UNIVERSIDAD LATINA DE COSTA RICA

FACULTAD DE INGENIERÍAS Y TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN TICs

ESCUELA DE INGENIERÍA ELÉCTRICA Y MECÁNICA

LICENCIATURA EN INGENIERÍA ELECTRÓNICA CON ÉNFASIS EN AUTOMATIZACIÓN

Trabajo Final de Graduación para optar por el título de Licenciatura de Ingeniería Electrónica con énfasis en Automatización

Implementación de sistema de detección y medición de cantidades de polvos mediante uso de herramientas de procesamiento visual

Autores:

Javier Fernando Robert Castro

&

Klever Anthony Rodríguez Madrigal

Heredia, Costa Rica

Mayo, 2022

UNIVERSIDAD LATINA DE COSTA RICA
POWERED BY Arizona State University

TRIBUNAL EXAMINADOR

Este proyecto titulado: Implementación de sistema de detección y medición de cantidades de polvos mediante uso de herramientas de procesamiento visual, por el estudiante: Javier Fernando Robert Castro & Klever Anthony Rodríguez Madrigal, fue aprobaba por el Tribunal Examinador de la carrera de Ingeniería Electrónica de la Universidad Latina de Costa Rica, Sede Heredia, como requisito para optar por el grado de Licenciatura en Ingeniería Electrónica con énfasis en Automatización y Control:

Ing. Christopher Ryan Russell Clark

Tutor

Ing. Luis Andres Brenes Oses

Lector

Ing. Oscar Jesús Delgado Jiménez

Representante

Heredia, 10 de mayo de 2022

Sres.

Miembros del Comité de Trabajos Finales de Graduación

SD

Estimados señores:

He revisado y corregido el Trabajo Final de Graduación, denominado: Implementación de sistema de detección y medición de cantiidades de polvos mediante uso de herramientas de procesamiento visual, elaborado por el estudiante Javier Fernando Robert Castro & Klever Anthony Rodríguez Madrigal puedan optar por Licenciatura en Ingeniería Electrónica con énfasis en Automatización y Control

Considero que dicho trabajo cumple con los requisitos formales y de contenido exigidos por la Universidad, y por tanto lo recomiendo para su defensa oral ante el Consejo Asesor.

Suscribe cordialmente,

Ing. Christopher Ryan Russell Clark

Tutor

Heredia, 10 de mayo de 2022

Sres.

Miembros del Comité de Trabajos Finales de Graduación

SD

Estimados señores:

He revisado y corregido el Trabajo Final de Graduación, denominado: Implementación de sistema de detección y medicón de cantidades de polvos mediante uso de herramientas de procesamiento visual, elaborado por el estudiante Javier Fernando Robert Castro & Klever Anthony Rodríguez Madrigal puedan optar por Licenciatura en Ingeniería Electrónica con énfasis en Automatización y Control

Considero que dicho trabajo cumple con los requisitos formales y de contenido exigidos por la Universidad, y por tanto lo recomiendo para su defensa oral ante el Consejo Asesor.

Suscribe cordialmente,

Ing. Luis Andrés Brenes Oses

Lector

M. L. Vilma Isabel Sánchez Castro
Bachiller y Licenciada en Filología Española. U.C.R.

A QUIEN INTERESE

Yo, Vilma Isabel Sánchez Castro, Máster en Literatura Latinoamericana, Bachiller y Licenciada en Filología Española, de la Universidad de Costa Rica; con cédula de identidad 6-054-080; inscrita en el Colegio de Licenciados y Profesores, con el carné N° 003671, hago constar que he revisado el siguiente documento. Y he corregido en él los errores encontrados en ortografía, redacción, gramática y sintaxis. El cual se intitula

IMPLEMENTACIÓN DE SISTEMA DE DETECCIÓN Y MEDICIÓN DE CANTIDADES DE POLVOS MEDIANTE USO DE HERRAMIENTAS DE PROCESAMIENTO VISUAL

JAVIER FERNANDO ROBERT CASTRO

LICENCIATURA DE INGENIERÍA ELECTRÓNICA CON ÉNFASIS EN AUTOMATIZACIÓN
FACULTAD DE INGENIERÍAS Y TECNOLOGÍAS DE LA

INFORMACIÓN Y COMUNICACIÓN TICs UNIVERSIDAD LATINA DE COSTA RICA

Se extiende la presente certificación a solicitud del interesado en la ciudad de San José a los ocho días del mes de mayo de dos mil veintidós. La filóloga no se responsabiliza por los cambios que se le introduzcan al trabajo posterior a su revisión.

M.L. Vilma Isabel Sánchez Castro
Máster en Literatura Latinoamericana. UCR.
Bachiller y Licenciada en Filología Esp. UCR.
Cédula 600540080-Carné 003671

Teléfonos 2227-8513. Cel 8994-76-93 Apartado 563-1011 Y griega
Correo electrónico: vilma_sanchez@hotmail.com-info@chavesysanchezfilologos.com
Página Web: Chaves y Sanchez filólogos
Waze Chaves y Sánchez filólogos

M. L. Vilma Isabel Sánchez Castro
Bachiller y Licenciada en Filología Española U.C.R.

A QUIEN INTERESE

Yo, Vilma Isabel Sánchez Castro, Máster en Literatura Latinoamericana, Bachiller y Licenciada en Filología Española, de la Universidad de Costa Rica; con cédula de identidad 6-054-080; inscrita en el Colegio de Licenciados y Profesores, con el carné N° 003671, hago constar que he revisado el siguiente documento. Y he corregido en él los errores encontrados en ortografía, redacción, gramática y sintaxis. El cual se intitula

IMPLEMENTACIÓN DE SISTEMA DE DETECCIÓN Y MEDICIÓN DE CANTIDADES DE POLVOS MEDIANTE USO DE HERRAMIENTAS DE PROCESAMIENTO VISUAL

KLEVER ANTHONY RODRÍGUEZ MADRIGAL

LICENCIATURA DE INGENIERÍA ELECTRÓNICA CON ÉNFASIS EN AUTOMATIZACIÓN

FACULTAD DE INGENIERÍAS Y TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN TICs
UNIVERSIDAD LATINA DE COSTA RICA

Se extiende la presente certificación a solicitud del interesado en la ciudad de San José a los ocho días del mes de mayo de dos mil veintidós. La filóloga no se responsabiliza por los cambios que se le introduzcan al trabajo posterior a su revisión.

M.L. Vilma Isabel Sánchez Castro
Máster en Literatura Latinoamericana, UCR.
Bachiller y Licenciada en Filología Esp. UCR.
Cédula 600540080-Carné 003671

Teléfonos 2227-8513. Cel 8994-76-93 Apartado 563-1011 Y griega
Correo electrónico: vilma_sanchez@hotmail.com-info@chavesysanchezfilologos.com
Página Web: Chaves y Sanchez filólogos
Waze Chaves y Sánchez filólogos

Declaración Jurada

Yo, Javier Fernando Robert Castro & Klever Anthony Rodríguez Madrigal estudiante de la Universidad Latina de Costa Rica, declaro bajo la fe de juramento y consciente de las responsabilidades penales de este acto, que soy el Autor Intelectual del Proyecto de Graduación , titulado:

Implementación de sistema de detección y medición de cantidades de polvos mediante uso de herramientas de procesamiento visual

Por lo que libero a la Universidad de cualquier responsabilidad en caso de que mi declaración sea falsa.

Firmo en San José, San José, Costa Rica, 11 de enero de 2022

Javier Fernando Robert Castro & Klever Anthony Rodríguez Madrigal

Licencia De Distribución No Exclusiva (carta de la persona autora para uso didáctico)

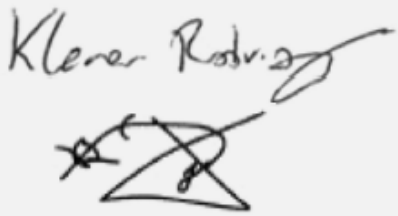
Universidad Latina de Costa Rica

Yo (Nosotros):	Javier Fernando Robert Castro Klever Anthony Rodríguez Madrigal
De la Carrera / Programa:	Licenciatura en Ingeniería Electrónica con énfasis en Automatización y Control
Modalidad de TFG:	Proyecto de Graduación
Titulado:	Implementación de sistema de detección y medición de cantidades de polvos mediante uso de herramientas de procesamiento visual

Al firmar y enviar esta licencia, usted, el autor (es) y/o propietario (en adelante el "**AUTOR**"), declara lo siguiente: **PRIMERO**: Ser titular de todos los derechos patrimoniales de autor, o contar con todas las autorizaciones pertinentes de los titulares de los derechos patrimoniales de autor, en su caso, necesarias para la cesión del trabajo original del presente TFG (en adelante la "**OBRA**"). **SEGUNDO**: El **AUTOR** autoriza y cede a favor de la **UNIVERSIDAD U LATINA S.R.L.** con cédula jurídica número 3-102-177510 (en adelante la "**UNIVERSIDAD**"), quien adquiere la totalidad de los derechos patrimoniales de la **OBRA** necesarios para usar y reusar, publicar y republicar y modificar o alterar la **OBRA** con el propósito de divulgar de manera digital, de forma perpetua en la comunidad universitaria. **TERCERO**: El **AUTOR** acepta que la cesión se realiza a título gratuito, por lo que la **UNIVERSIDAD** no deberá abonar al autor retribución económica y/o patrimonial de ninguna especie. **CUARTO**: El **AUTOR** garantiza la originalidad de la **OBRA**, así como el hecho de que goza de la libre disponibilidad de los derechos que cede. En caso de impugnación de los derechos autorales o reclamaciones instadas por terceros relacionadas con el contenido o la autoría de la **OBRA**, la responsabilidad que pudiera derivarse será exclusivamente de cargo del **AUTOR** y este garantiza mantener indemne a la **UNIVERSIDAD** ante cualquier reclamo de algún tercero. **QUINTO**: El **AUTOR** se compromete a guardar confidencialidad sobre los alcances de la presente cesión, incluyendo todos aquellos temas que sean de orden meramente institucional o de organización interna de la **UNIVERSIDAD SEXTO**: La presente autorización y cesión se regirá por las leyes de la República de Costa Rica. Todas las controversias, diferencias, disputas o reclamos que pudieran derivarse de la presente cesión y la materia a la que este se refiere, su ejecución, incumplimiento, liquidación, interpretación o validez, se resolverán por medio de los Tribunales de Justicia de la República de Costa Rica, a cuyas normas se someten el **AUTOR** y la **UNIVERSIDAD**, en forma voluntaria e incondicional. **SÉPTIMO**: El **AUTOR** acepta que la **UNIVERSIDAD**, no se hace responsable del uso, reproducciones, venta y distribuciones de todo tipo de fotografías, audios, imágenes, grabaciones, o cualquier otro tipo de

presentación relacionado con la **OBRA**, y el **AUTOR**, está consciente de que no recibirá ningún tipo de compensación económica por parte de la **UNIVERSIDAD**, por lo que el **AUTOR** haya realizado antes de la firma de la presente autorización y cesión. **OCTAVO**: El **AUTOR** concede a **UNIVERSIDAD.,** el derecho no exclusivo de reproducción, traducción y/o distribuir su envío (incluyendo el resumen) en todo el mundo en formato impreso y electrónico y en cualquier medio, incluyendo, pero no limitado a audio o video. El **AUTOR** acepta que **UNIVERSIDAD.** puede, sin cambiar el contenido, traducir la **OBRA** a cualquier lenguaje, medio o formato con fines de conservación. **NOVENO**: El **AUTOR** acepta que **UNIVERSIDAD** puede conservar más de una copia de este envío de la **OBRA** por fines de seguridad, respaldo y preservación. El **AUTOR** declara que el envío de la **OBRA** es su trabajo original y que tiene el derecho a otorgar los derechos contenidos en esta licencia. **DÉCIMO**: El **AUTOR** manifiesta que la **OBRA** y/o trabajo original no infringe derechos de autor de cualquier persona. Si el envío de la **OBRA** contiene material del que no posee los derechos de autor, el **AUTOR** declara que ha obtenido el permiso irrestricto del propietario de los derechos de autor para otorgar a **UNIVERSIDAD** los derechos requeridos por esta licencia, y que dicho material de propiedad de terceros está claramente identificado y reconocido dentro del texto o contenido de la presentación. Asimismo, el **AUTOR** autoriza a que en caso de que no sea posible, en algunos casos la **UNIVERSIDAD** utiliza la **OBRA** sin incluir algunos o todos los derechos morales de autor de esta. **SI AL ENVÍO DE LA OBRA SE BASA EN UN TRABAJO QUE HA SIDO PATROCINADO O APOYADO POR UNA AGENCIA U ORGANIZACIÓN QUE NO SEA UNIVERSIDAD U LATINA, S.R.L., EL AUTOR DECLARA QUE HA CUMPLIDO CUALQUIER DERECHO DE REVISIÓN U OTRAS OBLIGACIONES REQUERIDAS POR DICHO CONTRATO O ACUERDO. La presente autorización se extiende el día** 10 **de** Mayo **de** 2022 **a las** 18:00

Firma del estudiante(s):



Agradecimientos

Este trabajo se realizó gracias a los grandes esfuerzos de muchas personas, el profesor Christopher Ryan Russell, mis compañeras de cursos Aurora y Ana, el apoyo incondicional de nuestras familias, mis mentores durante el proceso de formación, a ese Klever Rodríguez que no se rindió y sacó fuerza de donde ya no quedaba y principalmente a mi compañero de carrera y proyectos Javier Robert, sin el apoyo de él y todas las personas mencionadas, esto solamente sería una idea y no una realidad, a todos ellos y todos los que no pude mencionar solamente me queda decirles:

Gracias por ayudarme a dar este gran paso.

Dedicatoria

Todo este proceso no se cumplió gracias solo a mi esfuerzo, esto es gracias a la paciencia de mi familia, el amor de ellos y la consideración de muchos que todo esto fue posible, este gran logro es dedicado a mis increíbles padres Rosi y Mario, porque de no ser por ellos, nada de esto podría hacerse realidad, a mi amada compañera de vida Jeimmy y la princesa de la casa Melanie, quienes me mantuvieron siempre centrado en ese norte, esa meta que estaba a solo unos pasos, también a mis hermanos y sus familias por siempre estar ahí para escucharme y mis grandes amigos que sin ese "Zafarrancho" no sería posible llegar hasta aquí.

Epígrafe

"Whatever you do in this life, it's not legendary, unless your friends are there to see it."

- Barney Stinson,
How I Met Your Mother season 9 Ep 17: "Sunrise"

Introducción

La actual investigación está enfocada en un estudio de algunas de las más enjuiciadas tendencias a nivel mundial en la rama de la automatización en el sector industrial, la investigación resulta innovadora por el hecho de que, a partir del estado de la cuestión, se pudo concluir que el conocimiento actual sobre este tema, en diferentes plantas a nivel mundial, en el caso de Costa Rica y México, es prácticamente nulo.

Los métodos de producción actuales son muy eficientes y precisos, pero al ir aumentando en precisión y eficiencia, se aumentan los costos de mantenimiento e implementación y, por ende, estos se trasladan al precio final del producto, por lo que un producto con procesos altamente tecnológicos es, de igual forma, altamente costoso. Por lo que se decide explorar la posibilidad de generar un algoritmo que pueda solventar situaciones en las que el uso de equipo automatizado pueda aumentar la calidad del producto y la eficiencia de la producción.

Muchos estudios han demostrado que el uso de tecnologías de visión de computadoras en el ámbito de la producción de materiales, comestibles y equipos electrónicos, entre otros, genera beneficios en la eficiencia de la producción, la calidad y durabilidad del producto final.

A lo largo de este documento se encontrará con descripciones y usos prácticos para herramientas que ya fueron desarrolladas, pero no necesariamente para el ámbito en el que se desenvuelve este proyecto, se pretende adaptar estas herramientas y unificar múltiples recursos para obtener un producto final que logre satisfacer las necesidades de los productores actuales con costos bajos, gracias al uso de sistemas de código abierto, equipos de uso cotidiano y ajustados para la propuesta específica y no sobredimensionados, generando costos extras.

Este trabajo final de grado pretende llevar al lector a conocer el proceso de la creación del sistema capaz de solucionar una situación que ha traído problemas a muchas líneas de

producción al presentar índices de eficiencia erróneos y con esto, ocultar los valores reales de material obtenido debido al método incorrecto de verificación de cantidad producto final.

Es bien sabido que el ser humano no es perfecto y tiende al error, ya sea conscientemente o no, por lo que utilizar personal humano, aunque esté capacitado, puede cometer errores en las líneas de producción que conlleven a pérdidas de dinero o de producto final, por lo que con el tiempo se ha popularizado más el uso de sistemas automatizados para cumplir las funciones que realizaban operadores, mejorando índices de producción, ya que se desperdicia menos materia prima, se maximizan los tiempos de producción y se aumenta las ganancias.

La importancia de los sistemas automatizados radica en su precisión, pero cuando se involucran variables del mundo físico a una máquina es cuando todo cambia, ya que la interpretación de las máquinas es muy distinta a la del ser humano, por lo que comprender a profundidad cómo hacer “entender a la máquina” lo que se desea, es necesario el estudio del campo de los sensores e intérpretes del mundo real al mundo de las máquinas y uno de estos intérpretes corresponde a las cámaras.

Las cámaras y la visión de computadoras vienen a solventar el problema de la interpretación de una situación que se da en el mundo real y mediante la interpretación de la cámara y el procesamiento de las imágenes por medio de los algoritmos de visión de computadoras, es posible automatizar prácticamente cualquier proceso en el que anteriormente, solo un ser humano podía realizar, ya que contaba con sus ojos para determinar texturas, colores, formas entre otras cosas que para las computadoras era prácticamente imposible.

Resumen

Este trabajo de investigación se basa en la exploración y estudio de métodos de análisis de imágenes en diferentes industrias de Costa Rica y México. En el estudio se evalúan los problemas que se encontraron en las visitas a plantas de producción y deficiencias en algunos procesos de producción que aún no han sido automatizados, gracias a una breve reseña de las experiencias en una línea de producción y los conocimientos adquiridos durante el periodo de trabajo en la industria. Lo anterior marcará el precedente para la elaboración del proyecto.

Cabe destacar que, se requiere un conocimiento básico en temas como programación en Python, sistemas embebidos, métodos de producción, mecánica de fluidos y las matemáticas que tienen intrínsecos estos temas, para comprender el proyecto en mención.

En una primera instancia se explica el algoritmo que se diseñó para solventar el problema encontrado y descrito anteriormente, este tema ahondará en el camino general que llevará el sistema para obtener las mediciones desde su punto inicial hasta la obtención del dato deseado. Por otro lado, también se explicará a profundidad y mediante lenguaje técnico, cada proceso dentro del algoritmo principal, por lo que se hará uso de múltiples conceptos y estructuras explicadas a lo largo del marco teórico.

Se analizarán los resultados obtenidos en la propuesta física creada, presentando las calibraciones hasta los valores obtenidos en mediciones de prueba, cálculos esperados y obtenidos y con esto, saber el porcentaje de error que se podría obtener al utilizar la herramienta diseñada.

Además de describir las posibles mejoras que a lo largo de la creación del algoritmo se determinarán y el ejemplo físico para lograr resultados más precisos, acabados más estéticos y mejor definidos, implementaciones en plantas ideales o reales y cuáles podrían ser los puntos débiles del diseño expuesto en el documento, ya que el mundo de la automatización y la mezcla de este mundo con las ventajas de la evolución tecnológica en constante mejora.

Seguidamente, se planteará un diseño e implementación de un sistema SCADA para la supervisión y el control de los equipos industriales en diversas etapas como: el desarrollo, la fabricación y la producción. En este caso se simularán equipos industriales: dos motores, uno supondría el ingreso del material al tanque y el otro, la salida del material del tanque. Estas variables también van representar los PLC, ya que se utilizan en el sistema de control de supervisión y adquisición de datos por medio de la arquitectura unificada OPC (OPC UA), siendo un protocolo de comunicación para aplicaciones de automatización industrial.

Además, todas las señales del proyecto, las de los dos motores a los que se le deben agregar las variables de volumen y masa del material que se trabajarán. Para lograr lo expuesto, se diseñará e implementará una interfaz hombre-máquina (HMI) que va a permitir a los operadores controlar los datos del proceso, vinculando las bases de datos del SCADA y los programas de software para suministrar información de gestión.

Una vez que se obtengan los datos de las variables anteriores ya expuestos en la interfaz hombre-máquina, se va a almacenar la información en bases de datos utilizando MySQL con el fin de llevar una trazabilidad de las diferentes variables y poder consultarlas en cualquier momento.

Finalmente, los datos ayudarán a realizar ciertos análisis en planta con la finalidad de mejorar los indicadores (evaluar posibles mejoras basados en evidencia numérica) para aumentar la producción de material, estos análisis se representarán por medio de gráficas que se podrá consultar desde el HMI.

Índice General

Índice de Tablas

Índice de Figuras

CAPÍTULO 1

1 Problema y Propósito

1.1 Síntoma

Tanto en Costa Rica como en México existen algunas limitaciones a la hora de realizar cierto tipo de mediciones en diferentes objetos. Muchos de estos problemas según nuestra experiencia en diferentes empresas, se deben a la falta de conocimiento de las nuevas tecnologías que se encuentran en el mercado y la dificultad de implementarlas rápidamente. Con el surgimiento de nueva tecnología a nivel global, más novedosa, las empresas carecen de protocolos para la correcta y rápida implementación, resultando en un desafío en el factor tiempo a la hora de integrarse.

La automatización de procesos ayuda a todo tipo de compañía en el proceso de la innovación digital, por lo tanto, si las empresas desean competir en sus mercados deben aceptar estas tecnologías e implementarlas de una forma correcta.

A nivel mundial la automatización es esencial para tramitar, modificar y adecuar los procesos de la empresa. Algunos de estos procesos pueden llegar a ser sumamente peligrosos para cualquier operador, ya que estarían situados en ambientes insalubres, realización de tareas en las cuales deben de manipular materiales peligrosos o incluso, manipulación de objetos pesados.

Considerando lo anterior, la automatización ayudaría a obtener una mayor productividad, ya que los operadores se pueden dedicar a otras tareas en la planta. Esto da una mayor confiabilidad al reducir la intervención humana, ya que disminuye la necesidad de realizar inspecciones y solventar cierto tipo de dificultades sin llevar a cabo ciertos métodos, experimentos, cálculos de flujos, de material y periodos de tiempo, entre otros, debido a una confiabilidad de los datos obtenidos de forma automática.

Finalmente, es necesario un control simplificado, ya que entre menos personal posea la empresa en esta área o departamento, las capacitaciones serán más rápidas y efectivas a la hora de transmitir el conocimiento de una manera eficiente a un grupo reducido de trabajadores que a un grupo más grande, en razón de que los operadores puedan lidiar de forma efectiva ante diferentes situaciones o monitorear diferentes procesos con el fin de valorar, subsanar o inspeccionar algún tipo de escenario efectivo o perjudicial durante una proceso sin necesidad de un particular presente en planta o bajo una supervisión casi nula.

1.2 Causas

Existen varias razones del porqué en Costa Rica los mecanismos de medición sufren una alta demora en el lanzamiento de las nuevas tecnologías. Incluso, dicho problema está también presente en varias industrias en México que por motivos legales no se pueden mencionar sus nombres, aunque se sabe que la situación en ese país es diferente a la de Costa Rica, por los avances que presentan en el ámbito de la automatización, de igual manera tienen, problemas por la forma en la cual realizan este tipo de mediciones, la cual completamente arcaica, realizando comprobaciones sin la utilización de ningún equipo o dispositivos de visión avanzados, por lo que solo se cuenta con una medición aproximada y calculada a simple vista por parte de un operador.

Al realizar las mediciones de esta forma manual, se exponen a un alto error humano y por ende, un margen de error importante en los indicadores que miden la eficacia general de los equipos, en ciertos reportes propios de cada empresa que por motivos legales no se pueden mencionar, los cuales afectan la toma de decisiones en la planta, con el fin de siempre mejorar este indicador, ya que dentro de más cerca se esté del 100%, se irá a tener una mayor producción en buena calidad, en el menor tiempo posible y sobre todo, sin paradas en las líneas.

Una gran cantidad de empresas a nivel internacional viven una irrealidad por una carencia de automatizar ciertos procesos delicados en la planta como el tema expuesto, en vista de que al realizar una medición de una forma totalmente antigua por parte de algún operador o jefe de turno, teniendo un margen de error excesivamente grande y, además, que lo puede manipular a conveniencia con el fin de aumentar el indicador general de los equipos (OEE) y así, perjudicarse por un mal trabajo o malos resultados durante su supervisión.

Una vez que los datos se pueden manipular o tienen un margen de error humano muy grande, se dará una afectación en la empresa en varios ámbitos, impactando la toma de decisiones de los jefes de planta o gerentes para la mejora continua, por el contrario, si ese

OEE es muy cercano al 100%, piensan que no hay mejora y se ha demostrado que a la hora de automatizar estos procesos, el indicador cae de un rango entre el 80 y 95% hasta un 20 y 30%, alertando a las personas tomadoras de decisión de la planta.

Los gerentes responsables de las plantas ven esto como una gran oportunidad para buscar cualquier cantidad de factores que afectan directamente a la producción, como podría ser un mal diseño o selección del cuello de botella, máquinas que presentan fallas constantes, mediciones más precisas y sin manipulación por parte de un operador, procesos químicos que se van a ver afectados por una calibración incorrecta o mal funcionamiento de algún sensor como podría ser un sensor de temperatura, entre otros factores que se pueden ir mejorando poco a poco para irse aproximando nuevamente al 100%.

Lo anterior es una situación bastante común en las plantas industriales, ya que se ha encontrado este problema en varias plantas, en las que se ha estado y que no se puede mencionar sus nombres por temas legales, las cuales pasan de procesos manuales a automáticos, porque se elimina una gran cantidad de los errores humanos y se presenta un escenario totalmente controlado con márgenes de error sumamente pequeños para un ambiente más controlado.

1.3 Pronóstico

Comparando las tecnologías para automatizar procesos de producción en el país con los países del primer mundo, se puede considerar que si no existe una reacción inmediata por parte de los dueños de diferentes empresas en diferentes ámbitos de producción, Costa Rica sufrirá un enorme rezago tecnológico, por ejemplo, en la empresa IPG (trabajo actual), la cual es una empresa costarricense que se dedica a automatizar procesos de producción, prefirieron migrar a otros países con un mercado más amplio en temas de automatización, donde encontraron más trabajo en países como en México, Colombia, Perú o Republica Dominicana, por decir los más recientes, afectando directamente al país, dado que muchas compañías prefieren enfocarse en mercados más competitivos por la cantidad de trabajo y la competitividad de salarios o pagos.

Es importante resaltar que en Costa Rica existe mucho profesional en automatización que es seleccionado por las empresas extranjeras. También lo es, que se trabaje en la cultura de automatización a nivel de las empresas industriales, pero, además, a nivel país, para que se desarrolle un verdadero interés en la automatización de procesos y que las personas no se sientan amenazadas por la automatización, si no que la vean como una herramienta para hacer más eficiente los sistemas y procesos de las empresas. Con la incursión de la automatización, se cuenta con otras necesidades de personal con capacidades a nivel técnico, necesarias para realizar varias tareas como el diseño de la línea de producción, la configuración de los controladores lógicos programados, diseño e implementación acerca de como la información de los controladores lógicos programables será compartida por medio de diferentes protocolos de comunicación a lo largos de una red y mantenimiento de los equipos, entre otras tareas.

1.4 Control al Pronóstico

Se llevan a cabo en la investigación, un análisis de sistemas automáticos de visión basado en un esquema metódico, en el cual, el proceso de visión computacional se puede exponer en tres grandes etapas con el fin de examinar y restringir objetos en el ambiente mediante el procesamiento de imágenes, por lo tanto, la visualización computacional vendría siendo el estudio de dichos métodos para comprenderlos y construir máquinas con capacidades equivalentes.

“Visión es un proceso que produce a partir de las imágenes del mundo exterior una descripción que es útil para observar y que no tiene información irrelevante” (Marr, 1982).

Las tres grandes etapas serían las siguientes:

- Procesamiento de bajo nivel: este se suele generalmente trabajar directamente con las imágenes para extraer propiedades como márgenes, gradiente, profundidad, disposición y color, entre otras propiedades más.
- Procesamiento de un nivel intermedio: este consiste, en su gran mayoría de casos, en apiñar los elementos obtenidos en el nivel anteriormente explicado, para obtener, en su mayoría, contornos y regiones, con el propósito de segmentación.
- Procesamiento de un alto nivel: este último nivel corresponde a la interpretación de las entidades obtenidas en los paralelismos anteriores, muy parecido al caso preliminar, con la diferencia de que se utilizan modelos y/o conocimiento con énfasis en el dominio.

Las etapas mencionadas no son del todo indispensables en cada sistema de visión y tampoco necesariamente secuenciales, pero permiten dar claridad y fragmentar los principales argumentos en una forma congruente.

Posteriormente, este análisis se complementa con una representación física a escala micro con el fin de corroborar la veracidad del algoritmo y el análisis de imágenes utilizadas en el proyecto.

Por último, se lleva a cabo un análisis extensivo por medio de reiteradas pruebas o simulaciones para obtener una gran cantidad de datos recolectados de forma totalmente experimental con el fin de poder comparar u obtener ciertos márgenes de error con respecto a lo teórico, de tal modo que se pueda concluir si se debe mejorar la depuración del código, el análisis de la imagen o simplemente, se está en un margen de error totalmente aceptable.

Estos datos van a dar confiabilidad al cliente o la persona encargada en la planta, ya que son absolutamente precisos con los cuales sí se podría llegar a tomar decisiones acertadas en la planta. Los datos pueden llegar a ser consultados por medio de reportes con el fin de facilitar el acceso a dicha información a los gerentes, supervisores o simplemente, operadores o jefes de turno en la planta.

1.5 Formulación del Problema

Uno de los principales problemas adjuntos en la actualidad a la suposición inicial de la monografía, reside en qué tan preparadas están las empresas de Costa Rica o mexicanas para enfrentarse a diferentes retos, como lo sería el cálculo de ciertos materiales de una manera manual y con más índice de error humano, utilizando nuevas tecnologías como serían los sistemas de visión automáticos, con el fin de automatizar estos procesos.

Es claro que se requiere un cambio requiere a nivel de equipamiento, con el cual se debe progresar de manera escalonada y con rapidez, lo que conlleva a invertir una gran suma de capitales, tanto en tecnologías como también en la búsqueda de personal capacitado en diferentes áreas como, por ejemplo, en el área de la programación, en este caso, en el lenguaje de programación Python o también, capacitarse en temas de sistemas de visualización automático.

Si se aspira a alcanzar una mejora continua en diferentes procesos manuales, no únicamente en el comentado en este trabajo, sino que hay infinidad de procesos industriales o de producción que puede automatizarse y, por ende, mejorarse cada vez más.

Las empresas de Costa Rica o de otras localidades deben conocer sobre la tecnología que conforma la automatización y aún más, si se hace un énfasis en los sistemas de conocimiento o análisis visual automático.

Las empresas deben de velar por la búsqueda de nuevos procesos y nuevas tecnologías y, sobre todo, los patrones, por las constantes capacitaciones a sus trabajadores, ya que son principalmente los futuros ingenieros, quienes estarán a cargo de la creación, mantenimiento y actualización de estas tecnologías o, mejor dicho, de estos procesos de automatización en las plantas.

El en caso de no implementar o darle la prioridad correspondiente a este tipo de tecnologías y a la mejora continua en las plantas, el país se podría quedar rezagado con respecto a los demás países que sí están apostando a estas tecnologías, porque saben que

ese es el futuro al que van todas las empresas de manufactura, grandes o medianas, a nivel mundial. Además de que hoy existe en el mundo y no es ningún secreto en el área de la tecnología, que esta área avanza a pasos agigantados cada día, más y más rápido, por lo tanto, se debe de estar siempre actualizado o por lo menos, conocer acerca de estas tecnologías, ya que a final de cuentas viene a mejorar los procesos o realizarlos con una mayor simplicidad, en un tiempo menor y mucho más efectivas.

1.6 Sistematización del Problema

Como parte de la sistematización que se plantea en esta investigación, se detallará una serie de soluciones con el fin de resolver el problema planteado, con el desarrollo del sistema automático de visión, para realizar todo el análisis correspondiente y el cálculo del volumen de las superficies.

También, con la implementación de un HMI para realizar la interacción hombre máquina para observar ciertos datos importantes como los kilogramos calculados en el momento preciso, capacidad del tanque, estado de las variables de los motores, entre otras cosas.

Por último, se va a asegurar la confiabilidad de los datos para poder realizar reportes y con base en estos reportes, poder tomar decisiones en la planta para comparar la producción real con respecto a la esperada.

1.7 Objetivo General

Diseñar un sistema capaz de medir la cantidad de un material en polvo mediante el análisis de imágenes obtenidas por una cámara en un entorno industrial y evaluar los datos obtenidos a lo largo de un periodo.

1.8 Objetivos Específicos

1. Desarrollar e implementar un algoritmo para la medición de un volumen a partir de una imagen capturada por computadora.
2. Crear un modelo a escala del sistema para la medición de un volumen a partir de una imagen capturada por computadora.
3. Diseñar e implementar de una interfaz amigable al usuario y un protocolo de comunicación industrial para la adaptación en una planta.

1.9 Estado Actual de la Investigación

Existen múltiples estudios sobre el análisis de imágenes a nivel volumétrico en el campo de la industria de la alimentación a nivel internacional, pero en el país no se está incursionando en este ámbito, por lo que se decidió investigar sobre este tema. Lo anterior es una de las razones por las cuales se realiza la investigación, complementada por las diferentes visitas a industrias mexicanas donde surgió la preocupación de que ciertos procesos industriales se realizan aun manualmente.

En las diferentes visitas que se realizaron en México, se pudo percibir que esto era un problema que se repetía, no utilizan sistemas de visualización automáticos en ciertos ambientes donde era totalmente factible su implementación, además de ser empresas sumamente grandes y a nivel de tecnología o en este caso, en temas de mejora continua y automatización de bastantes procesos en la planta que carecían siempre de esta herramienta.

En dichas plantas se encontraron los mismos problemas como que se comentaron anteriormente, ya que varios procesos que se realizan de forma arcaica, es decir, mediciones completamente manuales, en las que el operador del turno, únicamente con la vista indicaba cuántos kilos estaban produciendo o mejor dicho tenía el silo.

Esta situación preocupó bastante, ya que era una forma totalmente imprecisa de dar una medición, donde todo que el resto de la planta estaba automatizada con una precisión impresionante, pero estos procesos que también son delicados, no lo estaban y todo por falta de conocimiento, presupuesto o un plan de acción para solucionar o automatizarlo.

Lo anterior dio camino a que se visualizara el problema y se intentara solucionarlo, debido a que nadie había invertido tiempo o dinero, ya que solucionar este problema contemplaba una complejidad enorme en diferentes ámbitos, como por ejemplo, contratar especialistas en el tema de visualización o análisis de imágenes, ya que dicha implementación no podría llevarse a cabo por cualquier ingeniero de la planta, a menos de que se le capacitara en estos temas, pero tampoco era una opción, porque no había un curso específico o

documentación para guiarse a la perfección, por lo tanto, se debía de realizar una investigación sumamente extensa.

Todos estos impedimentos motivaron bastante a investigar y buscar documentación referente al tema con el fin de solucionar este problema, crear un sistema y un algoritmo con el cual se tuviera un conjunto de componentes capaces de realizar mediciones precisas por medio del análisis de imágenes, es decir, todo un reto.

Pero eso no era todo, ya que parecía que los datos obtenidos se debían de compartir, debido a que el operador los utiliza en sus reportes y también en la realización de ciertos cálculos matemáticos que afectaban la producción esperada y por ende, estos resultados se los transmitía a su superior, de tal modo que también se pensó en implementar un HMI con el fin de poder consultar la información que se está procesando en el análisis visual en esa pantalla amigable para el operador, donde podría consultar los kilogramos en ese momento, el volumen correspondiente, capacidad del tanque, variables de los motores (si están funcionando o no) y finalmente un reporte, porque esta información se almacena en una base de datos, todo esto para simplificar el trabajo del operador, automatizar el proceso con un equipo económicamente módico, ya que se utilizaron programas gratuitos y sobre todo, con una precisión bastante alta.

1.10 Metodológica

La metodología que se va a emplear en el presente trabajo investigativo se basa en una primera instancia en una en la investigación cuantitativa, ya que este puede validarse con los modelos y principios científicos por medio del diseño experimental que se utiliza para la obtención de mediciones de variables con el fin de establecer un comportamiento o, verificar o descartar alguna hipótesis o también, la modelación matemática, debido a que involucra el análisis de algún comportamiento mediante ecuaciones o cálculos matemáticos.

En una segunda instancia en una investigación cualitativa, porque la investigación desarrollada es completa, con el fin de extraer datos conocimiento o información de Internet en forma de libros y blogs, entre otros.

Es por esto que se puede decir que la metodología empleada para esta investigación es híbrida, puesto que combina tanto las metodologías cualitativas como las cuantitativas. Es importante destacar que últimamente, los científicos optan por las metodologías compuestas porque les permite llevar a cabo indagaciones cualitativas y cuantitativas en paralelo.

Con el pasar de los años y como producto del constante avance de la tecnología, la forma de ejecutar el análisis de imágenes ha cambiado considerablemente, en razón de que no es necesario invertir una gran cantidad de dinero para poder realizar análisis complejos y de calidad, por cuanto se encuentran cámaras en el mercado bastante potentes a un precio cómodo. De igual modo, los cálculos matemáticos bastante avanzados y complicados se pueden realizar con una mayor simplicidad al recurrir a librerías o complementos que ayudan drásticamente en esta parte y finalmente, encontrar suficiente documentación en Internet (libros o blogs, entre otros) que ayudan a entender el funcionamiento de los procesos, teoría y significados que se necesitan para una realizar correctamente esta implementación.

CAPÍTULO 2

2 Marco Teórico

2.1 Marco Situacional

La investigación se basa en la exploración y estudio de métodos de análisis de imágenes de industrias de Costa Rica y México. En el estudio se evalúan los problemas que se encontraron en las visitas a las plantas de producción y las deficiencias en algunos procesos de producción que aún no han sido automatizados, gracias a una breve reseña de las experiencias en una línea de producción y los conocimientos adquiridos durante el periodo de trabajo en la industria.

Lo anteriormente comentado marcará el precedente para la elaboración de la idea del proyecto, ya que en las diferentes visitas que se realizaron a lo largo del tiempo se han visto diferentes problemas que pueden ser fácilmente solucionados con un poco de tecnología y automatización.

Alguno de estos problemas está siempre presente en las líneas de producción, como, por ejemplo, en el cálculo de acumulaciones entre dos equipos que se realizan manualmente, un operador da un aproximado de esta acumulación y gracias a esta medición, se toman decisiones como aumentar la velocidad, seguir en velocidad nominal o irse a sobrevelocidad.

Este problema se puede solucionar o mejor dicho se solucionó implementado varios sensores a lo largo de las líneas y realizando ciertos cálculos matemáticas de áreas y pesos para saber con una mayor precisión cuál es la acumulación entre dos máquinas de una forma más precisa. El problema anterior también se puede solucionar fácilmente con un sistema de cámaras que calculen de forma automática las acumulaciones presentes en la línea.

Otro problema que se pudo observar en otra planta de producción mexicana fue cómo llevaban las fallas de algunas máquinas para indicar ciertos estados, por ejemplo, la pérdida de capacidad, paros, microparos o paros reactivos, entre un montón más. En algunos casos, los operadores apuntaban estas fallas con su tiempo de inicio y fin, en un papel y lo ingresaban

días después, incluso en algunos casos se vio como ese papel se les perdía y tenían que inventar los problemas de la línea. Además, en otros casos, el operador no estaba en la línea por estar haciendo otras tareas y, por ejemplo, el operador pudo haber empezado a la 1:00 p. m. y se dio cuenta de este error hasta las 4:00 p. m. que regresó a la línea y obviamente, no sabía cuál había sido el tiempo real cuando inició (1:00 p. m.) y simplemente apuntaba 3:30 p. m., afectando completamente el tiempo de fallo de esa máquina y, por lo tanto, los reportes irían con información errónea.

Esto también se pudo solucionar automatizando el proceso en las diferentes líneas de producción, realizando cierta configuración en los PLC y tomando esas variables para saber justamente cuándo no está funcionando alguna máquina e indicar inmediatamente a los operadores por medio del HMI, volviéndolo un sistema más robusto y sin fallos a la hora de calcular ese tiempo.

El otro problema que también se notó, fue el problema de mediciones de material en la gran mayoría de polvos en los diferentes tanques, porque los operadores debían subir a lo alto de estos, poniéndose en riesgo de sufrir algún accidente y una vez que estaban en posición y por medio de un cálculo totalmente arcaico, indicaban un valor correspondiente a los kilogramos totales en ese momento. Dicho cálculo era totalmente arcaico y sumamente impreciso, puesto que era una aproximación por parte del operador a lo que él creía que tenía el tanque en ese momento, afectando muchos indicadores en la planta.

Conversando con los operadores y altos mandos de la empresa se percibió que no existía toda una forma de solucionarlo, habían realizado ciertos estudios y varias propuestas, pero ninguna era del todo viable por diferentes motivos y el principal era el económico y sobre todo de tiempo.

Lo cual llevó a la tarea de investigar cómo se podía solucionar dicho problema de una manera efectiva, fácil en el sentido de que la instalación de la herramienta fuera de forma sencilla y sobre todo a un costo mínimo, utilizando tecnologías o sistemas de visión

automáticas sin ningún costo o uno mínimo, como sería el caso de utilizar programas de código abierto o simplemente gratuitos.

La implementación de estas tecnologías es inevitable en estos países donde no se pueden dar el lujo de realizar mediciones u obtener datos muy delicados para la planta o la toma de decisiones con el fin de mejorar su rendimiento.

Cualquier beneficio que logre obtener estas empresas siempre será bien recibido, ya que, en el primer caso, al obtener las acumulaciones precisas, pueden tomar decisiones para regular las velocidades de los equipos y tratar de evitar lo mayor posible, sus paros, es decir, evitar a toda costa detener los motores, debido a que al arrancarlos nuevamente y en reiteradas ocasiones, pueden generar todo tipo de problemas.

En el segundo caso se va a un tener control más preciso del inicio y la duración de algunos problemas que pueden presentar las máquinas en algún momento de la producción, además de informar directamente a los operadores para que realicen las acciones requeridas, como avisarle a mantenimiento, ya que la falla pudo haber sido provocado por algún componente que debe de cambiarse.

Finalmente, el problema sobre el cual se va a tratar en este trabajo, asegurará un cálculo del volumen y por dentro de la masa del producto, por medio de sistemas automáticos de visión con el fin de obtener datos más precisos en la producción de algún material, así se va a tener una real y más precisa que se pueda comparar con la esperada y de esta forma, tomar decisiones más acertadas en diferentes áreas, como por ejemplo, si el valor de la producción esperada es un valor cercano a la real; en caso de ser así, se puede concluir que la planta está trabajando de una forma, pero contrariamente, el gerente de planta puede empezar a investigar en qué se falló y se tuvo una producción tan baja.

2.2 Marco Teórico

2.2.1 Configuración de Recursos.

La parte más importante de este trabajo yace en la capacidad de la máquina de capturar la información del entorno físico y estimar una medición, este tema se centra en cómo la computadora logra “ver” el mundo exterior y esto se describe en este subtema.

2.2.2 Visión de Computadora.

Aristóteles Visión indica que es saber qué hay y dónde está mediante la vista, otros autores mencionan que la visión es recuperar, de la información que se captura con los sentidos, las características físicas del mundo.

En el libro *“Vision : a computational investigation into the human representation and processing of visual information”* se menciona que: visión es un proceso que produce a partir de las imágenes del mundo exterior, una descripción que es útil para el observador y que no tiene información irrelevante (Marr, 1982).

De la cual se puede recatar dos aspectos muy importantes que servirán a lo largo de este capítulo, los cuales son: que la descripción es útil para el observador y que no tiene información irrelevante. Esto es importante, debido a que la base de este proyecto es ponerle especial atención a la superficie y las irregularidades que se encuentren. Para comprender el punto de “Ser útil para el observador”, antes se tiene que entender cómo realmente “ve” la máquina.

2.2.3 Partes de la Visión.

Un detalle por entender es que los sistemas de visión deben contener al menos los siguientes aspectos básicos para el funcionamiento:

- Dispositivo de captura
- Memoria
- Procesador

- Monitor o dispositivo de salida

En el ámbito de la electrónica y como parte de los requisitos básicos de lectura de este documento está la comprensión del funcionamiento de todos estos elementos, entendiendo que los dispositivos de captura serán las cámaras o sensores que capturarán el medio del mundo real y lo transformará en señales eléctricas, estas serán almacenadas en la memoria RAM para su procesamiento posterior y que los monitores serán la forma de representar los datos procesados para que el usuario identifique las modificaciones aplicadas en el postprocesado.

2.2.4 Proceso de Captura de Imagen.

Las imágenes realmente se almacenan en la máquina mediante una trasformación de las propiedades físicas que percibe el lente de cámara y una digitalización de esa información, una de las principales propiedades utilizadas es la radiación de luz o el brillo que refleja un objeto, el cual se localiza mediante un sistema de coordenadas (x, y) y se le asigna un espacio en un plano de dos dimensiones.

Una de las formas más básicas de análisis de imágenes es mediante las conocidas en escalas de grises, debido a que la obtención del brillo en un punto específico es mucho más sencilla cuando solamente se tiene una escala de un solo color. Una forma de representar las imágenes es mediante la función:

$$I = f(x, y) \tag{1}$$

Donde I es la intensidad del brillo en un punto específico dado por el par de coordenadas X, Y.

Pero también es posible capturar e interpretar imágenes que no sean en escala de grises, completamente a color y esto se hace mediante la descomposición de cada punto en tres colores básicos: rojo, verde y azul, el parámetro a evaluar en este caso sería la longitud de onda que refleja cada objeto por lo que al dividirlo en “qué cantidad de rojo, azul y verde” tiene cada punto se logra obtener un grupo de coordenadas multiespectral de la siguiente forma:

$$I = f(x, y) \tag{2}$$

$$I = [frojo(x, y), fazul(x, y), fverde(x, y)] \tag{3}$$

Como lo se mencionó anteriormente, estos valores pueden ser digitalizados, por lo que mediante una discretización o análisis de cada punto de la imagen y Luis Enrique Sucar propone que la forma de discretizar estas imágenes es mediante la multiplicación de la función con un arreglo bidimensional de funciones delta, quedando de la siguiente forma: (Sucar, 2011)

$$f_s(x, y) = \iint_{-\infty}^{\infty} f(x, y), \delta(x - x_0, y - y_0), dx, dy \tag{4}$$

Usualmente se utilizan un bloque de números entre 0 y 255, por lo que una intensidad se puede representar en un arreglo de valores entre estos datos y cada punto sea representado por este valor, esto es lo que se conoce como pixel.

Los valores de cada elemento de la imagen representan desde niveles oscuros de luminosidad hasta valores claros. El nivel más oscuro es el valor más bajo del intervalo y viene representado por el negro y el nivel más claro se representa con el blanco y es el más alto.

Para colocarlo en palabras simples, para que una imagen sea capturada y reconstruida en una máquina depende de los fenómenos de reflexión y refracción de la luz en un objeto y cómo esta luz es recibida por el lente de una cámara para que, mediante el proceso descrito anteriormente, discretice esos fenómenos captados por el lente y traduzca a un lenguaje que la máquina entienda.

2.2.5 Iluminación.

Este proceso, al ser tan dependiente del comportamiento de los objetos a la luz, es de vital importancia el uso de una iluminación adecuada del entorno para captar la menor cantidad de información desechable, es decir, poder recuperar de la mejor forma estos puntos de brillo o

saturación de una longitud de onda (color) para aprovechar la mayor cantidad de puntos y obtener un porcentaje más alto de la imagen para reconstruir, puesto que los distintos tipos de fuentes de iluminación pueden variar desde las longitudes de onda que puede abarcar en su transmisión, las temperaturas que pueden alcanzar y las fluctuaciones en el tiempo que puede tener su emisor.

Existen muchos tipos de iluminación para utilizar, los cuales dependen del entorno, el objeto y la necesidad que se tenga, el autor Enrique Alegre destaca un grupo importante de tipos o técnicas de iluminación (Alegre, 2016).

Entre los tipos de iluminación se puede encontrar, la iluminación natural y la artificial, pero como es bastante claro, la natural tiene múltiples problemas como el tiempo y las condiciones que se requieren para aprovecharla, las cuales no son compatibles con la intención de la industria para un funcionamiento ininterrumpido, por lo que no será caso de estudio en este momento.

Por otro lado, está la iluminación artificial, la cual puede venir de múltiples fuentes, tipos y al ser fácilmente controlable, es posible aplicar técnicas para mejorar los resultados en las mediciones. Alegre menciona entre los tipos de iluminación artificial lo que se detalla en la Tabla 1.

Tabla 1

Características de cada uno de los tipos de iluminación artificial.

Tipo de iluminación	Ventajas	Inconvenientes
Incandescente /Halógena	• Bajo coste y fáciles de utilizar. • Permiten ajustar la intensidad de luz.	• Desprenden una gran cantidad de calor. • Su espectro se centra en el rojo siendo deficiente para

Tipo de iluminación	Ventajas	Inconvenientes
	• Oscila 50 veces por segundo.	azules verdes o amarillos.
Fluorescentes	• Se calienta menos que el incandescente • Su espectro se centra en los colores del ojo humano • La duración está estimada en torno a 10.000 horas.	• La longitud de onda de la luz cambia con el uso. • Para que sean válidos en aplicaciones industriales tienen que trabajar a una frecuencia del orden de 25 KHz.
Led	• Gran durabilidad (100.000 horas). • Posibilidad de encender y apagar solamente en el tiempo de captura de la imagen. • Fácil elección de la longitud de onda de la fuente de luz dentro del espectro visible e infrarrojo. • Las fuentes de luz se pueden construir en multitud de formas.	• Precio
Láser	• • Se utilizan para generar luz estructurada con forma diversas tales como líneas, líneas paralelas, líneas cruzadas, retículas, puntos y matriz de puntos. Para generar las formas se	• Precio.

Tipo de iluminación	Ventajas	Inconvenientes
	utilizan ópticas específicas. • Están disponibles en multitud de longitud de ondas desde el visible al infrarrojo cercano. • Dado que el ojo humano es muy sensible al verde, un diodo láser en esta longitud de onda genera un mejor contraste en los bordes especialmente sobre superficies rojas.	

Nota. Tomado de Conceptos y Métodos en Visión por Computador. (Alegre, 2016, pág. 13)

Dependiendo del entorno a iluminar, es muy importante tomar en cuenta las ventajas y desventajas que tenga cada fuente de iluminación para aprovechar al máximo la detención en el mismo.

2.2.6 Técnicas de Iluminación.

Además de tomar en cuenta la fuente a utilizar, es de igual importancia la técnica para distribuir esa luz a lo largo de los objetos o zonas para evitar que aparezcan problemas de saturación, contrastes altos o brillos excesivos y que estos problemas provoquen que los sistemas no sean capaces de aplicar los algoritmos para el correcto procesado de las imágenes.

Se consideraría una buena iluminación a aquella que no produzca saturaciones en brillos o contrastes en la imagen, al igual que aquella que no generase sombras, debido a que todos estos fenómenos provocan que múltiples pixeles pierdan fiabilidad e importancia a la

hora de la reconstrucción de la imagen y se conviertan en pixeles de poco interés, por lo que se han considerados como innecesarios y se obvien de las reconstrucciones.

Para evitar que se produzcan los fenómenos descritos anteriormente, se estudiaron múltiples técnicas para distribuir la luz sobre una superficie de maneras adecuadas, tomando en consideración los detalles como la superficie a evaluar, la exposición del lente a la luz, el tiempo que el lente debe estar abierto capturando la imagen, la sensibilidad a la luz que tenga el lente el tamaño del área a evaluar entre otros detalles.

Entre esas técnicas se podría pensar en las siguientes.

- Direccional: se utilizan varios puntos de iluminación en ángulos de incidencia que no sean paralelos ni perpendiculares al eje de la cámara, esta técnica produce brillos indeseados y sombras debido a la aplicación de varias fuentes de luz, pero es una técnica barata y se puede adaptar a cualquier ambiente, esta técnica es muy utilizada en la industria del cine y la fotografía.
- Lateral o “darkfield”: en este caso, la fuente de luz se encuentra en un ángulo de 0 grados con el objeto y, por ende, a 90 grados del eje de la cámara, pero al aplicar la luz en esta forma es probable que se generen sombras indeseadas. Este tipo de técnica es ideal para la detección de texturas, ya que es más eficiente para la detección de grietas o acumulaciones en las superficies y en los objetos.
- En forma de anillo: esta técnica está muy de moda últimamente con el uso de los “anillos de iluminación” que popularizaron las redes sociales con sus videos cortos, esta técnica consiste en iluminar mediante un anillo y un ángulo de incidencia en la misma dirección del eje de la cámara, colocando la cámara en el centro del anillo, gracias a esta práctica es posible reducir casi en su totalidad las sombras y resaltar los detalles centrales de los objetos, el problema principal

de esta técnica se encuentra en que se producen reflejos indeseados en la superficie

- Difuminada: la técnica de difusión se utiliza cuando se necesita disminuir la cantidad lumínica en la superficie, se trata de aplicar un lente o cristal que suavice la cantidad de luz que incide en la superficie, esto ocasionará que se reduzcan los brillos y las sombras, pero al llevar un objeto extra (el lente difusor) requiere más espacio. Si se aplica en un ángulo alineado con el eje de la cámara, se le conoce como difusión axial, pero en este caso pierde intensidad al aplicar el lente difusor.
- Estructurada: si se utilizan medios como el láser es posible generar patrones de iluminación y esta técnica permite que, se incida sobre una superficie con un patrón específico obteniendo la estructura real del objeto o superficie, esta técnica es utilizada principalmente para cortes o sistemas que necesiten de precisión milimétrica siempre que no sea dependiente del color, ya que así se pierde la capacidad de detallar colores.
- Contraluz: si se coloca dos fuentes de luz en direcciones opuestas apuntando al objeto de interés, se obtendrá esta técnica con la que se puede detallar bordes de manera más precisa, pero perdiendo la capacidad de ver detalles de superficies.

Como se puede observar, depende de la necesidad que se tenga, es posible usar alguna técnica, por lo que entender a profundidad la situación y aplicación del sistema es crucial para la selección del tipo de iluminación y la técnica para aplicar en cada entorno.

Al igual que con cada fenómeno de la naturaleza, es posible calcular la intensidad lumínica de una fuente que se puede aprovechar para obtener los mejores resultados. La intensidad de luz que despende una fuente se puede calcular mediante la siguiente forma:

$$I = d\emptyset/d\omega \quad (5)$$

Esta fórmula dará el flujo por ángulo sólido en watts/steradian.

También es posible calcular el flujo que incide en un área específica mediante la fórmula:

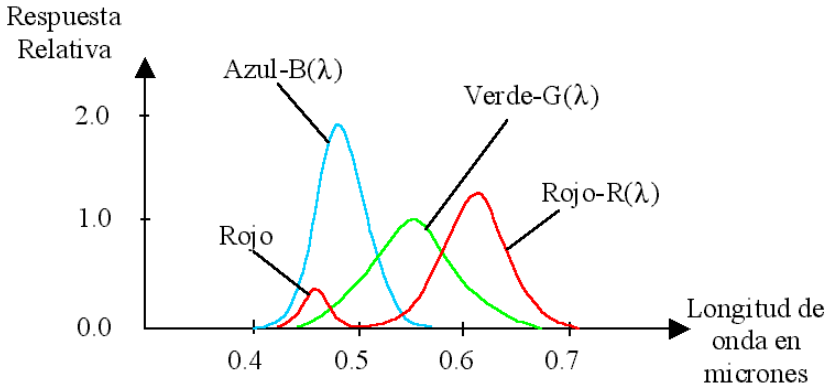
$$E = d\emptyset/dA \quad (6)$$

2.2.7 Color en la Imagen.

Otro término para tomar en cuenta es el color en una imagen, se sabe que el color es un fenómeno que se percibe gracias a las distintas longitudes de onda que capta el ojo humano que permite diferenciar toda la gama de colores que se conoce, mediante los tres receptores de longitudes que se capta.

Figura 1

Respuesta relativa según longitud de onda en micrones.



Nota. Tomado de Ingeniería de Sistemas y Automática, UMH. (Ingeniería de Sistemas y Automática UMH, 2022).

Gracias al conocimiento que se ha adquirido hasta ahora, se puede partir del hecho que existen dos formas en las que se capta una imagen, por medio de las longitudes de onda o frecuencias y por medio de los puntos de brillo en la imagen y esto es lo que postula Sucar en su documento en el que describe que los dos modelos en los que se pueden trabajar las

imágenes, el modelo RGB (Red/Green/Blue) basado en frecuencias y el modelo HSI (Hue/Saturation/Intensity) basado en espacio.

2.2.8 Modelo RGB.

El modelo RGB, como sus siglas lo describen, corresponde al modelo que se basa en la detección de las longitudes de onda por pixel para detectar su color, partiendo de los tres receptores que se encuentran en el ojo humano rojo, verde y azul. Es posible crear una normalización de estos valores mediante el uso de una regla de suma de triadas, es decir:

$$Cantidad\ de\ color\ = \frac{Color\ x}{(cantidad\ de\ rojo + cantidad\ de\ verde + cantidad\ de\ azul)} \qquad (7)$$

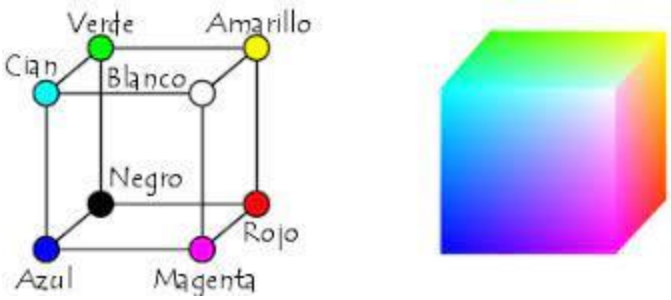
Estas cantidades serán representadas a nivel de máquina mediante la discretización de cada color en valores dentro de 0 y 255 formando el valor básico del pixel en el modelo RGB. La suma de todos los colores en este modelo da como resultado el color blanco. En los dispositivos actuales que despliegan imagen, con la combinación de estos tres colores es posible recrear cualquier color de la gama nombrada de colores.

Entre más cantidad de pixeles en una imagen, mejor definición general de colores tendrá, debido a que el valor unitario mínimo será reducido y para describir un cambio de un color a otro en un grupo de pixeles requerirá más de estos para la transición.

Este modelo puede representarse mediante un cubo donde el vértice de origen sería el color negro y el vértice opuesto sería el color blanco y partiendo del origen, cada uno de los ejes resultantes serían los colores que conforman el pixel.

Figura 2

Matriz de colores RGB.



Nota. Tomado de Instrumento virtual para estudiar la Representación de color RGB en máquinas de Visión con Labview. (Fuentes-Hernández, 2019, pág. 1008).

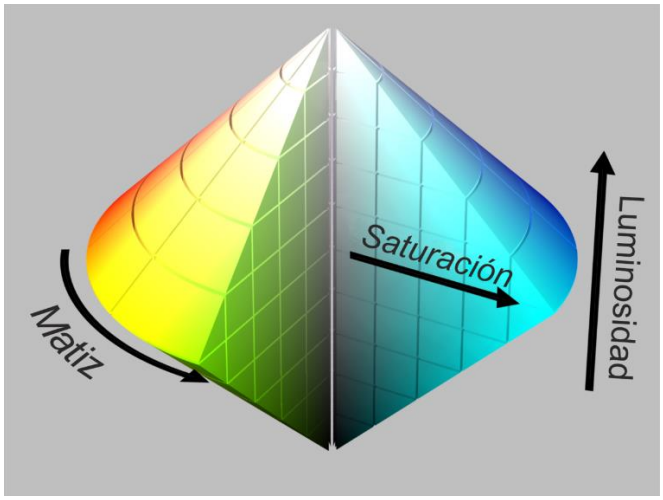
2.2.9 El Modelo HSI.

Este modelo se basa en el funcionamiento básico del ojo humano, ya que este codifica el brillo o la intensidad del punto, la saturación o el inverso a la cantidad de blanco en una imagen y, por último, el croma o matiz, que es atributo de una luz visible por el cual se diferencia o se asemeja a los colores primarios: rojo, verde y azul. El término también se usa para referirse a colores que no tienen tinte ni sombra añadidos.

El modelo HSI, al igual que el RGB es posible representarlo, pero tiene una forma distinta mediante un cono en el que la altura viene dada por la intensidad o luminosidad, el radio del cono es la saturación y el ángulo desde el origen es el croma o matiz.

Figura 3

Representación del modelo HSI



Nota. Tomado de sitio web PNGWING. (PNGWING, 2022).

Como se ha mencionado anteriormente, todos los fenómenos físicos pueden ser tratados a nivel matemático y este no es la excepción. Dependiendo de las necesidades de análisis, es necesario un modelo o el otro, por lo que existe una forma matemática de convertir un modelo en el otro el cual ayuda a describir Sucar en su libro (Sucar, 2011).

Si se tiene un modelo RGB y se necesitan sus componentes, HSI se realiza de la siguiente forma:

$$H = \cos^{-1}\left(\frac{\frac{1}{2} * (R-G) + (R-B)}{\sqrt{(R-G)^2 + (R-B) * (G-B}}\right) \quad (8)$$

$$S = 1 - \left(\frac{3 * [\min(R,G,B)]}{R+G+B}\right) \quad (9)$$

$$SI = \frac{1}{3} * (R+G+B) \quad (10)$$

Así como si se tiene un modelo HSI es posible convertirlo en RGB de la siguiente forma:

$$B = I * (1-S) \quad (11)$$

$$R = I * \left[1 + \frac{S * \cos H}{\cos(60° - H)}\right] \quad (12)$$

$$G = 3I - (R+B) \quad (13)$$

2.2.10 Elementos de sistema de visión.

Antes de pasar al procesado directo de la imagen y el amplio mundo que envuelve los temas de la visión de computadoras, es necesario comprender cómo funcionan estos sistemas a nivel de algoritmo, es decir, cuál es la estructura que cumplen estos sistemas para comenzar y terminar el análisis de una imagen desde su captura hasta la obtención de la información requerida.

Además del algoritmo, también es importante entender las partes físicas que componen estos sistemas, según se detalla a continuación. Tal y como lo describe David Marr en su libro, la visión es un proceso, por lo que tener claro el algoritmo con el que se desarrolla, es de suma importancia para sobrellevar un proyecto basado en este proceso (Marr, 1982). Marr describe que este proceso se puede derivar en dos grandes bloques, en el bloque de visión y el bloque de procesamiento.

El bloque de visión puede subdividirse en tres puntos más que serían el objetivo del proceso, lo cual es lo que se desea obtener al “ver” mediante la máquina y lo plantea mediante la definición de las metas y las estrategias para alcanzar esta meta.

Luego se plantea la representación del algoritmo para describir cómo llevar a cabo a nivel teórico este objetivo, es decir, qué pasos se debe seguir, cuáles transformaciones y qué cálculos deben llevar dichas transformaciones. Para finalizar, se centra en el método sobre cómo se resolverá el algoritmo a nivel físico, qué se necesita para resolver esas transformaciones y qué datos se obtendrá de estos.

2.2.11 El bloque del procesamiento.

Es posible procesar una imagen realizándolo de distintas formas y al igual que la programación en distintos lenguajes que también hay niveles para hacerlo, se podría subdividir este bloque en tres niveles para el desarrollo de un procesamiento, que se podrían ver

dependientes entre sí, aunque necesariamente no lo son, incluso, podrían ser cíclicos o retroalimentados entre ellos.

Los niveles se pueden definir entre bajo, medio y alto. En los que los bajos hacen referencia a la extracción de información y características de los pixeles o las unidades básicas, como, por ejemplo, degradación de bordes o captura de orillas entre otros. Los niveles intermedios involucran el agrupamiento de elementos obtenidos, para revisar regiones específicas de la imagen o segmentaciones de esta. En los niveles más altos se encuentra la valoración de objetos mediante los modelos de redes neuronales o conocimientos adquiridos.

2.2.12 Mejoramiento de la Imagen.

Uno de los cuestionamientos que se hizo es que, si la imagen iba ser nítida o no, para ello se tuvo que investigar en varios lugares y el libro más acertado para este proyecto fue el de Visión Computacional de Enrique Sucar.

El autor indica lo siguiente: “El objetivo de visión de bajo nivel o procesamiento temprano es hacer transformaciones directamente sobre la imagen para obtener información de las propiedades físicas de los objetos que están en ella y que sean de mayor utilidad para los siguientes niveles de visión” (Sucar, 2011, pág. 15). Dentro del mejoramiento de imagen, Sucar indica que los principales atributos que se debe considerar para tener una imagen nítida son:

- Discontinuidades u orillas
- Color
- Textura
- Gradiente y profundidad

También manifiesta “De tal forma, que podemos pensar que, de la imagen original, se obtendrá una nueva imagen por cada característica que se extraiga de la imagen, llamadas imágenes intrínsecas” (Sucar, 2011, pág. 15).

2.2.13 Aumento Lineal del Contraste.

Consiste en las transformaciones lineales, ya que se tiene una correspondencia o, mejor dicho, una función lineal de los valores, en este caso, de la intensidad de los pixeles de la imagen de salida con respecto a la imagen de entrada, con el fin de obtener el negativo de una imagen, aumentar o disminuir la intensidad y finalmente, aumentar el contraste.

Utilizando el valor de intensidad mínimo y máximo en una imagen, se puede aumentar su contraste, entonces el principio aquí es tratar de llevar el valor mínimo cero y el máximo a 255, pensando en imágenes monocromáticas.

Así, obtenido cierto factor o pendiente, ya que esta transformación crea que las intensidades se espacien de acuerdo con estos factores, por lo tanto, se podría decir que el factor para este acrecentamiento lineal de contraste se puede expresar de la siguiente forma:

$$\mathrm{C(x,y)} = ((\mathrm{I(x,y)} - \min)/(\max - \min) * 255) \qquad (14)$$

Donde I (x; y) es la imagen que procesar y C (x; y) es la imagen con aumento lineal del contraste.

Por lo tanto, si se regresa a la idea planteada anteriormente se puede verificar cómodamente que para I (x; y) en min, C (x; y) resulta cero, ya que el numerador en este caso vendría siendo igual a cero y luego para I (x; y) en máximo, C (x; y) resulta en 255, ya que utiliza un cociente igual a 1 (Sucar, 2011, pág. 18).

2.2.14 Filtrado en el Dominio Espacial.

Se utilizarán las técnicas o filtros que se encuentran en la página 23 del libro del señor Sucar para evitar el pixeleado de las imágenes que se van a trabajar durante el proyecto. La siguiente figura 4 ilustra un ejemplo de una máscara de 3 x 3, que se centra sobre el pixel de interés, de forma que el nuevo valor del pixel dependa de los pixeles que cubre la máscara.

Figura 4

Ejemplo de mascara de 3x3

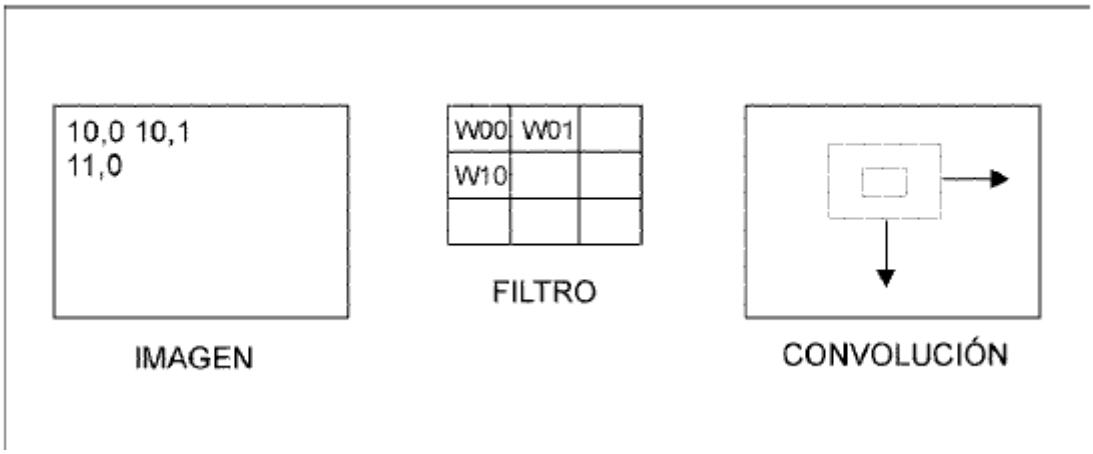
$w_{1,1}$	$w_{1,2}$	$w_{1,3}$
$w_{2,1}$	$w_{2,2}$	$w_{2,3}$
$w_{3,1}$	$w_{3,2}$	$w_{3,3}$

Nota. Tomado de Libro Visión Computacional. (Enrique Sucar, 2011, pág. 23).

En la figura 5 se ilustra el proceso de filtrado o convolución con la máscara.

Figura 5

Filtrado en el dominio espacial



Nota. Tomado de Libro Visión Computacional. (Enrique Sucar, 2011, pág. 24).

2.2.15 Filtrado Adaptable.

"Uno de los problemas al aplicar filtros pasa bajo o de suavizamiento para eliminar ruido, es que también se pueden eliminar atributos de la imagen que son importantes para las siguientes etapas de visión. Como veremos en el siguiente capítulo, las orillas o bordes en la imagen son muy importantes, y estos tienden a emborronarse al aplicar un filtro de suavizamiento" (Sucar, 2011, pág. 30).

El filtro de mediana intenta preservar las orillas mientras que suaviza (promedia) regiones homogéneas. Aunque da mejores resultados que un filtro promedio, el filtro de mediana no logra resultados óptimos en el compromiso de preservar orillas y eliminar ruido. Por ello se han desarrollado otras técnicas más sofisticadas entre las que destacan:

- Difusión anisotrópica
- Campos aleatorios de Márkov
- Filtrado Gaussiano no-lineal
- Filtrado Gaussiano adaptable. (Sucar, 2011, pág. 25)

2.2.16 Detección de Orillas.

"Diversos experimentos psicobiológicos han mostrado que el sistema visual humano utiliza una amplia gama de fuentes de información, tales como las sombras, proporciones, longitudes, color, curvatura e intensidades" (Sucar, 2011, pág. 35).

Uno de los inconvenientes que se tuvo a la hora de realizar este tipo de detección fue si realmente la computadora con la ayuda de la cámara estaba detectando orillas, pues es una tarea particularmente importante para esta aplicación. El autor explica la forma adecuada de solucionar este problema: "La información de orillas es procesada por el sistema visual primario, en donde se encuentran células especializadas que responden a las discontinuidades. La visión humana utiliza las orillas de manera jerárquica, agrupándolas y utilizando la experiencia visual hasta poder reconocer objetos más complicados que líneas, tales como rostros y objetos geométricos" (Sucar, 2011, pág. 36).

2.2.17 Visión Tridimensional.

Uno de los factores más importantes de este trabajo era no perder información cuando se pasa una imagen tridimensional a una bidimensional, para ello existen alternativas para poder obtener la tercera dimensión que se ha perdido en el proceso de dimensión, según en su libro Visión Computacional, una alternativa es "usar dos imágenes mediante visión

estereoscópica. Otras consisten en utilizar otras propiedades de la imagen como sombreado o textura para obtener un estimado de la profundidad, o al menos de la profundidad relativa (gradiente)" (Sucar, 2011, pág. 83).

2.2.18 Visión en estereoscópica.

Se decide utilizar el siguiente enfoque para obtener información de 3D o profundidad.

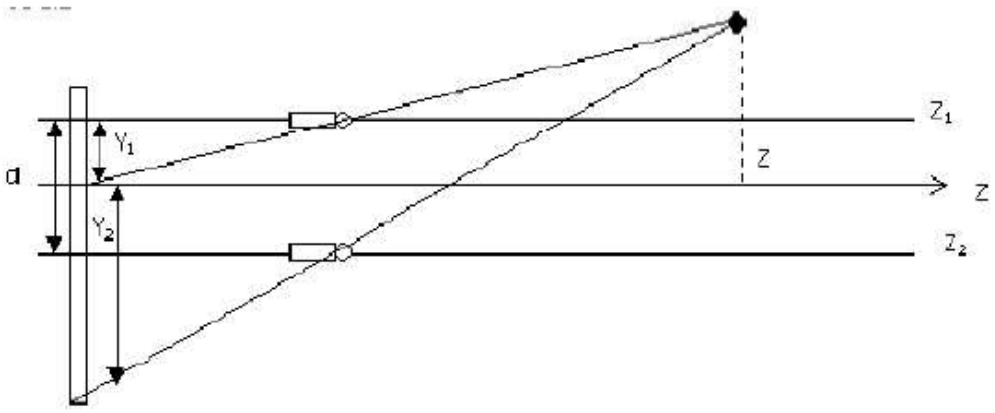
Una forma de recuperar la tercera dimensión es utilizar dos (o más) imágenes, en análoga con los sistemas de visión biológicos. Se colocan dos cámaras en posiciones distintas a una distancia conocida para obtener dos imágenes de cada punto de donde se puede recuperar su posición en 3-D, según se detalla en la figura 6.

El algoritmo básico consiste en cuatro etapas:

1. Obtener dos imágenes separadas por una distancia d.
2. Idéntica puntos correspondientes.
3. Utilizar triangulación para determinar las dos líneas en 3D en las que está el punto.
4. Intersecar las líneas para obtener el punto en 3D. (Sucar, 2011, pág. 35)

Figura 6

Visión estereoscópica. Un punto (z) tiene dos proyecciones diferentes en las cámaras, y1; y2. Las cámaras están separadas por una distancia d.



Nota. Tomado de Libro Visión Computacional. (Enrique Sucar, 2011, pág. 35).

2.2.19 Visión en 3D

Atendiendo ahora en el objetivo principal del proyecto, luego de entender cómo realmente la máquina “ve”, la forma en la que es posible procesar y mejorar una imagen, las capacidades de un sistema de visión para captar las profundidades y las innumerables posibilidades que abre el mundo del procesamiento de imágenes, se debe hacer énfasis en los procesos de evaluar los entornos, no en dos dimensiones, sino en tres, creando un nuevo campo de estudio en el proyecto.

2.2.20 Reconstrucción de objetos y superficies en 3D.

La reconstrucción de objetos en 3D según se ha demostrado por muchos autores de libros y publicaciones científicas, se debe basar en el tipo de medio que captura la imagen y de su forma de entregar la imagen y generalmente trabajar en tres grandes procesos: la adquisición de la imagen, procesamiento de par de imágenes (ya sea estereoscópicas o imágenes de profundidad) y el modelado.

Al tener aplicaciones tan amplias, la reconstrucción de objetos y texturas en 3D se han adoptado para optimizarse en cada ámbito en las que se necesite, por lo tanto, existen muchas formas de realizar el procedimiento de transformar un objeto real en una imagen tridimensional almacenada en las computadoras.

Volviendo al tema de los métodos de interpretar la información brindada por el dispositivo de captura, según Enrique Alegre, se puede encontrar que se podrían adquirir los datos mediante nubes de puntos o “PointCloud”, arreglos de profundidad, mallas de polígonos e imágenes de profundidad (Alegre, 2016).

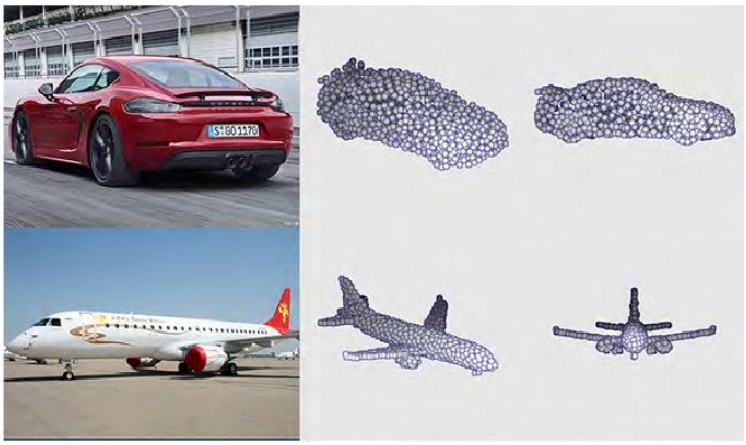
2.2.21 Nubes de puntos.

Las nubes de puntos o PointCloud es una representación mediante puntos definidos por las coordenadas cartesianas (X, Y, Z) que describen la imagen en un ambiente tridimensional.

Existen dos tipos de formas de almacenar estos datos, mediante un patrón desorganizado donde se almacenan todos los posibles vértices sin un orden lógico en un archivo en el que se separarán los datos de coordenadas por comas y no tendrían ningún tipo de información de la topología, esta forma de almacenar los datos es popular cuando no se necesita gran precisión ni gran eficiencia del sistema, pero sí, velocidad a la hora de reconstruir el objeto, ya que el sistema solo se preocupa por colocar el vértice en su lugar.

Figura 7

Ejemplo de objetos reconstruidos con nubes de puntos.



Nota. Tomada de la publicación RealPoint3D: An Efficient Generation Network for 3D Object Reconstruction from a Single Image. (Zhang, 2019).

Si se almacenan los datos mediante una relación topológica y dando una estructura a los vértices normalmente mediante una matriz, se le conoce como una nube de puntos organizada o estructurada, esta forma de almacenar y representar la nube de puntos es la más común, ya que cuenta con una mayor precisión en el modelado de objetos, es muy usada en los escáneres modernos de tres dimensiones,

Como se menciona anteriormente, existen distintas formas de capturar esta información, las dos más importantes son mediante el uso de láser o la tecnología LiDAR y el uso de la fotografía estereográfica.

2.2.21.1 Nubes de puntos mediante LiDAR.

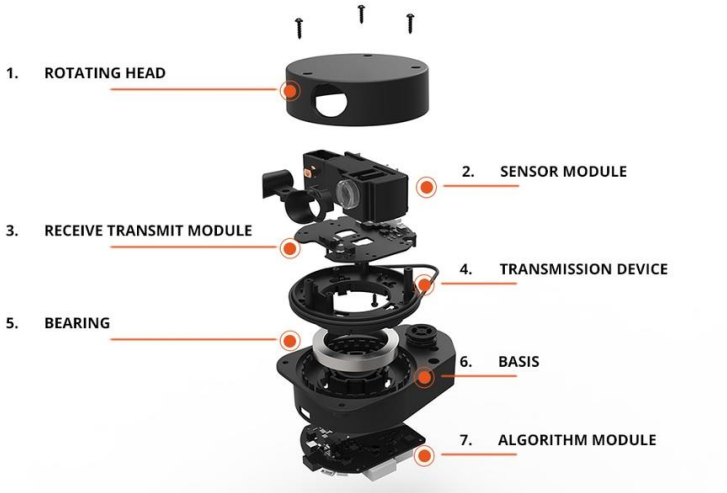
LiDAR se refiere a Ligth Detection and Ranging, es decir, detección y medición de rango de la luz y como su nombre lo explica, es el procedimiento por el cual se puede escanear un área para detectar una distancia mediante el tiempo que tarda una luz pulsante y reflejarse en un objeto.

Esta técnica de captura de imagen permite que se cree una nube de puntos de precisión milimétrica gracias al uso de tecnologías de iluminación láser (descritas anteriormente) siendo una de las tecnologías más comunes en la captura de imágenes geográficas, por su precisión a largas distancias.

El problema de esta técnica es que el costo es elevado, debido a la construcción del dispositivo que se encarga de enviar el haz de luz y hacerlo pulsante; en la figura 8 se puede apreciar la construcción de un dispositivo LiDAR y se nota que cuenta con la tapa rotativa que mediante el movimiento convierte un haz constante en una luz pulsante, lo compone principalmente un transmisor de luz, generalmente un láser, un sensor detector, rodamientos para permitir el giro de los componentes y su base (Li, 2019).

Figura 8

Dispositivo LiDAR

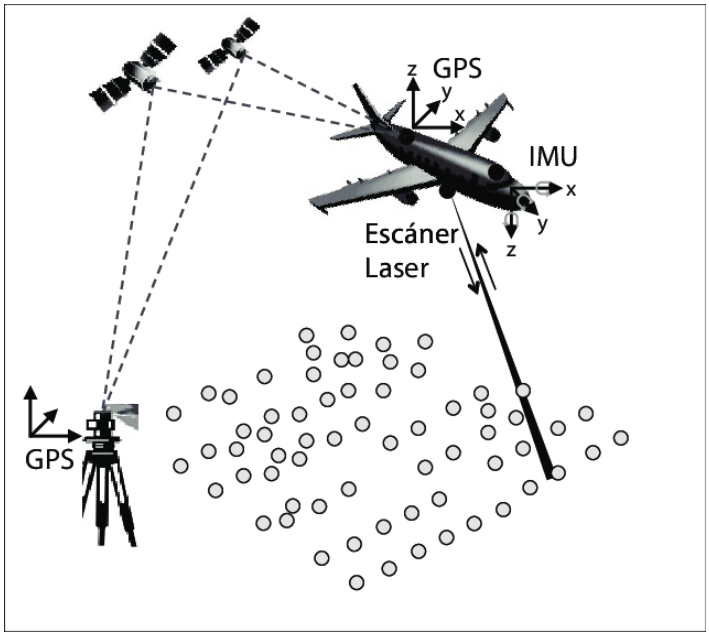


Nota. Tomado de sitio Web Direct Industry. (Industry, s.f.).

El funcionamiento en una aplicación geográfica es mediante el aerotransportado, utilizando un GPS para localizar los puntos donde se realizó la medición y almacenando esos puntos relacionados con la inspección de tiempo que tardó el haz de luz en viajar hasta la superficie y volver, es lo que compone la nube de puntos que se obtiene en esta técnica.

Figura 9

Funcionamiento en una aplicación geográfica



Nota. Tomado de Evaluación de la exactitud posicional vertical de una nube de puntos topográficos LiDAR usando topografía convencional como referencia. (Salinas-Castillo, 2014).

2.2.21.2 Nube de puntos basado en fotografía estereográfica.

Como se mencionó anteriormente, la visión estéreo o fotografía estereográfica es un pilar para la detección de profundidades en el ambiente de visión por computadora y para la reconstrucción de objetos y superficies en tres dimensiones no es la excepción.

Para que esta reconstrucción sea eficiente, es necesario que el mapa de profundidades sea preciso y esto se logra con la calibración correcta de las capturas para que al aplicar la

correlación de imágenes se encuentre la disparidad adecuada sin que sea afectado por sombras o puntos defectuosos en las imágenes.

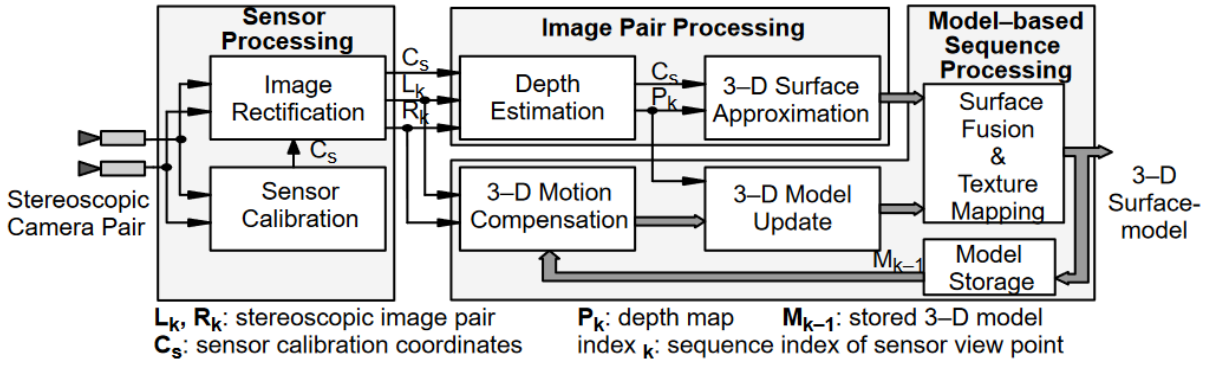
Parte de lo mencionado en anteriores títulos, la toma de las imágenes de izquierda y derecha para su respectiva diferencia y crear el mapa de profundidades es necesaria la captura de las imágenes rectificadas y mediante sacar la disparidad.

Con el mapa de profundidades y proyectando los puntos correlacionados en la imagen es posible crear una nube de puntos basados en las coordenadas correlacionadas y las profundidades calculadas para así poder aproximar una superficie colocando los puntos en el ambiente de tres dimensiones.

El profesor alemán de ciencias computacionales Reinhard Koch, en su publicación de 1995, expresa que para capturar y representar una superficie mediante la técnica de estereografía es necesario seguir un procedimiento estructurado que consta de tres grandes bloques, censar de la imagen, el procesamiento del par de imágenes y el modelado a partir de la secuencia de postprocesado de la imagen en la que se mezclan las aproximaciones obtenidas en el cálculo de profundidades y la estimación de la superficie con la textura mapeada en las primeras etapas del proceso, que se puede apreciar en la figura 10 (Koch, 1995).

Figura 10

Diagrama de proceso de reconstrucción de superficie 3-D



Nota. Tomado de la publicación 3–D Surface Reconstruction from Stereoscopic Image Sequences. (Koch, 1995).

La ventaja del uso de este algoritmo para la reconstrucción de superficies es la precisión que se puede obtener y el bajo costo que se tiene al utilizarla, ya que no precisa de elementos externos para su aplicación más que de la cámara y el sistema de procesamiento.

2.2.22 Calibración de la cámara

"La calibración de cámara geométrica, también conocida como resección de cámara, estima los parámetros de una lente y un sensor de imagen de una cámara de imagen o video. Puede usar estos parámetros para corregir la distorsión de la lente, medir el tamaño de un objeto en unidades mundiales o determinar la ubicación de la cámara en la escena. Estas tareas se utilizan en aplicaciones como la visión artificial para detectar y medir objetos. También se utilizan en robótica, para sistemas de navegación y reconstrucción de escenas en 3D" (The MathWorks, 2022).

2.2.23 Comunicación OPC

La compañía Atvise en su documentación indica lo siguiente: "La Arquitectura Unificada de Comunicaciones de Plataforma Abierta (OPC UA) es un protocolo de comunicación entre maquinas que permite intercambiar información y datos entre una máquina y otra de una manera simplificada" (Atvise, s.f.).

Figura 11

Diferencias entre OPC UA y OPC convencional

	OPC UA	OPC original
Sistema operativo	Cualquier sistema operativo, ya sea Android, Linux o Windows	Tecnología Windows (COM / DCOM)
Seguridad	Encriptación de las comunicaciones mediante certificados de seguridad	Comunicaciones sin encriptar
Tecnología	Puede residir dentro del PLC y formando parte del software de este	Debe instalarse sobre una máquina
Modelo de información	Combina todas ellas en una sola tecnología	Diferentes tecnologías para lidiar con cada tipo de fuente de datos y no todas pueden comunicarse entre si
Firewall	Muy sencillo de configurar	Muy complejo de configurar

Nota. Tomado de sitio web Atvise. (Atvise, s.f.).

2.3 Hipótesis

Los sistemas de visualización automáticos en Costa Rica o México, según los requerimientos establecidos anteriormente, no están preparados para ser implementados por lo menos internamente, utilizando recursos de la planta, como podría ser el caso de realizar dicha ejecución con los ingenieros o también, operadores de la planta, ya que necesitarían un conocimiento bastante complejo en este ámbito y, además, en el de la programación. Esto podría ser factible si se realizara algún tipo de capacitación específica, pero las empresas no lo ven rentable por diferentes razones.

Cuando se hace referencia a este tipo de nuevas tecnologías globales, se está hablando de programación en lenguaje Python, conocimientos: en sistemas de visualización, en redes, en PLC como también en sus protocolos como el caso del protocolo de comunicación OPC-UA, conocimiento en sistemas SCADA, como también en HMI o en bases de datos como en realización de reportes, entre otros temas complejos.

2.4 Limitaciones

Este proyecto posee una serie de limitaciones de todo tipo, debido a que, al utilizar diferentes programas, componentes y tener factores correspondientes a la fábrica totalmente inesperados, como también puede afectar el tipo de material con el que se vaya a trabajar en ese momento. Esto hace de que este proyecto tenga una gran cantidad de variables que afectan este sistema, algunas son totalmente solucionables y otras limitaciones no tienen ninguna solución y debe adaptarse a ellas.

En una primera instancia, se habla de los factores presentes en la planta como podría ser las vibraciones de los equipos que se encuentran en la misma sala, esto puede llegar a afectar al equipo de visión que se encuentra unido al tanque, porque el mismo tanque va a tener una vibración que afectaría a la cámara y, por ende, al cálculo que está realizando. La emisión de luces fuera de las calculadas para el silo o, mejor dicho, la iluminación que se tendrá dentro del silo afectaría completamente el análisis que estaría realizando este sistema automático de visión. El material por evaluar, si es granular del tipo arroz o frijoles, crea espacios entre las interacciones de granos, lo que produce volúmenes que no pueden ser calculados porque falla la densidad. En estos casos puede haber menos masa en un mismo volumen por el acomodo de los granos.

Además, otro factor importante que se debe tener en cuenta es el color interno del silo, porque este puede afectar significativamente, debido a que, si el color del silo es blanco y en ese momento están produciendo un material del mismo color, puedo afectar directamente a este sistema de visión, provocando muchos problemas de lectura de la materia.

En una segunda instancia se tendría los componentes a utilizar como las especificaciones y características del ordenador o Single Board Computer, en este caso se utiliza la Jetson Nano de NVIDIA, puesto que es un ordenador de inteligencia artificial, pequeño y potente que le brinda el rendimiento informático para ejecutar cargas de trabajo de inteligencia artificial modernas a un tamaño, potencia y costo sin precedentes justo lo que se

necesita para implementar este proyecto. Además, la cámara que se utiliza tiene una limitación con el alcance de 35 m afectando ciertas implementaciones.

Finalmente, algunas limitaciones del proyecto a modo de maqueta, ya que el tamaño del silo pudiese ser en algunos casos muy pequeño o grande y afectaría directamente al sistema de medición, puesto que se debería de realizar una calibración o configuración diferente una de otra.

2.5 Alcances

El alcance de este proyecto será implementar un sistema automático de visión de imágenes, utilizando diferentes herramientas. Lo anteriormente comentado es posible utilizando teoría básica de visión de computadora y temas básicos de reconstrucción de superficies en 3D. Además, de conocimientos avanzados en el lenguaje de programación Python, conocimiento en redes de comunicación, configuración de PLC y sus protocolos de comunicación y finalmente, conocimiento en bases de datos para crear ciertos reportes y análisis ciertos parámetros de la producción.

Si estas industrias analizadas de manufactura requieren adoptar este tipo de tecnologías de automatización en sus operaciones, con el fin de poder mejorar en diferentes aspectos de la fábrica con el fin de saber si tienen el conocimiento correcto para la toma de decisiones que vendrían a mejorar cada vez más y más el funcionamiento de la planta y sus operaciones.

Así mismo, la última etapa del proyecto se direcciona al desarrollo de una maqueta que vendría a simular el sistema estudiado, pero en un microámbito controlado para poder proyectar a una planta macro, afectando la menor cantidad de variables.

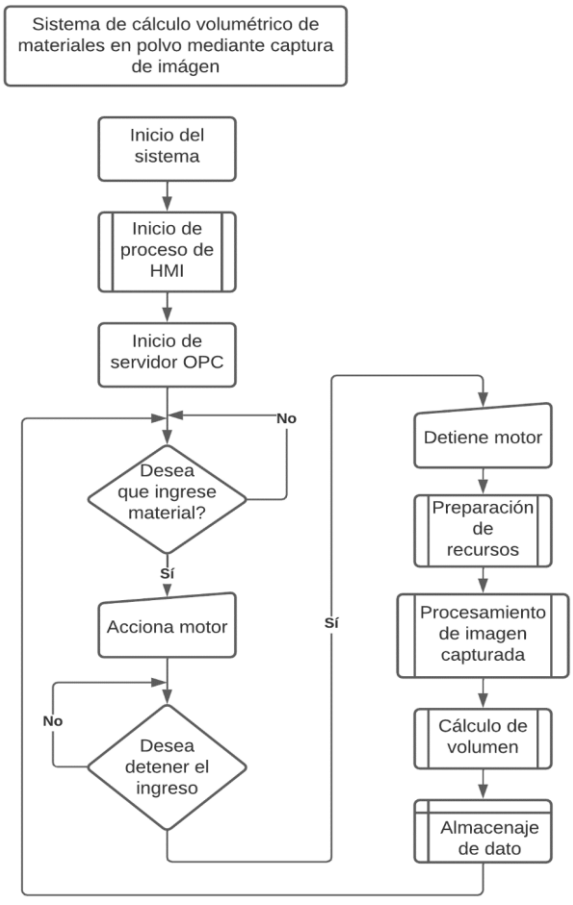
CAPÍTULO 3

3 Desarrollo

Se diseñó el diagrama presentado en la figura 12 para el desarrollo de esta investigación, con el fin de describir el algoritmo del sistema de cálculo de medición volumétrico de materiales en polvo mediante captura de imagen.

Figura 12

Diagrama del flujo del sistema



Con base en lo expuesto en capítulos anteriores, buscar la forma de solucionar el problema de la falta de precisión en la toma de medidas obtenidas por los operarios en los procesos de fabricación de productos alimenticios y materias primas que se encuentran en polvo, es la motivación para realizar este proyecto.

La primera decisión tomada para crear este proyecto fue la de utilizar un método que fuera accesible para las empresas, es decir, un sistema que tuviera costos bajos y una implementación sencilla, por lo que el uso de sistemas de código abierto, de acceso gratuito iba a rebajar esos costos, pero necesitarían un poco más de investigación, además de una optimización específica para el proyecto.

Librerías como OpenCV, DepthAI o Open3D, entre otras, así como el uso de programas con licencias gratuitas como Ignition para el desarrollo del HMI o MySQL para el almacenaje de los datos son la base de este trabajo, por lo que puede ser fácilmente adaptado, modificado e implementado en múltiples soluciones de la industria, cambiando un poco la programación y ajustándolo al entorno de la línea de producción, es aplicable a prácticamente cualquier situación en la que la medición por medio de cámaras sea requerida sin la necesidad de invertir altas sumas de dinero, trayendo beneficio a la industria en general.

Se debe aterrizar la idea del alcance de este proyecto específico, por lo que solamente se hará énfasis en materiales en polvo o sólidos granulares, como lo serían el azúcar, la pectina o el arroz, este último siendo utilizado en las pruebas, ya que es el que se puede considerar como de fácil alcance y más limpio de trabajar, aunque el uso de azúcar es más preciso, pero el almacenaje del azúcar provoca aglomeración de insectos en los lugares de pruebas, además, que, debido a la humedad, se endurecía y formaba cristales que afectaban las mediciones.

Como se describe en la figura 12, la forma en la que se pensó el proceso inicialmente, a grandes rasgos, es mediante el uso de varias etapas en la que se pueden apreciar las de los procesos del HMI, la etapa del servicio OPC para comunicación, accionamiento del sistema de captura de imagen, procesamiento de la imagen, cálculo del volumen aproximado y, por último, el almacenamiento y tratado de los datos. A lo largo de los próximos temas se evalúa a profundidad cada una de estas partes.

Como inicio del proyecto se seleccionó el uso de un sistema HMI, ya que se debe poder observar en tiempo real, cuál es el valor que se está midiendo, además de poder interactuar con los equipos involucrados, por lo que, la mejor forma de realizar esta interfaz es mediante la programación y diseño de una pantalla con la que los operarios van a poder accionar manualmente el ingreso de material y detener su tránsito a voluntad. Por otra parte, otra de las ventajas de esta decisión de diseño es que puede ser fácilmente escalable y modificable, además de presentar de manera amigable un proceso que en esencia no es tan sencillo de comprender si no se tiene los conocimientos adecuados.

Debido a que se debe integrar un sistema de comunicación con los operarios y un sistema de cálculos y procesamientos, es necesario un protocolo de comunicación industrial para el intercambio de datos entre las dos grandes partes del proyecto, el HMI y la parte de procesamiento de la imagen y se decidió por utilizar el protocolo de OPC, para aprovechar la capacidad de envío de información, la facilidad de integración y la sencillez de uso.

Luego de pensar en la orientación de los sistemas que se desea utilizar, viene el punto de la forma en la que se conseguirá el objetivo principal, por lo que se barajaron muchas ideas, desde el uso de escaneo de superficies mediante el uso de láser y medición ultrasónica, entre otras, siempre centrados en que el principal problema por atacar es la superficie del material, tal y como se describe en el documento Flujo y Atascos de un Medio Granular en la Descarga de Silos, el patrón de caída de los polvos y medios granulares es altamente irregular y no es fácil de predecir, puesto que en ocasiones tienden a tener comportamientos de cualquiera de los estados de la materia, pero con la particularidad de que toman la forma del recipiente que los almacena, por esta razón, se puede aprovechar este comportamiento y adelantarse a su aproximación volumétrica, pero esto se explicará más adelante (Janda-Galán, 2008, pág. 2)

Al tener en cuenta que la superficie es irregular y no es fácilmente predecible, se indica que se debe calcular el valor del volumen de la superficie en el momento de hacer la medición,

tomando en consideración el punto más bajo de la superficie y el más alto para esto y así formando un plano imaginario que sería similar al de la siguiente figura 13:

Figura 13

Representación de un material dividido por planos imaginaros

Este plano dividirá el contenido en dos partes, una con un plano imaginario que formará una figura geométrica con volumen fácilmente calculable mediante métodos simples y otra con la superficie irregular del material a la que se le aplicará el algoritmo de aproximación.

Parte importante de la explicación de este algoritmo es que luego de obtener el valor del volumen, se puede trabajar en metros cúbicos convertir a cantidad de masa mediante un cálculo simple, utilizando densidades aproximadas o exactas de los materiales y los volúmenes que se obtienen a la salida del sistema.

Para la detección de todo el sistema, se decide utilizar una cámara para “ver” el material, por lo que gran parte de este trabajo estará enfocada en explicar el método utilizado para capturar esa imagen, cómo es su procesamiento y cuál es el resultado que se espera de esa captura.

Luego de obtener esta captura, procesarla y calcular el volumen, se procederá a enviar esa información a una base de datos donde se realizará un almacenamiento para lograr

generar reportes *a posteriori* y con esto, tener un mejor control del histórico de la planta en la línea específica de producción.

3.1 Descripción de las partes del sistema

Este sistema, como ya se ha descrito, al pasar por varias etapas está formado por múltiples dispositivos que se encargarán de resolver cada una del proceso desarrollado, estos dispositivos son físicos y de software:

1- Cámara para capturar la imagen.

2- Máquina que procesará la imagen capturada y llevará el paso de cálculos.

3- Iluminación para el entorno

4- Interfaz máquina-humano.

3.1.1 Elección de cámara.

Un componente esencial para el proyecto es la cámara, ya que será la responsable de la calidad de la información que se capture y procese posteriormente, por lo que se barajó entre las opciones las cámaras de bajo costo y módulos simples como los descritos en la figura 14, estas cámaras cuentan con características básicas y ocho megapixeles y una resolución UHD, características que se adaptaban a los alcances del proyecto, pero al avanzar en la investigación se notó que era necesario el uso de la visón en estéreo para poder obtener las profundidades, por lo que una única cámara no era suficiente, se necesitaba al menos dos con características específicas, además de una tercera para las tomas centrales del objetivo.

Figura 14

Cámara compatible con Raspberry y Jetson Cámara IMX219-77 de 8 MP para NVIDIA Jetson Nano Developer Kit 3280 × 2464 Resolución con 8 megapíxeles IMX219 Sensor

Nota. Tomado de sitio web Amazon. (Amazon LLC, 2022).

Con la aparición de este inconveniente, surgió la tarea de investigar las posibilidades existentes en el mercado y en las herramientas que se encontraban para la calibración, ajuste e implementación de las cámaras en este entorno, encontrando que colocar dos cámaras individuales de este tipo propiciaba un mayor problema de ajuste, imprecisión en las mediciones y, además, un problema a la hora de instalarlas en los silos de ser el caso de implementar el proyecto en la industria.

Figura 15

Ejemplos de cámaras de profundidad a) Dual IMX219 b) Kinect de Microsoft c) Intel Realsense D435

La solución que se obtuvo fue la selección de una cámara con visión estéreo ya integrada, es decir, que cuente con la calibración de distancia fija entre focos y como mínimo, con los dos lentes ya integrados, además de la electrónica necesaria para poder capturar la imagen de ambas cámaras y enviarla por un solo puerto. De este tipo de cámaras se encuentran muchas clases en el mercado, varias de ellas se muestran en la figura 16, pero con librerías poco desarrolladas o costos elevados, por lo que se decantó por la que ofrece el grupo de OpenCV, la cámara Luxonis OAK-D.

Figura 16

Luxonis OAK-D

Nota. Tomado de sitio web Luxonis. (Luxonis, 2022).

Entre las múltiples ventajas con las que cuenta esta cámara, una de las más importantes es que posee una unidad de procesamiento de video Movidius Myriad X (Movidius, an Intel Company., 2017) de Intel que ayudará a alivianar el procesamiento del equipo elegido para correr el programa principal. Incluyendo una capacidad de 4Tflops de poder de procesamiento, capacidades de detectar objetos en dos y tres dimensiones, procesar imágenes de profundidad y postprocesado de imágenes para recortar, unir y desligar imágenes.

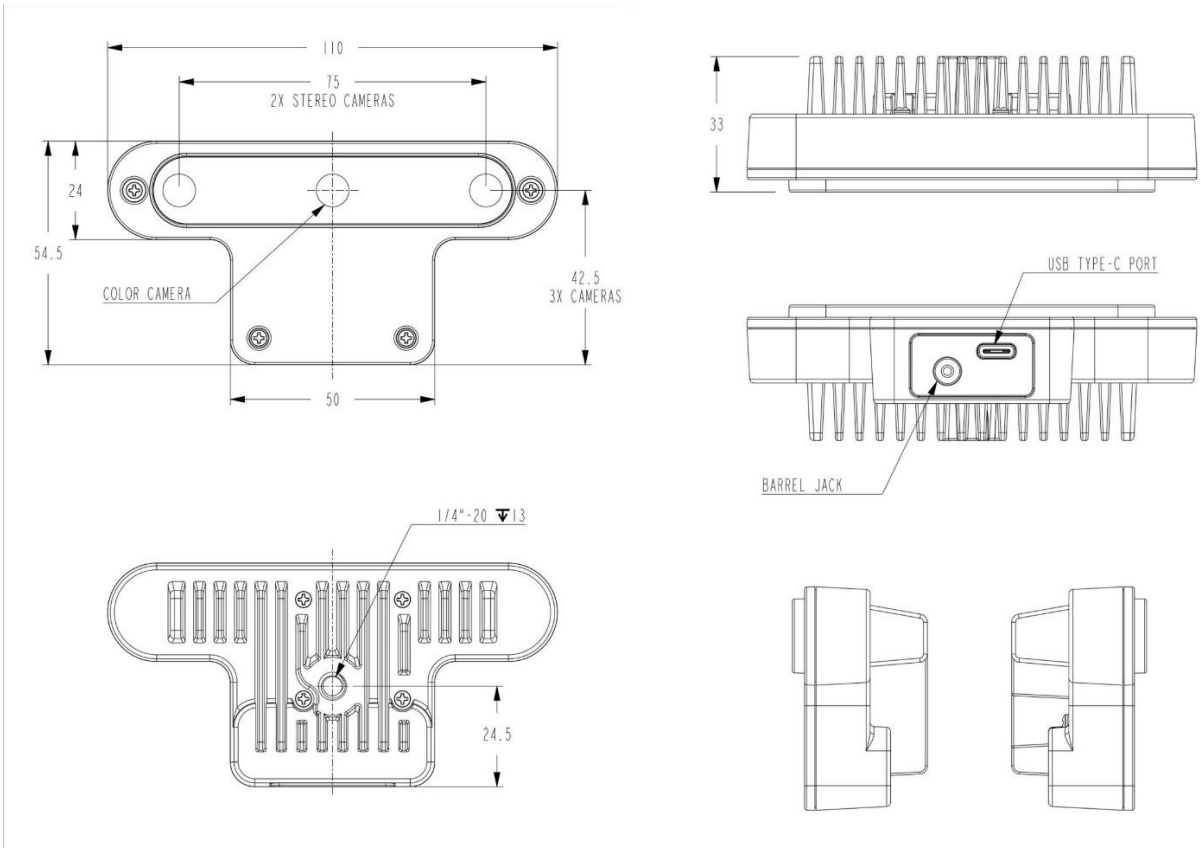
Esta cámara cuenta con dos lentes de tipo OV9282 para profundidad que consiste en cámaras que captan la información en escala de grises de 1 megapixel cada uno, obteniendo una resolución máxima de 1280x800 a 120FPS y capacidad de adaptar el foco desde una distancia de 7.5 cm hasta los 35 m (detalle que brinda versatilidad a la hora de implementar en ambientes industriales diversos)

Además de los dos lentes para el procesamiento de visón profunda, esta cámara cuenta también con un lente RGB Sony IMX378 con la capacidad de captar una resolución máxima de 4K a 60FPS que brinda, además de la información de color de la imagen, información de su textura para detectar superficies con mayor precisión.

Otra ventaja que se nota es el tamaño, descrito en la figura 17 y la construcción de la cámara, que permite tenerla en lugares donde las condiciones son agresivas y no tendrá mucho deterioro, además, de ser necesario es posible el cambio del *case* que la contiene sin incurrir en gastos mucho mayores de mantenimiento.

Figura 17

Detalle de tamaño y construcción de OAK-D



Nota. Tomado de sitio web Luxonis. (Luxonis, 2022).

El costo de esta cámara es de $199, por lo que, con base en sus características, la hacen óptima para el desarrollo del proyecto. La razón principal por la que se decide el uso de la OAK-D es por la integración a la librería Depthai que aunada al VPU Myriad X reduce el tiempo de diseño del código y facilita la creación del algoritmo, ya que los procesos son solamente métodos en la librería y no se deben desarrollar desde cero.

3.1.2 Selección de dispositivo de computación.

Se barajaron múltiples ideas para el equipo que se encargará de ser el centro de mando del sistema, tomando en cuenta los requisitos de la aplicación a crear, por ejemplo, potencia de procesamiento, capacidad de conexiones, conectividad a redes y adaptabilidad, entre otras.

Existe gran cantidad de dispositivos capaces de correr el sistema diseñado, al ser basado en un lenguaje popular y pasar la mayor cantidad de la carga de procesamiento a la cámara, es posible adaptar este sistema a múltiples dispositivos, de los cuales, los tres principales que se conoce son: el Raspberry Pi, la placa de Nvidia Jetson Nano y la placa de Intel NUC. Cada una cuenta con sus ventajas, desventajas y características que las hacen ideales para ciertos trabajos, por lo que ayudados por el cuadro comparativo y las comparaciones que facilitó el sitio web www.contec.com,se toma la decisión acerca de cuál placa utilizar.

Tabla 2

Características de la Nvidia Jetson Nano

Features	**Raspberry Pi 4 Model B**	**Jetson Nano™ Developer Kit**	**Intel® NUC 11 Pro Board NUC11TNBv7**
Procesador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	CPU: Quad-core ARM A57 @ 1.43 GHz GPU: 128-core Maxwell	Intel® NUC Board with 11th Generation Intel® Core™ Processors
Memoria	2GB, 4GB or 8GB LPDDR4-3200 SDRA microSD (not included)	4 GB 64-bit LPDDR4 25.6 GB/s microSD (not included)	64 GB, DDR4-3200 1.2V SO-DIMMs
Tamaño	85mm x 56mm (3.3” x 2.2”)	69mm x 45mm (2.7” x 1.77”)	101mm x 101mm UCFF (4" x 4")
I/O	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet 2	2x MIPI CSI-2 DPHY lanes Gigabit	PCI Express, 4 USB Ports, 1 SATA Port, Integrated LAN, 1x

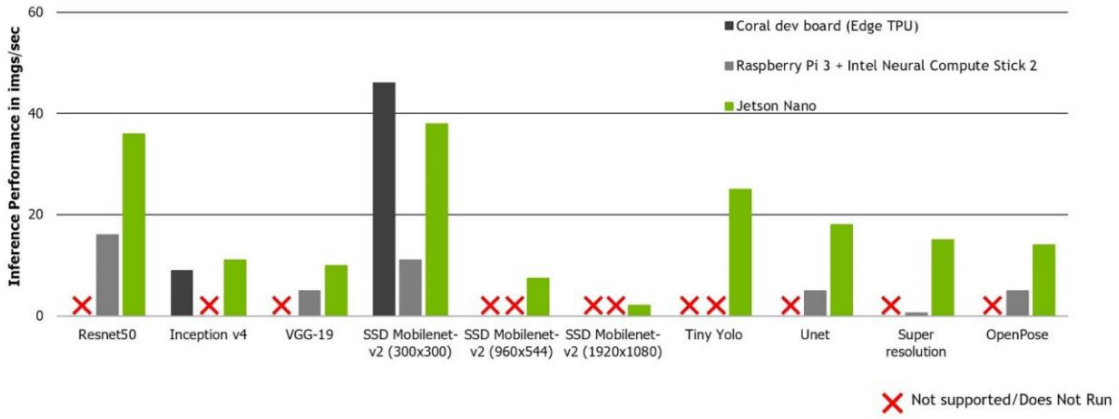
Features	Raspberry Pi 4 Model B	Jetson Nano™ Developer Kit	Intel® NUC 11 Pro Board NUC11TNBv7
	USB 3.0 ports; 2 USB 2.0 ports. 2 × micro-HDMI ports (compatible with Dual displays) Micro-SD card slot for loading operating system and data storage	Ethernet, M.2 Key E HDMI and display port 4x USB 3.0, USB 2.0 Micro-B	Thunderbolt™ 4, 1x Thunderbolt™ 3 Compatible with dual and triple displays.
Sistema Operativo	Raspberry Pi OS (previously called Raspbian) is the recommended operating system	Linux NVIDIA JetPack SDK	Windows 10, 64-bit*, Windows 10 IoT Enterprise*, Red Hat Linux*, Ubuntu 20.04
Consumo al 100% de uso	6 watts	10 watts	49 watts
Precio	Desde $125	Desde $298	Desde $649

Nota. Tomado del sitio web Contec. (Contec, 2021).

Un *benchmarking* realizado por el foro hackaday.com en 2019, colocó la placa Nvidia Jetson Nano como la mejor opción entre sus homólogas de otras marcas (Coral de Google y Raspberry Pi 3).

Figura 18

Benchmarking de capacidades de procesamiento

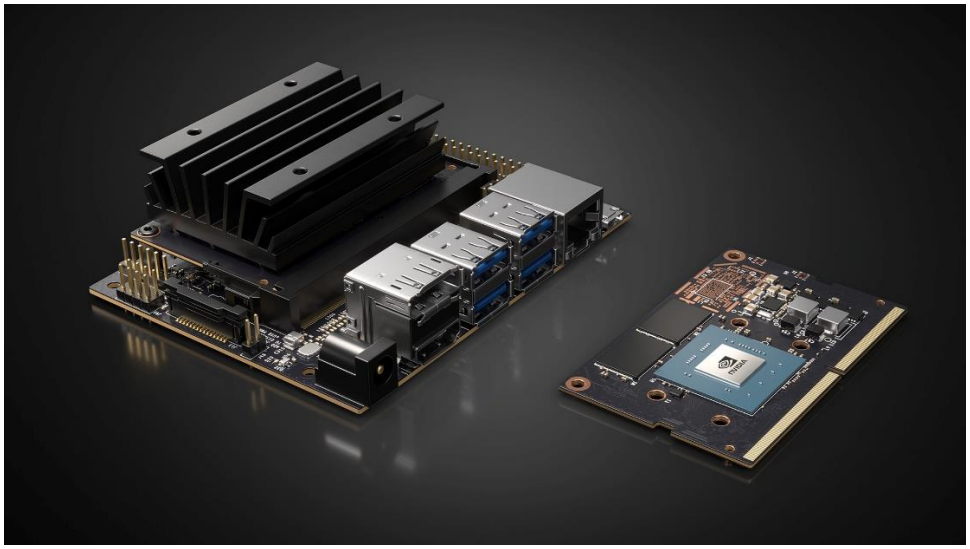


Nota. Tomado del sitio web Hackaday. (Hackaday, 2019).

Gracias a la experiencia adquirida en cursos como sistemas embebidos y sistemas en tiempo real y a la investigación hecha, se decide que la mejor placa para este sistema es la Nvidia Jetson Nano, debido que cuenta con una cantidad de memoria suficiente, poder de procesamiento adecuado y puertos para conectar la cámara vía USB3.0 para obtener su máxima capacidad sin sobredimensionar el dispositivo por un precio accesible. Para que este componente se mantuviera seguro, se realiza la impresión de un *case* en una impresora 3D para asegurar la integridad del equipo.

Figura 19

Nvidia Jetson Nano



Nota. Tomado del sitio web Hackaday. (Hackaday, 2019).

3.1.3 Discriminación de método de iluminación.

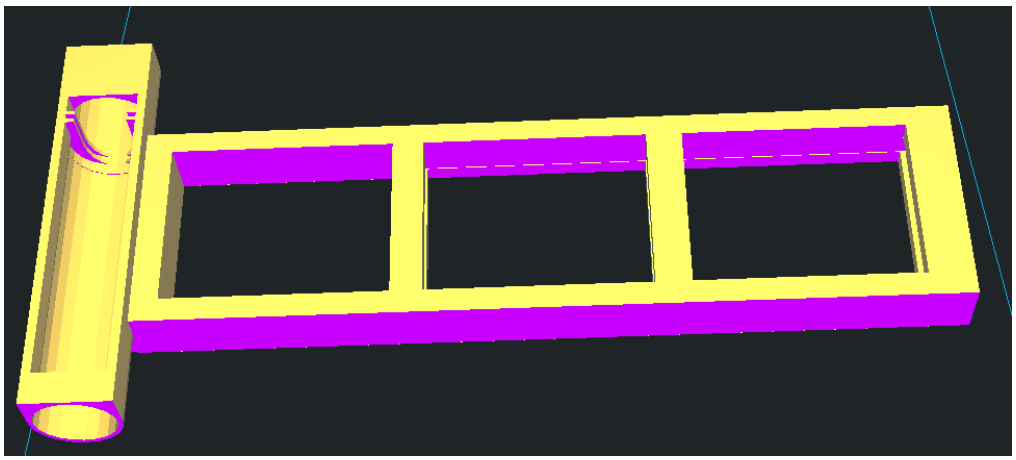
El método de iluminación que se elija va a depender del ambiente donde se implemente el proyecto y como se vio en capítulos anteriores, es de suma importancia tomar en cuenta las características del sitio por iluminar y la superficie que reflejará la luz, para así no sobredimensionar la cantidad de luz o los ángulos de incidencia.

Con base en las características de la habitación donde se realizaron las pruebas, se adoptó por realizar una iluminación por medio de luces LED con la técnica de iluminación lateral mezclada con direccional, ya que se modificó el ángulo de incidencia para ser cercano a los 90 grados de la superficie, esto se hizo de esta forma pensando en que, si se intervenía de alguna forma el silo para iluminarlo de manera lateral, habría que invertir en luces laterales para el silo que podrían alterar la medición, elevar los costos de construcción del proyecto, además de los costos de consumo energético general en la planta, sin pensar en los problemas que podría causar en el producto, la exposición a cierta temperatura del led si este llega calentar.

Como se muestra en la figura 20, se diseñó una base para colocar el dispositivo que iba a iluminar el contenedor del material mediante el servicio web para diseño de piezas 3D ThinkerCAD. Esta pieza se imprimió en una impresora 3D para asegurarnos que la luz que iba a incidir en el contenedor iba a tener el ángulo deseado y la cantidad lumínica necesaria para capturar la imagen de manera correcta.

Figura 20

Pieza diseñada para sostener la luz sobre la cámara lista en el sistema Slicer de Creality

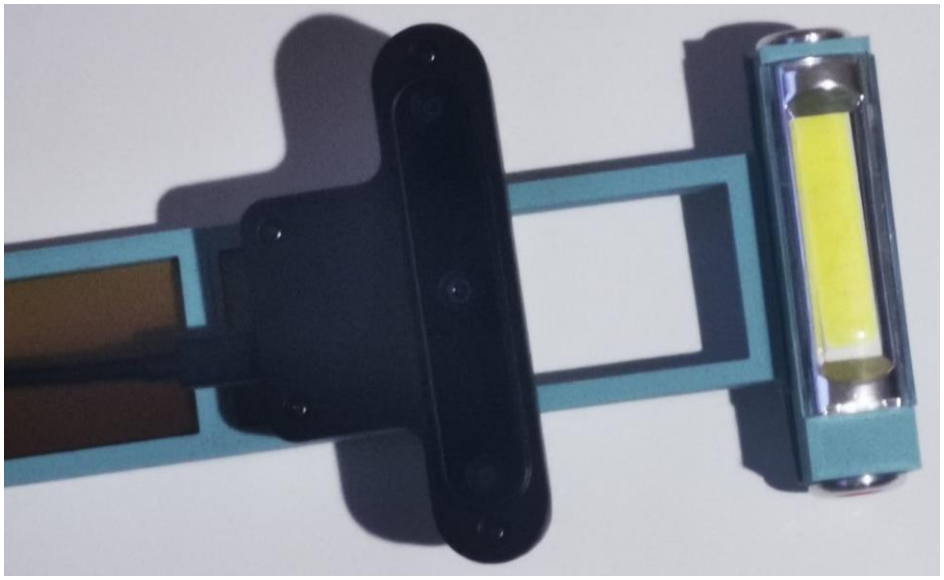


Se decidió no utilizar la tecnología LiDAR, que habría sido ideal para estas aplicaciones debido a su precisión y facilidad de implementación en sistemas de reconstrucción de superficies mediante nubes de puntos, debido a su alto costo, pero se tiene claro que puede ser una clara oportunidad de mejora en el diseño del sistema, por lo que se deja abierto el campo de estudio en esa área en caso de continuar con el proyecto a futuro.

La lámpara seleccionada consta de 16 LED con una capacidad lumínica de 50Lm cada una y una potencia general de 5W, se instaló un circuito para reducir la potencia a la mitad, por lo que se cuenta con dos potencias lumínicas en caso de querer utilizarlo en condiciones de mucha claridad de día o condiciones nocturnas. En la figura 21 se muestra el soporte de la luz terminada con la cámara, listos para comenzar a medir en el ambiente de laboratorio.

Figura 21

Soporte de la iluminación terminado junto con la cámara OAK-D

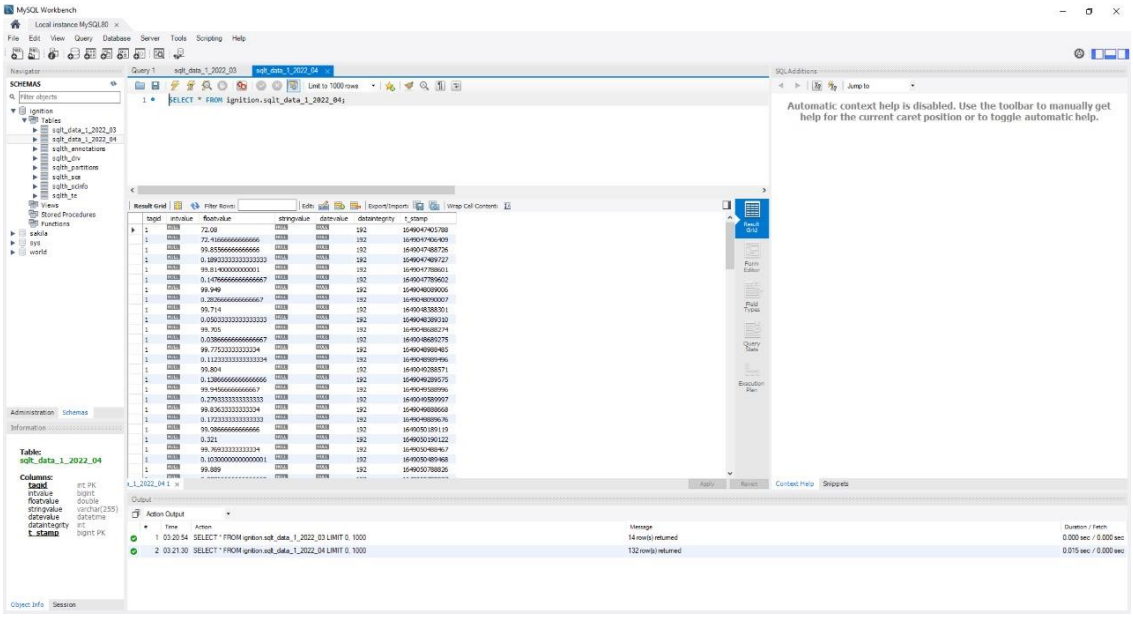


3.1.4 Interfaz HMI.

La selección para la interfaz HMI fue bastante sencilla, ya que, debido a la experiencia laboral de uno de los integrantes del proyecto, el uso de Ignition fue casi inmediatamente la primera opción. El entorno de Ignition es bastante amigable como se lo puede ver en la figura 22.

Figura 22

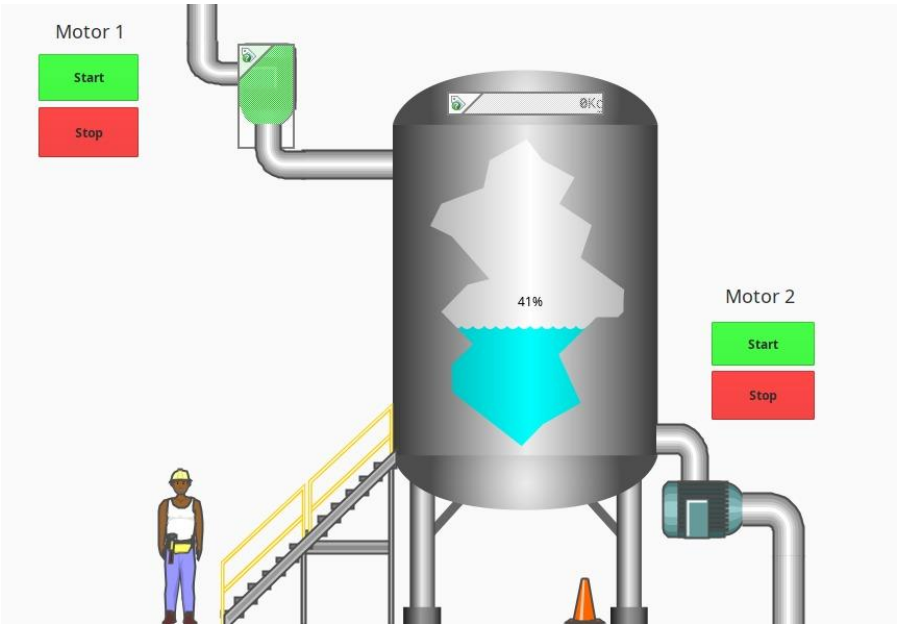
Entorno de trabajo de Ignition



Uno de los primeros pasos fue el desarrollo de la pantalla principal, que debe contener la estimación del volumen del tanque para evitar problemas de rebalse, los botones que controlarán los motores de ingreso de material y expulsión de este. En la figura 23 se puede detallar que la interfaz es simple y amigable con el usuario, por lo que la capacitación para su uso como operario es sencilla.

Figura 23

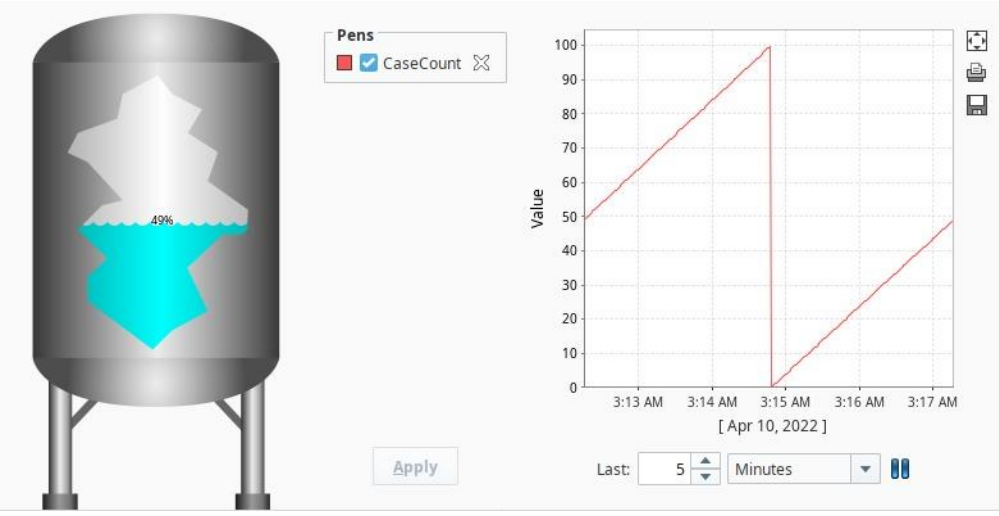
Pantalla principal de HMI de sistema



Al ser una interfaz sencilla, se debe ir aumentando los niveles de complejidad para los distintos funcionarios y que no solo sea una pantalla para operarios, sino que desde la misma se pueda obtener datos históricos de medición, por lo que se diseñó una pantalla donde se muestre de forma gráfica, los cambios de estados del tanque y la cual se detalla en la figura 24.

Figura 24

Pantalla de análisis gráfico de cambios de material en el contenedor.



Y si se aumenta el nivel de detalle que se quiere en los reportes, se entrará a las bases de datos que se crearon para ver con marcas de tiempo, todas las variables que cambiaron y sus valores a lo largo del funcionamiento del sistema.

Figura 25

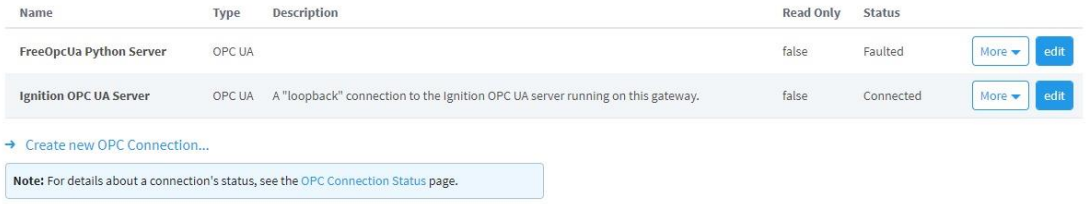
Vista detallada de los datos almacenados con marcas de tiempo

tagid	intvalue	floatvalue	stringvalue	datevalue	dataintegrity	t_stamp
1	NULL	72.08	NULL	NULL	192	1649047405788
1	NULL	72.41666666666666	NULL	NULL	192	1649047406409
1	NULL	99.85566666666666	NULL	NULL	192	1649047488726
1	NULL	0.18933333333333333	NULL	NULL	192	1649047489727
1	NULL	99.81400000000001	NULL	NULL	192	1649047788601
1	NULL	0.14766666666666667	NULL	NULL	192	1649047789602
1	NULL	99.949	NULL	NULL	192	1649048089006
1	NULL	0.2826666666666667	NULL	NULL	192	1649048090007
1	NULL	99.714	NULL	NULL	192	1649048388301
1	NULL	0.05033333333333333	NULL	NULL	192	1649048389310
1	NULL	99.705	NULL	NULL	192	1649048688274
1	NULL	0.03866666666666667	NULL	NULL	192	1649048689275
1	NULL	99.77533333333334	NULL	NULL	192	1649048988485
1	NULL	0.11233333333333334	NULL	NULL	192	1649048989496
1	NULL	99.804	NULL	NULL	192	1649049288571
1	NULL	0.13866666666666666	NULL	NULL	192	1649049289575
1	NULL	99.94566666666667	NULL	NULL	192	1649049588996
1	NULL	0.2793333333333333	NULL	NULL	192	1649049589997
1	NULL	99.83633333333334	NULL	NULL	192	1649049888668
1	NULL	0.1723333333333333	NULL	NULL	192	1649049889676
1	NULL	99.98666666666666	NULL	NULL	192	1649050189119
1	NULL	0.321	NULL	NULL	192	1649050190122
1	NULL	99.76933333333334	NULL	NULL	192	1649050488467
1	NULL	0.10300000000000001	NULL	NULL	192	1649050489468
1	NULL	99.889	NULL	NULL	192	1649050788826

Como se sabe, es posible conectar el sistema Ignition a servidores OPC UA y esta configuración es sumamente sencilla, solamente colocando el identificador del equipo a consultar dentro del parámetro de “nombre” y automáticamente el sistema reconoce el equipo remoto y le solicita constantemente el estado de los *tags* que tiene.

Figura 26

Conexión de sistema a servidor OPC UA remoto



3.2 Selección de Librerías

Se contará con librerías que fueron las herramientas básicas que permitieron la resolución de los algoritmos presentados en este proyecto, al ser un sistema que se desarrolló basado en Python. Entre las librerías utilizadas, las más importantes y orientadas al proyecto son:

- OpenCV
- Open3D
- Depth AI
- Numpy.
- Pycloud.

Se describe rápidamente cada librería para tener una idea básica a la hora de narrar las futuras partes del algoritmo y utilizar las partes del código diseñado como apoyo para la explicación.

OpenCV

Esta librería es un conjunto de funciones diseñadas por Intel para el desarrollo de programas basados en visión de computadoras. Es una biblioteca de software libre que cuenta con múltiples repositorios (la mayoría en GitHub) para el crecimiento colaborativo de los estudios que se realizan en el ámbito de los sistemas de visión artificial. Esta librería cuenta con módulos que facilitan la captura de las imágenes, procesado (aplicación de filtros, transformadas entre otras funciones) y monitorización de estas (OpenCV, 2022).

Open3D

La biblioteca Open3D es un complemento para OpenCV, cuenta con módulos que tratan datos en 3D y para la respuesta rápida del software diseñado, gracias a que está altamente optimizado y configurado para la paralelización. Con esta librería se realizan los cálculos de la nube de puntos y la conversión de esa en mallas entre otras funciones (Open 3D.org, 2022).

Depth AI

Depth AI es la biblioteca que ayudará a realizar las funciones con el Myriad X inmerso en la cámara, por lo que es fundamental la comprensión de su funcionamiento y sus módulos, dado que la construcción del software se basa en la captura del mapa de profundidad realizado mediante esta biblioteca. Explicado directamente de los desarrolladores:

> Es una plataforma de inteligencia artificial espacial, que permite a los robots y computadoras percibir el mundo como un ser humano, qué objetos o características son, y dónde se encuentran en el mundo físico. Se centra en la combinación de estas 5 características clave:
>
> - Inteligencia artificial.
> - Visión por computador.
> - Percepción de profundidad (Estéreo, ToF).
> - Rendimiento (alta resolución y FPS, múltiples sensores).
> - Solución integrada de bajo consumo.
>
> La plataforma DepthAI se basa en Movidius VPU: es un ecosistema completo de hardware, firmware y API de software personalizados. Lo mejor de todo es que es modular y puede integrar esta tecnología en sus productos. (Depth, 2020)

Este sistema basa su funcionamiento en “Nodos” y cada módulo cuenta con sus configuraciones para uso, entradas y salidas y la gracia de esta plataforma es crear la red adecuada para obtener las imágenes que sea posible procesar, se podría comparar su funcionamiento a un lego que debe ser armado en un orden específico para que tenga un sentido el código, ya que no se podría aplicar un filtro a una imagen si no se ha colocado el “bloque” de captura o no se ha configurado los parámetros del filtro que se desea aplicar. Conforme se avance en la explicación se describirán los módulos utilizados.

Numpy/Pycloud/Math

Estas librerías son básicamente las utilizadas para cálculos de matrices, algoritmos numéricos y manejo de datos dentro del programa, como se sabe, Python es un lenguaje de tipado dinámico, por lo que si se trata de aplicar operaciones aritméticas a dos variables de tipos distintos, se obtendrán datos que no concordarán con las necesidades de otros módulos, entonces aquí es donde entra la librería Numpy a normar las matrices y darles tamaños, codificaciones y estructuras estándares a los resultados de cálculos realizados.

3.3 Comunicación entre los sistemas

Se ha observado a lo largo de este capítulo que el sistema está formado por distintas partes y entre ellas están el sistema de captura de imágenes (que lo compone la placa Nvidia Jetson y la cámara) y el sistema de interpretación de datos/HMI (conformado por el servidor que contendrá el HMI y la base de datos de MySQL) y es necesario el intercambio de información entre el ambas partes, tomando en cuenta que se encontrarán en un ambiente industrial, se debe aplicar un protocolo de comunicación industrial que se adapte a otras partes de la planta, porque podría tener sistemas de monitoreo unificados, por lo que se tiene que diseñar un sistema que se adapte de manera adecuada a cualquier situación.

Para este caso, se volvió a hacer uso de la experiencia adquirida a lo largo de los años en la industria, pensando en utilizar protocolos como Modbus o Profibus, pero el sistema que

mejor se adapta al HMI y con el que se estaba familiarizados era OPCUA. Para aplicar OPCUA en un entorno de Linux, se utiliza la librería llamada FreeOPCUA que cuenta con los mecanismos necesarios para realizar conexiones servidor-cliente o servidor-servidor para el intercambio de información de *tags*, que en este caso se mantendrá en constante evaluación del estado motores de las puertas para saber en qué momento se debe capturar la imagen para calcular el volumen y entregar la masa del material.

3.4 Preparaciones previas al sistema

Antes de comenzar el proceso de medición, es necesario conocer bien el material que se va a ingresar al silo, por lo que el cálculo de una densidad que se nombrará “aparente”, se torna uno de los primeros pasos para la explicación del proceso que se realiza para la puesta en marcha.

Para las pruebas en campo de este sistema, se utilizó arroz 80% grano entero para el análisis de masa, lo que trajo bastantes problemas, ya que no es sencillo conocer la densidad de este producto, por lo que se procedió a realizar un cálculo aproximado de la densidad.

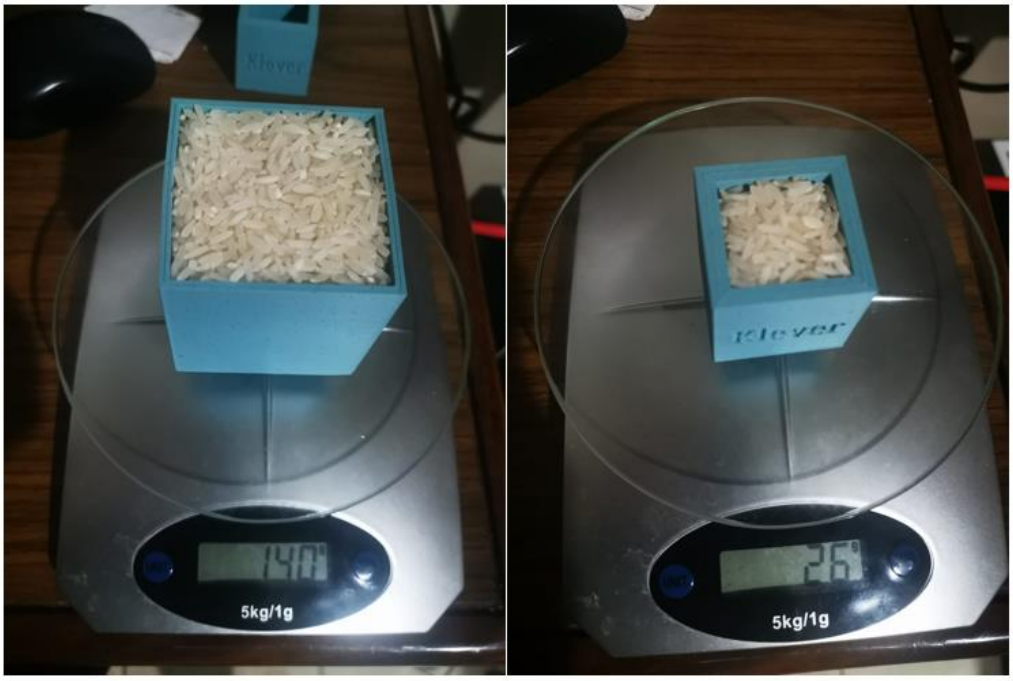
Se sabe que la densidad de un material es la relación de la cantidad que entra en un contenedor, eliminando lo más que se pueda el aire contenido y las impurezas, por lo que se procedió a calcular mediante el uso de cajas y su medición con precisión milimétrica (calibrador Vernier de 0.02 mm de precisión) y se utilizó una balanza para medir la cantidad de material que entró en estos cubos y aplicando una simple fórmula:

$$\rho = \frac{Masa\ (Kg)}{Volumen\ (m^3)} \tag{15}$$

Y con esto se calculó el valor más aproximado de la densidad del arroz del 80% grano entero.

Figura 27

Mediciones de densidades aparentes con cubos calibrados.



Como se observa en la foto, se realizaron medidas en dos cubos de distintos tamaños descritos en la figura 27. De los valores obtenidos se obtuvo la siguiente información:

Tabla 3

Densidades calculadas para el arroz 80% grano entero

Cubo	**Volumen interno** (cm^3)	**Cantidad de arroz medido** (g)	**Densidad calculada.** $(^{g}/_{cm^3})$
1 (4cm de lado)	31.5	26	0.8571
2 (6cm de lado)	166.375	140	0.8415

Con estos datos se calcula el promedio y se obtiene que la densidad aparente es de 0.8493 g/cm^3

Teniendo en cuenta que el material que se está utilizando es un grano, se sabe que la forma y tamaño de cada uno es muy variable y se generan microespacios vacíos entre los granos que provocan variaciones en las densidades calculadas. También se realizaron mediciones con un material más fino, en este caso, se midió con los mismos cubos, un polvo de dulce de tapa y se obtuvieron los siguientes resultados:

Tabla 4

Densidades para el polvo de tapa de dulce procesado

Cubo	**Volumen interno** (cm^3)	**Cantidad de dulce de tapa medido** (g)	**Densidad calculada.** $(^{g}/_{cm^3})$
1 (4cm de lado)	31.5	17	0,53968254
2 (6cm de lado)	166.375	103	0,6190834

Con el caso del segundo material, se ve que la variación es muy amplia y se decidió desestimar las mediciones. Con base en estos datos, se concluye utilizar el arroz con 80% grano entero para las mediciones y tener menor margen de error al buscar los resultados.

3.5 Funcionamiento del sistema

Se utilizarán varias herramientas para explicar el funcionamiento del sistema:

- Diagramas de flujos de procesos principales.
- Partes del código.
- Imágenes que describan los nodos.

Al comenzar, como lo muestra el diagrama de la figura 12, se procede con la inicialización del HMI que se encuentra en el equipo remoto, que cuenta con la configuración anteriormente explicada para la conexión con la placa Nvidia Jetson, por lo que el siguiente paso es el inicio del servicio OPC para poder compartir información desde la Jetson hacia el HMI que funcionará también como servidor (como se sabe, una de las ventajas de OPCUA es

que, es posible la comunicación entre dos servidores, lo que permite modificar ambas variables del otro). Esto se puede observar en la parte inicial del código. Como se aprecia en las líneas de código, el sistema esperará hasta que se dé el cierre de la compuerta de ingreso de material.

Figura 28

Código inicial configuración de servidor OPC

```
#-*-*-*-*-*-*-*-*-* Inicio del programa*-*-*-*-*-*-*-*-*-**-*
# Se comienza corriendo el servidor OPC UA
server = Server()
#Esta dirección cambia depende la conexión en el wifi
server.set_endpoint("opc.tcp://192.168.1.210:12345")
server.register_namespace("Room1")
#El servidor está basado en objetos por lo que se crea un cuarto para almacenar los objetos
objects = server.get_objects_node()
#El primer objeto es el calculador de volumen, se crea el objeto y se le asigna la variable respectiva
Volumencalc = objects.add_object(2,"Calculador de Volumen")
vol = Volumencalc.add_variable(2,"Volumen", 20)
#Creamos un segundo objeto que será nuestro trigger, es decir el activador del software desde el HMI
Trig = objects.add_object(2,"Trigger")
state = Trig.add_variable(2, "Start Mesure", True)
state.set_writable()
vol.set_read_only()
#Ciclo principal donde consulta si es necesario medir o no
try:
    print ("Start server")
    server.start()
    print("Server Online")
    while True:
        check = state.get_value()
        check2 = vol.get_value()
        print("Estado del motor: " +str(state.get_value()))
        if check:
            VolumenCalculado = str (Volumen())
            vol.set_value(VolumenCalculado)
            print ("Volumen Calculado: " +str(vol.get value()))
            sleep(2)
        else:
            print("Motor no activo")
            sleep(2)
finally:
    server.stop()
    print("Server Offline")
```

En el servidor se compartirán las variables del cálculo del volumen y del inicio de programa, esta variable será la que se presionará en el HMI para decirle al mecanismo que la medición debe empezar. Dentro de los parámetros que se encuentran en esa configuración están:

- El nombre con el que se identificará el servidor en la red

- El cuarto o subnivel que contendrá un grupo de objetos de evaluación.
- Los objetos que contendrán las variables que a su vez contendrán los valores que se quiere compartir y sus respectivas propiedades.

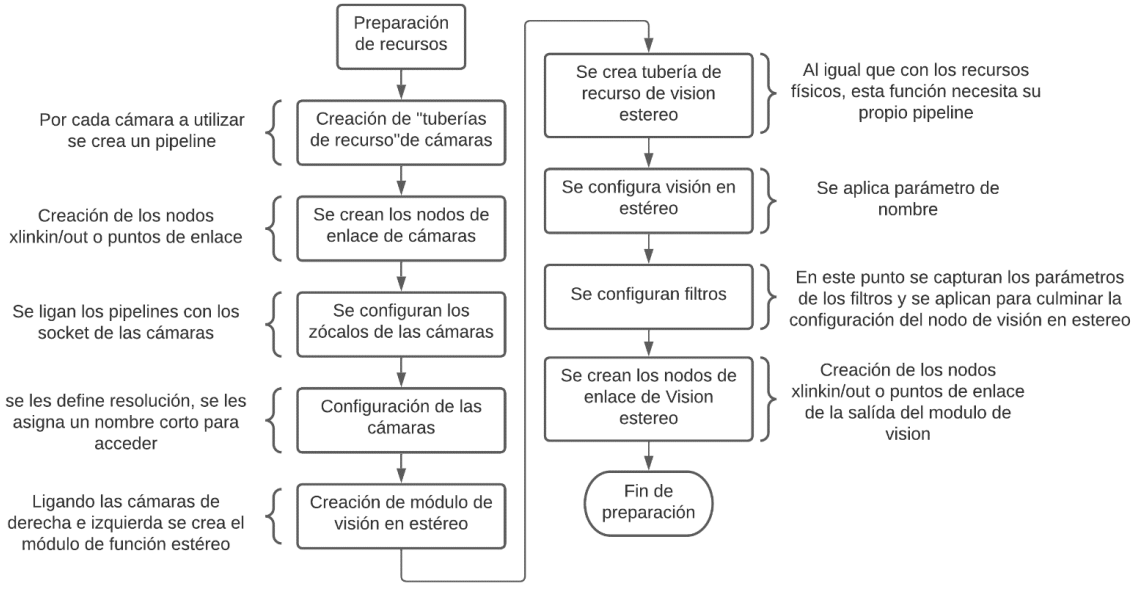
Este proceso se realizó utilizando la librería FreeOPCUA que contiene los métodos necesarios para el adecuado funcionamiento del servicio (FreeOPCUA, 2022).

3.6 Proceso de preparación de recursos

Luego de que mediante del servicio OPC, el sistema activa la función de medición y basado en el método de reconstrucción de superficies en 3D que expone Koch, anteriormente explicado, se debe preparar inmediatamente el recurso de la cámara para la captura de la imagen, esto se realizará con los módulos que se encuentran en Depth AI, recordando que la cámara cuenta con su propio procesador de imágenes, el subproceso se describirá mediante el diagrama de la figura 29:

Figura 29

Diagrama de subproceso de preparación de recursos



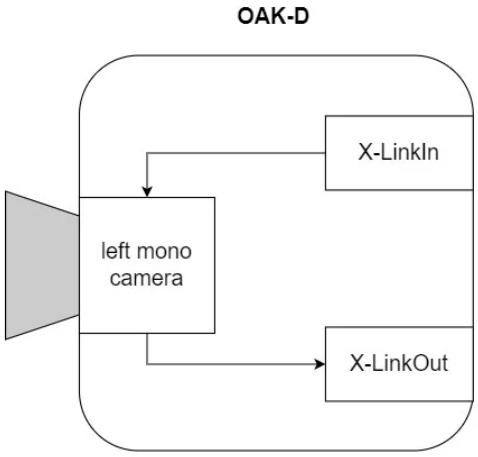
Para entender mejor este proceso es necesario entender la forma en la que trabaja Depth AI, con el que lo primero que se debe hacer es crear un *pipeline* que contendrá la

información de los recursos utilizados; por cada recurso físico a utilizar se creará el *pipeline* que intercambiará la información con la cámara mediante la creación de JSON y se enviarán vía Xlink.

Al crear métodos de enlace de información, se debe crear los nodos que se enlazarán, esto dependerá de la función que se necesita que cumpla la cámara, por ejemplo, para la creación de un nodo que entregue el video de la cámara izquierda seguirá el patrón detallado en la figura 30.

Figura 30

Ejemplo de enlace de nodos en Depth AI en la cámara OAK-D



Nota. Tomado de sitio web LearnOpenCV. (LearnOpenCV.com, 2022).

Como se muestra, se crea el *pipeline*, se le asigna el recurso físico (en este ejemplo, la cámara izquierda) y se enlaza con la salida, para que el procesamiento de las imágenes que se ingresen sea dentro de la cámara. Esto se realiza utilizando los módulos de la librería Depth AI quedando de la siguiente forma demostrada en la figura 31:

Figura 31

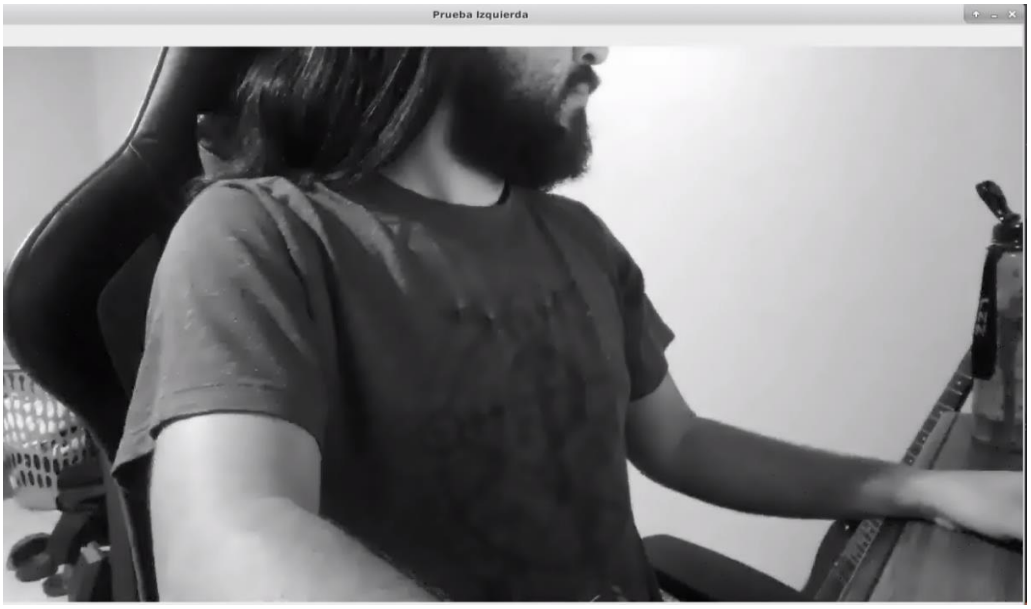
Código básico de uso de recursos y salida de imagen Depth AI

```
import depthai as dai
pipeline = dai.Pipeline()

mono = pipeline.createMonoCamera()
mono.setBoardSocket(dai.CameraBoardSocket.LEFT)

xout = pipeline.createXLinkOut()
xout.setStreamName("left")
mono.out.link(xout.input)

with dai.Device(pipeline) as device:
    queue = device.getOutputQueue(name="left")
    while True:
        frame = queue.get()
        imOut = frame.getCvFrame()
```



Esto se explica, ya que el resto del algoritmo está basado en estos enlaces y estas salidas de imágenes parciales, por lo que se debe dejar claros estos puntos hasta ahora. Como el algoritmo necesita utilizar todas las cámaras, se debe configurar los *pipelines* mediante un método para poder invocarlo en múltiples puntos del sistema.

Figura 32

Método para preparar los recursos

```
def create_pipelines(self):
    #Selección de resolución de cámaras
    rgb_resolution = dai.ColorCameraProperties.SensorResolution.THE_1080_P
    depth_resolution = dai.MonoCameraProperties.SensorResolution.THE_400_P
    pipeline = dai.Pipeline()
    # Se definen los recurso físicos )Nodos de entrada)
    camRgb = pipeline.createColorCamera()
    left = pipeline.createMonoCamera()
    right = pipeline.createMonoCamera()
    stereo = pipeline.createStereoDepth()
    # Se crean los nodos de salida
    depthOut = pipeline.createXLinkOut()
    depthOut.setStreamName("depth")
    rgbOut = pipeline.createXLinkOut()
    rgbOut.setStreamName("rgb")
    # Configuración de cámaras
    camRgb.setResolution(rgb_resolution)
    camRgb.setInterleaved(False)
    camRgb.setBoardSocket(dai.CameraBoardSocket.RGB)
    mono_camera_resolution = depth_resolution
    left.setResolution(mono_camera_resolution)
    left.setBoardSocket(dai.CameraBoardSocket.LEFT)
    right.setResolution(mono_camera_resolution)
    right.setBoardSocket(dai.CameraBoardSocket.RIGHT)
    #Configuración de filtros de postproceso
    self.configureDepthPostProcessing(stereo)
    stereo.setDepthAlign(dai.CameraBoardSocket.RGB)
    # Enlazamos los nodos de salida con los de entrada
    left.out.link(stereo.left)
    right.out.link(stereo.right)
    stereo.depth.link(depthOut.input)
    camRgb.video.link(rgbOut.input)
    # Book-keeping
    streams = [depthOut.getStreamName(), rgbOut.getStreamName()]
    rgb_image_size = (camRgb.getResolutionWidth(), camRgb.getResolutionHeight())
    depth_image_size = (right.getResolutionWidth(), right.getResolutionHeight())

    return pipeline, streams, rgb_image_size, depth_image_size
```

Con este método se cumplen los requerimientos necesarios para poder utilizar los tres sensores de la OAK-D, pero aparte de esta configuración, es necesario adelantar la creación de los nodos de postprocesado

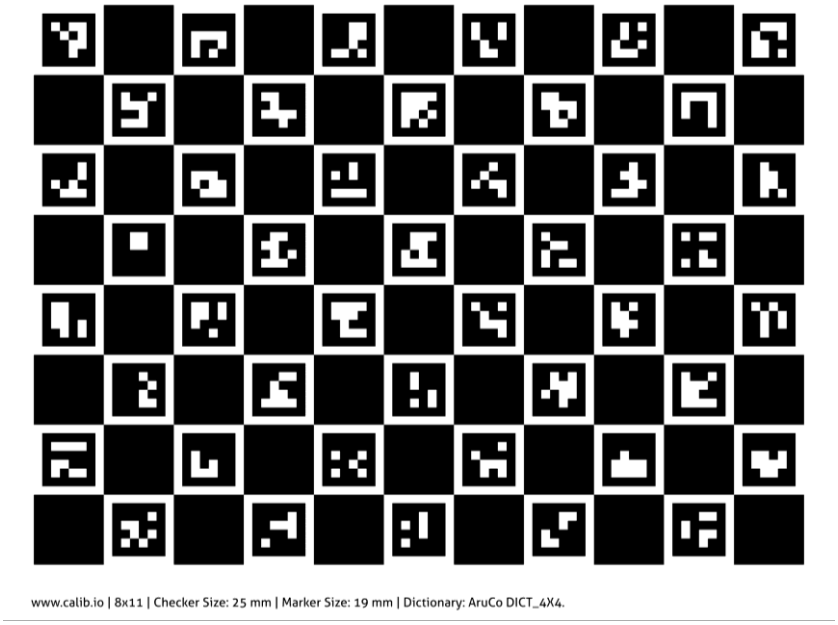
Dentro de este método se invoca también el nodo de visión en estéreo, este método es un modelo de postprocesado de imágenes que, utilizando la teoría explicada en capítulos anteriores, entrega una matriz que contendrá el mapa de profundidad necesario para los siguientes pasos.

Para comprender el nodo de visión en estéreo, se vuelve a la teoría de Depth AI, en la que se explica que el nodo StereoDepth se encarga de realizar los cálculos necesarios para que, teniendo la imagen de los dos sensores situados en los bordes, se pueda obtener un mapa de profundidades mediante el cálculo de disparidad entre la imagen rectificada de la cámara derecha y la izquierda, utilizando el método explicando anteriormente de correlación de puntos.

Antes de entrar al tema del módulo de visión en estéreo, se debe hablar sobre la calibración de los sensores de la OAK-D, que se realizará para obtener el margen de error que tienen los lentes debido a que, en el proceso de construcción de la cámara, es posible que se muevan ligeramente piezas o en el transporte, un golpe produzca una variación en la distancia que separa las cámaras o en los sensores como tal, dejando cada cámara con un porcentaje de error único.

Figura 33

Patrón de ajuste para OAK-D brindado por Depth AI

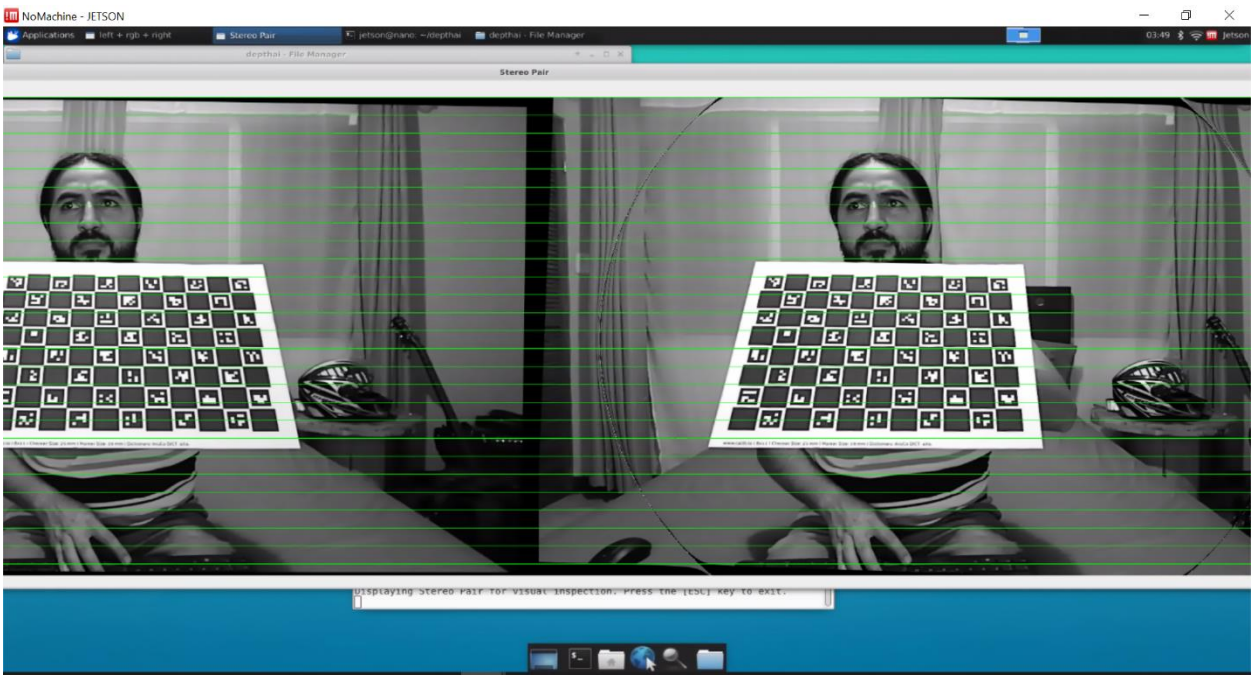


Nota. Tomado de sitio web Luxonis. (Luxonis, 2020).

El proceso de calibración se realiza mediante el uso del patrón brindado por los creadores del dispositivo que se aprecia en la figura 33 y siguiendo una serie de pasos que básicamente consisten en tomar imágenes en diferentes posiciones y aplicar el algoritmo de correlación de puntos, calculando con esto la relación real que debería tener la imagen de la cámara derecha con la izquierda, teniendo en cuenta que la separación de los centros de las cámaras es de 7.5cm y comparando con la diferencia que se ha medido mediante las capturas tomadas en el proceso, como se aprecia en la figura 34, se puede observar el algoritmo de correlación y epipuntos aplicado en la imagen capturada. Si observa detalladamente, entre los cuadros correlacionados con las líneas, se ve claramente la coincidencia entre las esquinas de los conjugados o que indica que el ajuste se realizó correctamente.

Figura 34

Resultado de calibración aplicando algoritmo de correlación

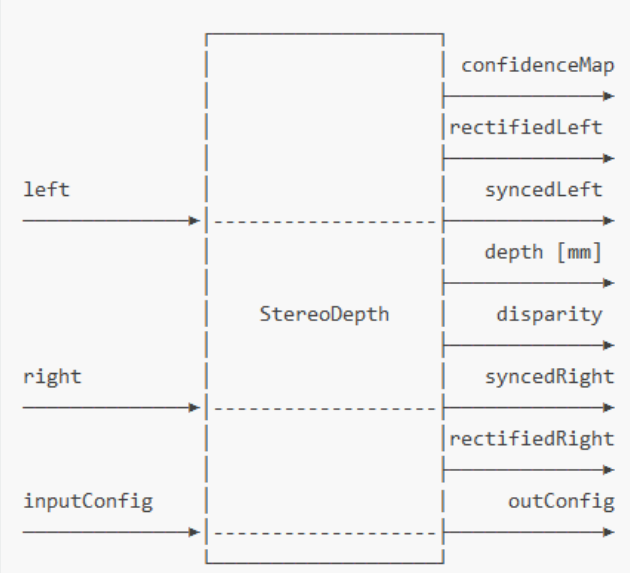


El resultado de este proceso es un valor de error epipolar que en este caso fue de 0.1333275%, lo que indica que la cámara se encuentra en óptimas condiciones para realizar las mediciones necesarias, además de brindarnos un número preciso para calcular el dato de

error general del sistema. Volviendo al tema del nodo StereoDepth, este se puede representar de la forma que describe la figura 35:

Figura 35

Estructura de nodo StereoDepth



Nota. Tomado de sitio web Luxonis. (Luxonis, 2020).

Donde en las entradas se tendrá las imágenes de las dos cámaras y los parámetros de configuración para ajustar que la medición de la profundidad estará adaptada al entorno y a la calibración de la cámara.

Para los parámetros se tiene una lista de posibles variables que afectarán la medición, las cuales están dadas por los entornos donde se realice la toma de datos, por lo que no pueden ser parámetros fijos y debido a esto se realiza una pantalla específica para la configuración de esos parámetros. Esta pantalla contiene los datos necesarios para crear una imagen adecuada al ambiente por medir y luego de asegurarse que los parámetros son los adecuados para el ambiente, se almacenan en un archivo de configuración al que será posible acceder luego, sin necesidad de configurar en cada medición. Los parámetros por configurar son los siguientes:

- Left-Right Check

- Extended disparity mode
- Subpixel mode
- Median filter
- Speckle Filter
- Temporal Filter
- Spatial Edge-Preserving Filter
- Decimation Filter

Estos son los nombres con los que se localizan dentro de la librería de Depth AI, pero se explicará rápidamente qué hace cada filtro para comprender los valores que se están modificando en la ventana de “Configuración de parámetros”.

LR-Check: elimina píxeles de disparidad mal calculados debido a oclusiones en los bordes de los objetos, calcula la disparidad que se da entre la imagen de la derecha con la de la izquierda, luego, al contrario, la de la izquierda con la de la derecha y con cada par de puntos se aplica la fórmula:

$$Disparidad_{L-R} d = (x, y)\ comparado\ con\ Disparidad_{R-L} = (x - d, y) \quad (16)$$

En esta configuración se le entrega un valor entero para que sea el umbral de este cálculo.

Extended disparity mode: permite detectar objetos de distancia más cercana de manera más eficiente. La disparidad por defecto está dada por un rango máximo de 96 a 191 y con esto la distancia mínima se calcula de la siguiente forma:

$$\text{Distancia mínina} = \frac{\text{Distancia focal} * \text{Distancia de línea base}}{95} \quad (17)$$

Donde HFOV en esta cámara por construcción es de 71.9 grados de visión. La distancia de línea base es la distancia que existe entre los dos lentes de la cámara (L y R), por lo que en esta cámara este valor es de 7.5 cm.

Activando el modo extenso de disparidad, este rango de pixeles de disparidad se amplía de 0 a 190. Con esto se reduce la distancia mínima de percepción calculada de la siguiente forma:

$$Distancia\ focal = \frac{ancho\ de\ la\ imagen\ en\ pixeles * 0.5}{\tan(\frac{HFOV * 0.5 * \pi}{180})} \quad (18)$$

Donde HFOV en la cámara por construcción es de 71.9 grados de visión. La distancia de línea base es la distancia que existe entre los dos lentes de la cámara (L y R), por lo que en esta cámara este valor es de 7.5 cm.

Activando el modo extenso de disparidad este rango de pixeles de disparidad se amplía de 0 a 190. Con esto se reduce la distancia mínima de percepción calculada de la siguiente forma:

$$\text{Distancia mínima} = \frac{\text{Distancia focal} * \text{Distancia de línea base}}{190} \quad (19)$$

$$\text{Distancia mínia} = \frac{\text{Distancia focal} * \text{Distancia de línea base}}{190} \quad \textbf{\textit{20)}}$$

Con este cambio, la distancia mínima se reduce ampliamente, pero tiene un costo, que afecta las oclusiones que pueden surgir al realizar la medición. En la configuración del nodo, esto solamente es un valor booleano que permitirá el cambio de las distancias mínimas, que se configuran en los siguientes puntos mediante estas fórmulas explicadas.

Subpixel mode: mejora la precisión y es especialmente útil para mediciones de largo alcance. También ayuda a estimar mejor las superficies normales, esta configuración se deja pensando en los silos de gran tamaño, donde el *extended disparity mode* o es eficiente y como es de esperarse, al activar el *subpixel mode* no es posible activar *extended disparity mode*.

El funcionamiento, en pocas palabras, de este modo es que usando el valor de disparidades por defecto (que como se observó antes, si no se aplica el modo extendido es de 96) por pixel lo fracciona para poder enviar ocho posibles valores dentro del mismo dato, es decir, haciendo cada pixel en tres bits fraccionados para obtener ocho niveles de profundidad, lo que recae en el aumento de pasos de profundidad. Para aplicar esta función solamente es necesario modificar una variable de tipo booleano, esto se recomienda en caso de medir en distancias mayores a los 70 cm.

Median Filter: este es un filtro básico de suavizado como se explicó en capítulos anteriores, se usa si la imagen tiene mucho ruido, por lo que se recomienda para mediciones cercanas, usarlo en mediciones de largas distancias podría afectar la estimación de las superficies.

Este filtro trabaja mediante el enmascarado, por lo que se puede modificar los tamaños de las máscaras para abarcar más pixeles por suavizado, terminando en una degradación mayor o menor de los bordes en la imagen. Los valores de las máscaras son solamente tres tipos: 3x3, 5x5 y 7x7. En la configuración es posible solamente apagarlo o encender el filtro y seleccionar la máscara deseada.

Speckle Filter: este filtro se utiliza para reducir el ruido granular. Cuando una región de la imagen se ve afectada por este tipo de ruido, se detecta debido a una gran variación de disparidad/profundidad entre los píxeles vecinos. El filtro usa un enmascaramiento como el del anterior y reduce considerablemente este ruido, entregando una imagen más limpia. Se recomienda su uso en ambientes con exceso de luz que crean sombras y cambios bruscos de color en las imágenes.

La configuración es mediante la activación o desactivación por medio de la variable booleana y el ingreso de un valor que sería el rango de búsqueda de esos cambios y su filtrado.

Temporal Filter: el filtro temporal funciona para crear un mapa de profundidad con valores persistentes en función a los fotogramas anteriores, es decir, el filtro permite el envío de un solo paso de información, ajustando los valores de profundidad mientras actualiza el resto de la imagen, en caso de que falten datos o los pixeles medidos sean 0 (para el cálculo de profundidad, un pixel de profundidad 0 es un pixel que el sistema reconoce como inválido), el filtro rellena esa información con la de fotogramas anteriores, con base en el factor de persistencia que se le facilita en la ventana de ajuste. Este filtro se recomienda cuando el volumen de mediciones es bajo y la imagen es estática, ya que el historial no ingresará datos erróneos o que afecten la medición.

Spatial Edge-Preserving Filter: el filtro de preservación especial de bordes funciona para llenar los píxeles de profundidad no válidos con píxeles de profundidad vecinos válidos. Realiza una serie de pasadas o iteraciones en una dimensión horizontales y verticales para mejorar la suavidad de los datos reconstruidos. Una mejor explicación para el algoritmo lo da Eduardo S. L. Gastal en su publicación “Domain Transform for Edge-Aware Image and Video Processing”.

Threshold Filter: este filtro es de los más importantes, ya que marcará la base de la medición y su tope, mediante el cálculo de las profundidades mencionado anteriormente, obteniendo dos valores, la distancia mínima a medir y la distancia máxima, por lo que el adecuado ajuste proporcionará mejores mediciones. La distancia mínima se calculará mediante la fórmula antes vista:

$$\text{Distancia mínima} = \frac{\text{Distancia focal} * \text{Distancia de línea base}}{disparidad\ en\ pixeles.} \qquad (21)$$

Donde ya se sabe que el cálculo de la distancia focal depende de las dimensiones de la imagen, la distancia de *baseline* de esta cámara es de 7.5 cm y la disparidad varía dependiendo del modo que se use entre extendido (190) o subpíxel (95). La distancia máxima se calcula mediante las dimensiones de la imagen y el HVOF de la siguiente fórmula:

$$\text{Distancia máxima} = \frac{Distancia\ de\ linea\ base}{2} * \tan\left(\frac{90 - HFOV}{Pixeles\ horizontales} * \frac{\pi}{180}\right) \quad (22)$$

Donde todos los datos fueron explicados con anterioridad, a excepción de los pixeles horizontales que se obtienen cuando se selecciona la resolución de la cámara (en caso de ser una imagen 1280x760 el valor usado sería 1280). En el programa de ajuste se colocarán mediante barras para mayor facilidad.

Decimation Filter: al activar este filtro se generará un submuestreo, el mapa de profundidad, lo que significa que reduce la complejidad de la escena de profundidad y permite que otros filtros se ejecuten más rápido. Este dato se colocará en la ventana de ajuste para modificar el factor de escalado, es decir, si el factor se deja en dos, se reducirá la escala del mapa de profundidad de 1280x800 a 640x400.

Logrando entender este complejo mapa de filtros, se podrá hacer adaptable muestro sistema a cualquier ambiente con solamente mover unas cuantas barras y darles clic a unos cuantos *checkboxes*.

Esta información de la pantalla de ajuste se guardará en un archivo de configuración como se mencionó antes y se configurará en el nodo de StereoDepth mediante una función que se observa en la figura 36, que almacenará las variables necesarias para pasarlas a la parte de la configuración del pipeline de StereoDepth y con esto, poder crea las colas de trabajo para los cálculos subsiguientes.

Figura 36

Proceso de parámetros de configuración de nodo StereoDepth

```
def configureDepthPostProcessing(self,stereoDepthNode):

    ConfNum = []
    ConfNum = self.configNumb()

    LRcheckthresh = ConfNum[0]
    confidenceThreshold = ConfNum[1]
    min_depth = ConfNum[2]
    max_depth = ConfNum[3]
    speckle_range = ConfNum[4]


    ConfCheck = []
    ConfCheck = self.configCheck()

    lrcheck = ConfCheck[0]
    extended = ConfCheck[1]
    subpixel = ConfCheck[2]


    stereoDepthNode.initialConfig.setConfidenceThreshold(confidenceThreshold)
    stereoDepthNode.initialConfig.setLeftRightCheckThreshold(LRcheckthresh)
    stereoDepthNode.initialConfig.setMedianFilter(dai.StereoDepthProperties.MedianFilter.KERNEL_5x5)
    stereoDepthNode.initialConfig.setBilateralFilterSigma(9)
    config = stereoDepthNode.initialConfig.get()
    config.postProcessing.speckleFilter.enable = True
    config.postProcessing.speckleFilter.speckleRange = speckle_range
    config.postProcessing.temporalFilter.enable = True
    config.postProcessing.spatialFilter.enable = True
    config.postProcessing.spatialFilter.holeFillingRadius = 1
    config.postProcessing.spatialFilter.numIterations = 1
    config.postProcessing.thresholdFilter.minRange = min_depth  # mm
    config.postProcessing.thresholdFilter.maxRange = max_depth  # mm
    config.postProcessing.decimationFilter.decimationFactor = 1
    config.censusTransform.enableMeanMode = True
    config.costMatching.linearEquationParameters.alpha = 0
    config.costMatching.linearEquationParameters.beta = 1
    stereoDepthNode.initialConfig.set(config)
    stereoDepthNode.setLeftRightCheck(lrcheck)
    stereoDepthNode.setExtendedDisparity(extended)
    stereoDepthNode.setSubpixel(subpixel)
    stereoDepthNode.setRectifyEdgeFillColor(0)
```

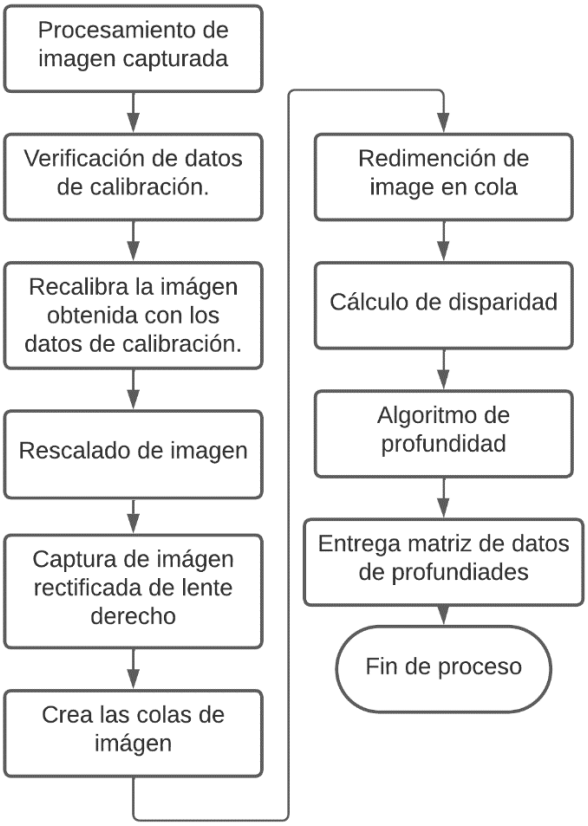
Para finalizar este subproceso, se crean los nodos de salida, asignando las configuraciones expuestas y con esto ya se puede comenzar la medición.

3.7 Proceso de imagen capturada

Se procederá a describir en esta parte del documento, el algoritmo que se aplicó para obtener de la imagen el mapa de profundidad para que en la siguiente etapa se cree una nube de puntos con este mapa. En la figura 37 se presenta el proceso de la imagen capturada.

Figura 37

Procesamiento de imagen capturada

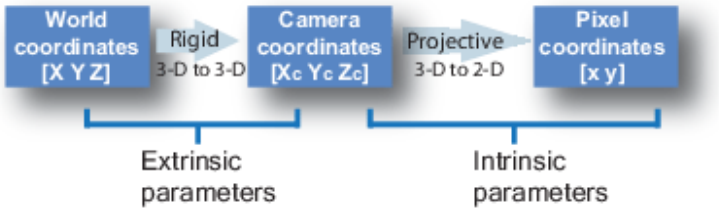


La mejor forma de entender esta parte del algoritmo es mediante las necesidades de las etapas, entonces, si se desea el mapa de profundidades, es necesario conseguir los parámetros de rectificación del punto de referencia y las dimensiones de las imágenes.

Una imagen rectificada es una imagen que pasó por el proceso de adaptación de la imagen, tomando en consideración las proyecciones al mundo real, las distorsiones que conlleva el uso de una cámara y las técnicas de proyectado de tres dimensiones a dos y viceversa. Como se muestra en la figura 38

Figura 38

Proceso de calibración de imágenes



Nota. Tomado de sitio web Mathworks. (Mathworks, 2022).

Para lograr que una imagen sea rectificada, se necesita una matriz que contenga la información para reajustar el mapa de profundidad a la imagen real, es decir, aplicar a las profundidades un arreglo para que se ajusten a las dimensiones del espacio físico real.

Para conseguir esta matriz de adaptación es necesario conocer los parámetros intrínsecos y extrínsecos de la cámara, por lo que con las funciones de captura de esa información que se pueden adquirir mediante Xlink, se calcula con procesos matemáticos las relaciones entre estos parámetros, según lo explicado por la documentación de Matlab de calibración de imágenes; para obtener el valor de la matriz es necesario aplicar el producto x entre la matriz de rotación/traslación y multiplicarla por la matriz de los intrínsecos (k) esta matriz resultante es conocida como la homografía.

$$H = \begin{bmatrix} R \\ t \end{bmatrix} X\ K \tag{23}$$

Al obtener la matriz de homografía, se invierte para adaptarla a la imagen real. El código para realizar este proceso se basa en la solicitud de las matrices mediante Xlink, tal y como se observa en la figura 39.

Figura 39

Código de matriz de homografía

```
calibData = device.readCalibration()
# PArametros intrinsecos
right_intrinsic = np.array(calibData.getCameraIntrinsics(dai.CameraBoardSocket.RIGHT, *depth_image_size))
right_distortion = np.array(calibData.getDistortionCoefficients(dai.CameraBoardSocket.RIGHT))
# Se necesita compensar la imagen ya que el calculo del mapa de profundidades #
# no es basado en la imágen de la camara derecha sin o en la rectificada de la derecha
right_rotation = np.array(calibData.getStereoRightRectificationRotation())
right_homography = np.matmul(np.matmul(right_intrinsic, right_rotation), np.linalg.inv(right_intrinsic))
inverse_rectified_right_intrinsic = np.matmul(np.linalg.inv(right_intrinsic), np.linalg.inv(right_homography))
rectified_right_intrinsic = np.linalg.inv(inverse_rectified_right_intrinsic)
# Información de la cámara RGB
_, rgb_default_width, rgb_default_height = calibData.getDefaultIntrinsics(dai.CameraBoardSocket.RGB)
rgb_orig_intrinsic = np.array(calibData.getCameraIntrinsics(dai.CameraBoardSocket.RGB))
# Reescalado de intrinsecos
rgb_intrinsic = rgb_orig_intrinsic.copy()
width_scaling = depth_image_size[0] / rgb_default_width
height_scaling = depth_image_size[1] / rgb_default_height
rgb_intrinsic[0, :] *= width_scaling
rgb_intrinsic[1, :] *= height_scaling
rgb_distortion = np.array(calibData.getDistortionCoefficients(dai.CameraBoardSocket.RGB))
# Obtención de parámetros extrínsecos.
right_to_rgb_extrinsic = np.array(calibData.getCameraExtrinsics(dai.CameraBoardSocket.RIGHT, dai.CameraBoardSocket.RGB))
```

Con esto calculado, se procede a reescalar la matriz para que coincida con el tamaño de la matriz de profundidad y se está listo para proceder con la proyección de nube de puntos.

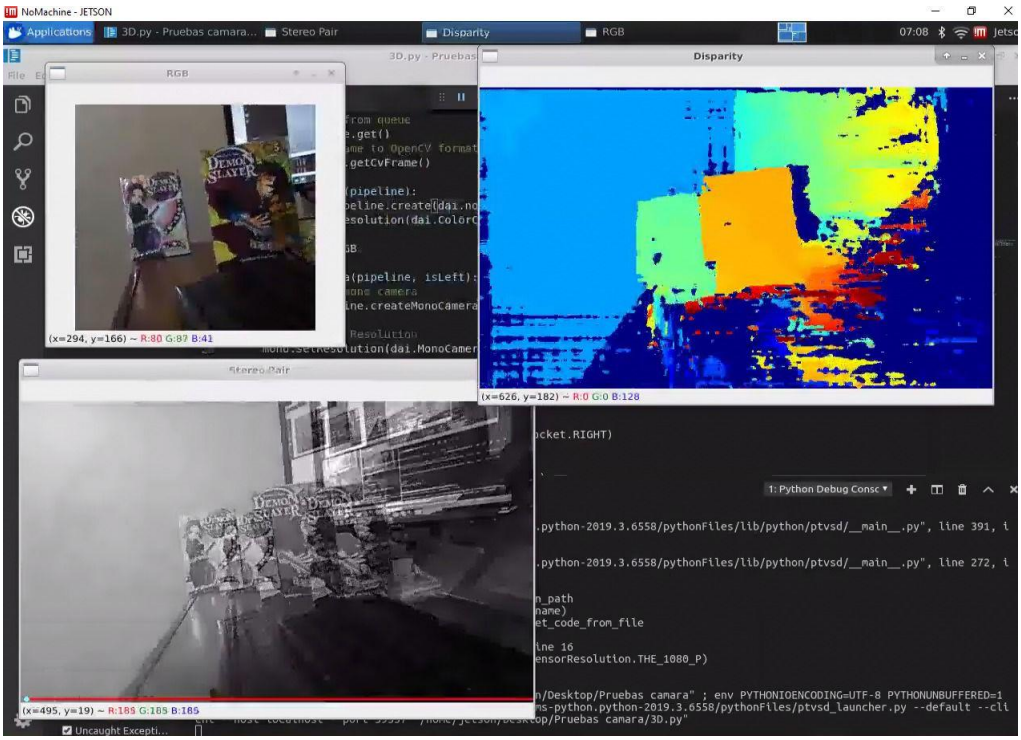
3.8 Trasformación de mapa de profundidad a nube de puntos

Se debe obtener primero para este paso el mapa de profundidad, esto es bastante sencillo, tomando en cuenta que ya se explicó qué información contiene este mapa (visión en estéreo) y cómo lo hace el sistema (nodo StereoDepth), por lo que invocando al *pipeline* de StereoDepth, se obtiene un *frame* con el mapa de profundidad, este *frame* se debe tener con el escalamiento homólogo al que se utiliza para escalar la matriz de homografía para no tener datos erróneos en la medición.

Este mapa se puede proyectar en una imagen mediante la aplicación de una máscara de mapeo de colores que, basado en un umbral predeterminado por cada mapa, se asigna un color a cada valor en la matriz de profundidad, dando como resultado lo que se muestra en la figura 40.

Figura 40

Ejemplo de imagen capturada con cálculo de profundidad

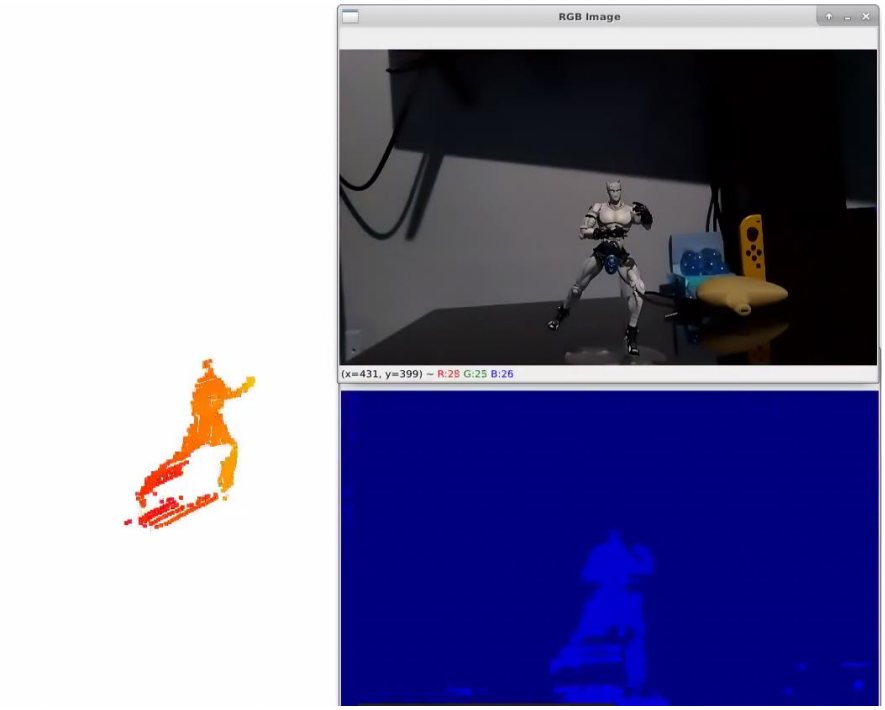


Uno de los puntos para mejorar el código fue utilizar clases para no cargar el código principal por lo que se utilizan los métodos que involucran la librería Open3D en una clase aparte, esto viene a afectar en el sistema, ya que se va a invocar esta clase con sus métodos para entregar el mapa de nubes de puntos.

Para crear una proyección de una nube de puntos es necesario el mapa de profundidad y la matriz de corrección (homografía) de la cámara para que el método “PointCloud create from Depth image” de la librería Open3D cree la equivalencia de la profundidad de cada punto versus la posición de ese punto en el espacio, para luego visualizarlo mediante el mapeo de colores que se explicó anteriormente como lo muestra la figura 41 (Open 3D.org, 2022).

Figura 41

Creación de nube de puntos de superficie de objeto. (Derecha- Nube de puntos), (superior izquierda- Imagen RGB), (Inferior izquierda- Mapa de profundidad)



Ya teniendo la nube de puntos es la hora de generar la superficie, esto es posible realizarlo mediante la aplicación de un algoritmo que convierta esa nube de puntos en malla de figuras (normalmente triangulares) que pueda representar de manera óptima la superficie a reconstruir.

Existen múltiples algoritmos que se pensó para este proceso como la reconstrucción mediante el algoritmo de Poisson, el uso del casco convexo o *convex hull*, el algoritmo de Ball-pivoting, o el algoritmo de Alpha Shape, cada uno tiene sus ventajas y desventajas, dependiendo del objeto o las variaciones que tengan sus capas, incluso, el tamaño del objeto puede marcar cuál algoritmo utilizar. El algoritmo elegido fue el del cálculo de casco convexo, debido a su simple aplicación y excelentes resultados.

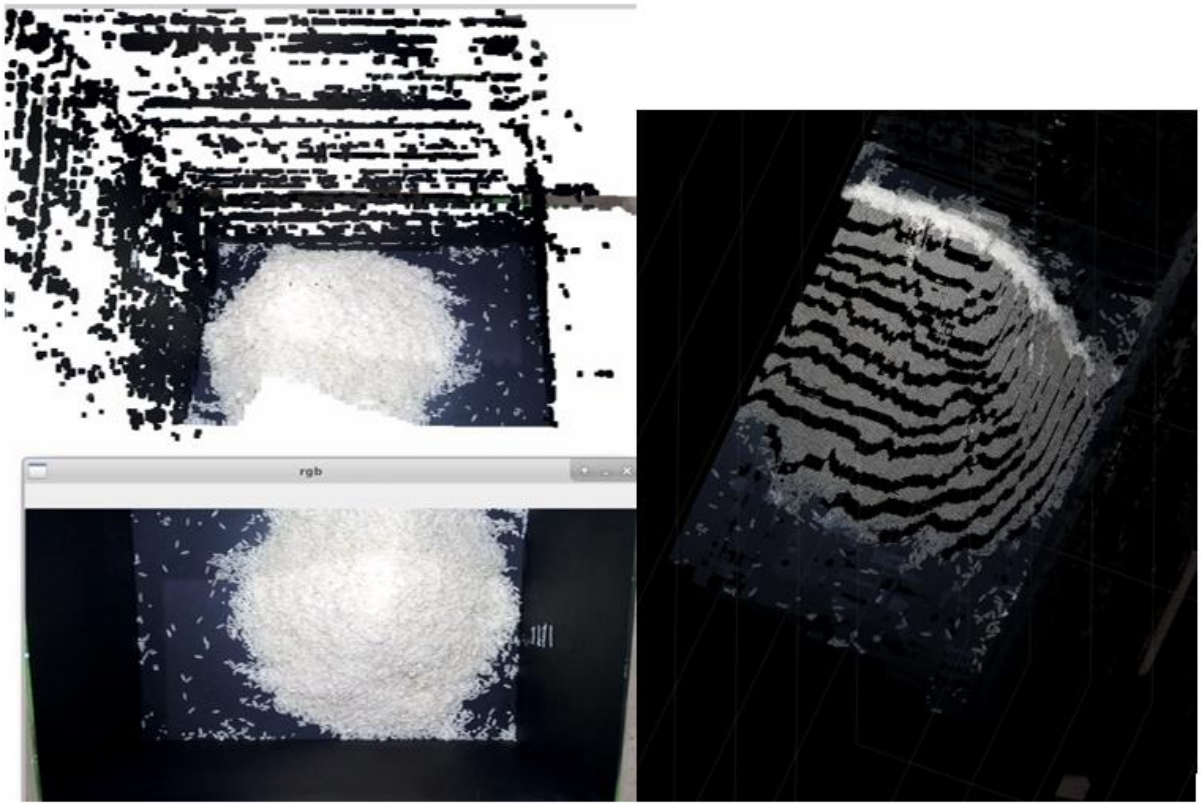
Para realizar este cálculo, lo mejor es tomar la nube calculada de puntos y filtrarla lo más que se pueda, para no tener ningún punto que provoque problemas a la hora de realizar la

reconstrucción cuando se seleccionen los puntos vecinos. Este proceso de depuración de la nube se realizó tomando la nube de puntos original y creando un plano de la superficie y una nube de puntos del resto de los puntos.

Se puede notar en la figura 42, que a la izquierda se puede observar el comportamiento de la imagen real y la proyección de puntos que muestra el sistema para monitorización y a la derecha es posible apreciar la misma nube de puntos que se almacenó como archivo .ply que contiene el conglomerado de los puntos y su posición espacial.

Figura 42

Imágenes de nubes de puntos en representaciones de open3d y Mathlab



Como se aprecia, hay demasiado ruido o puntos que no sirve tener en la imagen, por lo que se debe eliminarlos y dejar solamente la porción de la imagen que sea útil, además, que se necesita centrar la imagen al origen de los ejes para poder calcular el volumen.

El método para recortar la imagen se centra en el mapeo de los puntos que no superan un umbral que se va a colocar (esto sería aplicarlo solo una vez, en el instante cuando se calibre el sistema inicialmente en el sitio) y separarlo de la porción de la imagen que se quiere medir.

Como se ha demostrado a lo largo del proyecto, el uso de las librerías de aporte de comunidad brinda muchísimas ventajas y este punto es en uno donde se demuestra, Open 3d cuenta con los módulos necesarios para mover y crear estos planos, por lo que solamente se debe seleccionar la altura de umbral y el cálculo lo hará de manera automática, separando el plano de la base de la imagen y la porción de interés, como se nota en la figura 43, en la que se ve que la base se pintó de color rojo y el área sobre la base tiene el color real, pero nuevamente se ve como aún hay cierto ruido, por lo que es necesario recortar esa áreas que la cámara capta y no interesan como las paredes de los contenedores o superficies lejanas al contenedor.

Figura 43

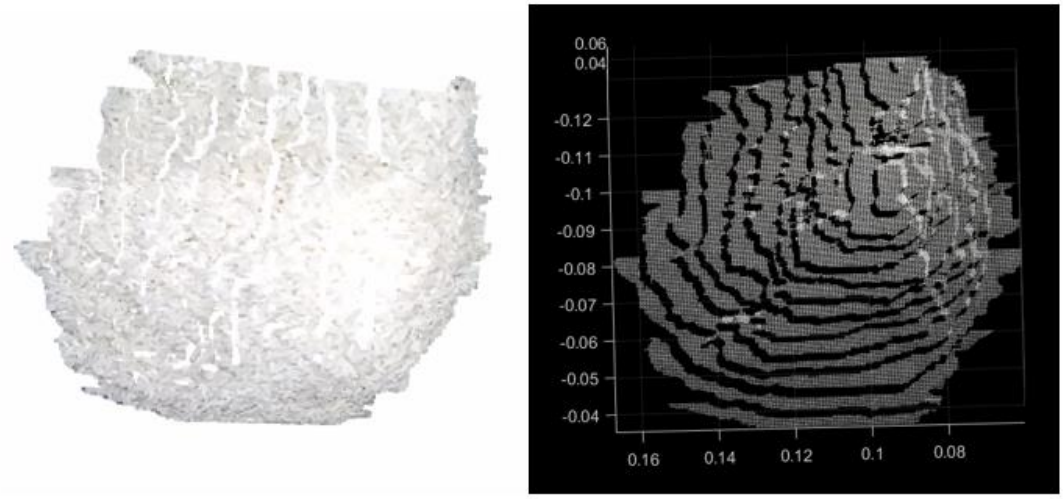
Separación de plano de base con superficie de interés

Cortar las paredes se realiza de manera sencilla, recortando la nube de puntos como se recortaría una imagen en algún punto donde no interesa cierto detalle, mediante la herramienta de cortado o “crop”.

Se crea el cubo que se usará para cortar la imagen y la herramienta de corte, luego se separa la base de la imagen y se obtiene solamente la superficie que se quiere medir como lo muestra la figura 44.

Figura 44

Nube de puntos recortada en Open3d (izquierda) y Mathlab (derecha)



El resultado obtenido es una nube de puntos densa con los puntos que se necesitan, únicamente que se vuelve más sencilla de manipular, ahora luego de este paso, el proceso avanza mediante la creación de la malla de triángulos que se realiza mediante el algoritmo de reconstrucción de Poisson, Open 3d cuenta con una trasformación muy eficiente para una nube de puntos a una malla, usando este algoritmo y solamente se necesita que los puntos en la nube se encuentren normalizados, esto quiere decir que lo que se hace con cada punto es convertirlo en un vector, dándole una orientación para que al aplicar el algoritmo de Poisson se logre entender cuáles áreas quedan dentro del polígono creado y cuáles fuera para poder aplicar la transformada al polígono específico. Al aplicar la función se obtiene la superficie que

se ha estado buscando desde el inicio para poder calcular el volumen y la cual es apreciada en la figura 45.

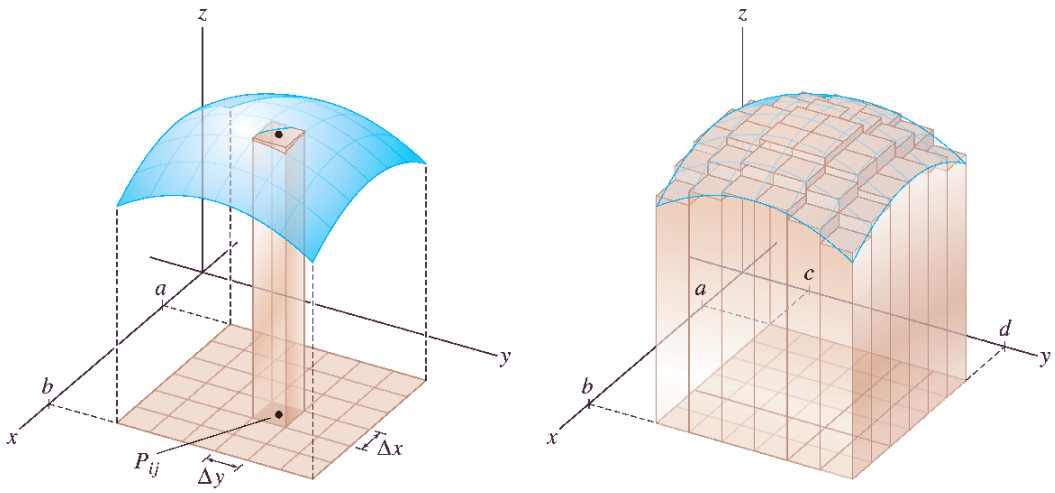
Figura 45

Reconstrucción de superficie por algoritmo de Poisson

Ya con la malla se debe usar las matemáticas y comenzar a calcular el volumen de esa superficie y un método para hacerlo es como se decía, mediante el cálculo del casco convexo que como lo describe Yong Liu en su publicación: “Accurate Volume Calculation Driven by Delaunay Triangulation for Coal Measurement” (Liu, 2021), este método describe que para calcular el volumen de una superficie creada por una malla de triángulos, se calcula el volumen de cada triángulo por separado y se suman para obtener la aproximación deseada y para calcular esto, como lo explica George B. Thomas en su libro de cálculo “Thomas’ Calculusearly Transcendentals” (Thomas, 2010), es simplemente el área de ese triángulo, asumiendo que su superficie es completamente plana por su altura, la cual se obtiene con el mapeo de profundidad que se hizo al inicio del sistema, como se puede observar en la figura 46 tomada del libro antes mencionado.

Figura 46

Aproximación de volumen por cálculo de casco convexo



Nota. Tomado de libro Calculusearly Transcendentals. (Thomas, 2010).

Nuevamente, una librería facilita el trabajo calculando esta compleja tarea y mediante el método Convex_hull.Volume() se obtiene el valor del volumen de la superficie en metros cúbicos. Esto será utilizado para que se multiplique por la densidad aproximada que se calculó mediante los cubos y así se obtendrá la masa aproximada que se encuentra en el contenedor.

Este dato es enviado al HMI para que se aplique a la imagen del tanque y que se almacene en el historial de movimiento de material para que pueda ser evaluado en futuras oportunidades y evaluar la producción.

Toda esa descripción del procedimiento muestra que existen muchas herramientas que pueden facilitar el trabajo en una planta y a mejorar enormemente cientos de procedimientos que hasta el momento contaban con un error humano y mediante la evolución de la tecnología se es capaz de avanzar en el desarrollo de la industria de producción.

Conclusiones

Las industrias de manufactura de Costa Rica y México deben de implementar más sistemas de automatización en los diferentes procesos presentes en la planta con el fin de mejorar sus capacidades y latencias.

A nivel de tecnologías y documentación se está preparado para realizar este tipo de configuración en las diferentes plantas. Se diseñó un sistema capaz de medir masa mediante la aproximación volumétrica de la superficie captada por una cámara.

El sistema implementado tiene un porcentaje de error sumamente bajo en la medición, que supera el porcentaje de error humano que podría presentarse y se estimó en las visitas realizadas (no se permitió acceder a los datos históricos de la planta debido a políticas de seguridad).

Se presentó un algoritmo capaz de realizar el cálculo correspondiente del volumen y masa de algún material específico por medio de un sistema automático de visión. Se presentó un modelo a escala que tiene un porcentaje de error sumamente bajo a la hora de comparar el valor teórico con el práctico.

Se le facilitó a la industria un modelo de medición con un costo bajo para solventar los problemas económicos, ya que este sistema tiene un costo menor a $500 dólares, cuando los sistemas del mismo perfil tienen un costo muchísimo más alto, además de que intervienen en la integridad de las estructuras de los sitos, mientras que este sistema es completamente adaptable sin afectar la integridad del contenedor.

Se logró crear interfaces amigables con los usuarios para la configuración y uso diario de los dispositivos involucrados en el sistema de cálculo. En este caso se hace referencia a HMI, ya que es una herramienta que utilizan los operadores todos los días, es decir, están completamente capacitados a utilizar este tipo de sistema.

Recomendaciones

Como se pudo observar con el estudio desarrollado, en este momento si estas empresas desean implementar el sistema de visión automática o implementar algún proceso de automatización en la planta por medio de visión con una alta precisión y un bajo costo, es necesario que implementen estas soluciones en diferentes áreas lo más pronto posible para mejorar la toma de decisiones o la mejora continua de la planta, aunque es posible, que se requiera la contratación de especialistas en este tema o la utilización del mismo personal, en este caso, ya capacitados en este tema de la planta con el fin de ahorrase la contratación de alguien externo.

Así mismo, es imperante verificar los procesos actuales de cada una de las líneas de producción, puesto que cada una puede funcionar o estar diseñada de una forma diferente o incluso estar adaptada para un producto especifico.

Además, es recomendable seguir optimizando el algoritmo de cálculo de volumen basado en las mejoras que se van creando a lo largo del tiempo con las investigaciones en el ámbito de la visión de computadora, calculo volumétrico de modelos en 3D y algoritmos de adquisición de datos mediante cámaras de visión profunda.

Por todo lo anterior, se recomienda a los patrones o jefes responsables, continuar capacitando a los responsables del sistema con el fin de siempre estar al día con las nuevas tecnologías y conocimientos, ya que, en el ámbito de la tecnología, todos los días hay algo nuevo que aprender y si no se está en un constante aprendizaje puede llegar a tal punto de que la tecnología utilizada llegue a ser obsoleta.

En la mayoría de las ocasiones, en Costa Rica o México, se tiene este tipo de temor a invertir en alguna tecnología nueva que no se ha probado en alguna otra fábrica, les da temor la nueva tecnología que no se haya probado en algún ambiente de producción, por lo tanto, siempre que se lanza un servicio esperan su tiempo y dependiendo de su desarrollo, se realizan las inversiones a nivel local, esto conlleva a que la tecnología utilizada en ese

momento pueda ya ser antigua, porque pueden haber pasado meses desde su salida, pero por el mismo temor, se implementa hasta un año después, por lo tanto, la calidad o el comportamiento esperado puede ser nefasto para la empresa. Por ello, la recomendación es siempre tratar de trabajar con la última tecnología para evitar este tipo de problemas.

En algunas plantas se encuentra el problema de que los tanques o silos están en zonas de la planta donde la conexión eléctrica es sumamente complicada o no existe del todo, por lo tanto, para evitar realizar una instalación eléctrica, se podría utilizar un sistema autosustentable con energía solar, ya que con este sistema propuesto se requiere una cantidad mínima de energía para trabajar correctamente.

Además de complementar este sistema de energía, creándole un sistema de movimiento que siga de cierta forma la posición del sol con el fin de siempre estar en la posición correcta para aprovechar la mayor posible la energía solar o incluso, se podría configurar en la compuerta un tipo de péndulo que mueva una bobina y se aproveche esa engería generada para alimentar el sistema.

Bibliografía

Alegre, E. (2016). *Conceptos y Métodos en Visión por Computador.* España: Comité Español de Automática.

Amazon LLC. (2022). *Amazon LLC*. Obtenido de https://www.amazon.com/-/es/IMX219-77-Jetson-Nano-Resoluci%C3%B3n-megap%C3%ADxeles/dp/B07W3HB9X4

Atvise. (s.f.). *Atvise*. Obtenido de https://atvise.vesterbusiness.com/news/que-es-opc-ua-arquitectura-unificada/#:~:text=La%20Arquitectura%20Unificada%20de%20Comunicaciones,otra%20de%20una%20manera%20simplificada

Contec. (25 de agosto de 2021). *Contec*. Obtenido de https://www.contec.com/support/blog/2021/210825_best-computer/

Depth. (2020). *Depth*. Obtenido de https://docs.luxonis.com/en/latest/"

FreeOPCUA. (2022). *Python OPCUA*. Obtenido de https://python-opcua.readthedocs.io/en/latest/

Fuentes-Hernández, C. A. (2019). *Instrumento virtual para estudiar la Representación de color RGB en máquinas de Visión con Labview.* México: Pistas Educativas.

Hackaday. (18 de marzo de 2019). *Hackaday*. Obtenido de https://hackaday.com/2019/03/18/hands-on-new-nvidia-jetson-nano-is-more-power-in-a-smaller-form-factor/

Industry, D. (s.f.). *LiDAR Navigation Module Outfits Drones, Robots And Autonomous Devices*. Obtenido de https://trends.directindustry.es/project-154791.html

Ingeniería de Sistemas y Automática UMH. (2022). *Capítulo 12 tutoría visión por computadora*. Obtenido de https://isa.umh.es/titere/tutorial/vision/cap12.htm

Janda-Galán, Á. (2008). *Flujo y Atascos de un Medio Granular en la Descarga de Silos.* Servicios de Publicaciones de la Universidad de Navarra.

Koch, R. (1995). 3-D surface reconstruction from stereoscopic image sequences. *Proceedings of IEEE International Conference on Computer Vision*, 109-114.

LearnOpenCV.com. (2022). *LearnOpenCV.com*. Obtenido de https://learnopencv.com/introduction-to-opencv-ai-kit-and-depthai/

Li, Z. (2019). LiDAR Navigation Module Outfits Drones, Robots And Autonomous Devices. *International Conference on Robotics and Automation (ICRA)*, 6905-6911.

Liu, Y. (2021). Accurate Volume Calculation Driven by Delaunay Triangulation for Coal Measurement. *Hindawi Scientific Programming, 2021*.

Luxonis. (2020). *Luxonis*. Obtenido de https://docs.luxonis.com/en/latest/pages/calibration/

Luxonis. (2020). *Nodo Stereo Depth*. Obtenido de https://docs.luxonis.com/projects/api/en/latest/components/nodes/stereo_depth/

Luxonis. (2022). *Luxonis OAK-D*. Obtenido de https://shop.luxonis.com/products/1098obcenclosure

Marr, D. (1982). *Vision : a computational investigation into the human representation and processing of visual information.* San Francisco: W.H. Freeman.

Mathworks. (2022). *What is a Camera Calibration*. Obtenido de https://la.mathworks.com/help/vision/ug/camera-calibration.html

Movidius, an Intel Company. (2017). *newsroom.intel.com.* Obtenido de https://newsroom.intel.com/wp-content/uploads/sites/11/2017/08/movidius-myriad-xvpu-product-brief.pdf

Open 3D.org. (2022). *Open3D.org*. Obtenido de http://www.open3d.org/

OpenCV. (2022). *OpenCV*. Obtenido de https://opencv.org/about/

PNGWING. (2022). *FreePNG*. Obtenido de https://www.pngwing.com/es/free-png-kfxse

Salinas-Castillo, W. E. (2014). Evaluación de la exactitud posicional vertical de una nube de puntos topográficos lidar usando topografía convencional como referencia. *Investigaciones Geográficas, Boletín del Instituto de Geografía*, 5-17. Obtenido de

https://www.researchgate.net/publication/273932070_Evaluacion_de_la_exactitud_posicional_vertical_de_una_nube_de_puntos_topograficos_Lidar_usando_topografia_convencional_como_referencia

Sucar, E. (2011). *Visión Computacional.* Puebla, México.

The MathWorks, I. (15 de marzo de 2022). *Mathworks*. Obtenido de https://www.mathworks.com/help/vision/ug/camera-calibration.html

Thomas, G. B. (2010). *Thomas Calculusearly Transcendentals.* Pearson Education.

Zhang, Y. (2019). RealPoint3D: An Efficient Generation Network for 3D Object Reconstruction from a Single Image. *IEEE ACCESS* , 57539-57549.

Glosario

- **HFVO.:** Horizontal Field of View, campo de visión horizontal, línea de vista en el horizonte, desde un punto de vista como referencia se marca un ángulo desde el punto de observación hasta el horizonte.
- **HMI:** interfaz Hombre-Máquina (**HMI**) es la interfaz entre el proceso y los operarios; se trata básicamente de un panel de instrumentos del operario. Es la principal herramienta utilizada por operarios y supervisores de línea para coordinar y controlar procesos industriales y de fabricación.
- **JSON**.: (JavaScript Object Notation) es un formato ligero de intercambio de datos.
- **Oclusión:** fenómeno se produce principalmente en los bordes de los objetos o en cambios bruscos de la geometría de su superficie, se produce cuando, desde el punto de vista de la cámara, hay puntos de la escena que están ocultos por otros o cuando existen ciertos puntos de la escena que producen sombras sobre otros, ocultando el patrón proyectado.
- **OPC-UA:** la arquitectura unificada **OPC** (**OPC UA**) es un **protocolo** de comunicación independiente del proveedor para aplicaciones de automatización industrial. Se basa en el principio cliente-servidor y permite una comunicación continua desde los sensores y actuadores individuales hasta el sistema ERP o la nube.
- **Parámetros intrínsecos:** los parámetros intrínsecos incluyen la distancia focal, el centro óptico, también conocido como punto principal y el coeficiente de inclinación.
- **Ruido granular:** en las imágenes se crea una marca muy pequeña de un color en el área que rodea un objeto, el ruido granular es la afectación de ese cambio de color en las mediciones de una imagen.
- **Xlink:** especificación que define el lenguaje de enlace XML (XML Linking) que permite insertar elementos en documentos XML para crear y describir enlaces entre recursos.

Utiliza la sintaxis XML para crear estructuras que pueden describir enlaces similares a los hipervínculos unidireccionales simples del HTML actual, así como enlaces más sofisticados.

Anexos

Anexo 1. Código principal

```
#!/usr/bin/env python3
import cv2
import depthai as dai
import numpy as np
import utils
import open3d as o3d
from os import remove
from pathlib import Path
from pyntcloud import PyntCloud
from opcua import Server
from time import sleep
import pandas as pd

no_pcl=False

if not no_pcl:
  try:
    from projector_3d import PointCloudVisualizer
  except ImportError as e:
    raise ImportError(
      f"\033[1;5;31mError occured when importing PCL projector: {e}")

datos = [0,0,0,0,0]

class Volumen():
#-*-*-*-*-*-*-*-*-*-*-*Función de alamcenaje de los parámetros en archivo TXT
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


  def
Almacenar_Config(self,tempConf,tempStra,tempProfMax,tempProfMin,tempSigma):
    f= open("ConfigFiltros.txt","w+")
    Lista_Atributos =
np.array([tempConf,tempStra,tempProfMax,tempProfMin,tempSigma])
    np.savetxt(f,Lista_Atributos, fmt="%s")
    f.close()

#-*-*-*-*-*-*-*-*-*-*-*Función de lectura de los parámetros en archivo TXT *-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

  def Leer_Config(self):
    tempdata = np.loadtxt('ConfigFiltros.txt')
    dataint = tempdata.astype(int)
    return dataint


#Configuración de post procesado

  # Parámetros de Visualización

  #La idea aquí es llamar estas funciones y estas variables llamarlas tempxxx
luego en la config
```

```
  #de stereodepth cargarlas en las variables que ya están.
  #Crear clase para estas funciones o selfs para meterlas en el codigo
principal o crear programa
  #autoejecutable que lea de un argumento que le diga si es de prueba o de
calculo...


  #Configuraciones de Stereo Depth
  def configcheck (self):
    templrcheck = True # Better handling for occlusions
    tempextended = True # Closer-in minimum depth, disparity range is doubled
    tempsubpixel = False # True # Better accuracy for longer distance,
fractional disparity 32-levels
    tempdownsample_pcl = False # downsample the pointcloud before operating
on it and visualizing
    CheckParam = [templrcheck,tempextended,tempsubpixel,tempdownsample_pcl]
    return CheckParam

  def confignumb(self):
    leido = self.Leer_Config()
    tempLRcheckthresh = 5
    tempconfidenceThreshold = leido[0]
    tempmin_depth = leido[3] # mm
    tempmax_depth = leido[2] # mm
    tempspeckle_range = 80
    tempstride = leido[1] # skip points in the depth image when projecting to
3d pointcloud, only matters if align_depth == False
    NumParam =
[tempLRcheckthresh,tempconfidenceThreshold,tempmin_depth,tempmax_depth,tempsp
eckle_range,tempstride]
    return NumParam

  #Función para aplicar los parámetros de los filtros de post proceso

  def configureDepthPostProcessing(self,stereoDepthNode):

    ConfNum = []
    ConfNum = self.confignumb()

    LRcheckthresh = ConfNum[0]
    confidenceThreshold = ConfNum[1]
    min_depth = ConfNum[2]
    max_depth = ConfNum[3]
    speckle_range = ConfNum[4]


    ConfCheck = []
    ConfCheck = self.configcheck()

    lrcheck = ConfCheck[0]
    extended = ConfCheck[1]
    subpixel = ConfCheck[2]


    stereoDepthNode.initialConfig.setConfidenceThreshold(confidenceThreshold)
    stereoDepthNode.initialConfig.setLeftRightCheckThreshold(LRcheckthresh)
```

```
stereoDepthNode.initialConfig.setMedianFilter(dai.StereoDepthProperties.Media
nFilter.KERNEL_5x5)
    stereoDepthNode.initialConfig.setBilateralFilterSigma(9)
    config = stereoDepthNode.initialConfig.get()
    config.postProcessing.speckleFilter.enable = True
    config.postProcessing.speckleFilter.speckleRange = speckle_range
    config.postProcessing.temporalFilter.enable = True
    config.postProcessing.spatialFilter.enable = True
    config.postProcessing.spatialFilter.holeFillingRadius = 1
    config.postProcessing.spatialFilter.numIterations = 1
    config.postProcessing.thresholdFilter.minRange = min_depth # mm
    config.postProcessing.thresholdFilter.maxRange = max_depth # mm
    config.postProcessing.decimationFilter.decimationFactor = 1
    config.censusTransform.enableMeanMode = True
    config.costMatching.linearEquationParameters.alpha = 0
    config.costMatching.linearEquationParameters.beta = 1
    stereoDepthNode.initialConfig.set(config)
    stereoDepthNode.setLeftRightCheck(lrcheck)
    stereoDepthNode.setExtendedDisparity(extended)
    stereoDepthNode.setSubpixel(subpixel)
    stereoDepthNode.setRectifyEdgeFillColor(0)


  # Función de creación de pipeline, recursos y configuración del device.

  def create_pipelines(self):
    #Selección de resolución de cámaras
    rgb_resolution = dai.ColorCameraProperties.SensorResolution.THE_1080_P
    depth_resolution = dai.MonoCameraProperties.SensorResolution.THE_400_P
    pipeline = dai.Pipeline()
    # Se definen los recurso físicos )Nodos de entrada)
    camRgb = pipeline.createColorCamera()
    left = pipeline.createMonoCamera()
    right = pipeline.createMonoCamera()
    stereo = pipeline.createStereoDepth()
    # Se crean los nodos de salida
    depthOut = pipeline.createXLinkOut()
    depthOut.setStreamName("depth")
    rgbOut = pipeline.createXLinkOut()
    rgbOut.setStreamName("rgb")
    # Configuración de cámaras
    camRgb.setResolution(rgb_resolution)
    camRgb.setInterleaved(False)
    camRgb.setBoardSocket(dai.CameraBoardSocket.RGB)
    mono_camera_resolution = depth_resolution
    left.setResolution(mono_camera_resolution)
    left.setBoardSocket(dai.CameraBoardSocket.LEFT)
    right.setResolution(mono_camera_resolution)
    right.setBoardSocket(dai.CameraBoardSocket.RIGHT)
    #Configuración de filtros de postproceso
    self.configureDepthPostProcessing(stereo)
    stereo.setDepthAlign(dai.CameraBoardSocket.RGB)
    # Enlazamos los nodos de salida con los de entrada
    left.out.link(stereo.left)
    right.out.link(stereo.right)
    stereo.depth.link(depthOut.input)
```

```
    camRgb.video.link(rgbOut.input)
    # Book-keeping
    streams = [depthOut.getStreamName(), rgbOut.getStreamName()]
    rgb_image_size = (camRgb.getResolutionWidth(),
camRgb.getResolutionHeight())
    depth_image_size = (right.getResolutionWidth(),
right.getResolutionHeight())

    return pipeline, streams, rgb_image_size, depth_image_size

  def __init__(self):

    Confstride = []
    Confstride = self.configNumb()

    stride = Confstride[5]

    ConfDown = []
    ConfDown = self.configCheck()

    downsample_pcl = ConfDown[3]

    pipeline, streams, rgb_image_size, depth_image_size =
self.create_pipelines()

    # Conectamos los nodos al dispositivo (ligar el pipeline al dispositivo)

    with dai.Device(pipeline) as device:
      # Se lee la última calibración del dispositivo.
      calibData = device.readCalibration()
      # PArametros intrínsecos
      right_intrinsic =
np.array(calibData.getCameraIntrinsics(dai.CameraBoardSocket.RIGHT,
*depth_image_size))
      right_distortion =
np.array(calibData.getDistortionCoefficients(dai.CameraBoardSocket.RIGHT))
      # Se necesita compensar la imagen, ya que el calculo del mapa de
profundidades #
      # no es basado en la imágen de la camara derecha sin o en la
rectificada de la derecha
      right_rotation =
np.array(calibData.getStereoRightRectificationRotation())
      right_homography = np.matmul(np.matmul(right_intrinsic,
right_rotation), np.linalg.inv(right_intrinsic))
      inverse_rectified_right_intrinsic =
np.matmul(np.linalg.inv(right_intrinsic), np.linalg.inv(right_homography))
      rectified_right_intrinsic =
np.linalg.inv(inverse_rectified_right_intrinsic)
      # Información de la cámara RGB
      _, rgb_default_width, rgb_default_height =
calibData.getDefaultIntrinsics(dai.CameraBoardSocket.RGB)
      rgb_orig_intrinsic =
np.array(calibData.getCameraIntrinsics(dai.CameraBoardSocket.RGB))
      # Reescalado de intrinsecos
      rgb_intrinsic = rgb_orig_intrinsic.copy()
      width_scaling = depth_image_size[0] / rgb_default_width
      height_scaling = depth_image_size[1] / rgb_default_height
```

```
      rgb_intrinsic[0, :] *= width_scaling
      rgb_intrinsic[1, :] *= height_scaling
      rgb_distortion =
np.array(calibData.getDistortionCoefficients(dai.CameraBoardSocket.RGB))
      # Obtención de parámetros extrínsecos.
      right_to_rgb_extrinsic =
np.array(calibData.getCameraExtrinsics(dai.CameraBoardSocket.RIGHT,
dai.CameraBoardSocket.RGB))

      #cv2.namedWindow("Depth Image")

      pcl_converter = PointCloudVisualizer(rectified_right_intrinsic,
*depth_image_size)

      pcl_frames = [None, None] # depth frame, color frame
      queue_list = [device.getOutputQueue(stream, maxSize=8, blocking=False)
for stream in streams]
      pixel_coords = utils.pixel_coord_np(*depth_image_size).astype(float)
      tiempo = 0
      sleep (10)
      for i, queue in enumerate(queue_list):
        name = queue.getName()
        image = queue.get()

        depth_frame = np.array(image.getFrame())
        h,w = depth_frame.shape
        pcl_frames[i] = cv2.resize(depth_frame, depth_image_size,
cv2.INTER_NEAREST)
      if any([frame is None for frame in pcl_frames]):
        print("Error: Need rgb AND depth frame to align!")

      elif pcl_frames[0] is None:
        print("Waiting on depth image to visualize")

      #quantized_depth_image = utils.quantizeDepthFrame(pcl_frames[0],
depth_scale_factor=5000)
      #cv2.imshow("Depth Image", quantized_depth_image)
      if not no_pcl:

 # ES aquí donde preparamos la imagen obtenida en un instante para
convertirla en una proyección de puntos
        PCP = pcl_converter.depth_to_projection(pcl_frames[0], stride=stride,
downsample=downsample_pcl)


        fil = Path('copy_of_fragment.ply').exists()
        if fil == True:
          remove('copy_of_fragment.ply')
          o3d.io.write_point_cloud("copy_of_fragment.ply",
PCP,write_ascii=True)
        else:
          o3d.io.write_point_cloud("copy_of_fragment.ply", PCP,
write_ascii=True)

        diamond = PyntCloud.from_file("copy_of_fragment.ply")
        convex_hull_id = diamond.add_structure("convex_hull")
        convex_hull = diamond.structures[convex_hull_id]
```

```
        diamond.mesh = convex_hull.get_mesh()
#Auí comienza la magia de tomar el ply, converirlo en una estructura sólida y
aproximar el volumen
        Filemesh = Path('diamond_hull.ply').exists()
        if Filemesh == True:
          remove('diamond_hull.ply')
          diamond.to_file("diamond_hull.ply", also_save=["mesh"])
        else:
          diamond.to_file("diamond_hull.ply", also_save=["mesh"])
#La función convex_hull calcula el aproximado del volumen bajo la superficie
creada, basado en el proceso de integración de múltiples puntos
        volume = convex_hull.volume
        self.value = volume
        #print(volume)
  def __str__(self):
    return str(self.value)


#-*-*-*-*-*-*-*-*-* Inicio del programa*-*-*-*-*-*-*-*-*-**-*
# Se comienza corriendo el servidor OPC UA
server = Server()
#Esta dirección cambia depende la conexión en el wifi
server.set_endpoint("opc.tcp://192.168.1.210:12345")
server.register_namespace("Room1")
#El servidor está basado en objetos por lo que se crea un cuarto para
almacenar los objetos
objects = server.get_objects_node()
#El primer objeto es el calculador de volumen, se crea el objeto y se le
asigna la variable respectiva
Volumencalc = objects.add_object(2,"Calculador de Volumen")
vol = Volumencalc.add_variable(2,"Volumen", 20)
#Creamos un segundo objeto que será nuestro trigger, es decir el activador
del software desde el HMI
Trig = objects.add_object(2,"Trigger")
state = Trig.add_variable(2, "Start Mesure", True)
state.set_writable()
vol.set_read_only()
#Ciclo principal donde consulta si es necesario medir o no
try:
  print ("Start server")
  server.start()
  print("Server Online")
  while True:
    check = state.get_value()
    check2 = vol.get_value()
    print("Estado del motor: " +str(state.get_value()))
    if check:
      VolumenCalculado = str (Volumen())
      vol.set_value(VolumenCalculado)
      print ("Volumen Calculado: " +str(vol.get_value()))
      sleep(2)
    else:
      print("Motor no activo")
      sleep(2)
finally:
  server.stop()
  print("Server Offline")
```

Anexo 2. Código de clase de pointcloud

```
#!/usr/bin/env python3


import numpy as np
import open3d as o3d
from pyntcloud import PyntCloud
import pandas as pd
import math
from scipy.spatial import Delaunay
from pyhull.delaunay import DelaunayTri

class PointCloudVisualizer():
  def __init__(self, intrinsic_matrix, width, height):
    self.pcl = None
    # transform from camera to world orientation (Note the absolute position
won't be correct)
    self.R_camera_to_world = np.array([[-1, 0, 0], [0, 1, 0], [0, 0, -
1]]).astype(np.float64)

    self.pinhole_camera_intrinsic = o3d.camera.PinholeCameraIntrinsic(width,
                                    height,
                                    intrinsic_matrix[0][0],
                                    intrinsic_matrix[1][1],
                                    intrinsic_matrix[0][2],
                                    intrinsic_matrix[1][2])
    self.vis = o3d.visualization.Visualizer()
    self.vis.create_window()
    self.isstarted = False

  def depth_to_projection(self, depth_map, stride=1, downsample=False):
    depth_o3d = o3d.geometry.Image(depth_map)
    if self.pcl is None:
      self.pcl = o3d.geometry.PointCloud.create_from_depth_image(depth_o3d,
self.pinhole_camera_intrinsic, stride=stride)
    else:
      pcd = o3d.geometry.PointCloud.create_from_depth_image(depth_o3d,
self.pinhole_camera_intrinsic, stride=stride)
      if downsample:
        pcd = pcd.voxel_down_sample(voxel_size=0.01)
      # Remove noise
      pcd = pcd.remove_statistical_outlier(30, 0.1)[0]
      self.pcl.points = pcd.points

    # Rotate the pointcloud such that it is in the world coordinate frame
(easier to visualize)
    self.pcl.rotate(self.R_camera_to_world,
center=np.array([0,0,0],dtype=np.float64))
    #self.downpcd = self.pcl.voxel_down_sample(voxel_size=0.05)

#self.downpcd.estimate_normals(search_param=o3d.geometry.KDTreeSearchParamHyb
rid(radius=0.1, max_nn=30))
    #self.distances = self.downpcd.compute_nearest_neighbor_distance()
    #self.avg_dist = np.mean(self.distances)
    #self.radius = 4*self.avg_dist
```

```
    #self.bpa_mesh =
o3d.geometry.TriangleMesh.create_from_point_cloud_ball_pivoting(self.downpcd,
o3d.utility.DoubleVector([self.radius, self.radius * 2]))
    #self.dec_mesh = self.bpa_mesh.simplify_quadric_decimation(160000000)
    self.PCD = self.pcl
    self.plane_model, self.inliers =
self.PCD.segment_plane(distance_threshold=0.01,
                       ransac_n=3,
                       num_iterations=1000)
    [a, b, c, d] = self.plane_model
    self.plane_pcd = self.PCD.select_by_index(self.inliers)
    self.plane_pcd.paint_uniform_color([1.0, 0, 0])
    self.stockpile_pcd = self.PCD.select_by_index(self.inliers, invert=True)
    self.stockpile_pcd.paint_uniform_color([0, 0, 1.0])
    self.plane_pcd = self.plane_pcd.translate((0,0,d/c))
    self.stockpile_pcd = self.stockpile_pcd.translate((0,0,d/c))
    self.cos_theta = c / math.sqrt(a**2 + b**2 + c**2)
    self.sin_theta = math.sqrt((a**2+b**2)/(a**2 + b**2 + c**2))
    u_1 = b / math.sqrt(a**2 + b**2 )
    u_2 = -a / math.sqrt(a**2 + b**2)
    self.rotation_matrix = np.array([[self.cos_theta + u_1**2 * (1-
self.cos_theta), u_1*u_2*(1-self.cos_theta), u_2*self.sin_theta],
                  [u_1*u_2*(1-self.cos_theta), self.cos_theta + u_2**2*(1-
self.cos_theta), -u_1*self.sin_theta],
                  [-u_2*self.sin_theta, u_1*self.sin_theta, self.cos_theta]])
    self.plane_pcd.rotate(self.rotation_matrix)
    self.stockpile_pcd.rotate(self.rotation_matrix)
    self.cl, self.ind =
self.stockpile_pcd.remove_statistical_outlier(nb_neighbors=30,
                           std_ratio=2.0)
    self.stockpile_pcd = self.stockpile_pcd.select_by_index(self.ind)

    return self.stockpile_pcd

  def rgbd_to_projection(self, depth_map, rgb, downsample=False):
    rgb_o3d = o3d.geometry.Image(rgb)
    depth_o3d = o3d.geometry.Image(depth_map)
    rgbd_image = o3d.geometry.RGBDImage.create_from_color_and_depth(rgb_o3d,
depth_o3d, convert_rgb_to_intensity=False, depth_trunc=6)
    if self.pcl is None:
      self.pcl = o3d.geometry.PointCloud.create_from_rgbd_image(rgbd_image,
self.pinhole_camera_intrinsic)
    else:
      pcd = o3d.geometry.PointCloud.create_from_rgbd_image(rgbd_image,
self.pinhole_camera_intrinsic)
      if downsample:
        pcd = pcd.voxel_down_sample(voxel_size=0.01)
      # Remove noise
      pcd = pcd.remove_statistical_outlier(30, 0.1)[0]

      self.pcl.points = pcd.points
      self.pcl.colors = pcd.colors

    # Rotate the pointcloud such that it is in the world coordinate frame
(easier to visualize)
    self.pcl.rotate(self.R_camera_to_world,
center=np.array([0,0,0],dtype=np.float64))
```

```
    self.rgbd = o3d.geometry.RGBDImage.create_from_color_and_depth(rgb_o3d,
depth_o3d, convert_rgb_to_intensity = False)
    self.pcda = o3d.geometry.PointCloud.create_from_rgbd_image(self.rgbd,
self.pinhole_camera_intrinsic)
    self.point_si = self.pcda.rotate(self.R_camera_to_world,
center=np.array([0,0,0],dtype=np.float64))
    o3d.io.write_point_cloud("si.ply", self.point_si)
    self.PCD = o3d.io.read_point_cloud("si.ply")
    self.axes = o3d.geometry.TriangleMesh.create_coordinate_frame()
    self.plane_model, self.inliers =
self.PCD.segment_plane(distance_threshold=0.01,
                       ransac_n=3,
                       num_iterations=10000)
    [a, b, c, d] = self.plane_model
    self.plane_pcd = self.PCD.select_by_index(self.inliers)
    self.plane_pcd.paint_uniform_color([1.0, 0, 0])
    self.stockpile_pcd = self.PCD.select_by_index(self.inliers, invert=True)
    self.stockpile_pcd.paint_uniform_color([0, 0, 1.0])
    self.plane_pcd = self.plane_pcd.translate((0,0,d/c))
    self.stockpile_pcd = self.stockpile_pcd.translate((0,0,d/c))
    self.cos_theta = c / math.sqrt(a**2 + b**2 + c**2)
    self.sin_theta = math.sqrt((a**2+b**2)/(a**2 + b**2 + c**2))
    u_1 = b / math.sqrt(a**2 + b**2 )
    u_2 = -a / math.sqrt(a**2 + b**2)
    self.rotation_matrix = np.array([[self.cos_theta + u_1**2 * (1-
self.cos_theta), u_1*u_2*(1-self.cos_theta), u_2*self.sin_theta],
                  [u_1*u_2*(1-self.cos_theta), self.cos_theta + u_2**2*(1-
self.cos_theta), -u_1*self.sin_theta],
                  [-u_2*self.sin_theta, u_1*self.sin_theta, self.cos_theta]])
    self.plane_pcd.rotate(self.rotation_matrix)
    self.stockpile_pcd.rotate(self.rotation_matrix)
    self.cl, self.ind =
self.stockpile_pcd.remove_statistical_outlier(nb_neighbors=30,
                         std_ratio=2.0)
    self.stockpile_pcd = self.stockpile_pcd.select_by_index(self.ind)
    #o3d.visualization.draw_geometries([self.stockpile_pcd])


    #o3d.visualization.draw_geometries([self.plane_pcd, self.stockpile_pcd,
self.axes])
    return self.stockpile_pcd

  def visualize_pcd(self):
    if not self.isstarted:
      self.vis.add_geometry(self.pcl)
      origin = o3d.geometry.TriangleMesh.create_coordinate_frame(size=0.1,
origin=[0, 0, 0])
      self.vis.add_geometry(origin)
      self.isstarted = True
    else:
      self.vis.update_geometry(self.pcl)
      self.vis.poll_events()
      self.vis.update_renderer()

  def close_window(self):
    self.vis.destroy_window()
```

Anexo 3. Código de clase de cálculo de volumen

Anexo 4. Código de GUI de ajuste

```
#!/usr/bin/env python3

import main
import sys
from time import sleep
import utils
import open3d as o3d
import depthai as dai
import numpy as np, cv2 as cv
cv2=cv
from PyQt5.QtCore import Qt

from PyQt5.QtWidgets import (
  QApplication,
  QDialog,
  QMainWindow,
  QMessageBox,
  QPushButton,
  QCheckBox,
  QLabel,
  QSlider,
  QWidget,
  QVBoxLayout,
  QHBoxLayout,
  QTabBar,
)


no_pcl=False

if not no_pcl:
  try:
    from projector_3d import PointCloudVisualizer
  except ImportError as e:
    raise ImportError(
      f"\033[1;5;31mError occured when importing PCL projector: {e}")

datos = [0,0,0,0,0]


#Clase principal de la ventana
class MainWindow(QMainWindow):

#-*-*-*-*-*-*-*-*-*-*Función de alamcenaje de los parámetros en archivo TXT
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


  def
Almacenar_Config(self,tempConf,tempStra,tempProfMax,tempProfMin,tempSigma):
```

```
    f= open("ConfigFiltros.txt","w+")
    Lista_Atributos =
np.array([tempConf,tempStra,tempProfMax,tempProfMin,tempSigma])
    np.savetxt(f,Lista_Atributos, fmt="%s")
    f.close()

#-*-*-*-*-*-*-*-*-*-*Función de lectura de los parámetros en archivo TXT *-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

  def Leer_Config(self):
    tempdata = np.loadtxt('ConfigFiltros.txt')
    dataint = tempdata.astype(int)
    return dataint


#Configuración de post procesado

  # Parámetros de Visualización

  #La idea aquí es llamar estas funciones y estas variables llamarlas tempxxx
luego en la config
  #de stereodepth cargarlas en las variables que ya están.
  #Crear clase para estas funciones o selfs para meterlas en el codigo
principal o crear programa
  #autoejecutable que lea de un argumento que le diga si es de prueba o de
calculo...


  #Configuraciones de Stereo Depth
  def configcheck (self):
    templrcheck = True # Better handling for occlusions
    tempextended = True # Closer-in minimum depth, disparity range is doubled
    tempsubpixel = False # True # Better accuracy for longer distance,
fractional disparity 32-levels
    tempdownsample_pcl = False # downsample the pointcloud before operating
on it and visualizing
    CheckParam = [templrcheck,tempextended,tempsubpixel,tempdownsample_pcl]
    return CheckParam

  def confignumb(self):
    leido = self.Leer_Config()
    tempLRcheckthresh = 4
    tempconfidenceThreshold = leido[0]
    tempmin_depth = leido[3] # mm
    tempmax_depth = leido[2] # mm
    tempspeckle_range = 10
    tempstride = leido[1] # skip points in the depth image when projecting to
3d pointcloud, only matters if align_depth == False
    NumParam =
[tempLRcheckthresh,tempconfidenceThreshold,tempmin_depth,tempmax_depth,tempsp
eckle_range,tempstride]
    return NumParam

  #Función para aplicar los parámetros de los filtros de post proceso

  def configureDepthPostProcessing(self,stereoDepthNode):
```

```
    ConfNum = []
    ConfNum = self.configumb()

    LRcheckthresh = ConfNum[0]
    confidenceThreshold = ConfNum[1]
    min_depth = ConfNum[2]
    max_depth = ConfNum[3]
    speckle_range = ConfNum[4]


    ConfCheck = []
    ConfCheck = self.configcheck()

    lrcheck = ConfCheck[0]
    extended = ConfCheck[1]
    subpixel = ConfCheck[2]


    stereoDepthNode.initialConfig.setConfidenceThreshold(confidenceThreshold)
    stereoDepthNode.initialConfig.setLeftRightCheckThreshold(LRcheckthresh)

stereoDepthNode.initialConfig.setMedianFilter(dai.StereoDepthProperties.Media
nFilter.KERNEL_5x5)
    stereoDepthNode.initialConfig.setBilateralFilterSigma(8)
    config = stereoDepthNode.initialConfig.get()
    config.postProcessing.speckleFilter.enable = True
    config.postProcessing.speckleFilter.speckleRange = speckle_range
    config.postProcessing.temporalFilter.enable = True
    config.postProcessing.spatialFilter.enable = True
    config.postProcessing.spatialFilter.holeFillingRadius = 2
    config.postProcessing.spatialFilter.numIterations = 2
    config.postProcessing.thresholdFilter.minRange = min_depth # mm
    config.postProcessing.thresholdFilter.maxRange = max_depth # mm
    config.postProcessing.decimationFilter.decimationFactor = 1
    config.censusTransform.enableMeanMode = True
    config.costMatching.linearEquationParameters.alpha = 0
    config.costMatching.linearEquationParameters.beta = 1
    stereoDepthNode.initialConfig.set(config)
    stereoDepthNode.setLeftRightCheck(lrcheck)
    stereoDepthNode.setExtendedDisparity(extended)
    stereoDepthNode.setSubpixel(subpixel)
    stereoDepthNode.setRectifyEdgeFillColor(0) # Black, to better see the
cutout


  # Función de creación de pipeline, recursos y configuración del device.

  def create_rgbd_pipeline(self):

    #Selección de resolución de cámaras

    rgb_resolution = dai.ColorCameraProperties.SensorResolution.THE_1080_P
    depth_resolution = dai.MonoCameraProperties.SensorResolution.THE_400_P

    pipeline = dai.Pipeline()

    # Se definen los recurso físicos )Nodos de entrada)
```

```
    camRgb = pipeline.createColorCamera()
    left = pipeline.createMonoCamera()
    right = pipeline.createMonoCamera()
    stereo = pipeline.createStereoDepth()

    # Se crean los nodos de salida
    depthOut = pipeline.createXLinkOut()
    depthOut.setStreamName("depth")
    rgbOut = pipeline.createXLinkOut()
    rgbOut.setStreamName("rgb")

    # Configuración de cámaras
    camRgb.setResolution(rgb_resolution)
    camRgb.setInterleaved(False)
    camRgb.setBoardSocket(dai.CameraBoardSocket.RGB)


    mono_camera_resolution = depth_resolution
    left.setResolution(mono_camera_resolution)
    left.setBoardSocket(dai.CameraBoardSocket.LEFT)
    right.setResolution(mono_camera_resolution)
    right.setBoardSocket(dai.CameraBoardSocket.RIGHT)

    #Configuración de filtros de postproceso
    self.configureDepthPostProcessing(stereo)
    stereo.setDepthAlign(dai.CameraBoardSocket.RGB)

    # Enlazamos los nodos de salida con los de entrada
    left.out.link(stereo.left)
    right.out.link(stereo.right)
    stereo.depth.link(depthOut.input)
    camRgb.video.link(rgbOut.input)

    # Book-keeping
    streams = [depthOut.getStreamName(), rgbOut.getStreamName()]
    rgb_image_size = (camRgb.getResolutionWidth(),
camRgb.getResolutionHeight())
    depth_image_size = (right.getResolutionWidth(),
right.getResolutionHeight())

    return pipeline, streams, rgb_image_size, depth_image_size

  def Videoplayer (self):

    Confstride = []
    Confstride = self.configNumb()

    stride = Confstride[5]

    ConfDown = []
    ConfDown = self.configcheck()

    downsample_pcl = ConfDown[3]

    pipeline, streams, rgb_image_size, depth_image_size =
self.create_rgbd_pipeline()
```

```
    # Conectamos los nodos al dispositivo (ligar el pipeline al dispositivo)

    with dai.Device(pipeline) as device:
      # Se lee la última calibración del dispositivo.
      calibData = device.readCalibration()
      # RIGHT CAMERA INFO
      right_intrinsic =
np.array(calibData.getCameraIntrinsics(dai.CameraBoardSocket.RIGHT,
*depth_image_size))
      right_distortion =
np.array(calibData.getDistortionCoefficients(dai.CameraBoardSocket.RIGHT))
      # This is needed because the depth image is made from the rectified
right camera image, not the right camera image
      # (although in practice, I did not see a big difference)
      right_rotation =
np.array(calibData.getStereoRightRectificationRotation())
      right_homography = np.matmul(np.matmul(right_intrinsic,
right_rotation), np.linalg.inv(right_intrinsic))
      inverse_rectified_right_intrinsic =
np.matmul(np.linalg.inv(right_intrinsic), np.linalg.inv(right_homography))
      rectified_right_intrinsic =
np.linalg.inv(inverse_rectified_right_intrinsic)
      # COLOR CAMERA INFO
      _, rgb_default_width, rgb_default_height =
calibData.getDefaultIntrinsics(dai.CameraBoardSocket.RGB)
      rgb_orig_intrinsic =
np.array(calibData.getCameraIntrinsics(dai.CameraBoardSocket.RGB))
      # update camera intrinsics based on new image size
      rgb_intrinsic = rgb_orig_intrinsic.copy()
      width_scaling = depth_image_size[0] / rgb_default_width
      height_scaling = depth_image_size[1] / rgb_default_height
      rgb_intrinsic[0, :] *= width_scaling
      rgb_intrinsic[1, :] *= height_scaling
      rgb_distortion =
np.array(calibData.getDistortionCoefficients(dai.CameraBoardSocket.RGB))
      # RIGHT --> RGB CAMERA INFO
      right_to_rgb_extrinsic =
np.array(calibData.getCameraExtrinsics(dai.CameraBoardSocket.RIGHT,
dai.CameraBoardSocket.RGB))

      #cv2.namedWindow("Depth Image")

      if not no_pcl:
        # setup pcl converter

        pcl_converter = PointCloudVisualizer(rectified_right_intrinsic,
*depth_image_size)

      # setup bookkeeping variables
      pcl_frames = [None, None] # depth frame, color frame
      queue_list = [device.getOutputQueue(stream, maxSize=8, blocking=False)
for stream in streams]
      pixel_coords = utils.pixel_coord_np(*depth_image_size).astype(float)
      tiempo = 0
      while tiempo <= 300:
        sleep(10)
        for i, queue in enumerate(queue_list):
```

```
            name = queue.getName()
            image = queue.get()

            depth_frame = np.array(image.getFrame())
            h,w = depth_frame.shape
            pcl_frames[i] = cv2.resize(depth_frame, depth_image_size,
cv2.INTER_NEAREST)
          if any([frame is None for frame in pcl_frames]):
            print("Error: Need rgb AND depth frame to align!")
            continue
          elif pcl_frames[0] is None:
            print("Waiting on depth image to visualize")
            continue
          #quantized_depth_image = utils.quantizeDepthFrame(pcl_frames[0],
depth_scale_factor=5000)
          #cv2.imshow("Depth Image", pcl_frames[0])
          if not no_pcl:
            pcl_converter.depth_to_projection(pcl_frames[0], stride=stride,
downsample=downsample_pcl)
            #deptho3d = o3d.geometry.Image(pcl_frames[0])
            #f2= open("testo.txt","w+")
            #testo = np.asarray(deptho3d)
            #np.savetxt(f2,testo, fmt="%s")
            #f2.close()
            pcl_converter.visualize_pcd()
          tiempo = tiempo + 1
          if tiempo == 300:
            break
        pcl_converter.close_window()
        dlg = QMessageBox(self)
        dlg.setWindowTitle("Revisión")
        dlg.setText("Si el video está OK guarde la config y corra el programa
principal")
        button = dlg.exec()





#-*-*-*-*-*-*-*-*-*-*-*Función INICIAL *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*

  def __init__(self):
    super().__init__()
    self.w = None
    self.init_ui()

#-*-*-*-*-*-*-*-*-*-*-*Función principal de UI *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*
  def init_ui(self):

    layout = QVBoxLayout()
    layoutsliders = QVBoxLayout()
    #Creación de objetos para agregar a la pantalla principal con sus
atributos y funciones.
```

```
    #Creación de los parámetros de los filtros, se crean las barras y se les
da el formato y se agregan labels
    # para que se sepa que parámetro estamos modificando.

#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-Parámetro de confidence*-*-*-*-*-*-*-*-
*****-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    conflayout = QHBoxLayout()
    self.Confidence = QSlider(Qt.Horizontal)
    self.Confidence.setMinimum(1)
    self.Confidence.setMaximum(255)
    self.Confidence.valueChanged.connect(self.vConf_change)
    self.confvallbl = QLabel(self)
    self.confvallbl.setMinimumWidth(40)
    self.Confidence.valueChanged.connect(self.updateconfvalLabel)
    self.conflbl = QLabel(self)
    self.conflbl.setMinimumWidth(40)
    self.conflbl.setText("Confidence")
    conflayout.addWidget(self.Confidence)
    conflayout.addSpacing(15)
    conflayout.addWidget(self.confvallbl)
    conflayout.addSpacing(15)
    conflayout.addWidget(self.conflbl)
    self.setLayout(conflayout)

#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-Parámetro de Strade*-*-*-*-*-*-*-*-*****-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    stradlayout = QHBoxLayout()
    self.Strade = QSlider(Qt.Horizontal)
    self.Strade.setMinimum(1)
    self.Strade.setMaximum(10)
    self.Strade.valueChanged.connect(self.vStrad_change)
    self.stradvallbl = QLabel(self)
    self.stradvallbl.setMinimumWidth(40)
    self.Strade.valueChanged.connect(self.updatestradvalLabel)
    self.stradlbl = QLabel(self)
    self.stradlbl.setMinimumWidth(40)
    self.stradlbl.setText("Strade")
    stradlayout.addWidget(self.Strade)
    stradlayout.addSpacing(15)
    stradlayout.addWidget(self.stradvallbl)
    stradlayout.addSpacing(15)
    stradlayout.addWidget(self.stradlbl)
    self.setLayout(stradlayout)

#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-Parámetro de Profundidad máxima*-*-*-*-*-*-*-
*-*-*****-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    Maxlayout = QHBoxLayout()
    self.Max_depth = QSlider(Qt.Horizontal)
    self.Max_depth.setMinimum(200)
    self.Max_depth.setMaximum(2000)
    self.Max_depth.valueChanged.connect(self.vMaxd_change)
    self.maxdvallbl = QLabel(self)
    self.maxdvallbl.setMinimumWidth(40)
    self.Max_depth.valueChanged.connect(self.updatemaxdvalLabel)
    self.maxlbl = QLabel(self)
```

```
        self.maxlbl.setText("Max Depth")
        Maxlayout.addWidget(self.Max_depth)
        Maxlayout.addSpacing(15)
        Maxlayout.addWidget(self.maxdvallbl)
        Maxlayout.addSpacing(15)
        Maxlayout.addWidget(self.maxlbl)
        self.setLayout(Maxlayout)

    #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*---*-Parámetro de coprofundidad minima*-*-*-*-*-
    *-*-*-****-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

        Minlayout = QHBoxLayout()
        self.Min_depth = QSlider(Qt.Horizontal)
        self.Min_depth.setMinimum(50)
        self.Min_depth.setMaximum(600)
        self.Min_depth.valueChanged.connect(self.vMind_change)
        self.mindvallbl = QLabel(self)
        self.mindvallbl.setMinimumWidth(40)
        self.Min_depth.valueChanged.connect(self.updatemindvalLabel)
        self.minlbl = QLabel(self)
        self.minlbl.setText("Min Depth")
        Minlayout.addWidget(self.Min_depth)
        Minlayout.addSpacing(15)
        Minlayout.addWidget(self.mindvallbl)
        Minlayout.addSpacing(15)
        Minlayout.addWidget(self.minlbl)
        self.setLayout(Minlayout)

    #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*---*-Parámetro de Sigma*-*-*-*-*-*-*-*-****-*-*-
    *-*-*-*-*-*-*-*-*-*-*-*-*-*

        Sigmalayout = QHBoxLayout()
        self.Sigma = QSlider(Qt.Horizontal)
        self.Sigma.setMinimum(1)
        self.Sigma.setMaximum(255)
        self.Sigma.valueChanged.connect(self.vSig_change)
        self.sigmavallbl = QLabel(self)
        self.sigmavallbl.setMinimumWidth(40)
        self.Sigma.valueChanged.connect(self.updatesigmavalLabel)
        self.siglbl = QLabel(self)
        self.siglbl.setText("Sigma")
        Sigmalayout.addWidget(self.Sigma)
        Sigmalayout.addSpacing(15)
        Sigmalayout.addWidget(self.sigmavallbl)
        Sigmalayout.addSpacing(15)
        Sigmalayout.addWidget(self.siglbl)
        self.setLayout(Sigmalayout)


    #*-*-*-*-*-*-*-*-*-*-*-Botón de video para mostrar el video resultante
    después de la config

        self.Video = QPushButton("Play Video")
        self.Video.setCheckable(False)
        self.Video.released.connect(lambda: self.VideoPlay())
```

```
#*-*-*-*-*-*-*-*-*-*-*-Botón de video para detener el video

    self.Videostop = QPushButton("Stop Video")
    self.Videostop.setCheckable(False)
    self.Videostop.released.connect(lambda: self.StopVideo())

#*-*-*-*-*-*-*-*-*-*-*-Botón de video para mostrar el video resultante
después de la config

    self.Leer = QPushButton("Leer Datos",self)
    self.Leer.setCheckable(False)
    self.Leer.released.connect(lambda: self.Read())


#-*-*-*-*-*-*-*-*-*-*-Botón de guardado para alamacenar los datos de las
barras

    self.Guardar = QPushButton("Guardar",self)
    self.Guardar.setCheckable(False)
    self.Guardar.released.connect(lambda: self.Saved(datos))


    #Agragado de cada uno de los objetos a la pantalla principal.

    self.botones = [self.Video,self.Videostop,self.Guardar,self.Leer]

    for w in self.botones:
      layout.addWidget(w)

    self.sliders = [conflayout,stradlayout, Maxlayout,Minlayout,Sigmalayout]

    for z in self.sliders:
      layoutsliders.addLayout(z)


    layout.addLayout(layoutsliders)

    self.widget = QWidget()
    self.widget.setLayout(layout)
    self.setCentralWidget(self.widget)
    self.setGeometry(300, 300, 350, 250)
    self.setWindowTitle("Ajuste")

 #-*-*-*-*-*-*-*-*-*- Funciones de apoyo para los objetos -*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*-*-*-*-.

  def vConf_change (self):
    datos[0] = str(self.Confidence.value())
  def vStrad_change (self):
    datos[1] = str(self.Strade.value())
  def vMaxd_change (self):
    datos[2] = str(self.Max_depth.value())
  def vMind_change (self):
    datos[3] = str(self.Min_depth.value())
  def vSig_change (self):
    datos[4] = str(self.Sigma.value())
```

```
  def updateconfvalLabel(self, value):
    self.confvallbl.setText(str(value))
  def updatestradvalLabel(self, value):
    self.stradvallbl.setText(str(value))
  def updatemaxdvalLabel(self, value):
    self.maxdvallbl.setText(str(value))
  def updatemindvalLabel(self, value):
    self.mindvallbl.setText(str(value))
  def updatesigmavalLabel(self, value):
    self.sigmavallbl.setText(str(value))


  def Saved (self, valores):
    DF = valores
    self.Almacenar_Config(DF[0],DF[1],DF[2],DF[3],DF[4])

  def Read (self):
    datos = self.Leer_Config()
    self.Confidence.setValue(int(datos[0]))
    self.Strade.setValue(int(datos[1]))
    self.Max_depth.setValue(int(datos[2]))
    self.Min_depth.setValue(int(datos[3]))
    self.Sigma.setValue(int(datos[4]))
  def VideoPlay(self):
    self._run_flag = True
    self.Videoplayer()



# Aquí seejecutan las funciones de inicio de programa donde se llaman a las
subfucniones-*-*-*-*-*-*-*
app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```

Hoja Guarda