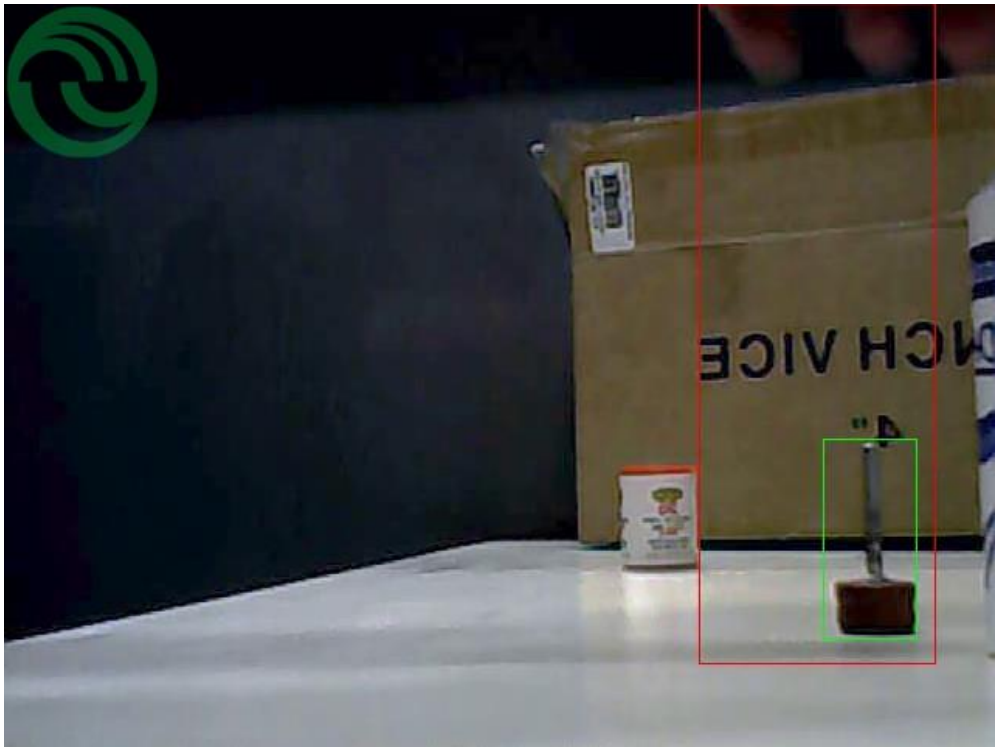


Figura 5 Error en la detección de un objeto cuando este no se encuentra

Código fuente:

El código fuente fue programado en C++ 11 y utilizando la biblioteca OpenCV bajo un entorno Linux Ubuntu 16.04. El código se dividió en dos clases: Escena y Objeto. La clase Objeto fue programada para aprender el objeto que se desea buscar a partir de las imágenes “template” y crear una descripción del mismo. La clase Escena tomando la descripción del objeto a buscar y un fotograma (la escena), busca el objeto dentro del mismo devolviendo la posición donde se encuentra. Este código fuente cambio en el año siguiente , por lo tanto no se anexa en este informe.

Algoritmo para ubicar un objeto

Del análisis realizado sobre el trabajo [29] ya mencionado se pudo plantear una referencia para diseñar el programa que permite extraer la posición de un objeto dentro del plano *(x,y)* de la imagen tomada por una cámara.

Adicionalmente, se realizó el estudio de alternativas para lograr una detección tridimensional dado que el manipulador necesita trasladar piezas entre cualquier punto espacial, por lo que es necesario detectar también la profundidad 'z'.

Se intenta realizar la medición de la tercera dimensión por medio del 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 utilizarlo para medir la profundidad.

A continuación se plantea una breve explicación del ensayo:

La técnica propuesta para detectar el eje de profundidad es la estereoscópica. La misma se basa en el sistema de visión humano. Se interpreta que el mecanismo utilizado para la estimación de la profundidad por el cerebro está basado en la disparidad horizontal de las imágenes visualizadas.

Dada la separación horizontal de los ojos *b* (en promedio de 5 a 6 Cm) las imágenes presentaran una disparidad mayor para objetos cercanos y al contrario para los lejanos. Es decir, mientras mayor sea la diferencia de distancia horizontal para un mismo objeto en las dos imágenes, significa que éste se encuentra más cerca de los ojos.

A nivel matemático, este mecanismo puede ser resuelto a través del método de triangulación.

En la figura 6 se puede observar un esquema básico, en el cual hay 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 debería ser la misma en ambas) se encuentran 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.*

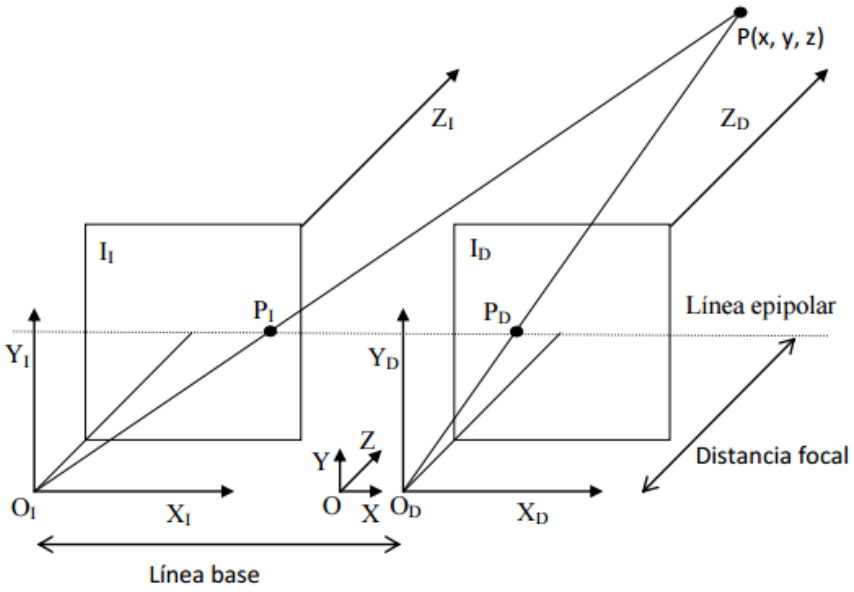


Figura 6. – Diagrama tridimensional para el cálculo de la distancia del punto P

A continuación, en la figura 7, se pueden observar los puntos PI y PD 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 PI y PD tiendan a ubicarse sobre el eje ZI y ZD

La disparidad del punto se calcula como XI - XD = d. Y en el ejemplo de un objeto ubicado en el infinito del eje Z hará que d valga 0.

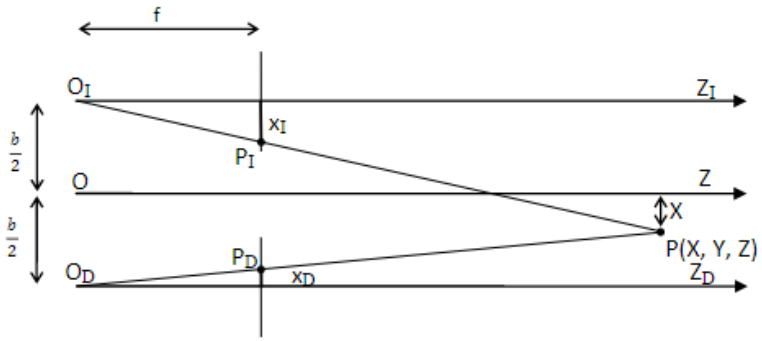


Figura 7 – Diagrama bidimensional para el cálculo de la distancia del punto P

Cálculo de la distancia del objeto sobre eje Z:

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

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

En la ecuación *Imagen Derecha* se asigna un signo negativo a XD para compatibilizar los sistemas de coordenadas de los ejes OI y OD.

Disparidad del punto P:

$$x_I\text{-}x_D = d$$

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)

El sistema es tan preciso como sea la detección bidimensional del objeto en cada imagen.

Para poner en práctica lo planteado se realizaron pruebas a nivel práctico, utilizando dos cámaras idénticas y una aplicación que procese las imágenes y realice los cálculos.

La aplicación utilizada se denomina Octave, es una herramienta de software libre que tiene una alta compatibilidad con el reconocido software matemático Matlab. De hecho, las bibliotecas de uno se pueden utilizar en el otro.

Se realizó un aprendizaje para la implementación del sistema estereoscópico en Octave utilizando el tutorial desarrollado por McCormick [31], quien propone un código (adjunto al final de este documento) que está divido en dos partes. El mismo, por un lado, realiza la identificación de los objetos en cada una de las imágenes del par estereoscópico, para luego calcular las diferencias de distancias horizontales. (disparidad). Luego, por medio de un sistema de representación bidimensional, se realiza lo que se denomina mapa de disparidad, que no es más que la representación en un plano del valor de disparidad en cada punto, codificado en un rango de colores.

En las figuras 8 a 10 se pueden observar las imágenes tomadas, y a continuación el mapa de profundidad calculado:

Figura 8. – Imagen cámara izquierda

Figura 9. – Imagen cámara derecha

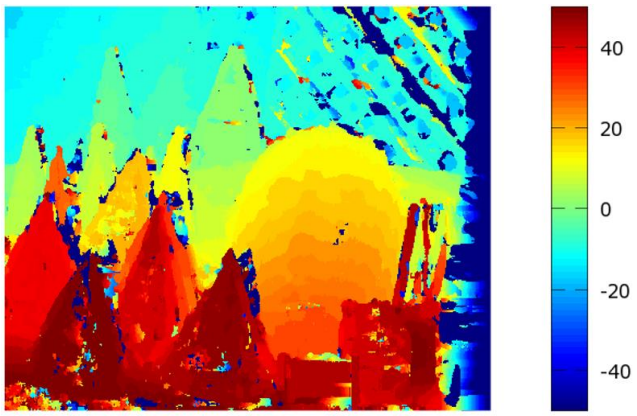


Figura 10. – Mapa de profundidad.

Se puede observar en la figura 10 a nivel gráfico la diferenciación de profundidad entre los objetos; el método gráfico utilizado les asigna a los objetos cercanos un matiz rojo y a los lejanos uno azul.

En el ensayo de laboratorio se pudo verificar que la calibración de las cámaras es esencial, dado que la falta de la misma genera errores que a nivel gráfico se pueden identificar como ruido.

A continuación, se da un ejemplo:

En la figura 11 y 12 se puede ver el par estereoscópico tomado con un conjunto de cámaras no calibradas.

Figura 11. – Imagen cámara Izquierda

Figura 12. – Imagen cámara derecha.

Dando el resultado de la figura 13:

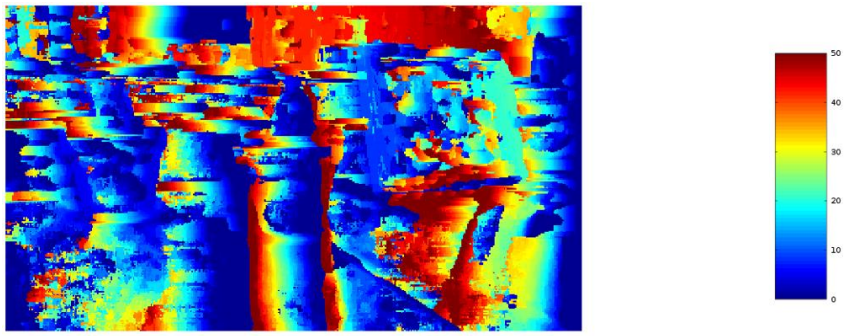


Figura 13. – Mapa de profundidad.

Por consiguiente, se considera necesario desarrollar una técnica de calibración de las cámaras.

En el transcurso de la investigación se pudo encontrar una herramienta que funciona para el programa Matlab. Por consiguiente el próximo paso propuesto es exportarla a Octave para su estudio, o en un mejor caso, desarrollar un método propio; por ejemplo, la utilización de un plano patrón cuadriculado paralelo a la lente de las cámaras y a una distancia conocida de ellas.

Adicionalmente, este método no calcula la distancia de los objetos en forma directa, si no que se consigue un valor proporcional a la profundidad (el valor de disparidad). Es necesario calcular el

valor Z para el objeto de interés, y para ello se utilizará la ecuación propuesta en párrafos anteriores:

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

Elección de la técnica estereoscópica:

El uso de la técnica estereoscópica puede resultar ventajoso desde el punto de vista de la simplicidad mecánica y económica, ya que con un sensor compacto (dos cámaras relativamente pegadas) se puede analizar un amplio rango del espacio sin generar obstrucciones. También se considera útil este estudio desde el punto de vista del aprendizaje para futuros desarrollos, como, por ejemplo, herramientas para la ayuda de personas con problemas de visión.

De todos modos, se considera importante la existencia de otros métodos a estudiar para el caso particular del brazo robot, como por ejemplo la complementación del análisis con un sistema Lidar. Se conoce con esta sigla a un dispositivo que permite determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser pulsado. La distancia al objeto se determina midiendo el tiempo de retraso entre la emisión del pulso y su detección a través de la señal reflejada.

16.3.- Año 2017

Gran parte del año 2017 fue dedicada, tal cual lo previsto, a los temas asociados con la detección y procesamiento de imágenes para el autómata en desarrollo, temas considerados como el objetivo fundamental del presente proyecto. Por el valor que se le asigna a este punto de la investigación, en apartados posteriores se desarrolla, en forma detallada, el trabajo realizado.

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 bibliotecas de procesamiento gráfico.

En cuanto a la electrónica destinada al control de motores, se completó el desarrollo del PLC destinado al control de motores del autómata y lectura de parámetros fisicos mediante sensores, dentro del marco del proyecto CYTMA 034 “Controlador Lógico Programable de bajo costo para aplicaciones robóticas”. Se desarrolló una segunda versión mejorada y se finalizó el diseño y pruebas de las bibliotecas destinadas al control de los periféricos.

En cuanto a la mecánica del autómata, se desarrolló una versión imprimible del brazo robot, con el objeto de bajar los costos de desarrollo del mismo y facilitar la trasferencia de la tecnología a unidades académicas y empresas. Esta versión pudo llevarse a cabo mediante la utilización de una impresora 3D adquirida por el grupo de investigación durante el periodo 2016. Diversos problemas ocurridos en el proceso de impresión impidieron completar la misma, lo que se espera concretar para mediados del presente año.

17.- Desarrollo de algoritmos para la detección y procesamiento de imágenes.

17.1.- Ensayos y pruebas preliminares

Como ya ha sido dicho, con el objetivo de avanzar en el desarrollo de algoritmos para la detección y procesamiento de imágenes, se procedió a la adquisición de una placa de desarrollo Nvidia Jetson TK1, la que se ilustra en la figura 14. La placa en cuestión está construida sobre la base de un procesador ARM® Cortex™-A15 y una unidad destinada al procesamiento gráfico de NVIDIA Kepler GPU con 192 núcleos CUDA. Cabe mencionar que CUDA es una arquitectura de cálculo paralelo de NVIDIA que aprovecha la gran potencia de la GPU (unidad de procesamiento gráfico) para proporcionar un incremento extraordinario del rendimiento del sistema.

Como inconveniente, presenta una incompatibilidad con OpenCL, por lo que hubo que considerar la utilización de la arquitectura CUDA. Después de instalarse todo el entorno de desarrollo se comenzó con programación del algoritmo DOT.

Fig. 14.- Placa Nvidia Jetson TK1

En un primer intento se compiló el código realizado anteriormente con mínimas modificaciones para ejecutarlo en la placa de desarrollo. Las consideraciones iniciales, basadas en que gran parte del código hace uso de la biblioteca OpenCV, y en que varios métodos utilizados están optimizados para la ejecución en una GPU, prometían o sugerían un muy buen rendimiento. Pero esto no fue así. La detección en un video de prueba superaba el tiempo de procesamiento, por mucho, a la ejecución del mismo analizando el mismo en una CPU.

Visto y considerando este inconveniente se comenzó a analizar el código con detenimiento. Se observó que la manera en cómo se estaba intentando utilizar la GPU no era la correcta, por lo que 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 primero que más demoraba en ejecutar, era el cálculo de gradientes. Este método se debe ejecutar fotograma a fotograma.

En el cálculo de gradientes las tareas a realizar son las siguientes, cada una de ellas depende de la anterior:

- Se convierte a escala de grises la imagen del fotograma a analizar.
- Se calculan el filtro Sobel por X e Y
- Se calcula el gradiente a partir de los filtros anteriores
- 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
- 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 utilizó la función optimizada para CUDA de OpenCV de manera correcta y debido a que se tienen que hallar los componentes en X y en Y, estas tareas se ejecutaron en paralelo. Sin embargo, si bien se obtuvo una mejora sustancial, la comprensión y el análisis de los tiempos globales de demora llevó a los responsables de la investigación a la conclusión de que la mejor manera de implementar el algoritmo DOT no era utilizando los métodos optimizados ofrecidos por OpenCV, sino a través del desarrollo de un kernel. Esto es 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 en la ejecución del algoritmo (“*overhead”)*. Por lo tanto, utilizar una GPU sin tener en cuenta lo anterior produce demoras y las mismas deben ser minimizadas.
- Se debe buscar en la imagen máximos locales en regiones de interés de NxN pixeles. En primer lugar OpenCV no ofrece una función optimizada que utilice CUDA para realizar esta tarea y en el caso de ofrecerla se debería tener en cuenta el tiempo de transferencia entre la memoria principal y la memoria de la GPU.

Por todo esto se procedió al diseño de un kernel que pudiera acelerar y hacer uso correcto de la GPU y así hallar los gradientes de manera rápida y correcta. El enfoque utilizado para resolver el problema fue dividir y solucionar el mismo en etapas parciales, debido a la falta de experiencia de programar en una GPU y en CUDA en particular.

Luego de comprender el funcionamiento y la programación en CUDA lo primero que se realizó fue desarrollar un kernel que permitiera aplicar el filtro Sobel a una imagen utilizando la GPU. Este paso fue fundamental ya que en un kernel se solucionaban los puntos 2 y 3 de la lista de tareas mencionada anteriormente.

Seguidamente se ensayó el comportamiento del kernel en el caso de que lo tratado fuera un video y los resultados fueron muy alentadores: en un tiempo mucho menor de lo esperado la aplicación del

filtro era calculado. Además se comprobó también que la placa aceptara el uso de una webcam la cual se utilizó con el algoritmo y funcionó de manera correcta. Por lo tanto la lista de tareas que hasta ahora se resume son las siguientes:

- Se convierte a escala de grises la imagen del fotograma a analizar.
- Se calculan el filtro Sobel por X e Y, y se calcula el gradiente (GPU).
- 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
- Se almacena el resultado.

Fig. 15.- Imagen de prueba para verificar el filtro Sobel, utilizando la típica imagen de Lena.

El paso siguiente consistió en la búsqueda del máximo gradiente en cuadrados 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 y así trabajar la imagen resultante del filtro anterior en cuadrados NxN, pudiendo hallar paralelamente los máximos de cada bloque. Esto es correcto pero el hecho de utilizar un único hilo para buscar el máximo gradiente desperdiciaba el poder de procesamiento de la GPU. Es por eso que después de investigar y 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: Para hallar un máximo se deben utilizar comparaciones. Si se utiliza un solo núcleo o un solo 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, es

decir la mitad, 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. La figura 16 ejemplifica de una manera clara el concepto de reducción que se programó.

A posteriori de haberse hallado el máximo, en el mismo kernel se programó el cálculo para transformar y expresar el ángulo medido en un byte, tal como lo proponen los autores del algoritmo DOT. Por lo tanto la lista de tareas que se resume hasta ahora son las siguientes:

- Se convierte a escala de grises la imagen del fotograma a analizar.
- Se calculan el filtro Sobel por X e Y , y se calcula el gradiente (GPU)
- 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 guarda el resultado (GPU).

Por último, debido a que existen demoras entre la ejecución de ambos kernels, estos se combinaron en uno solo. Todas estas mejoras dejan así un aumento de velocidad realmente considerable respecto a las mismas tareas ejecutadas en una CPU o utilizando, en parte la GPU, a través de OpenCV.

Ya solucionado el problema generado por la demora del método que calcula los gradientes, se siguió 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.

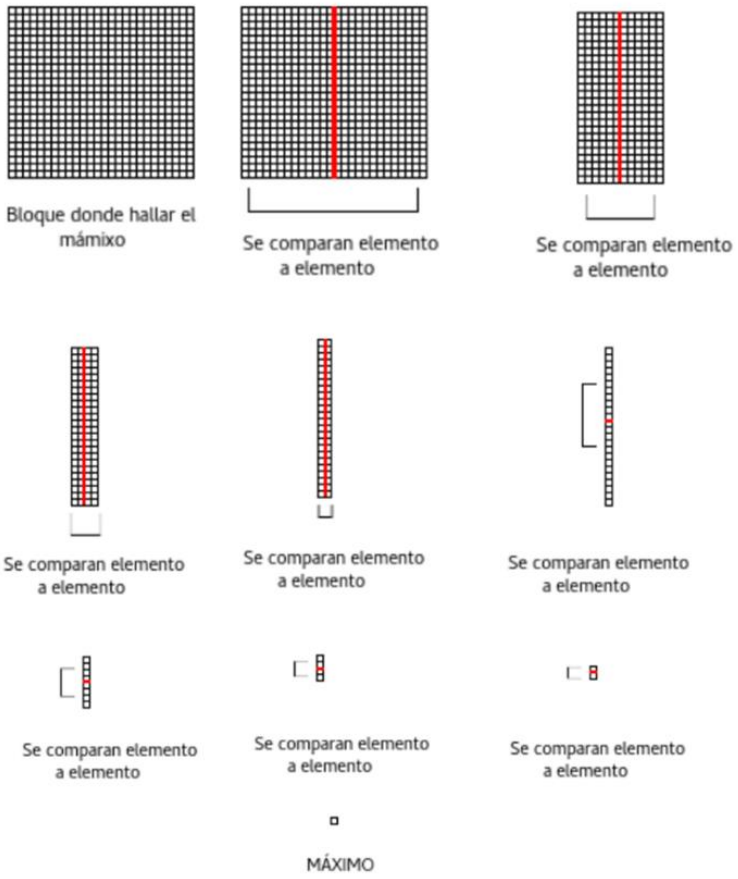


Fig. 16.- Ejemplificación de una reducción de una matriz utilizando una GPU

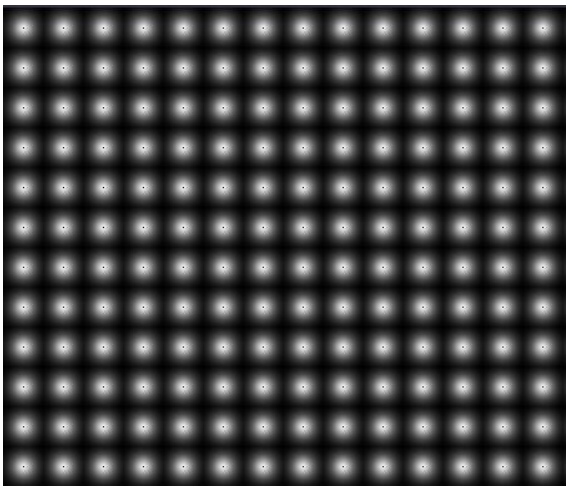


Fig. 17.- Porción de la imagen de prueba para la búsqueda de máximos (versión no optimizada). Se pueden observar los máximos hallados que se encuentran en los puntos negros en cada subdivisión.

Se debe tener en cuenta que la placa Nvidia Jetson TK1 posee un micro controlador 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 decodifica 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. El esquema de la figura 17 muestra cómo se ejecutan los diversos algoritmos.

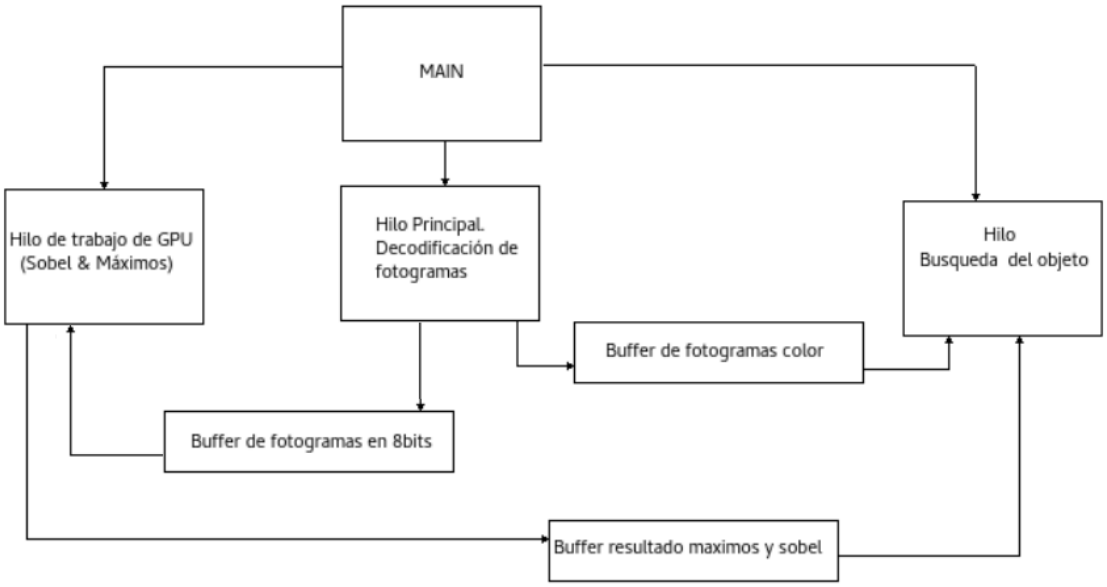


Fig. 18.- Esquema de ejecución de algoritmos

Solamente resta optimizar la búsqueda del objeto con la GPU denominado “Match Template”.

17.2.- Optimización de velocidad de la búsqueda

En primer lugar a priori se entiende que no se puede sobrecargar más a la GPU en este punto. Por lo tanto la búsqueda del objeto se debe realizar en la CPU. El algoritmo DOT plantea que lo que se tiene realizar para encontrar un objeto es buscar la coincidencia más probable entre la escena y los objetos que se aprendieron. Para poder hacer esto se necesita utilizar una ventana deslizante (que recorre toda la escena) y busca 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 ángulos 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)
```

Es evidente que la complejidad computacional es alta y de alguna manera habría que minimizarla. Después de observar la manera en cómo funcionaba el algoritmo, se llegó a la siguiente conclusión: Se podría aumentar la eficiencia del algoritmo si solamente se comprobara la coincidencia de ángulos solo en los lugares donde en la plantilla realmente existen ángulos y estos no son indefinidos 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 (en ese caso se estaría bajando la resolución de búsqueda).

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 todo plantilla se guardará solamente el valor del/los ángulos y la posición en la plantilla. 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 valido 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 sustanciales, aunque la pregunta es: Si bien la GPU está ocupada, no está completamente ocupada todo el tiempo ¿Sería factible realizar la búsqueda de coincidencias ("match template") en la GPU? ¿Sería más rápido?

17.3.- El problema de los cambios drásticos de reconocimientos

El algoritmo DOT no es perfecto y en las pruebas se observó más de una vez que quizás por falta de plantillas de objetos la detección no se realizaba en el lugar correcto. Aun así se entiende que una manera de evitar este comportamiento consiste en suavizar de alguna manera la detección brusca, teniendo en cuenta que los objetos no pueden aparecen en otro lugar de un momento a otro. Para solucionar este problema se implementó la siguiente solución: Si se detectó un objeto en una posición que supera un determinado porcentaje del lugar detectado, de manera progresiva se irá corriendo la región de interés donde se supone que esta el objeto en dirección de la nueva región de interés. De esta manera se obtuvo un buen resultado como se puede observar en la imagen a continuación, a la izquierda se encuentra la detección sin el método implementado para evitar los cambios drásticos de reconocimientos y a la derecha la detección con el método implementado.

17.4.- El problema de las falsas detecciones

En el transcurso de los ensayos del algoritmo con el video de pruebas se pudo observar que, si bien existía un mejor desempeño respecto al algoritmo programado el año anterior, seguían ocurriendo errores en la detección. Analizando el problema se arribó a la siguiente conclusión: el programa que se dedica a aprender los objetos, si bien estaba bien programado, no era el correcto. El

inconveniente surge a partir de que en ese programa el filtro Sobel utilizado era el proporcionado por OpenCV. Este algoritmo es ligeramente diferente al programado en la GPU, y al comparar los ángulos en la detección existían diferencias. Además también existía una diferencia sustancial en la manera de hallar la orientación de los ángulos entre el algoritmo programado en la GPU y en el programa que se encarga del aprendizaje, diferencia que fue subsanada.

17.5.- Pruebas en ambiente real

A continuación se presentan algunas imágenes de momentos en los que el algoritmo produjo resultados apropiados, en videos que se tomaron en ambiente real con ruido de fondo. Es importante recalcar que fueron algunos momentos debido a que no se utilizaron demasiadas plantillas para realizar la detección. Las imágenes de las escenas son grises debido a que se consume demasiada memoria si estas se las carga a memoria en color. El exceso de consumo de la misma generaba como consecuencia que el sistema operativo "matara" al proceso de detección.

Fig. 20. Algoritmo reconociendo el envase de pegamento

Fig. 21. Algoritmo presentando una falla en la detección

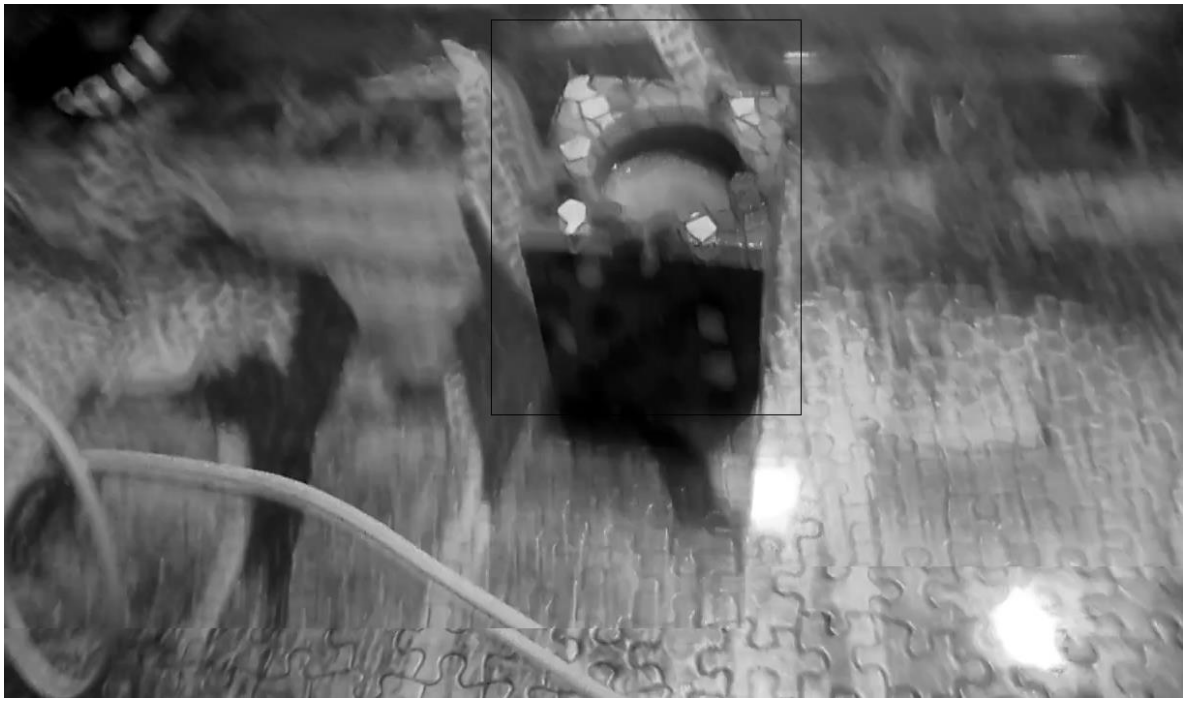


Fig. 22. Algoritmo detectando el cubo

Fig. 23. Algoritmo falla por falta de plantillas. En este video en particular la tasa de fallas fue muy alta. El algoritmo malinterpretaba el ruido de fondo (rompecabezas)

Fig. 24. Tomando un video donde se reconocía el cubo en la implementación en CPU se trató de reconocer el cubo. En rojo la detección de esta implementación y en azul la de la anterior implementación. Cabe destacar que se utilizaron diferentes plantillas

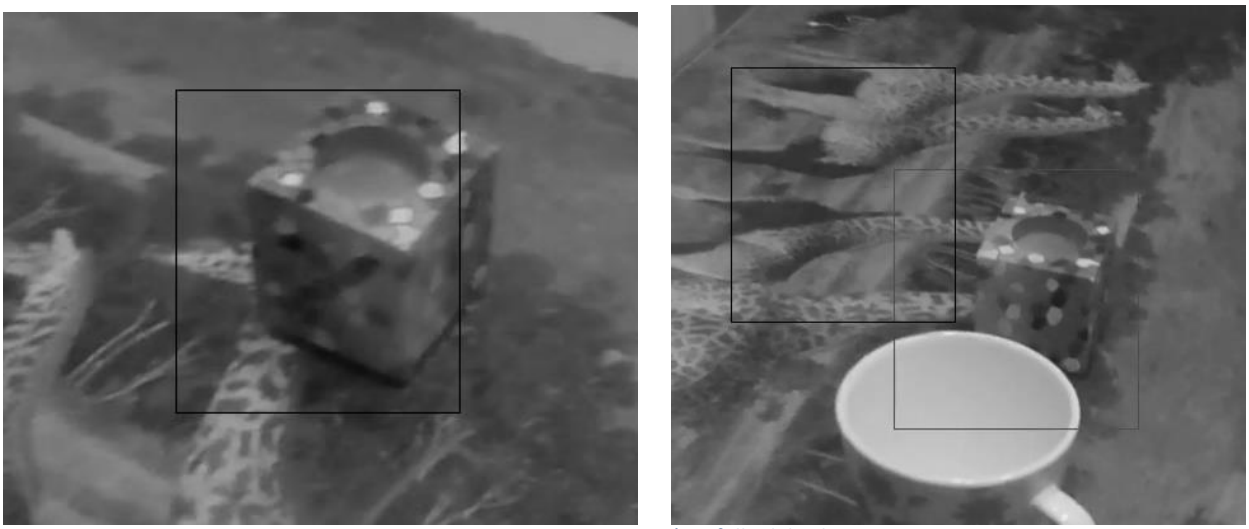


Fig. 25. Detección y fallo del cubo

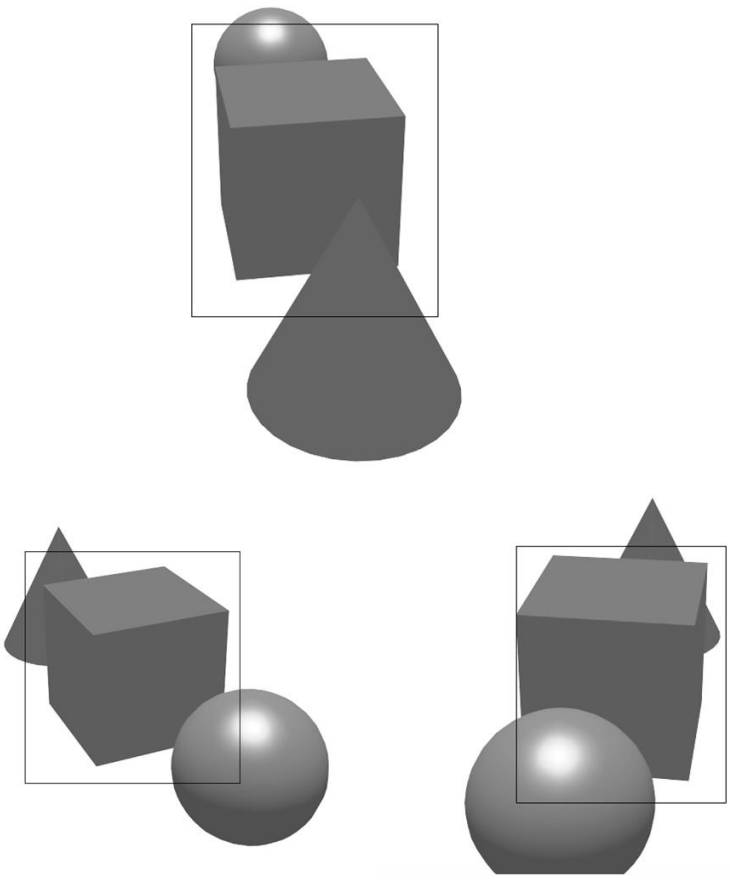


Fig. 26. Detecciones correctas en el video de prueba que se utilizó. Observar la detección con oclusiones

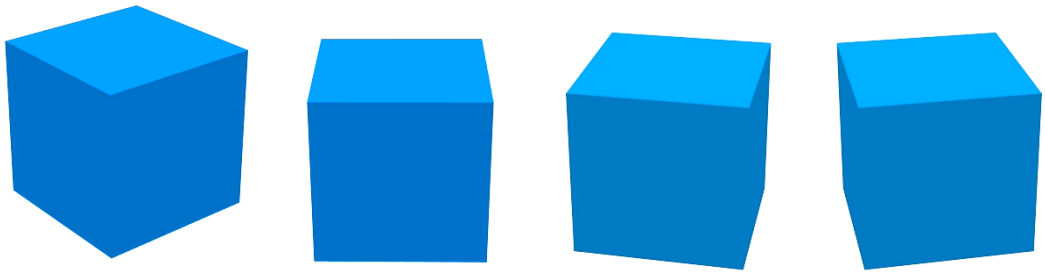


Fig. 26. Plantillas que se utilizaron para la detección de las imágenes anteriores. El video de prueba no tuvo falsos positivos.

17.6.-Calibración de las cámaras (Detección de profundidad)

De las actividades desarrolladas durante el año 2016 había quedado pendiente un método para calibrar las cámaras, por lo que se procedió a su implementación. Este desarrollo se llevó a cabo de forma paralela al desarrollo del algoritmo DOT, por lo cual, para algunos puntos de este desarrollo, se utilizó otro método de búsqueda y detección de objetos.

Para la calibración de las cámaras se utilizó en primera instancia el método de alineación intrínseca, que implica tomar, en tiempo real, videos de las cámaras del par estereoscópico y lograr una alineación lo más precisa posible mediante la utilización de puntos comunes de las imágenes como referencia.

Para ello se podría utilizar una escena patrón convenientemente diseñada: por ejemplo, un plano patrón símil tablero de ajedrez. La realidad es que en la práctica, la observación de puntos comunes entre dos imágenes de una misma escena es suficiente ya que las correcciones finales las realiza el algoritmo calibrador. Lo ideal es que en un principio el ajuste sea realizado a grandes rasgos lo más preciso posible para que luego el algoritmo de calibración tenga mejores resultados.

Una vez hecho este ajuste, se comienza con el cálculo de las matrices de calibración, para lo que se utiliza la biblioteca de OpenCV que contiene clases para calibración y reconstrucción de imágenes 3D: “Camera calibration and 3D reconstruction”. Para la calibración se utilizó la aplicación *calibrate_cameras*, quel forma parte de la biblioteca StereoVision (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 tomas en distintas perspectivas del mismo. 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 contienen 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 identificación de objetos. Esto se puede resolver con el algoritmo DOT que se está desarrollado. Más allá de dicho algoritmo posiblemente se pueda complementar con alguno de “*block matching*” para que una vez identificado un mismo objeto en cada una de las imágenes de manera individual, se puedan corregir las posibles diferencias en la ubicación bidimensional utilizando una ventana de búsqueda mucho más reducida que en el caso de

una escena completa. Claro está que todo proceso extra atenta también contra la performance del sistema. Habrá que evaluar los beneficios y buscar, si es necesario, una solución de compromiso.

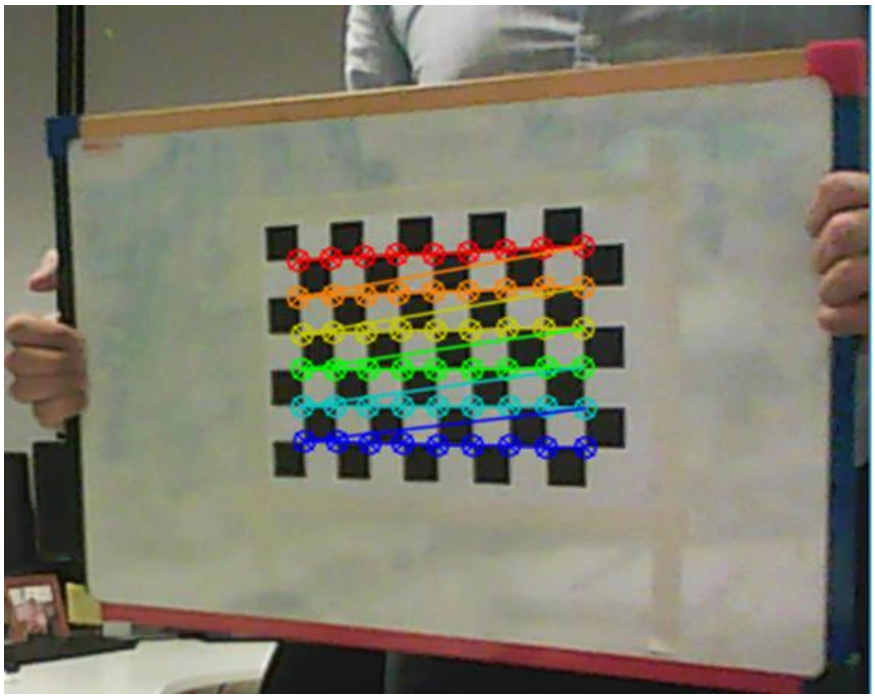


Fig. 27. Vértices detectados por el algoritmo de calibración "findChessboardCorners" de la biblioteca OpenCV

Fig. 38. Tomas realizadas desde distintos ángulos

17.7-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 este 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 píxeles 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 verifica en paralelo la "calidad" del mapa de disparidad resultante:

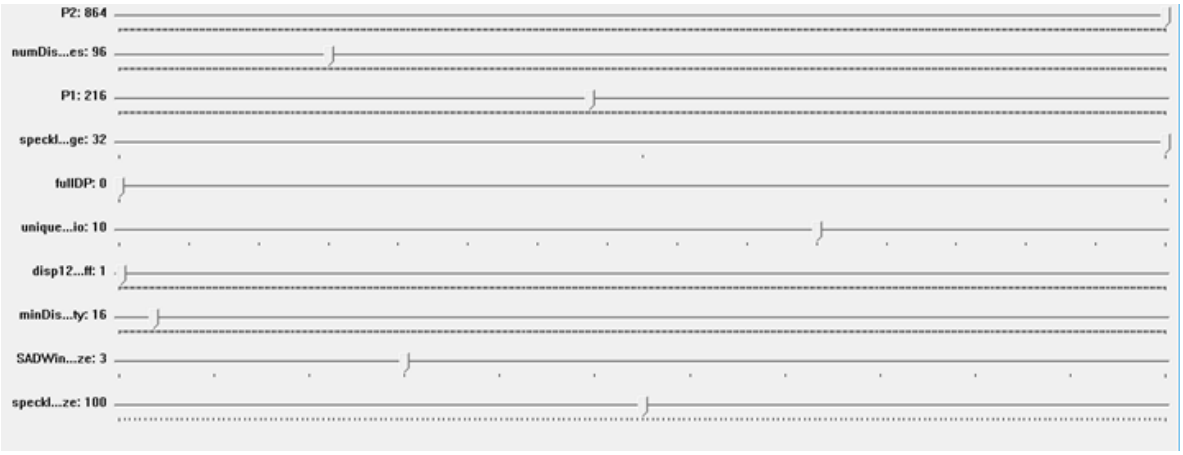


Fig. 49. Ajustes seleccionados

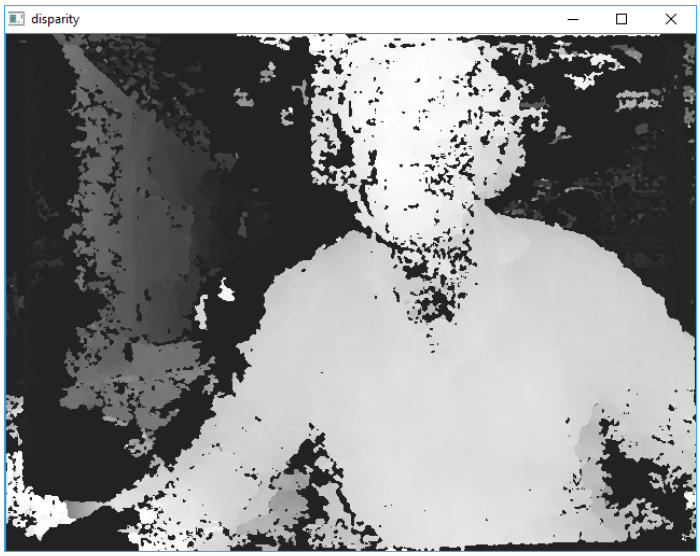


Fig. 30. Mapa de disparidad obtenido

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*" y da como resultado una matriz de 3 dimensiones que contiene los vectores (también de 3 dimensiones) correspondientes a la información del color de cada pixel. Como resultado se puede observar la siguiente nube de puntos (Figura 31):

Fig. 31. Un par de imágenes (imagen estéreo)

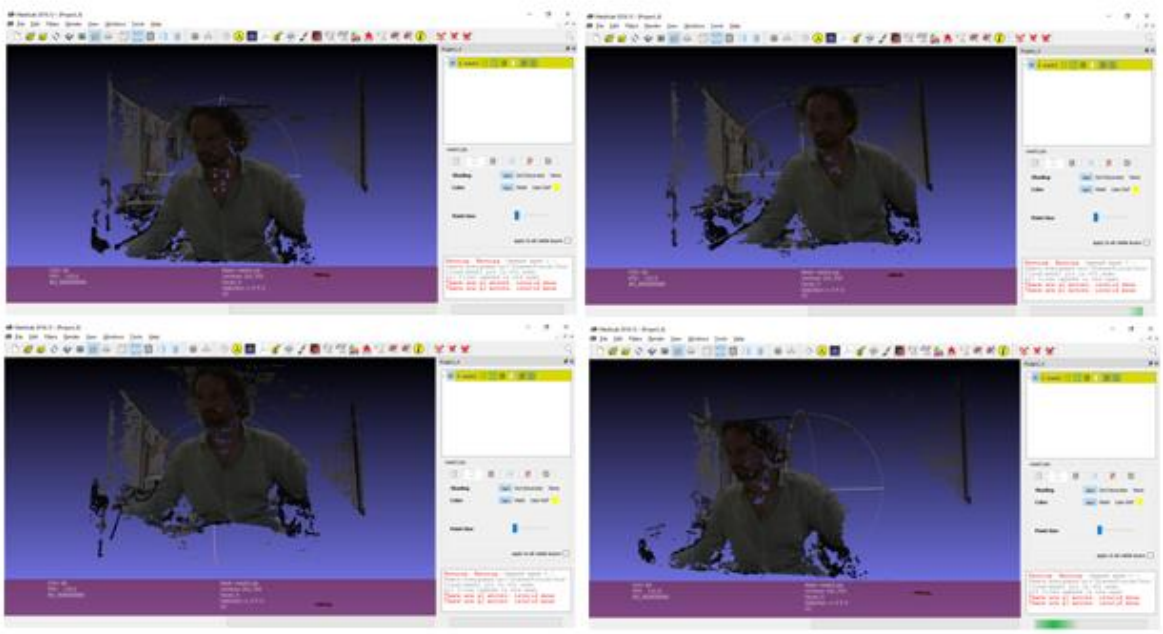


Fig. 32. Resultado de la nube de puntos, a partir de la imagen estéreo

17.8.- Conclusión

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 en forma de pregunta a resolver sobre el final del apartado "Optimización de velocidad de la búsqueda", 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 esa sería una mejora a esta implementación de DOT. Otro punto importante a destacar es que el tiempo de reconocimiento de un objeto depende de la cantidad de plantillas y de la dimensión del video. Esto no debería ser así aunque es una característica intrínseca del algoritmo.

Una de las ideas que se podría proponer para la mejora del algoritmo es que no se utilice la ventana deslizante sino que por algún medio se detecten los posibles candidatos a la imagen de la plantilla buscada y luego se realice 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 detección 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.

17.9.- Diagramas estructurales del software de detección desarrollado

En los diagramas siguientes, se plantea la estructura de los programas desarrollados para la detección de imágenes en dos dimensiones, aquí descriptos. Los códigos fuente se encuentran incorporados al presente informe como Anexo I.

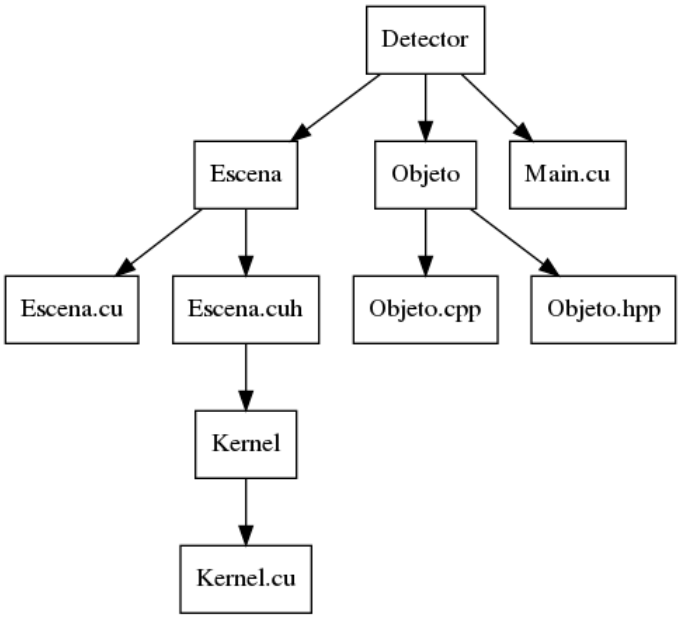
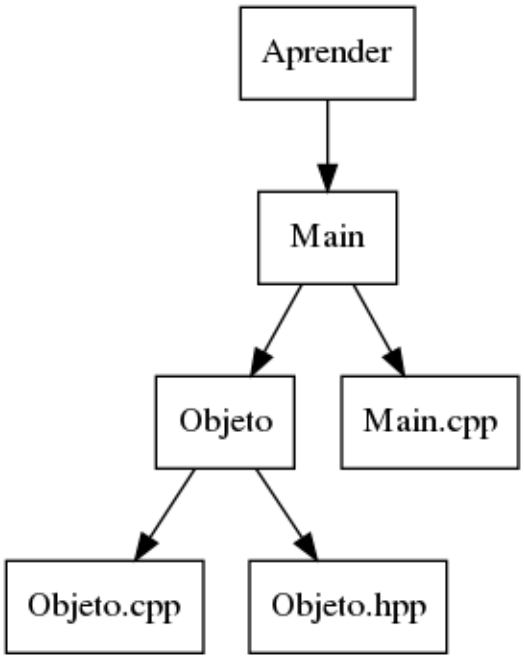


Fig. 15. Esquema de la estructura del módulo detector.



18.- Evaluación del equipo de trabajo

Las dificultades mencionadas en los apartados iniciales afectaron en cierto grado el avance del proyecto de investigación, según las tareas planteadas en el cronograma original, pero no impidieron que los integrantes del grupo se dedicaran a tareas de investigación asociadas con el proyecto, avanzando sobre temas afines o alterando el orden de los avances previstos.

El grupo de investigación ya ha demostrado, desde los años anteriores, su alto grado de integración y responsabilidad en el desarrollo de las tareas asignadas, por lo que es redundante establecer una nueva calificación individual de los mismos.

De todos modos, vale la pena hacer algunas menciones referentes a la actividad y dedicación de los integrantes del grupo de investigación.

El Sr. Martín Ferreyra Birón continuó acaparando conocimientos en el área de detección y análisis de imágenes, no solamente mediante la colaboración permanente con los integrantes del grupo en todas aquellas necesidades afines a este proyecto y a otros, sino también con una adquisición de conocimientos tal que lo convirtió en parte indispensable del presente proyecto.

La incorporación del Ing. Alberto Míguens permitió un avance importante sobre el tema de las técnicas de reconocimiento y evaluación de imágenes, en el que se encontraba trabajando el Sr. Ferreyra Birón. Dichos avances permitieron avanzar sobre el estudio de alternativas para lograr una detección tridimensional, necesaria para que el manipulador pueda cumplir con la función de trasladar piezas entre cualquier punto espacial. El Ing. Miguens demostró capacidad profesional y, lo que también es importante, dedicación y participación en el equipo de trabajo. La incorporación del Ing. Míguens potenció la actividad del Sr. Ferreyra Birón en el tema específico al que se hace referencia.

Los Ings. Alejandro Martínez y Nahuel Nieva continuaron con el desarrollo, puesta en marcha y mejora de los controladores electrónicos requeridos para el manejo del autómata. Tras el desarrollo, implementación y puesta en marcha del primer prototipo, utilizaron el año para revisar algunas condiciones del diseño, lo que permitió lograr un segundo prototipo, actualmente en producción, para ser usado como elemento de control definitivo en el proyecto.

La incorporación del Ing. Alejandro Pérez Aráuz perfeccionó los conocimientos del grupo de investigación en lo que hace a los aspectos mecánicos del proyecto. Debe hacerse notar que, hasta la incorporación del mencionado profesional, no había en el grupo integrantes con formación en mecánica, lo que obligó a los Ings. Nicolás Molina Vuistaz y Gustavo Sagarna (fundamentalmente este último) a perfeccionarse en estos temas ajenos a su formación académica.

El Ing. Pérez Aráuz diseñó e implementó modelos de brazos robóticos, utilizando elementos disponibles en el laboratorio de trabajo, o aportados por él mismo, para permitir suplir la carencia de los elementos originalmente diseñados, carencia determinada por las mencionadas demoras en la acreditación de los subsidios originalmente previstos. Cabe destacar, de todos modos, que el

mencionado profesional abandonó el grupo de investigación a principios de 2017, dado que fue invitado a participar de otros proyectos de investigación dentro del Departamento de Ingeniería e Investigaciones Tecnológicas. Al no poder completar sus desarrollos, los mismos no pudieron ser aplicados en el proyecto en cuestión.

El Ing. Sagarna, liberado en parte del desarrollo mecánico por la incorporación del Ing. Pérez Aráuz, pudo avanzar sobre otras áreas del proyecto, para lo cual, entre otras actividades, participó en la última edición del año de la Escuela de Sistemas Embebidos, llevada a cabo en la Universidad Nacional de Tucumán. Se espera que con el comienzo del nuevo año académico, el Ing. Sagarna trasmita los conocimientos adquiridos al resto de los integrantes del grupo de investigación que lo requieran.

El Ing. Molina Vuistaz, como codirector del proyecto, participó del mismo mediante la supervisión y apoyo a los distintos integrantes del grupo. Fue, en definitiva, responsable del seguimiento de la mayoría de los desarrollos llevados a cabo por el resto del grupo de investigación.

19.- Participación de integrantes del grupo de investigación en eventos científicos.

19.1.- 7ma Escuela para la Enseñanza de Sistemas Embebidos

Como ya ha sido dicho, el Ing. Gustavo Sagarna participó como asistente a la 7ma Escuela para la Enseñanza de Sistemas Embebidos, que se llevó a cabo en la Universidad Nacional de Tucumán, en la localidad de Horco Molle, en el mes de noviembre de 2016.

En dicha escuela, el Ing. Sagarna participó de un curso sobre Programación de la Computadora Industrial Abierta Argentina (CIAA) y su versión educativa, EDU-CIAA, uitilizando CIAA-Firmware y RTOS (Free-OSEK).

Se trata de la programación en sistemas embebidos utilizando un sistema operativo de tiempo real de código abierto.

El curso estuvo a cargo de la Red Universitaria de Sistemas Embebidos (RUSE) del CONFEDI. Es una actividad orientada a docentes universitarios cuyo principal objetivo fue que se puedan incorporar en su formación conceptos elementales sobre la utilización de sistemas operativos de Tiempo Real en sistemas embebidos.

La participación del Ing. Sagarna en dicho curso se relaciona como que en la actualidad existen muy pocas aplicaciones que empleen sistemas embebidos basados en microcontroladores de 32 bits y que no utilicen un Sistema Operativo de Tiempo Real (RTOS, por sus siglas en inglés). El RTOS es básicamente una biblioteca de software que es aprovechada por el código del usuario a fin de sumar ciertas características al diseño de su aplicación.

El Profesor a cargo de la catedra fue el Ing. Esteban Daniel Volentini. Las clases fueron teórico-prácticas, donde se planteó, entre otras cosas, la aplicación de los conocimientos adquiridos sobre la solución de problemas reales. El curso tuvo una duración de 40 horas y se desarrolló, a jornada completa, entre el 31 de octubre y el 4 de noviembre de 2016.

Conocimientos adquiridos:

- Aprendizaje de las herramientas y métodos de desarrollo utilizadas actualmente en los proyectos colaborativos de código abierto.
- Incorporación de los servicios de un RTOS como evolución de la programación tradicional.
- Diferencias entre RTOS dinámicos o estáticos y motivos de la elección.
- Motivos de las diferentes políticas de scheduling que implementan los RTOS a diferencia de los Sistemas Operativos de propósito general.
- Framework de testing para mejorar la calidad del software desarrollado.
- Diferentes prácticas.

El motivo de la elección del curso se debió a que, como parte del desarrollo del manipulador objeto del presente proyecto, se requiere la implemenetación de un de similares caracteristicas al

Hardware de dicho curso. La idea es implementar en futuras actualizaciones de este y otros proyectos un sistema operativo en tiempo real. Para ello se requiere comprender algunos aspectos que fueron resueltos en el Firmware del curso en cuestion.

20.- Conclusiones finales.

En cuanto a los objetivos que se plantearon, se cumplieron en forma satisfactoria a pesar de los inconvenientes ya mencionados en los párrafos anteriores, dado que durante 2016 se logró la construcción de un prototipo de brazo mecánico, con las características adecuadas para realizar los ensayos de los algoritmos de control y se inició el diseño y puesta en marcha de los algoritmos necesarios para la detección de objetos y cálculo de su distancia, aptos para continuar con las actividades planificadas para la segunda etapa del proyecto.

Durante el año 2017 el proyecto avanzó fundamentalmente sobre estos algoritmos de detección mencionados, como puede verse en el apartado correspondiente. En este aspecto, el avance en la investigación se puede considerar como muy satisfactorio, dado que lograron plantearse e implementarse algoritmos que, en su etapa de ensayo, mostraron cumplir con la función para la que fueron desarrollados.

En lo que hace a la parte electrónica del proyecto, se logró completar el desarrollo de los controladores necesarios, los que, también, se ensayaron y se pusieron en marcha en forma satisfactoria.

Queda pendiente, aunque no es el objetivo fundamental de este proyecto, la finalización del prototipo inicial del autómata controlado por visión. El aspecto mecánico del mismo, que requiere de mecanismos de calidad y precisión, se ha visto limitado en su avance por la escasez de recursos económicos.

Durante el año 2017, y al lograrse finalmente la concreción del subsidio tan postergado desde el Ministerio de Ciencia y Tecnología, se pudo plantear la compra de los sistemas mecánicos de precisión necesarios para lograr la calidad requerida en el movimiento de los brazos del robot.

Estos insumos, tan imprescindibles, están en proceso de compra al cierre de este informe, y serán recibidos en algún momento del año 2018. En esta situación, el desarrollo del brazo robótico podrá completarse razonablemente bajo el marco del programa PICTO, según convenio entre el mencionado ministerio y la Universidad Nacional de La Matanza, en un plazo estipulado en 24 meses a partir de la asignación y puesta a disposición del subsidio ministerial, lo que ocurrió en febrero de 2017.

21.- Bibliografía utilizada

21.1.- Libros:

[1] Robotics, Vision and Control: Fundamental Algorithms in MATLAB (Springer Tracts in Advanced Robotics). Peter I. Corke (Nov 3, 2011) ISBN-13: 978-3642201431, Springer.

[2] Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series). Howie Choset, Kevin M. Lynch, Seth Hutchinson and George A. Kantor (May 20, 2005) ISBN-13: 978-0262033275, Mit Pr.

[3] Introduction to Robotics: Mechanics and Control (3rd Edition). John J. Craig (Aug 6, 2004) ISBN-13: 978-0201543612, Prentice Hall.

[4] Robot Modeling and Control. Mark W. Spong (Nov 18, 2005) ISBN-13: 978-0471649908, John Wiley & Sons.

[5] Professional Microsoft Robotics Developer Studio (Wrox Programmer to Programmer). Kyle Johns (May 19, 2008) ISBN-13: 978-0470141076, Wrox.

[6] Springer Handbook of Robotics. Bruno Siciliano and Oussama Khatib (Jun 27, 2008) ISBN-13: 978-3540239574, Springer.

[7] Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.(tercera edición), Thomas Bräunl (Oct 24, 2008) ISBN-13: 978-3540705338, Springer.

[8] Experimental Robotics: The 10th International Symposium on Experimental Robotics (Springer Tracts in Advanced Robotics).Oussama Khatib, Vijay Kumar and Daniela Rus (Dec 1, 2010) ISBN-13: 978-3642096105, Springer.

[9] Handbook of Industrial Robotics, 2nd Edition. Shimon Y. Nof (Feb 16, 1999) ISBN-13: 978-0471177838, John Wiley & Sons.

[10] Industrial Robotics: How to Implement the Right System for Your Plant. Andrew Glaser (Aug 1, 2008.) ISBN-13: 978-0831133580, Industrial Pr Inc.

[11] IEEE/RSI International Conference on Intelligent Robots and Systems Proceedings -IEEE Robotics & Automation Society (Feb 2001) ISBN-13: 978-0780363496, I.E.E.E.Press (28 de febrero de 2001).

[12] 2006 IEEE Conference on Robotics, Automation and Mechatronics (Nov 14, 2008) ISBN-13: 978-1424400256, Institute of Electrical & Electronics Engineers (IEEE)

[13] Digital Signal Processing (4th Edition). John G. Proakis, Dimitris K Manolakis (Author) (Apr 7, 2006) ISBN-13: 978-0131873742, Prentice Hall.

[14] Understanding Digital Signal Processing. Richard G. Lyons (Nov 6, 1996) ISBN-13: 978-0201634679, Addison Wesley.

[15] Discrete-Time Signal Processing (3rd Edition). Alan V. Oppenheim (Aug 28, 2009). ISBN-13: 978-0131988422, Addison Wesley Pub Co Inc.

[16] Signal Processing and Linear Systems.B. P. Lathi (Feb 24, 2000) ISBN-13: 978-0195219173, OUP USA.

[17] The DSP Handbook: Algorithms, Applications and Design Techniques. Andrew Bateman (Oct 26, 2002) ISBN-13: 978-0201398519, Prentice Hall.

[18] Real-Time Digital Signal Processing from MATLAB® to C with the TMS320C6x DSPs, Second Edition. Thad B. Welch (Dec 22, 2011) ISBN-13: 978-1439883037, CRC Press,

[19] DSP Filter Cookbook (Electronics Cookbooks). John Lane (Dec 1, 2000) ISBN-13: 978-0790612041, Premier Pr.

[20] Introduction to Mechatronic Design. J. Edward Carryer (Dec 31, 2010) ISBN-13: 978-0131433564, Prentice Hall.

[21] System Dynamics: Modeling and Simulation of Mechatronic Systems. Dean C. Karnopp, Donald L. Margolis and Ronald C. Rosenberg (Jan 3, 2006) ISBN-13: 978-0471709657, John Wiley & Sons, cuarta edición.

[22] Mechatronics & Machine Tools. Hindustan Machine Tools Limited and Hmt Limited (Dec 31, 1998) ISBN-13: 978-0071346344, McGraw Hill Higher Education.

21.2.- Tesis y trabajos adicionales.

[23] Design Optimization in Industrial Robotics - Methods and Algorithms for Drive Train Design. Marcus Pettersson, Department of Management and Engineering, Division of Machine Design, Linköpings universitet SE-581 83 Linköping, Sweden, Linköping 2008.

[24] Identification, Diagnosis, and Control of a Flexible Robot Arm Mans Ostring, Division of Automatic Control, Department of Electrical Engineering Linköpings universitet, SE-581 83 Linköping, Sweden, Linköping 2002.

[25] Dynamic Modeling and Simulation of Robot Manipulators The Newton-Euler Formulation Herman Høifødt, Master of Science in Engineering Cybernetics.

[26] On Kinematic Modelling and Iterative Learning Control of Industrial Robots. Johanna Wallén, Division of Automatic Control, Department of Electrical Engineering. Linköping University, SE-581 83 Linköping, Sweden.

[27] Stereo Vision Tutorial http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/

[28] Camera Calibration and 3D Reconstruction.
http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html
[29] Dominant orientation templates for real-time detection of texture-less objects, Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, Nassir Navab .

http://ieeexplore.ieee.org/document/5539908/?reload=true

[30] Técnicas de visión estereoscópica para determinar la estructura tridimensional de la escena, Autor: Martín Montalvo Martínez, Director: Gonzalo Pajares Martinsan Curso académico 2009/2010, Máster en Investigación en Informática, Facultad de Informática, Universidad Complutense de Madrid. http://eprints.ucm.es/11350/1/Técnicas_de_visión_estereoscópica_para_determinar_la_estructura_tridimensional_de_la_escena.pdf

[31] Stereo Vision Tutorial, Chris Mc Cormick, http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/

ANEXO I

CÓDIGO FUENTE CORRESPONDIENTE AL PROGRAMA DE DETECCIÓN DE IMÁGENES.

Código y dependencias del programa Detector

Kernel/Kernel.cu

```
#define BLOQUE 8
#define INTENSIDAD 10

struct
{
 int pos;
 unsigned char max;
 }typedef est;


static __global__ void SobelYMaximo(unsigned char imagen[],unsigned char resultado[],int dx[],int dy[],int ancho,int alto)
{
 int col = blockIdx.x*blockDim.x+threadIdx.x;
 int fil = blockIdx.y*blockDim.y+threadIdx.y;

 if(col>BLOQUE && col<ancho-BLOQUE && fil>BLOQUE && fil<alto-BLOQUE)
 {
 __shared__ est maximo [BLOQUE][BLOQUE];
 int indice = col + fil*ancho;

 int indice_1=indice+1;
 int indice_m1=indice-1;
 int indice_ancho=indice+ancho;
 int indice_mancho=indice-ancho;
 int res;
 int x;
 int y;

 x=(-2*imagen[indice_m1]
 -imagen[indice_m1-ancho]
 -imagen[indice_ancho-1]
 +imagen[indice_1+ancho]
 +2*imagen[indice_1]
 +imagen[indice_1-ancho])/9;


 y=(-imagen[indice_m1-ancho]
 -2*imagen[indice_mancho]
 -imagen[indice_mancho+1]
 +imagen[indice_ancho-1]
 +2*imagen[indice_ancho]
 +imagen[indice_ancho+1])/9;

 res=(unsigned char) sqrtf (x*x+y*y);

 if(res<INTENSIDAD)
```

```
res=0;

dx[indice]=x;
dy[indice]=y;

if(res>255)
res=255;

maximo[threadIdx.x][threadIdx.y].max=res;
maximo[threadIdx.x][threadIdx.y].pos=indice;
__syncthreads();

if(threadIdx.x<=(BLOQUE/2)-1)
{
for(int i=BLOQUE-1;i>0;i/=2)
{
if(threadIdx.x<=(i/2))
{
if(maximo[threadIdx.x][threadIdx.y].max<maximo[i-threadIdx.x][threadIdx.y].max)
{
maximo[threadIdx.x][threadIdx.y].max=maximo[i-threadIdx.x][threadIdx.y].max;
maximo[threadIdx.x][threadIdx.y].pos=maximo[i-threadIdx.x][threadIdx.y].pos;
__syncthreads();
}
}
}

__syncthreads();

for(int i=BLOQUE-1;i>0;i/=2)
{
if(threadIdx.y<=(i/2))
{
if(maximo[0][threadIdx.y].max<maximo[0][i-threadIdx.y].max)
{
maximo[0][threadIdx.y].max=maximo[0][i-threadIdx.y].max;
maximo[0][threadIdx.y].pos=maximo[0][i-threadIdx.y].pos;
__syncthreads();
}
}
}

__syncthreads();

if(threadIdx.x==0 && threadIdx.y==0)
{
unsigned char bit;
int lx=dx[maximo[0][0].pos];
int ly=dy[maximo[0][0].pos];
```

```
if(lx==0 && ly==0)
bit=128;
else
{
float angulo=(atan2f(lx,ly)*180)/M_PI;
if(angulo<0)
angulo+=360;

if(angulo>=0 && angulo<=25.714)
bit=1;
else
{
if(angulo>25.714 && angulo<=51.428)
bit=2;
else
{
if(angulo>51.428 && angulo<=77.142)
bit=4;
else
{
if(angulo>77.142 && angulo<=102.856)
bit=8;
else
{
if(angulo>102.856 && angulo<=128.57)
bit=16;
else
{
if(angulo>128.57 && angulo<=154.284)
bit=32;
else
bit=64;
}
}
}
}
}
}

resultado[blockIdx.x+blockIdx.y*(ancho/BLOQUE)]=bit;
}
}
}
}
```


Escena/Escena.cuh

```
#pragma once
#include <time.h>
```

```
#include <iostream>
#include <unistd.h>
#include <math.h>
#include "../Kernel/Kernel.cu"

using namespace std;

class Escena
{
 public:
 Escena(int alto,int ancho);
 void Preparar(unsigned char *fotograma,unsigned char * resultado);
 void Preparar_tiempo(unsigned char *fotograma,unsigned char *resultado);

 private:
 void Tiempos(double inicio,double fin);
 unsigned char *cuda_result;
 unsigned char *cuda_image;
 unsigned char *result;
 unsigned char *fotograma;
 double fin;
 double inicio;
 double suma;
 size_t bytes_int;
 size_t bytes_res;
 size_t bytes;
 dim3 bloques;
 dim3 rejilla;
 int *cuda_result_y;
 int *cuda_result_x;
 int cantidad;
 int ancho;
 int alto;
 int tiempo;

};
```


Escena/Escena.cu

```
#include "Escena.cuh"

Escena::Escena(int fil,int col)
{
 this->alto=fil;
```

```
 this->ancho=col;
 bytes=sizeof(unsigned char) * fil*col;
 bytes_int=sizeof(int) * fil*col;
 bytes_res =sizeof(unsigned char) * (fil/BLOQUE)*(col/BLOQUE);
 fotograma=(unsigned char *) malloc(sizeof(unsigned char)*fil*col);;

 bloques.x=BLOQUE;
 bloques.y=BLOQUE;
 rejilla.x=(int)ceil(col/bloques.x);
 rejilla.y=(int)ceil(fil/bloques.y);

 cudaMalloc(&cuda_image,bytes);
 cudaMalloc(&cuda_result,bytes_res);
 cudaMalloc(&cuda_result_x,bytes_int);
 cudaMalloc(&cuda_result_y,bytes_int);

 suma=0;
 tiempo=0;
 cantidad=0;
}
```



```
void Escena::Preparar(unsigned char *fotograma,unsigned char *resultado)
{
 cudaMemcpy(cuda_image,fotograma,bytes,cudaMemcpyHostToDevice);
 SobelYMaximo<<<rejilla,bloques>>>(cuda_image,cuda_result,cuda_result_x,cuda_result_y,ancho,alto);
 cudaMemcpy(resultado,cuda_result,bytes_res,cudaMemcpyDeviceToHost);
}

void Escena::Preparar_tiempo(unsigned char *fotograma,unsigned char *resultado)
{
 cudaMemcpy(cuda_image,fotograma,bytes,cudaMemcpyHostToDevice);
 inicio=clock();
 SobelYMaximo<<<rejilla,bloques>>>(cuda_image,cuda_result,cuda_result_x,cuda_result_y,ancho,alto);
 fin=clock();
 cudaMemcpy(resultado,cuda_result,bytes_res,cudaMemcpyDeviceToHost);
 Tiempos(inicio,fin);
}

void Escena::Tiempos(double inicio,double fin)
{
 double tiempo=((double)(fin - inicio) / CLOCKS_PER_SEC);
 suma+=tiempo;
 cantidad++;
 cout<<"Promedio: "<<suma/cantidad<<"|Tiempo: "<<tiempo<<endl;
```

```
}
```

/Objeto/Objeto.hpp

```
#pragma once
#define BLOQUE 8
#include <math.h>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include "opencv2/opencv.hpp"


using namespace std;
using namespace cv;


struct
{
 uchar gradiente;
 int alto;
 int ancho;


}typedef estructura_auxiliar_guardar;


struct
{
        short int x;
        short int y;
        uchar z;// Fuerza o intensidad del gradiente
        uchar angulo;

}typedef Gradiente;


struct
{
        int alto;
        int ancho;

}typedef Dimension;

struct
{
 Dimension dimensiones;
```

```
 estructura_auxiliar_guardar *modelo;
 int gradientes;
}typedef Modelo;


bool Comparacion(const Gradiente &aux1,const Gradiente &aux2);
bool ComparacionIgual(const Gradiente &aux1,const Gradiente &aux2);
bool ComparacionIgualPotencia(const Gradiente &aux1,const Gradiente &aux2);

class Objeto
{
 public:
 Objeto();
 void Aprender(string imagen);
 void Guardar(string direccion);
 vector<Modelo> Abrir(string direccion);
 vector<Modelo> Abrir(char *direccion);

 private:
 int l;
 int k;
 string direccion;
 list< Dimension> dimensiones;
 list< list< unsigned char >> gradientes;
 vector<Modelo> resultado;


 unsigned char Maximos(list<Gradiente> maximos);
 void Sobel(Mat imagen,Mat gradiente,Mat dx,Mat dy);
 void Corrimientos(string imagen,Mat *resultado);
 void Angulo(Gradiente *auxiliar);
 Mat Completar(Mat imagen);


};

/Objeto/Objeto.cpp

#include "Objeto.hpp"
#define CANTIDAD 9
#define UMBRAL 10

bool Comparacion(const Gradiente &aux1,const Gradiente &aux2)
{
        if(aux1.z>aux2.z)
                return true;
        else
                return false;
}
```

```
bool ComparacionIgual(const Gradiente &aux1,const Gradiente &aux2)
{
		if(aux1.angulo==aux2.angulo)
			return true;
		else
			return false;
}

bool ComparacionIgualPotencia(const Gradiente &aux1,const Gradiente &aux2)
{
		if(aux1.z==aux2.z)
			return true;
		else
			return false;
}


Objeto::Objeto()
{
 l=BLOQUE;
}

void Objeto::Aprender(string imagen)
{
 //Las nueve imagenes corresponden a
 //los corrimientos
 Mat imagenes [CANTIDAD];
 Mat gradiente[CANTIDAD];
 Mat gradientex[CANTIDAD];
 Mat gradientey[CANTIDAD];
 list <Gradiente> maximos;


 Corrimientos(imagen,imagenes);

 for(int i=0;i<CANTIDAD;i++)
 {
 gradiente[i].create(imagenes[i].rows,imagenes[i].cols, CV_8U);
 gradientex[i].create(imagenes[i].rows,imagenes[i].cols, CV_16SC1);
 gradientey[i].create(imagenes[i].rows,imagenes[i].cols, CV_16SC1);
 }



 for(int h=0;h<CANTIDAD;h++)
	{
 Gradiente auxiliar;
 Dimension dimension;

 vector < list <uchar> > modelo;
```

```
list <uchar> angulos;

                cout<<"Aprendiendo objeto "+imagen+" (Shift:"<<h<<"/8)"<<endl;

//Se aplica el mismo algoritmo pero en la CPU, en el caso del aprendizaje
//no es vital el tiempo de procesamiento
Sobel(imagenes[h],gradiente[h],gradientex[h],gradientey[h]);


imshow("Cuadro",gradiente[h]);
if( (char) waitKey(1)== 's')
{

}


dimension.alto=gradiente[h].rows/l;
dimension.ancho=gradiente[h].cols/l;

                for(int i=0;i<=gradiente[h].rows-l;i+=l)
                {
                        for(int j=0;j<=gradiente[h].cols-l;j+=l)
                        {
                                for(int m=0;m<l;m++)
                                {
                                        for(int n=0;n<l;n++)
                                        {
                                                auxiliar.z=gradiente[h].at<uchar>(i+m,j+n);
                                                auxiliar.x=gradientex[h].at<short
int>(i+m,j+n);
                                                auxiliar.y=-1*gradientey[h].at<short
int>(i+m,j+n);
                                                maximos.push_back(auxiliar);
                                        }
                                }

                                angulos.push_back(Maximos(maximos));
                                maximos.clear();
                        }
                }

                gradientes.push_back(angulos);
dimensiones.push_back(dimension);
        }

}


void Objeto::Corrimientos(string imagen,Mat *resultado)
{
```

```
Mat img=imread(imagen,CV_LOAD_IMAGE_COLOR);
	cvtColor(img, img, CV_BGR2GRAY);

	for(int i=0;i<CANTIDAD;i++)
	{
		resultado[i].create(l*2+img.rows,l*2+img.cols,CV_8UC(1));
		resultado[i]=Completar(resultado[i]);
		resultado[i].setTo(cv::Scalar(255,255,255));

		switch(i)
		{
			case 0:
				img.copyTo(resultado[i](Rect(l, l, img.cols, img.rows)));
				break;
			case 1:
				img.copyTo(resultado[i](Rect(l/2, l, img.cols, img.rows)));
				break;
			case 2:
				img.copyTo(resultado[i](Rect(l+l/2, l, img.cols, img.rows)));
				break;
			case 3:
				img.copyTo(resultado[i](Rect(l, l/2, img.cols, img.rows)));
				break;
			case 4:
				img.copyTo(resultado[i](Rect(l, l+l/2, img.cols, img.rows)));
				break;
			case 5:
				img.copyTo(resultado[i](Rect(l/2, l/2, img.cols, img.rows)));
				break;
			case 6:
				img.copyTo(resultado[i](Rect(l+l/2, l+l/2, img.cols, img.rows)));
				break;
			case 7:
				img.copyTo(resultado[i](Rect(l/2, l+l/2, img.cols, img.rows)));
				break;
			case 8:
				img.copyTo(resultado[i](Rect(l+l/2, l/2, img.cols, img.rows)));
				break;

		}
	}
}


void Objeto::Sobel(Mat imagen,Mat gradiente,Mat dx,Mat dy)
{
```

```
for(int alto=BLOQUE;alto<imagen.rows-BLOQUE;alto++)
{
for(int ancho=BLOQUE;ancho<imagen.cols-BLOQUE;ancho++)
{
dx.at<short int>(alto,ancho)= ( -1*imagen.at<uchar>(alto-1,ancho-1)
-2*imagen.at<uchar>(alto,ancho-1)
-1*imagen.at<uchar>(alto+1,ancho-1)
+imagen.at<uchar>(alto+1,ancho+1)
+2*imagen.at<uchar>(alto,ancho+1)
+imagen.at<uchar>(alto+1,ancho+1)
)/9;


dy.at<short int >(alto,ancho)=( -1*imagen.at<uchar>(alto-1,ancho-1)
-2*imagen.at<uchar>(alto-1,ancho)
-1*imagen.at<uchar>(alto-1,ancho+1)
+imagen.at<uchar>(alto+1,ancho-1)
+2*imagen.at<uchar>(alto+1,ancho)
+imagen.at<uchar>(alto+1,ancho+1)
)/9;



gradiente.at<uchar>(alto,ancho)= (uchar) sqrtf( dy.at<short int>(alto,ancho)* dy.at<short
int>(alto,ancho)+ dx.at<short int>(alto,ancho)* dx.at<short int>(alto,ancho));

if(gradiente.at<uchar>(alto,ancho)>255)
gradiente.at<uchar>(alto,ancho)=255;

if(gradiente.at<uchar>(alto,ancho)<UMBRAL)
gradiente.at<uchar>(alto,ancho)=0;
}
}
}

Mat Objeto::Completar(Mat imagen)
{

        if(imagen.cols%l==0)
        {
                if(imagen.rows%l==0)
                        return imagen;
                else
                {
                        Mat destino(imagen.rows+(l-imagen.rows%l),imagen.cols,CV_8UC(1));
                        destino.setTo(cv::Scalar(0,0,0));
                        imagen.copyTo(destino(Rect(0,0, imagen.cols, imagen.rows)));
                        return imagen;
                }
        }
```

```
        else
        {
                if(imagen.rows%l==0)
                {
                        Mat destino(imagen.rows,imagen.cols+(l-imagen.cols%l),CV_8UC(1));
                        destino.setTo(cv::Scalar(0,0,0));
                        imagen.copyTo(destino(Rect(0,0, imagen.cols, imagen.rows)));
                        return destino;
                }
                else
                {
                        Mat                     destino(imagen.rows+(l-imagen.rows%l),imagen.cols+(l-
imagen.cols%l),CV_8UC(1));
                        destino.setTo(cv::Scalar(0,0,0));
                        imagen.copyTo(destino(Rect(0,0, imagen.cols, imagen.rows)));
                        return destino;
                }

        }

}

void Objeto::Angulo(Gradiente *auxiliar)
{
        float angulo;

        if((*auxiliar).y==0 && (*auxiliar).x==0)
        {
                (*auxiliar).angulo=128;
                return;
        }

 angulo=(atan2f((*auxiliar).x,(*auxiliar).y)*180)/M_PI;

 if(angulo<0)
 angulo+=360;

 if(angulo>=0 && angulo<=25.714)
 (*auxiliar).angulo=1;
 else
 {
 if(angulo>25.714 && angulo<=51.428)
 (*auxiliar).angulo=2;
 else
 {
 if(angulo>51.428 && angulo<=77.142)
 (*auxiliar).angulo=4;
 else
 {
 if(angulo>77.142 && angulo<=102.856)
```

```
(*auxiliar).angulo=8;
else
{
if(angulo>102.856 && angulo<=128.57)
(*auxiliar).angulo=16;
else
{
if(angulo>128.57 && angulo<=154.284)
(*auxiliar).angulo=32;
else
(*auxiliar).angulo=64;
}
}
}
}
}

        return;
}

unsigned char Objeto::Maximos(list<Gradiente> maximos)
{
 unsigned char angulo=0;
 unsigned char aux;

        list<Gradiente> auxiliar;
        maximos.sort(Comparacion);
        maximos.unique(ComparacionIgualPotencia);

        for(unsigned int i=0;i<7 && i<maximos.size();i++)
        {
                Gradiente tmp=maximos.front();
                Angulo(&tmp);
                auxiliar.push_back(tmp);
                maximos.pop_front();
        }

        auxiliar.unique(ComparacionIgual);
        auxiliar.sort(Comparacion);

 for(unsigned int i=0;i<auxiliar.size();i++)
 {
 aux=auxiliar.front().angulo;
 angulo|=aux;
 auxiliar.pop_front();
 }

        return angulo;
}
```

```
void Objeto::Guardar(string direccion)
{

 vector <estructura_auxiliar_guardar> gradientesg;
 estructura_auxiliar_guardar auxiliar_s;
 int cantidad=0;
 ofstream archivo;
 unsigned int modelos=gradientes.size();
 cout<<"Guardando en "<<direccion<<endl;
 archivo.open (direccion,std::ios_base::app);


 for(unsigned int i=0;i<modelos;i++)
 {
 list<unsigned char> lista_gradientes=gradientes.front();

 for(unsigned int j=0;j<lista_gradientes.size();j++)
 {
 Dimension dimension=dimensiones.front();


 for(int alto=0;alto<dimension.alto;alto++)
 {
 for(int ancho=0;ancho<dimension.ancho;ancho++)
 {
 auxiliar_s.gradiente=lista_gradientes.front();


 if(auxiliar_s.gradiente!=128)
 {
 cout<<(int) auxiliar_s.gradiente<<"|";
 auxiliar_s.alto=alto;
 auxiliar_s.ancho=ancho;
 gradientesg.push_back(auxiliar_s);
 cantidad++;
 }
 else
 cout<<"0|";

 lista_gradientes.pop_front();
 }
 cout<<endl;
 }
 cout<<endl;

 archivo<<"*"<<cantidad<<endl;
 archivo<<dimension.alto<<"|"<<dimension.ancho<<endl;

 for(int k=0;k<cantidad;k++)
 {
```

```
archivo<<to_string(gradientesg[k].gradiente)<<"|"<<gradientesg[k].alto<<"|"<<gradientesg[k].ancho<<en
dl;
 }


 cantidad=0;
 gradientesg.clear();
 dimensiones.pop_front();
 }
 gradientes.pop_front();
 }

 archivo.close();
}


vector<Modelo> Objeto::Abrir(string direccion)
{
 int alto;
 int ancho;
 int gradiente_a;
 int cantidad_de_gradientes;
 string linea;
 ifstream archivo;
 estructura_auxiliar_guardar auxiliar;


 archivo.open (direccion);

 if(archivo.is_open() == false)
 cout<<"No abierto"<<endl;
 else
 cout<<"Abierto"<<endl;


 while(getline(archivo,linea))
 {
 Modelo estructura_modelo;

 sscanf(linea.c_str(),"*%d",&cantidad_de_gradientes);
 getline(archivo,linea);
 sscanf(linea.c_str(),"%d|%d",&alto,&ancho);

 estructura_modelo.dimensiones.alto=alto;
 estructura_modelo.dimensiones.ancho=ancho;
 estructura_modelo.gradientes=cantidad_de_gradientes;
 estructura_modelo.modelo=(estructura_auxiliar_guardar                                    *)
malloc(sizeof(estructura_auxiliar_guardar)*cantidad_de_gradientes);
```

```
 for(int i=0;i<cantidad_de_gradientes;i++)
 {
 getline(archivo,linea);
 sscanf(linea.c_str(),"%d|%d|%d",&gradiente_a,&auxiliar.alto,&auxiliar.ancho);
 estructura_modelo.modelo[i].gradiente=(uchar) gradiente_a;
 estructura_modelo.modelo[i].alto=auxiliar.alto;
 estructura_modelo.modelo[i].ancho=auxiliar.ancho;
 }
 resultado.push_back(estructura_modelo);

 }
 archivo.close();

 return resultado;
}

vector<Modelo> Objeto::Abrir(char * direccion)
{
 string dir;
 dir=direccion;
 return Abrir(dir);
}
```

Main/Main.cu


```
#include <queue>
#include <iostream>
#include <pthread.h>
#include <semaphore.h>
#include <math.h>
#include "../../Aprender/Objeto/Objeto.hpp"
#include "../Escena/Escena.cuh"
#include "opencv2/opencv.hpp"
#include "opencv2/core/core.hpp"
#include "opencv2/highgui/highgui.hpp"
#define SALTO 2

using namespace std;
using namespace cv;

int fil;
int col;
queue <uchar *> fotogramas;
queue <uchar *> maximos;
queue <Mat> fotograma_color;
sem_t semaforo_fotogramas;
sem_t semaforo_maximos;
sem_t semaforo_fotograma_color;
sem_t semaforo_fotograma_buffer;
```

```
void Verificar(int cantidad,char **cadenas);
void *Busqueda(void * datos);
void *Hilo_GPU(void *);
void Crear_Busqueda(string direccion);


int main (int cantidad,char **objetos)
{
 Verificar(cantidad,objetos);

 bool seguir=true;
 int cantidadf=0;
 Mat gris;
 Mat fotograma_inicial;
 VideoCapture *video=0;

 sem_init(&semaforo_maximos,0,1);
 sem_init(&semaforo_fotogramas,0,0);
 sem_init(&semaforo_fotograma_color,0,1);
 sem_init(&semaforo_fotograma_buffer,0,30);

 //Esto deberia estar parametrizado
 video= new VideoCapture("Video/Plasticola.m4v");
 (*video)>>fotograma_inicial;

 col=fotograma_inicial.cols;
 fil=fotograma_inicial.rows;

 for (int i=0;i<30;i++)
 {
 Mat fotograma;

 (*video)>>fotograma;
 uchar *ugris=(uchar *) malloc(sizeof(uchar)*fil*col);
 cvtColor( fotograma, gris, CV_BGR2GRAY );
 cvtColor( fotograma, fotograma, CV_BGR2GRAY );
 memcpy(ugris,gris.data,sizeof(uchar) * fil*col);
 fotogramas.push(ugris);
 sem_post(&semaforo_fotogramas);

 sem_wait(&semaforo_fotograma_color);
 fotograma_color.push(fotograma);
 sem_post(&semaforo_fotograma_color);

 }

 for(int i=1;i<cantidad;i++)
 Crear_Busqueda(objetos[i]);
```

```
 pthread_t hilo_gpu;
 pthread_create( &hilo_gpu, NULL, Hilo_GPU, NULL);


 while(seguir==true)
 {

 sem_wait(&semaforo_fotograma_buffer);
 Mat fotograma;
 (*video)>>fotograma;

 cantidadf++;

 if(fotograma.rows!=0)
 {
 uchar *ugris=(uchar *) malloc(sizeof(uchar)*fil*col);
 cvtColor( fotograma, gris, CV_BGR2GRAY );
 cvtColor( fotograma, fotograma, CV_BGR2GRAY );
 memcpy(ugris,gris.data,sizeof(uchar) * fil*col);
 sem_wait(&semaforo_fotogramas);
 fotogramas.push(ugris);
 sem_post(&semaforo_fotogramas);

 sem_wait(&semaforo_fotograma_color);
 fotograma_color.push(fotograma);
 sem_post(&semaforo_fotograma_color);
 }
 else
 seguir=false;

 }

 pthread_join(hilo_gpu, NULL);

}


void Verificar(int cantidad,char **cadenas)
{
 if(cantidad==1)
 {
 cout<<"Faltan parametros"<<endl;
 cout<<"Modo de uso: "<<cadenas[0]<<" <direccion de archivo de aprendizaje .dot>"<<endl;
 exit(1);
 }
}

void *Hilo_GPU(void *dummy)
{
 cout<<"Ejecute"<<endl;
```

```
Escena escena(fil,col);
int nulos = 0;
size_t bytes = sizeof(unsigned char) * fil*col;
size_t bytes_res = sizeof(uchar) * (fil/BLOQUE)*(col/BLOQUE);
unsigned char *fotograma = (uchar *) malloc(bytes);

while(true)
{
sem_wait(&semaforo_fotogramas);
if( fotogramas.size() > 0 )
{
nulos=0;
fotograma=fotogramas.front();
}
else
nulos++;
sem_post(&semaforo_fotogramas);

if(nulos==0)
{

unsigned char *resultado = (uchar *) malloc(bytes_res);
escena.Preparar(fotograma,resultado);
free(fotograma);

sem_wait(&semaforo_maximos);
maximos.push(resultado);
sem_post(&semaforo_maximos);

sem_wait(&semaforo_fotogramas);
fotogramas.pop();
sem_post(&semaforo_fotogramas);

sem_post(&semaforo_fotograma_buffer);

}

}

return NULL;
}


void Crear_Busqueda(string direccion)
{
pthread_t busqueda;
pthread_create( &busqueda, NULL, Busqueda, (void *) direccion.c_str());
}
```

```
void *Busqueda(void * datos)
{

 int cantidad_s;
 int ancho_anterior;
 int alto_anterior;
 bool inicio=false;
 int s_modelo=0;
 int alto_final;
 int ancho_final;
 int cantidad_de_gradientes=0;
 int maxima_cantidad_de_gradientes=0;
 size_t bytes_res = sizeof(uchar) * (fil/BLOQUE)*(col/BLOQUE);
 Objeto objeto;
 unsigned char *aux=(unsigned char *) malloc(sizeof(unsigned char)*fil*col);
 unsigned char mmaximos[fil/BLOQUE][col/BLOQUE];
 vector<Modelo> modelos=objeto.Abrir((char *)datos);
 Mat resultadof;

 while(true)
 {
 sem_wait(&semaforo_maximos);
 if(maximos.size()>0)
 {
 aux=maximos.front();
 }
 sem_post(&semaforo_maximos);



 for(int alto=0;alto<fil/BLOQUE;alto++)
 for(int ancho=0;ancho<col/BLOQUE;ancho++)
 mmaximos[alto][ancho]=aux[(col/BLOQUE)*alto+ancho];

 sem_wait(&semaforo_maximos);
 if(maximos.size()>0)
 {
 maximos.pop();
 }
 sem_post(&semaforo_maximos);



 for(unsigned int modelo=0;modelo<modelos.size();modelo++)
 {
 estructura_auxiliar_guardar *plantilla=modelos[modelo].modelo;
 Dimension dimensiones=modelos[modelo].dimensiones;

 for(int altoimagen=0;altoimagen<fil/BLOQUE-dimensiones.alto;altoimagen+=SALTO)
 {
```

```
for(int anchoimagen=0;anchoimagen<col/BLOQUE-dimensiones.ancho;anchoimagen+=SALTO)
{
for(int
gradientes_en_el_modelo=0;gradientes_en_el_modelo<modelos[modelo].gradientes;gradientes_en_el_m
odelo++)
{

if(mmaximos[altoimagen+plantilla[gradientes_en_el_modelo].alto][anchoimagen+plantilla[gradientes_en
_el_modelo].ancho]!=128 && (plantilla[gradientes_en_el_modelo].gradiente &
mmaximos[altoimagen+plantilla[gradientes_en_el_modelo].alto][anchoimagen+plantilla[gradientes_en_el
_modelo].ancho])!=0 )
cantidad_de_gradientes++;
}

if(cantidad_de_gradientes!=0 && cantidad_de_gradientes>maxima_cantidad_de_gradientes)
{
cantidad_s=modelos[modelo].gradientes;
s_modelo=modelo;
alto_final=altoimagen;
ancho_final=anchoimagen;
maxima_cantidad_de_gradientes=cantidad_de_gradientes;
}

cantidad_de_gradientes=0;
}
}
}


if(maxima_cantidad_de_gradientes>0)
{

if(((maxima_cantidad_de_gradientes*100)/(cantidad_s/2))>45)
{

if(maxima_cantidad_de_gradientes>1 && inicio==false)
{
ancho_anterior=ancho_final;
alto_anterior=alto_final;
inicio=true;
}
else
{
if(inicio==true)
{
if(ancho_final>ancho_anterior*1.08)
ancho_final=ancho_anterior*1.08;

if(ancho_final<ancho_anterior*0.92)
ancho_final=ancho_anterior*0.92;
```

```
if(ancho_final==0)
ancho_final=1;


if(alto_final>alto_anterior*1.08)
alto_final=alto_anterior*1.08;

if(alto_final<alto_anterior*0.92)
alto_final=alto_anterior*0.92;


if(alto_final==0)
alto_final=1;


ancho_anterior=ancho_final;
alto_anterior=alto_final;
}
}


sem_wait(&semaforo_fotograma_color);
if(fotograma_color.size()>0)
{
resultadof=fotograma_color.front();
fotograma_color.front().release();
fotograma_color.pop();
}
sem_post(&semaforo_fotograma_color);



if(resultadof.rows>0)
{
cv::rectangle(resultadof,Point(ancho_final*BLOQUE,
alto_final*BLOQUE),Point(ancho_final*BLOQUE+modelos[s_modelo].dimensiones.ancho*BLOQUE,
alto_final*BLOQUE+modelos[s_modelo].dimensiones.alto*BLOQUE),Scalar(0, 0,0));
imshow("Cuadro",resultadof);
}

if( (char) waitKey(1)== 's')
{
}

maxima_cantidad_de_gradientes=0;
}
else
{
sem_wait(&semaforo_fotograma_color);
```

```
if(fotograma_color.size()>0)
{
resultadof=fotograma_color.front();
fotograma_color.front().release();
fotograma_color.pop();
}
sem_post(&semaforo_fotograma_color);

imshow("Cuadro",resultadof);
if( (char) waitKey(1)== 's')
{

}
}
}
else
{
sem_wait(&semaforo_fotograma_color);
if(fotograma_color.size()>0)
{
fotograma_color.front().release();
fotograma_color.pop();
}
sem_post(&semaforo_fotograma_color);

}
}

return NULL;
}
```


Makefile

```
all:Main

Main:    Escena.o Main.o Objeto.o
	@nvcc -o Detectar Main.o Objeto.o Escena.o `pkg-config --cflags --libs opencv` --use_fast_math -O3
-Xptxas -O3 --use_fast_math


Escena.o: ./Escena/Escena.cu ./Escena/Escena.cuh
	@echo [Compilando Escena.o]
	@nvcc -c --std=c++11 ./Escena/Escena.cu -use_fast_math -O3 -Xptxas -O3 --use_fast_math
	@echo [Terminado Escena.o]

Main.o: ./Main/Main.cu
	@echo [Compilando Main.o]
	@nvcc -c --std=c++11 ./Main/Main.cu -use_fast_math -O3 -Xptxas -O3 --use_fast_math
```

```
	@echo [Terminado Main.o]


Objeto.o: ../Aprender/Objeto/Objeto.cpp ../Aprender/Objeto/Objeto.hpp
	@echo [Compilando Objeto.o]
	@nvcc -std=c++11 -c -O3 ../Aprender/Objeto/Objeto.cpp -o Objeto.o
	@echo [Terminando Objeto.o]

clean:
	@echo [Limpiando]
	@rm -f *.o
	@rm -f Detectar
```

Código y Dependencias del programa Aprender

Main.cpp

```
#include <iostream>
#include "../Objeto/Objeto.hpp"

using namespace std;


int main (void)
{
 Objeto objeto;

 //Imágenes a aprender (Esto es un codigo de
 //prueba, en realidad este codigo deberia estar parametrizado
 //pasando los valores por el ejecutable, o la carpeta donde se encuentran
 //las imagenes
 objeto.Aprender("../ImagenesCubo/CA.png");
 objeto.Aprender("../ImagenesCubo/CB.png");
 objeto.Aprender("../ImagenesCubo/CC.png");
 objeto.Aprender("../ImagenesCubo/CD.png");

 objeto.Guardar("Cubo.dot");
 objeto.Abrir("Cubo.dot");
}
```


Nota: Los archivos Objeto.hpp Objeto.cpp son los mismos que se utilizan en el programa Detector

Makefile

```
all:Main

Main:   Objeto.o Main.o
        @echo [Linkeando]
        @g++ -std=c++11 -o Aprender Main.o Objeto.o `pkg-config --libs opencv` -g
        @echo [Limpiando]
        @rm -f *.o

Main.o: ./Main/Main.cpp
        @echo [Compilando Main.o]
        @g++ -std=c++11 -c -O3 ./Main/Main.cpp -o Main.o -Wall -g

Objeto.o: ./Objeto/Objeto.cpp ./Objeto/Objeto.hpp
        @echo [Compilando Objeto.o]
```

```
	@g++ -std=c++11 -c -O3 ./Objeto/Objeto.cpp -o Objeto.o -Wall -g

clean:
	@rm -f *.o
	@rm -f Aprender
	@echo Proyecto Limpio
```

Nota: Al no existir necesidad en velocidad de procesamiento en el programa aprender no se utilizaron las opciones de optimización y se dejaron incorporadas en el código las opciones de depuración.

ANEXO II

PLANOS MECÁNICOS.