DESARROLLO E IMPLEMENTACIÓN DE UN MÓDULO PARA LA GENERACIÓN DE IMÁGENES DE RANGO Y DE TEXTURA DE CAMPO AMPLIO MEDIANTE PROCESAMIENTO DE IMÁGENES DE MICROSCOPÍA DE ALTA RESOLUCIÓN

MICHAEL ANDRÉS ÁLVAREZ NAVARRO
JOHNNY ARIEL ACUÑA LAMUS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

DESARROLLO E IMPLEMENTACIÓN DE UN MÓDULO PARA LA GENERACIÓN DE IMÁGENES DE RANGO Y DE TEXTURA DE CAMPO AMPLIO MEDIANTE PROCESAMIENTO DE IMÁGENES DE MICROSCOPÍA DE ALTA RESOLUCIÓN

MICHAEL ANDRÉS ÁLVAREZ NAVARRO
JOHNNY ARIEL ACUÑA LAMUS

Trabajo de Grado para optar al título de Ingeniero Electrónico e Ingeniero de Sistemas e Informática, respectivamente.

Director
MSc. LOLA XIOMARA BAUTISTA ROZO

Codirector
Ph.D. ARTURO PLATA GÓMEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

AGRADECIMIENTOS

A Dios, a mis padres Carlos Álvarez y Aide Navarro por sus consejos y guía durante todos estos años, a mi hermana María Alejandra, a Andrea Angarita por su apoyo y acompañamiento durante el desarrollo del presente trabajo de grado.

A la profesora Lola Xiomara Bautista y al profesor Arturo Plata Gómez por su dirección y asesoría.

A la Escuela de Eléctrica y Electrónica por nutrirme de conocimientos durante el tiempo de estudiante, y a la Escuela de Matemáticas por permitirme trabajar con ellos en los laboratorios de cómputo

A todos mis amigos: David Chacón, David Abreo, Diego Coba, Tama, Yohana Galvis, Paula Pedroza, Nelson Rueda, Fabio Ruiz, el grupo de auxiliares de matemáticas y todos los demás que no alcanzo a nombrar en estos tres reglones.

¡¡¡Gracias!!!

Michael Andrés Álvarez Navarro

AGRADECIMIENTOS

A Dios por todas aquellas personas y eventos que puso en mi camino y que dejaron en mi algo para crecer como persona.

A mis padres, Manuel Acuña Ramírez y Gilma Bertha Lamus Villarreal por todo su apoyo y ayuda en este largo proceso de formación, y sus consejos para la vida. A mi hermano, Helmer Omar Acuña Lamus por aguantarme todas las rabietas por estrés por realización de trabajos de la U, colaboración en los oficios de la casa y compartir ratos de esparcimiento. Y a Leydi Johana Arenales Mancilla por su acompañamiento, comprensión y cariño que me dio en más de una ocasión fuerzas cuando las necesitaba.

A la profesora Lola Xiomara Bautista Rozo por aceptar ser mi directora de trabajo de grado y su inmensa paciencia, además de su orientación en el desarrollo del mismo. Al profesor Arturo Plata Gómez, por compartir sus conocimientos y asesorarnos en el desarrollo de este trabajo y por mostrarme muchas de las maravillas que se pueden hacer con procesamiento de imágenes.

A la escuela de Ingeniería de Sistemas e Informática, por todos sus profesores que me compartieron sus conocimientos teóricos y experiencias para la vida que me llevo conmigo. A la secretaría de la escuela por la ayuda ofrecida para resolver los a los problemas que tuve.

A todos mis amigos y compañeros de las escuelas de sistemas, matemáticas y física, con quienes compartí clases y momentos de ocio, además de aprender cosas de la vida con ellos.

A todos ellos, ¡¡¡Muchas, pero muchas Gracias!!!

Johnny Ariel Acuña Lamus

TABLA DE CONTENIDO

pág.

INDICE DE FIGURAS

pág.

RESUMEN

TITULO: DESARROLLO E IMPLEMENTACIÓN DE UN MÓDULO PARA LA GENERACIÓN DE IMÁGENES DE RANGO Y DE TEXTURA DE CAMPO AMPLIO MEDIANTE PROCESAMIENTO DE IMÁGENES DE MICROSCOPÍA DE ALTA RESOLUCIÓN.*

AUTORES:

MICHAEL ANDRÉS ÁLVAREZ NAVARRO**

JOHNNY ARIEL ACUÑA LAMUS***

PALABRAS CLAVE: Microscopia Digital, Imágenes de Rango, Imagen de Textura, Profundidad de Campo Amplio, Mosaico de Imágenes, Imágenes de Campo Amplio.

DESCRIPCIÓN:

El estudio de materiales inorgánicos y tejidos orgánicos realizado por medio de imágenes digitales tomadas por un microscopio óptico automatizado, presenta retos debido a las limitantes de carácter físico y tecnológico. Superar estos límites requiere del uso de técnicas como el procesamiento digital de imágenes y de ingeniería de software.

El objeto de estudio de este proyecto se enfoca a superar uno de estos retos: Generación de imágenes de mosaico, dados un conjunto de imágenes de profundidad de campo extendido (y opcionalmente otro conjunto de imágenes de rango) que han sido tomados con un microscopio óptico automatizado de alta resolución.

La investigación realizada conlleva al diseño y desarrollo de un algoritmo generador de imágenes de mosaico, el cual hace uso de varias técnicas de procesamiento de imágenes: extracción y comparación de características, rotación, traslación y proyecciones geométricas, entre otras. Además, se hace un estudio del proceso de toma de imágenes, de la plataforma tecnológica ofrecida por el software usado en el laboratorio para el control y manejo del microscopio, y de paquetes de software disponibles para el procesamiento digital de imágenes y programación de algoritmos, con el fin de conocer las posibles estrategias de implementación a usar y seleccionar una para tener el algoritmo desarrollado en un ambiente productivo, usable para los investigadores.

* Trabajo de grado

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Lola Xiomara Bautiista Rozo. Codirector: Arturo Plata Gómez

*** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías de Sistemas e Informática. Director: Lola Xiomara Bautiista Rozo. Codirector: Arturo Plata Gómez

ABSTRACT

TITLE: DEVELOPMENT AND IMPLEMENTATION OF A MODULE FOR THE GENERATION OF RANGE AND TEXTURE LARGE FIELD IMAGES BY HIGH-RESOLUTION MICROSCOPY IMAGE PROCESSING.[*]

AUTHORS:

MICHAEL ANDRÉS ÁLVAREZ NAVARRO[**]

JOHNNY ARIEL ACUÑA LAMUS[***]

KEYWORDS: Digital Microscopy, Range Images, Texture Image, Extended Depth of Field , Image Mosaic, Large Field Image

DESCRIPTION:

The study of inorganic materials and organic tissues made by means of digital images taken by an automated optical microscope, presents challenges due to limitations of physical and technological nature. Overcoming these limitations requires the use of techniques such as digital image processing and software engineering.

The object of study of this project focuses on the overcoming of one of such challenges: Mosaic Imaging generation given a set of extended depth field images (and optionally another set of range images) that have been taken with a high resolution automated optical microscope.

The research done leads to the design and development of an algorithm capable of generating mosaic images, which makes use of several techniques of image processing: feature acquisition and comparison, rotation, translation and geometric projections, among others. Additionaly, a study of the process of taking the images of the sample, the technological platform offered by the software used in the laboratory for the control and use of the microscope, and software packages available for the digital image processing techiniques and for algorithm programming is done in order to know the available implementation strategies which can be used and select one for making the developed algorithm to be in a production environment, available to the researchers.

[*] Bachelor Thesis

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Lola Xiomara Bautiista Rozo. Codirector: Arturo Plata Gómez

[***] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías de Sistemas e Informática. Director: Lola Xiomara Bautiista Rozo. Codirector: Arturo Plata Gómez

INTRODUCCIÓN

El presente trabajo de grado forma parte de una secuencia de proyectos realizados por el grupo de óptica y tratamiento de señales GOTS, en el cual se busca desarrollar e implementar un algoritmo para la generación de imágenes de rango amplio en microscopia. Las imágenes tomadas con el microscopio óptico sólo capturan una parte de la muestra y para apreciar toda la escena es necesario mover el área observada, por ello son importantes las imágenes de rango amplio ya que permiten visualizar los detalles de toda la muestra.

Aplicando técnicas de ingeniería de software, es posible desarrollar un algoritmo que permita fusionar imágenes de una misma muestra para obtener una imagen de rango amplio, que también se puede implementar como un módulo en el software que usa el microscopio del laboratorio de investigación en óptica y tratamiento de señales de la Escuela de Física de la Universidad Industrial de Santander.

Con este módulo se busca superar las limitantes del módulo actual que está en uso y que asista al investigador en la toma de imágenes de las muestras para así generar imágenes de rango amplio pertinentes para el investigador.

Las tres primeras secciones del presente trabajo tratan de las generalidades del proyecto: Objetivos, descripción del problema y alcance. La cuarta sección trata del marco de referencia donde está expuesto los detalles del microscopio y la teoría básica del procesamiento de imágenes. La quinta sección es la explicación de la metodología usada para el desarrollo de este trabajo. La sexta sección lista con una breve descripción las herramientas computacionales usadas. La séptima sección describe el algoritmo desarrollado además de explicar cómo se realiza su implementación. La octava sección contiene pruebas de tiempo realizadas al algoritmo desarrollado. Las últimas secciones son conclusiones, recomendaciones, bibliografía y anexos.

1. OBJETIVOS

1.1 Objetivo General

Desarrollo e implementación de un algoritmo para la generación de imágenes de rango y de textura, de campo amplio, mediante procesamiento de imágenes de microscopia de alta resolución.

1.2 Objetivos Específicos

- Diseñar un algoritmo para la generación de imágenes de campo amplio con nivelación de textura, empleando técnicas de procesamiento digital de imágenes.
- Desarrollar un módulo para el procesamiento y generación de imágenes en mosaico con nivelación de textura para un microscopio óptico de alta resolución.
- Implementar el módulo para la generación de imágenes en mosaico en el sistema de control de un microscopio automatizado xyz.

2. DESCRIPCIÓN DEL PROBLEMA

Uno de los microscopios en el laboratorio GOTS de la sede de Guatiguará cuenta con su respectivo software que permite cierto procesamiento de las imágenes tomadas a través de diversos módulos que se pueden comprar para el software o se pueden programar en el Visual Basic Application incorporado.

Uno de los módulos que tienen es el de generación de imágenes en mosaico el cual actualmente no es capaz de producir imágenes de la calidad requerida por las siguientes razones: La imagen resultante no es continua, es decir, se nota donde se unieron las imágenes que forman el mosaico, el módulo de auto enfocado no funciona con el hardware actual y por lo tanto las imágenes individuales no quedan correctamente enfocadas, y, no fusiona imágenes con información de profundidad.

El módulo de auto enfocado sería reemplazado por uno que realiza captura de imágenes de rango de profundidad, por lo tanto el módulo de generación de imágenes en mosaico a realizar debe usar el módulo anteriormente mencionado para obtener las imágenes básicas y fusionar no sólo las imágenes de textura sino también unir la información de profundidad.

3. ALCANCE DEL PROYECTO

El desarrollo del algoritmo contempla la creación de un módulo de software que permita su uso particularmente en el laboratorio del GOTS que se encuentra en el Parque Tecnológico de Guatiguará, donde el computador de trabajo corre el sistema operativo de Windows XP.

4. MARCO DE REFERENCIA

4.1 Microscopia Digital

4.1.1 Sistema de Adquisición: El Grupo de Investigación en Óptica y Tratamiento de Señales (GOTS) cuenta con un laboratorio en la sede de Guatiguará de la UIS, en el cual dentro de su dotación de equipos, cuenta con el microscopio Axio Imager.Z1m de la casa fabricante Carl Zeiss (figura 1). El sistema óptico cuenta con objetivos tipo Epiplan-Neofluar con aumentos de 5X – 20X – 50X – 100X, módulo de campo claro, C-DIC (*circular-diferencial interferometer contrast*) y TIC (*total interferometer contrast*). La tabla 1 muestra las características de los objetivos.

Figura 1. Microscopio Imager.Z1m

Fuente: Carl Seizz Microscopy GmbH. Axio Imager 2. [PDF, online p 26]. Alemania. http://applications.zeiss.com/C125792900358A3F/0/1BF248DEEBCE457FC12579060047C0D4/$FILE/70-2-0020_e.pdf

Tabla 1. Características de los Objetivos del Microscopio Imager.Z1m

OBJETIVO	5X	20X	50X	100X
APERTURA NUMÉRICA	0,13	0,5	0,8	0,9
RESOLUCIÓN LATERAL	2,34 µm	0,61 µm	0,38 µm	0,33 µm
RESOLUCIÓN AXIAL	29,58 µm	2 µm	0,78 µm	0,61 µm

Fuente: CHACON, Carlos. Reconstrucción Tridimensional por Microscopía de Contraste Diferencial con Polarización Circular. Bucaramanga 2013. Trabajo de Grado (Maestría en Física). Universidad Industrial de Santander. Facultad de Ciencias. P 21.

Las características mecánicas más importantes radican en los desplazamientos que el microscopio Imager.Z1m puede realizar, gracias a que cuenta con una platina con actuadores piezoeléctricos tanto a lo largo del eje óptico (dirección Z) como en los ejes transversales $X-Y$. El máximo desplazamiento $X-Y$ de la platina es $85mm$ y $130mm$ respectivamente, con paso de $200nm$ y desplazamiento en Z de $120mm$ a pasos de $10nm$. En la figura 2 se muestra el sistema mecánico del microscopio Imager.Z1m.

4.1.2 Pre-procesamiento de imágenes: Una imagen es un conjunto de matrices bidimensionales donde cada matriz representa una capa de color o la intensidad de color en un punto del objeto. El color que representa cada matriz depende del modelo de color que se esté utilizando. El modelo de color RGB (Red, Green, Blue) es la composición del color en términos de la intensidad de los colores primarios de la luz.

El tratamiento digital de imágenes para la creación de mosaicos obedece a diferentes técnicas, que no solo consisten en la comparación de pixeles. Es necesario implementar algoritmos de reconocimiento de patrones que utilizan

transformadas que permitan superponer la imagen generando la sensación de continuidad.

Figura 2. Sistema Mecánico Imager.Z1m

Fuente: Carl Seizz Microscopy GmbH. Op. cit., p 22.

4.2 Algoritmo SIFT (Lowe, 2004) [1]

SIFT es el acrónimo de Scale Invariant Feature Transformation, es un algoritmo publicado por David Lowe en 1999, después de esta fecha se han realizado varias modificaciones e implementaciones. Este algoritmo es una transformada que se encarga de extraer características distintivas de las imágenes invariantes a cambios de escala, es decir que puede reconocer la misma característica en diferentes vistas de un objeto.

El algoritmo se realiza mediante 4 pasos: Construcción de Pirámides de Scale-Space, Localización de puntos claves, Asignación de orientación y Descriptor de Puntos claves.

[1] LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints [Online]. *Int. J. Comput. Vision*, nov 2004, p. 91-110. <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>

4.3 Homografía

"Las imágenes de puntos 3D sobre un plano están relacionadas por una homografía, por tanto se dice que un plano induce una homografía entre las vistas"[2]. En el caso de la composición de imágenes lo que se busca es encontrar una transformación que convierta la zona de similitud en una homografía entre las imágenes.

4.4 RANSAC[3]

El método iterativo de consenso de muestras aleatorias, o RANSAC por sus siglas en inglés, identifica y retira los valores atípicos de un conjunto de datos y estima un modelo matemático deseado usando los datos del subconjunto resultante.

Esta característica de identificar datos atípicos y retirarlos del conjunto de datos, permite hallar comunalidad entre dos conjuntos de datos. En el caso de procesamiento digital de imágenes, con RANSAC encontramos los puntos en común de dos imágenes distintas.

El método RANSAC consiste en los siguientes pasos:

1. Seleccionar aleatoriamente un subconjunto del conjunto de datos.

2. Ajustar el subconjunto seleccionado a un modelo.

3. Determinar el número de datos atípicos.

4. Repetir los pasos del 1 al 3 por un número determinado de iteraciones.

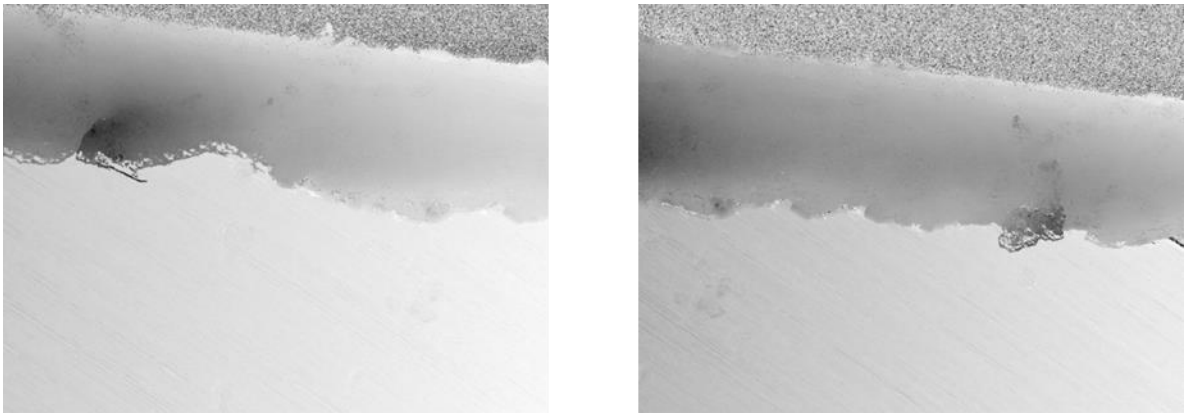
[2] Instituto de Ingeniería Eléctrica [PDF, online]. Uruguay, Montevideo, Universidad De La República. 2002. Estimación de la Geometría en Visión por Computador. p1. Disponible en: http://iie.fing.edu.uy/investigacion/grupos/gti/cursos/egvc/material/tema-7.pdf

[3] MathWorks [online]. Using RANSAC for estimating geometric transforms in computer vision. Disponible en: http://www.mathworks.com/discovery/ransac.html

4.5 Imagen de Rango

La información de la posición de focalización, para cada pixel a medida que se avanza bajo las capas de profundidad generan la Imagen de Rango[4]. La figura 3 muestra imágenes de rango tomadas a una pieza de metal.

Figura 3. Imágenes de Rango de una muestra de metal.



4.6 Imagen de Textura

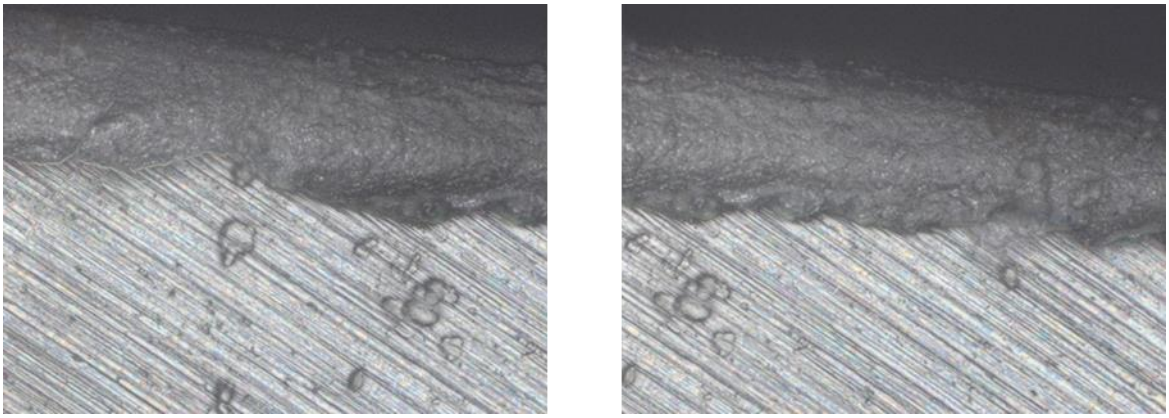
Las imágenes de Textura son producto de la reflectividad del material, más la topografía y la iluminación, donde el sistema óptico también determina la profundidad de campo y la resolución de la imagen.

El concepto detrás de estas imágenes de textura es el de imágenes de campo de profundidad extendido o *extended depth of field (DOF)*. En la microscopia, el detalle de los objetos a estudiar es más grueso que el DOF del microscopio, de ahí la importancia de expandir el DOF y así poder armar las imágenes de textura, de otro modo, sólo se podría focalizar una parte del objeto de estudio y no se podría apreciar completamente. La figura 4 muestra las correspondientes

[4] OSABE, Taito. et al., "A study on joint progressive coding of range data and texture images." [online]. Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on, vol.2.

Imágenes de Textura para la muestra de metal de la cual se obtuvieron las Imágenes de Rango.

Figura 4. Imagenes de Textura correspondientes a la muestra de metal.



Las figuras 5 y 6 corresponden a los mosaicos de cada par de imágenes; las de Rango y las de Textura.

Figura 5. Mosaico Imágenes de Rango.

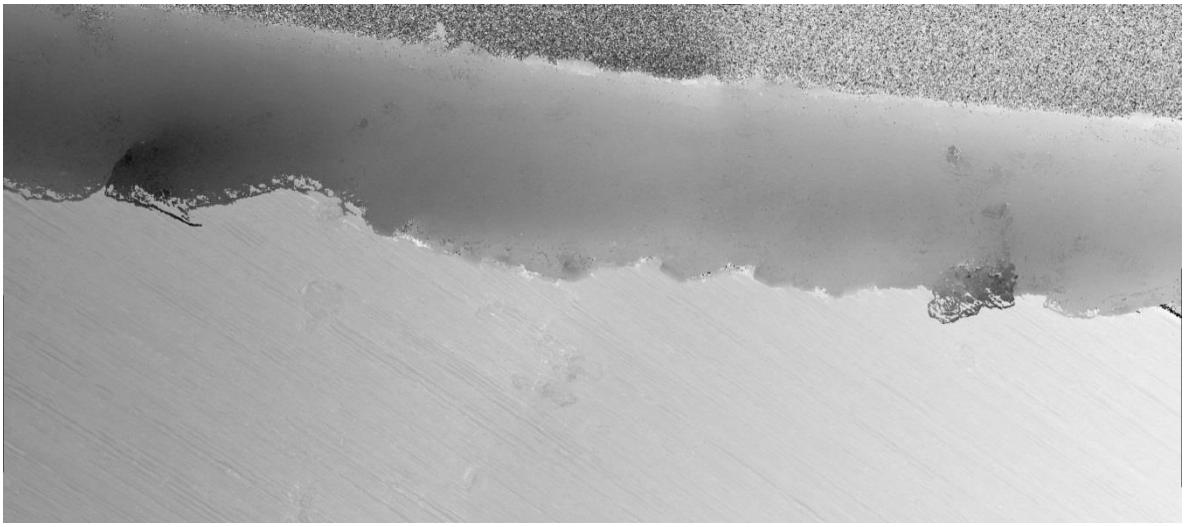
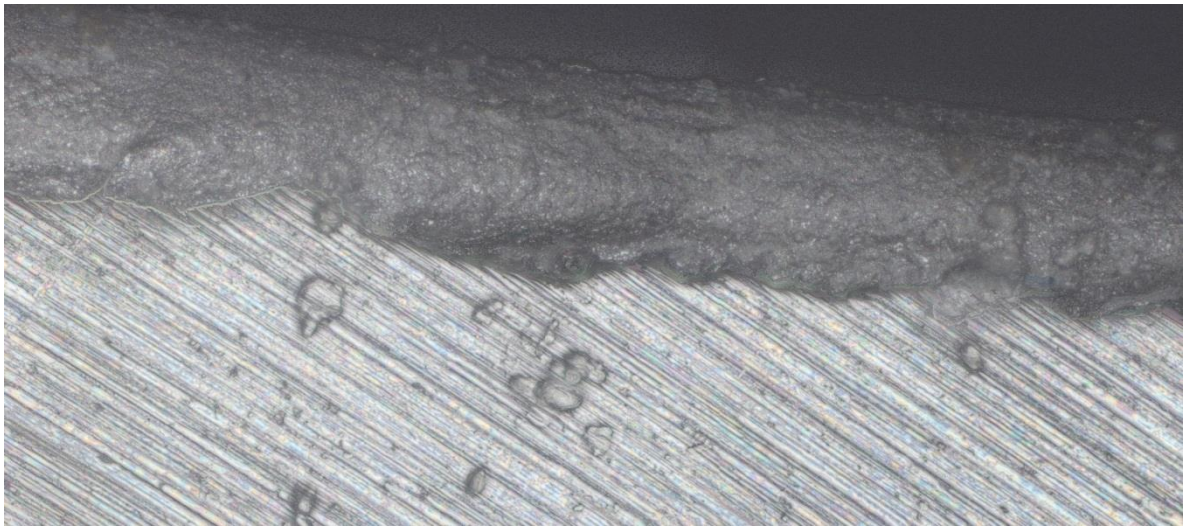
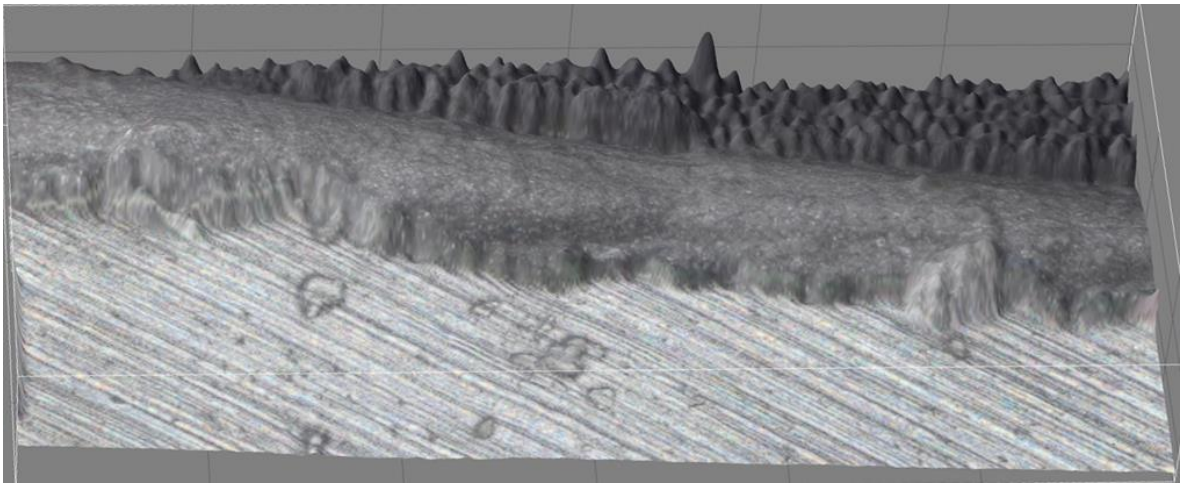


Figura 6. Mosaico de Imágenes de Textura.



Con las imágenes de rango y profundidad de campo extendido, se puede hacer una recreación tridimensional de la muestra que permite estudiar con detalle el material o tejido de una forma que no se puede con el ojo humano. Se presenta en la figura 7 la representación tridimensional de dicha muestra.

Figura 7. Representacion 3D del Mosaico.



4.7 Noción de algoritmo

La parte más importante en este trabajo de grado es la realización de un algoritmo para la generación de imágenes de campo amplio, o también referidas aquí como imágenes en mosaico. Por lo tanto es importante repasar la noción de lo que es un algoritmo. "*La noción de algoritmo es básica a toda programación de computadores*" Es una de las citas que hace Fant sobre el tema.

Una definición típica de algoritmo es la siguiente:

- Un algoritmo debe ser una secuencia de operaciones paso a paso.
- Cada operación debe ser definida con precisión.
- Un algoritmo debe terminar en un número finito de pasos.
- Un algoritmo debe dar efectivamente una solución correcta.
- Un algoritmo debe ser determinista en que dada las mismas entradas, siempre producirá los mismos resultados.

Esta típica definición, como expone Fant, no se aplica del todo para los campos de las ciencias de la computación y explica: "*Cualquier programa con un 'bug' no puede ser un algoritmo pero es generalmente aceptado que ningún programa significativo puede demostrarse que es libre de 'bugs'*". Por lo que se dice que todo programa de computadora es como mínimo un semi-algoritmo, esto es, cumple con casi toda la definición de lo que es un algoritmo. En este trabajo se seguirá la noción de semi-algoritmo, basándose en que no hay forma de asegurar que será libre de bugs. Adicionalmente, siguiendo la definición de Haberman et al[5], Definimos a un algoritmo como:

[5] HABERMAN, Bruria, et al. Is it really an algorithm: the need for explicit discourse? [online]. Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, New York, NY, USA, 2005, pp. 74– 78.

“Es la solución a un problema de carácter computacional, que cumple correctamente con las precondiciones y post-condiciones especificada por el problema y que es conciso y eficiente.”

4.8 Modelo de objetos componentes (COM)

El modelo de objetos componentes, o mejor conocido por sus siglas en inglés COM (Component Object Model) es un estándar que especifica un modelo de objetos y requerimientos de programación para permitir la comunicación entre objetos. COM es un estándar que se aplica después que un programa está en código máquina, por lo cual decimos que es un estándar binario.

Dichos objetos pueden estar en el mismo o diferentes procesos de la misma máquina local o entre máquinas remotas. Estos objetos no sólo pueden presentar estructuras muy diferentes, sino que también pueden estar escritos en diferentes lenguajes de programación.

Un objeto de software es un conjunto de datos y funciones para manipular estos datos. Estas funciones son llamadas métodos, sólo mediante una o más de ellas se puede acceder a los datos del objeto COM.

Una interfaz es un conjunto de estos métodos, y la única forma de acceder a ellos es mediante un puntero a memoria de esta interfaz. Por esta razón, los requerimientos del lenguaje de programación para el estándar COM, es que se puedan usar estructuras de punteros a memoria y llamar funciones a través de punteros. Lenguajes como C, C++, Java y VBScript, entre otros, pueden ser usados para crear objetos COM, y en particular C++ simplifica su implementación.

El uso de componentes COM permite extender la funcionalidad de programas sin estar atados al lenguaje de programación nativo de la plataforma. Por el ejemplo, el software de control del microscopio permite extender su funcionalidad mediante Visual Basic Application (abreviado VBA), que como su nombre lo indica, las

sintaxis del lenguaje es aquella de Visual Basic, pero con COM, podemos escribir módulos en lenguajes diferentes.

Ventajas adicionales incluyen la extensión de librerías de código, y de superación de límites del lenguaje que se esté usando. Siguiendo con el ejemplo de visual basic, eso quiere decir que nuevos módulos pueden hacer uso de librerías de C++ o Matlab, aparte de las existentes para VBA, además se puede hacer uso de instrucciones específicas de los procesadores centrales o gráficos para aumentar la velocidad de los algoritmos.

5. Metodología

Las metodología usada para el desarrollo del módulo es una combinación, tomando ciertas libertades, de las metodologías Investigación-Acción Participativa y el modelo de proceso adaptativo para el desarrollo ágil expuesto por Rizwan y Hussain en el artículo "*An adaptive software development process model*". Aunque al principio se tenía una concepción para aplicar esta metodología, la realidad del trabajo conllevó a tomar caminos diferentes a los planeados inicialmente.

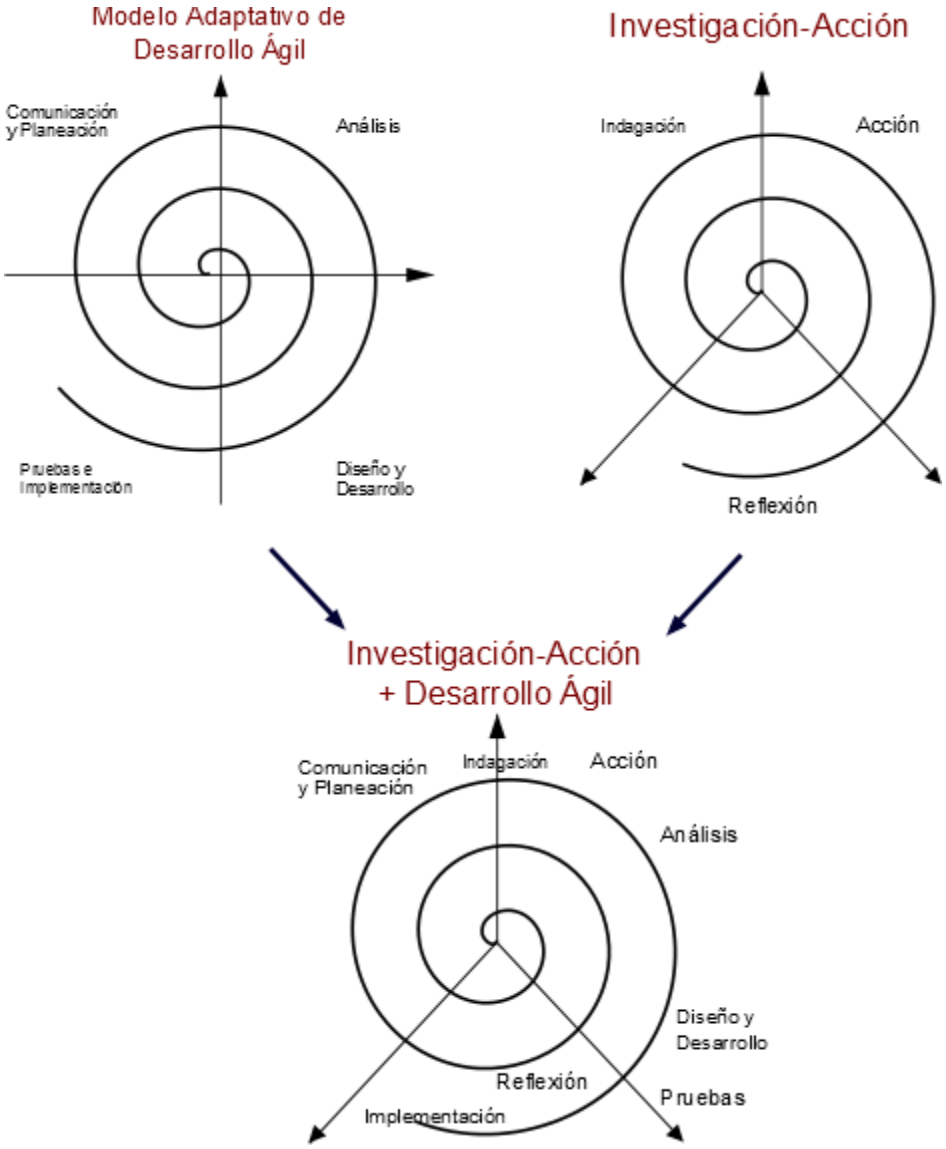
Al graficar ambas metodologías en relación al tiempo, obtenemos gráficos en espiral por lo que unirlas en una sola es directo. En las siguientes subsecciones se explica con detalle en que consiste cada paso de las metodologías combinadas. El resultado de ilustra en la figura 8.

5.1 Comunicación y Planeación

En esta fase se realiza un estudio del problema. De este se extraen los requerimientos mínimos funcionales del aplicativo a desarrollar, se traza un plan de acción y se indaga a más profundidad sobre el problema.

- ✓ *Comunicación*: Para este caso en particular, los desarrolladores se ponen en contacto con los investigadores del Laboratorio de óptica de física, quienes tras charlas casuales y entrevistas informales, darán a conocer el problema que tienen entre manos y que los desarrolladores pueden dar solución. Se consolida una primera idea general del problema a resolver.
- ✓ *Planeación*: El problema se descompone en tareas más pequeñas. Se les asigna un tiempo para cumplir con estas. La planeación es flexible y tiene que responder y acomodarse a los avances que se tienen en el desarrollo, los nuevos requerimientos mínimos funcionales que se van descubrimientos y las limitantes que tras prueba y error se van encontrando en el camino.

Figura 8. Combinación de metodologías de desarrollo y de investigación.



Fuente: Basada en el gráfico de Investigación-Acción de Rizwain y Hussain.

- ✓ *Indagación*: En esta etapa se realizan las investigaciones teóricas pertinentes para buscar una solución al problema, además de elegir las tecnologías apropiadas según el conocimiento de los desarrolladores y plataformas de software disponibles. Un criterio importante para la selección de tecnologías a usar es que tenga buena documentación y tenga trayectoria.

5.2 Acción

Esta fase de la metodología concentra gran parte del trabajo de campo a realizar para la solución del problema y satisfacción de los objetivos. Se compone de las siguientes etapas:

- ✓ *Análisis*: Se analizan las entradas y salidas de los algoritmos estudiados en la etapa de indagación, así como también del proceso que se busca automatizar. Esto permite conocer qué parámetros interactúan con el usuario y otro sistema, y cuáles parámetros no se deben tocar con el fin de reducir la probabilidad de errores y con el ánimo de automatizar las tareas. El resultado de este análisis termina en un bosquejo preliminar de lo que se espera que pueda hacer el aplicativo y de las posibles estrategias de implementación.
- ✓ *Diseño*: Partiendo del bosquejo preliminar del aplicativo, se realizan los diseños de arquitectura, de clases y demás que se consideren necesarios. Se usa UML para los diagramas de diseños. En esta etapa también el bosquejo preliminar se convierte en un diseño preliminar de interfaz.
- ✓ *Desarrollo*: Con los diseños realizados se prosigue a implementarlos de manera funcional. Para evitar hacer más tediosa la etapa de documentación, se sugiere comentar todo código escrito y hacer uso de una sintaxis de comentarios de documentación, como los usados por javadocs o doxygen.
- ✓ *Documentación*: En esta etapa, se verifica la documentación de lo desarrollado hasta el momento. Todo lo que no esté bien documentado se terminará de documentar adecuadamente. Se sugiere que la documentación cumpla con los siguientes criterios:
  - Dividido en áreas temáticas.
  - Describe la API desarrollada.
  - Presenta ejemplo de uso de la API desarrollada.

5.3 Reflexión

En esta fase se realizan las pruebas del software y con base en ello se reflexiona sobre los resultados obtenidos con el fin de decidir si se han cumplido con los objetivos trazados o se retorna a la fase de planeación, tras describir los nuevos problemas que se hayan encontrado.

- ✓ *Pruebas*: Se realizan pruebas para medir la efectividad del aplicativo con otro de referencia si este existe. Si no hay aplicativo de referencia, se consideran las expectativas del usuario como criterios de evaluación de desempeño. Se tabulan los resultados obtenidos.
- ✓ *Reflexión*: Se estudian los resultados obtenidos con el fin de determinar el cumplimiento de los objetivos trazados. Si se considera que no se han cumplido uno o más objetivos, se debe identificar la razón de ello como un problema a resolver y se inicia un nuevo ciclo de la metodología.
- ✓ *Puesta en marcha*: En esta etapa se instala y/o actualiza el software o módulo desarrollado en el ciclo listo para ser usado de forma productiva por la entidad que lo pidió.

5.4 Entrega

No necesariamente una fase de la metodología, sin embargo es un procedimiento formal donde se entrega a la entidad interesada los aplicativos desarrollados que están listos para ser usados de forma productiva. Consistente en obtener un visto bueno de que lo desarrollado es lo que se pidió y se ha dejado funcionando.

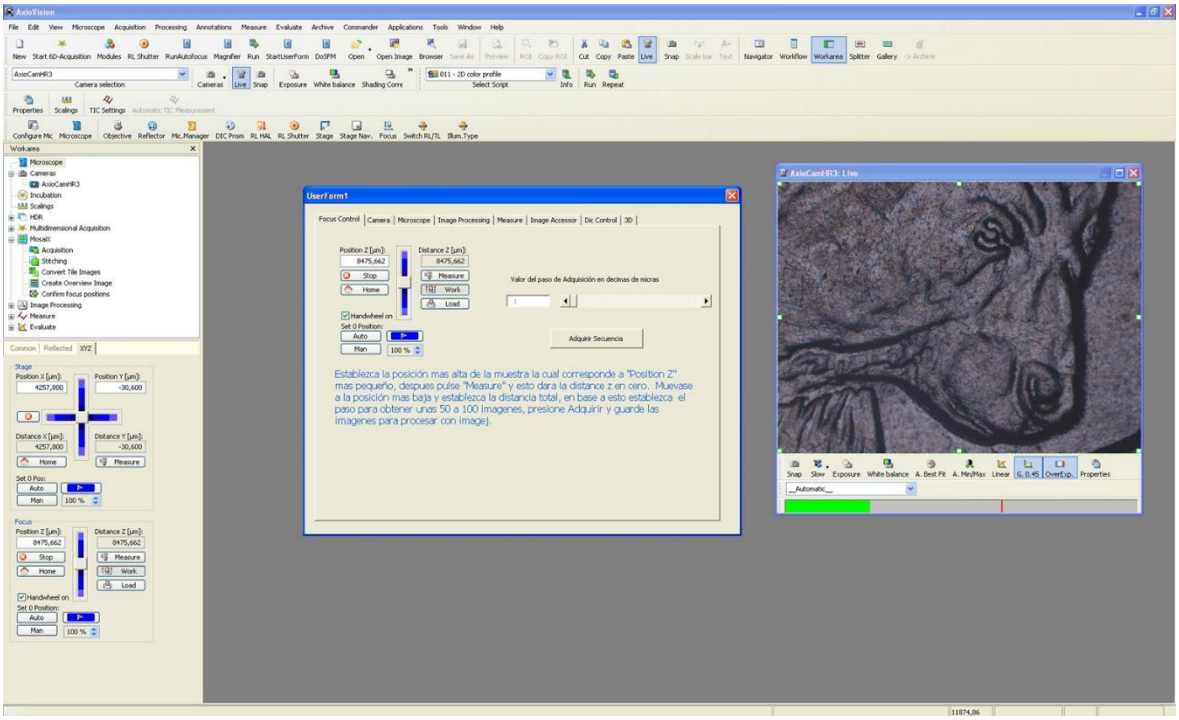
Si se han cumplido con todos los ciclos planeados, futuros ciclos necesarios de investigación y desarrollo serán necesarios para el mantenimiento del aplicativo, o para agregar más características o mejoras.

6. Herramientas computacionales utilizadas

6.1 AxioVision

AxioVision es la plataforma que viene incorporada con el microscopio, este software viene diseñado por módulos con funciones específicas como Adquisición de Imágenes, Archivado imágenes, Visualización en tiempo real y Balance de Blancos entre otras. Es totalmente compatible con el hardware del microscopio Carl Zeiss y posee un módulo de Visual Basic Aplication para extensiones de funcionalidad, macros y proyectos que el usuario de implementar en el sistema. La interfaz principal de AxioVision se presenta en la figura 9.

Figura 9. Interfaz de AxioVision.



6.2 Visual Basic Application

Visual Basic Application (VBA) era la tecnología ofrecida por Microsoft para ser incrustada en aplicaciones que actuarían como host (o máquina virtual) que ejecuta el código compilado de VBA. AxioVision cuenta con una versión de VBA para programarle módulos adicionales y extender su funcionalidad.

Cabe destacar que esta tecnología hoy en día sólo es usada por la suite ofimática de Microsoft, ya que VBA ha sido reemplazado por Visual Studio Tools for Applications (VSTA).[6]

6.3 Matlab

MATLAB ® es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, visualización y programación.[7] Dentro de la interfaz de MATLAB se pueden crear modelos y desarrollar algoritmos con herramientas y funciones ya implementadas de tal forma que es más agradable para el usuario la programación en este entorno, obteniendo la solución en tiempos comparables a otros lenguajes como C / C + + o Java ™.

6.4 Matlab NE Builder

Es un módulo de Matlab que encapsula un proyecto en un componente .NET o COM. El componente actúa como cualquier componente nativo de lenguajes .NET y además es distribuido con el Matlab Compiler Runtime (MCR) que permite tener un mínimo ambiente para ejecutar programas de Matlab en equipos que no lo tienen instalado.[8]

---

[6] OPPENHEIMER, Diego. Office Blogs [online] [citado el 15 de julio de 2014]. Clarification on VBA Support. Disponible en <http://blogs.office.com/2008/01/17/clarification-on-vba-support/>

[7] MathWorks [online]. MATLAB the Lenguage of Technical Computing. Product Overview. Disponible en <http://www.mathworks.com/products/matlab/>

[8] MathWorks [online]. MATLAB Builder NE: for Microsoft .NET framework. Overview. Disponible en <http://www.mathworks.com/products/netbuilder/>

NE Builder es usado para implementar el algoritmo generador de mosaicos usado en este proyecto. Cabe destacar que requiere de una instalación de visual c++ para que compile el proyecto en un componente COM.

6.5 OpenCV

OpenCV es una librería de C++ para computación visual. Tiene interfaces para otros lenguajes como Python y Java. Es particularmente usado en este proyecto por el "cosedor" de imágenes (image stitching) el cual une imágenes para crear una panorámica o imagen de rango amplio. Se usa como referencia para comparar con el algoritmo desarrollado.

Es mucho más efectivo el OpenCV Stitcher si se sigue su flujo de trabajo y se tiene un equipo con procesador multinúcleo y tarjeta gráfica con opencl o Nvidia CUDA.[9][10]

6.6 Poco C++

Es un conjunto de librerías de C++ que permiten entre otras cosas, programar conectividad entre aplicaciones, capacidad de multihilo, creación de servicios Windows, compresión de archivos, etc.

Se usa Poco C++ junto con OpenCV para crear una aplicación de servidor para demostrar una de las posibles estrategias de implementación, además de poder tomar medidas de comparación entre el algoritmo desarrollado y el Image Stitcher. Por lo tanto se usando las clases para sockets, multihilo y temporizadores de Poco C++ para crear este aplicativo.

[9] OpenCV [online]. OpenCV API Reference: Stitching Pipeline. [citado el 28 de julio de 2014] Disponible en <http://docs.opencv.org/modules/stitching/doc/introduction.html>
[10] OpenCV [online]. OpenCV API Reference: GPU Module Introduction. [citado el 28 de julio de 2014] Disponible en <http://docs.opencv.org/modules/gpu/doc/introduction.html>

6.7 Visual C++

El IDE de Microsoft para programación con C++. Existen varias versiones, pero es de particular interés la versión gratuita y la versión profesional.

La versión gratuita es conocida como Visual C++ Express y trae lo mínimo para trabajar. Es suficiente para programar la implementación servidor del image stitcher usando OpenCV y Poco C++.

La versión profesional trae más herramientas, sin entrar en mucho detalle, provee las librerías y automatización necesaria para crear componentes COM, que son de nuestro interés. Es necesario tener esta versión para que el Matlab NE Builder pueda compilar sin problemas los componentes COM de los proyectos.

7. Implementación de Algoritmos de Campo Amplio

Para la implementación del algoritmo es necesario conocer el proceso que se lleva a cabo en el laboratorio para la toma de las imágenes de entrada, ya que como se ha venido diciendo, si estas imágenes no son de calidad, la imagen final no será lo que se espera obtener.

7.1 Proceso de toma de imágenes

El proceso inicia con la muestra de la cual se desea obtener la imagen a través del microscopio, debe calibrarse correctamente para tomar las capturas. El investigador debe conocer el sistema óptico y corregir la saturación de luminosidad presente en la muestra, el balance de blancos y elegir el objetivo adecuado para la muestra entre otros.

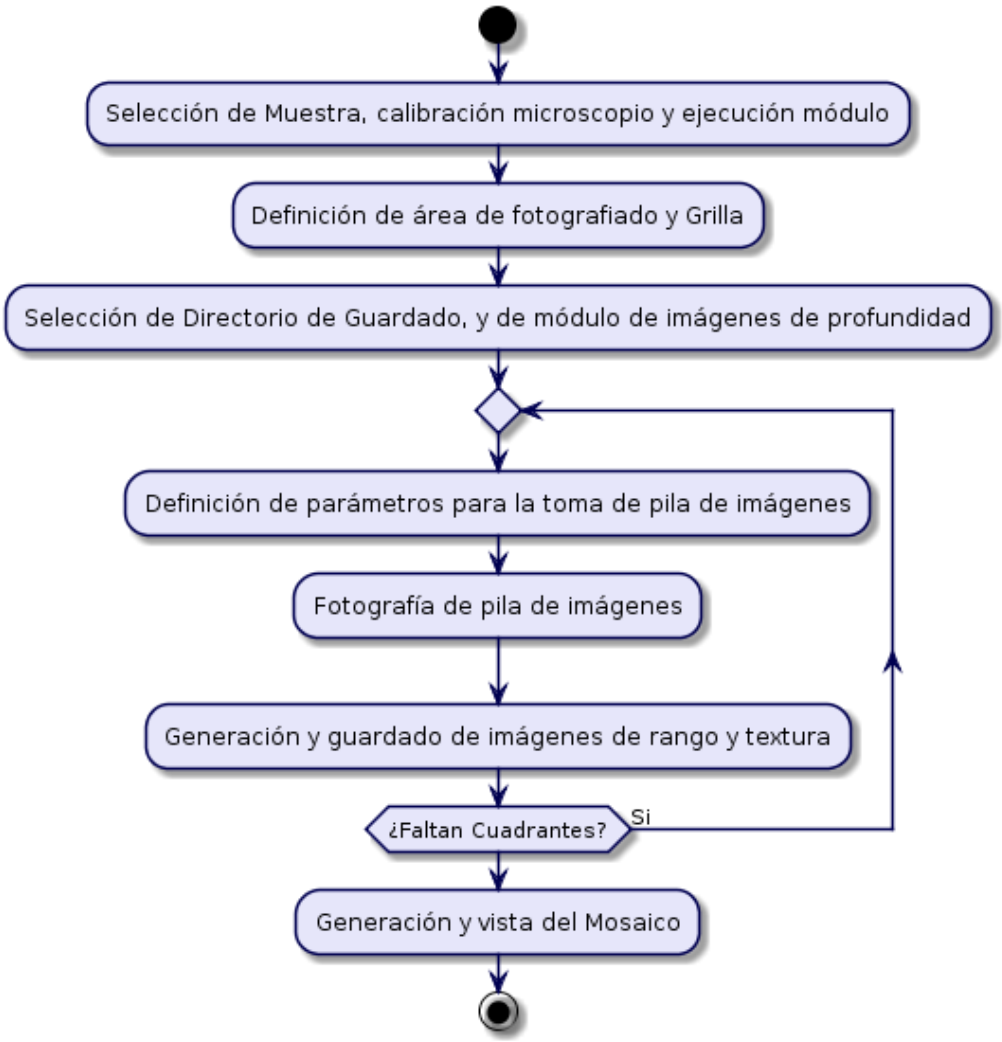
El investigador estará interesado en tener una imagen de toda la muestra, sin embargo el microscopio sólo puede tomar una sección de esta, por lo que será necesario tomar varias imágenes para cubrir toda la muestra.

La cámara del microscopio sólo se puede trasladar en coordenadas rectangulares, dos dimensiones únicamente. Por ello para tomar todas las imágenes necesarias para fotografiar toda la muestra, se debe definir una cuadrícula o grilla rectangular la cual barrerá el microscopio para tomar todas las imágenes necesarias.

Debido a limitantes de carácter físico, cuando se usa uno de los objetivos de más de 5x, cuando se va a tomar la imagen de uno de los cuadrantes de la grilla, no se verá enfocada la muestra visible en esos momentos. Para superar esa limitante se requiere tomar varias imágenes del mismo cuadrante con variación axial (eje z), cada una de las cuales enfoca una parte diferente. Al unir estas imágenes se obtiene lo que se define como una imagen de profundidad de campo extendida (EDF) y una imagen de rango.

El investigador deberá seleccionar el módulo ya existente en la plataforma de AxioVision para la creación de imágenes de profundidad. Cada vez que se obtiene una imagen de profundidad, está ya se podría pasar al algoritmo para la generación de mosaicos a la espera de otra para unir. Esto significa que hay dos estrategias disponibles para generar el mosaico, o bien procesar todas las imágenes en un solo lote o procesarlas a medida que van estando disponibles. La figura 10 resume este proceso.

Figura 10. Diagrama del proceso para toma de Imágenes.



7.2 Módulo de imágenes de profundidad

No se entrará en mucho detalle sobre este módulo, ya que ha sido material de estudio para un anterior trabajo de grado[11]. A continuación se expondrá lo que necesita el módulo para que el resultado producido, sea el esperado.

Como se ha mencionado anteriormente, el microscopio al tener mayor aumento para la toma de imágenes, sólo podrá tener enfocado una parte de la muestra. Para enfocar distintas partes del mismo cuadrante, el microscopio cuenta con un motor que mueve la plataforma donde se encuentra la muestra para acercarla o alejarla del lente, enfocando distintas partes del mismo cuadrante.

Se requiere una pila de imágenes que enfocan todas las partes posibles de la muestra, para fusionarlas en la imagen de profundidad que requiere el generador de mosaicos. Esta fusión de imágenes da como resultado una imagen totalmente enfocada, probablemente a color, de la muestra tomada y una segunda imagen, también totalmente enfocada, pero a escala de grises que guarda información de profundidad de los puntos.

La adquisición de la pila de imágenes se automatiza conociendo la distancia mínima y máxima de la muestra al lente del microscopio y definiendo un delta de desplazamiento para esta distancia. El número de imágenes necesarias es definido por la siguiente fórmula:

$$n_i = \frac{D_{máx} - D_{mín}}{\Delta D}$$

En donde, $Dmáx$ es la Distancia focal máxima, $Dmín$ es la Distancia focal mínima, ΔD es el aumento o delta de desplazamiento, y n_i es el número total de imágenes.

[11] BARRETO, Natalia. RODRIGUEZ Carlos. Análisis y desarrollo de un algoritmo para la generación de imágenes de profundidad de campo extendido basado en microscopia de alta resolución. Trabajo de grado Ingeniero de Sistemas. Bucaramanga. UNIVERSIDAD INDUSTRIAL DE SANTANDER. Facultad Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. 2013. 85p.

Entre más pequeño sea el desplazamiento, mayor número de imágenes tendremos para fusionar. El tamaño del delta de desplazamiento depende más que todo del lente usado y de la muestra, sin embargo se puede definir una delta lo suficientemente pequeño que funcionará para cualquier caso con el precio de que habrá ocasiones donde se toman más imágenes de las necesarias para fusionar en la imagen de profundidad.

El algoritmo generador de mosaicos puede generar un mosaico con la imagen de la muestra, y otro mosaico con la información de profundidad de los puntos de la muestra. Estas dos imágenes se pueden combinar usando un módulo o programa externo para generar una vista tridimensional de la muestra. En este caso, la imagen a color sería llamada imagen de textura, y la imagen a escala de grises, la imagen de profundidad.

7.3 Imágenes de Entrada para el Mosaico

El algoritmo requiere de un grupo de imágenes que comparten un conjunto de puntos similares con una o más imágenes diferentes. Estas se unen en una panorámica o mosaico. Si el conjunto de puntos similares no existen entre imágenes, le será imposible al algoritmo unirlas, y entre más puntos similares existan, mucho mejor. La figura 11 presenta imágenes cuya continuidad muestra la zona de similitud del 10% entre imágenes.

Otra característica de las imágenes individuales que se debe cumplir es que deben ser totalmente enfocadas.

Si las imágenes no son de profundidad de campo extendido para objetivos superiores a 5x, y/o no existe intersección de puntos entre imágenes, el algoritmo fallará y no dará resultados que sirvan, es decir, se aplica el principio de “basura-entra, basura-sale”.

Teniendo previamente las imágenes de rango y de profundidad de campo extendido, el procedimiento para la creación del mosaico se describe por el diagrama de flujo de las figura 12 y 13. Cabe resaltar que el algoritmo opera para pares de imágenes, asumiendo que estas tienen una zona de similitud.

Figura 11. Zona de Similitud entre imágenes.

Figura 12. Diagrama de flujo del proceso mosaico.

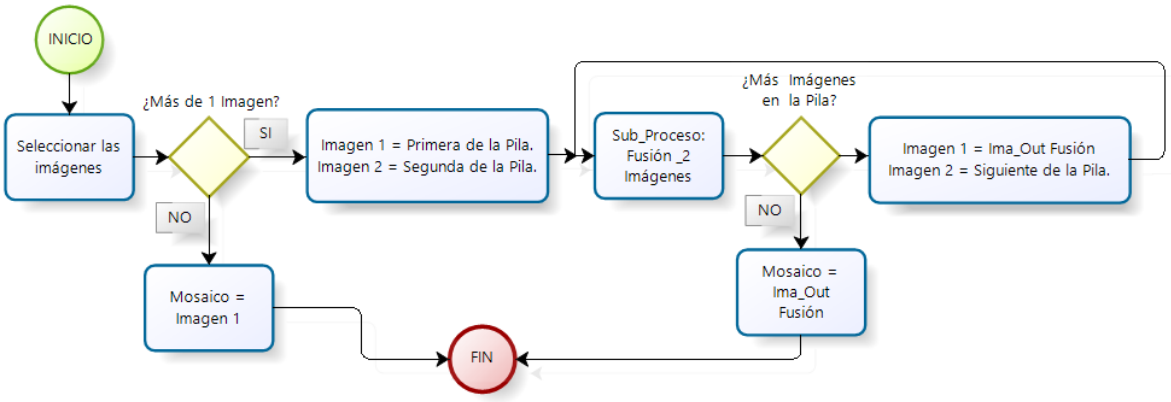
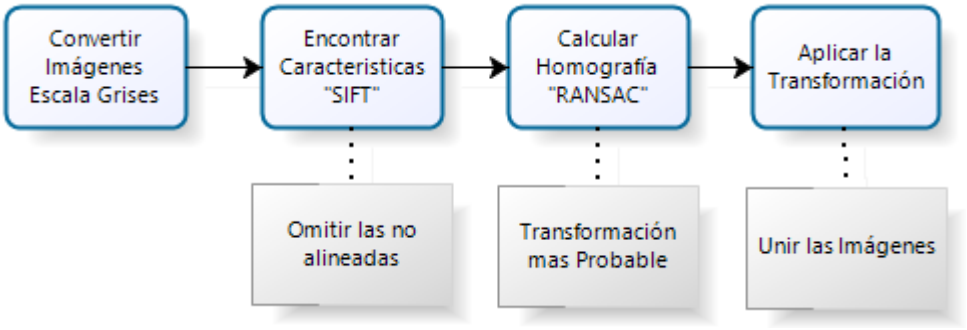
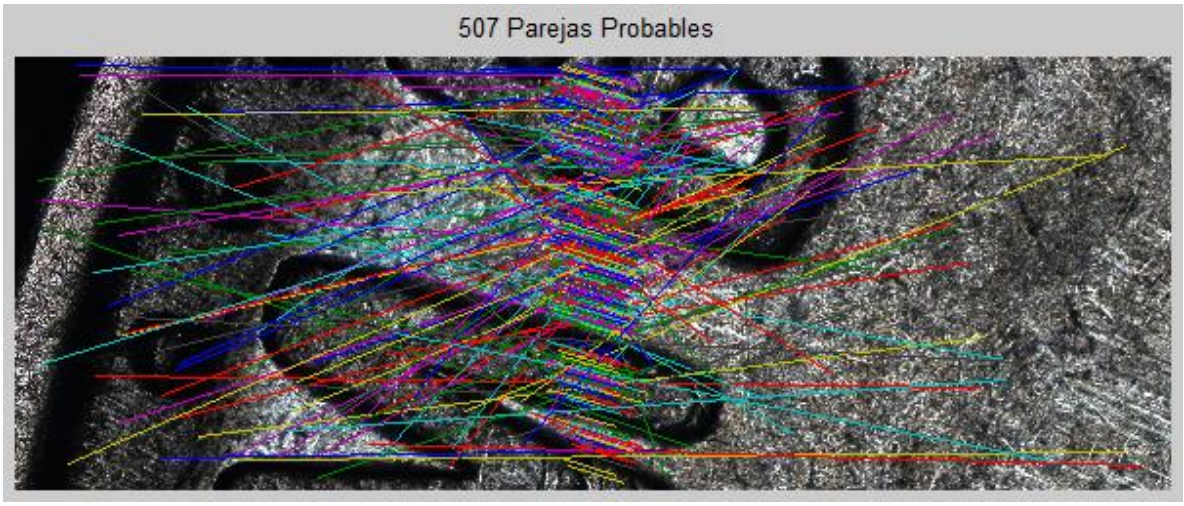


Figura 13. Detalle del Sub-proceso Fusión_2_Imágenes.



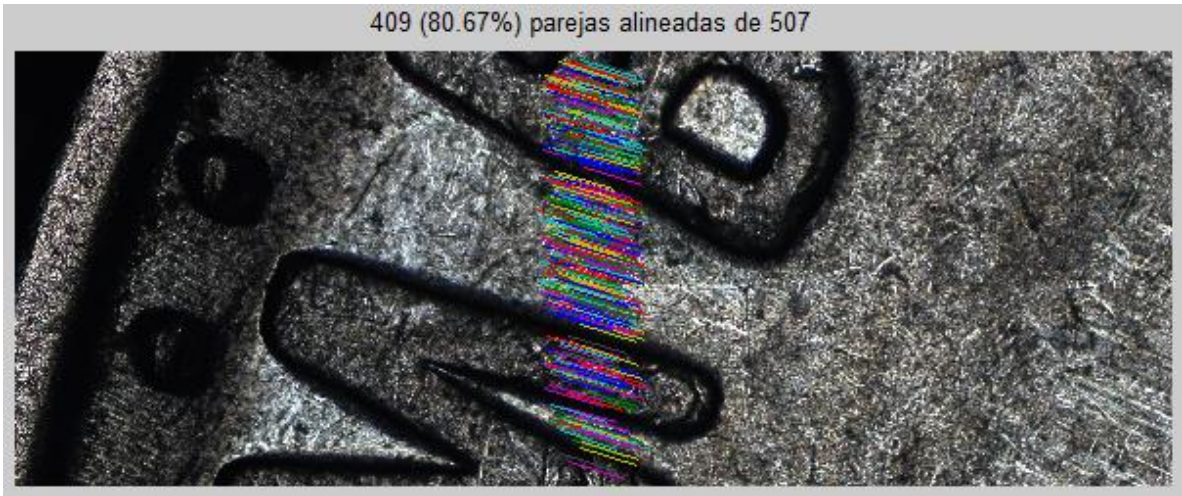
El primer paso para realizar el mosaico entre las imágenes es encontrar los puntos característicos, esto se realiza con la técnica de SIFT descrita anteriormente. La figura 14 presente los puntos clave de un par de imágenes continuas con zona de similitud (figura 11), además se relaciona los puntos clave de una imagen con descriptores similares a los puntos clave de la otra por medio de una línea recta, las rectas se pintan en diferentes colores para diferencias puntos clave entre sí.

Figura 14. Puntos Clave Imágenes Consecutivas.



Los puntos clave que se encuentren dentro de la zona de similitud comparten ciertas características de linealidad. La figura 15 ilustra este resultado.

Figura 15. Características Alineadas en la zona de similitud.



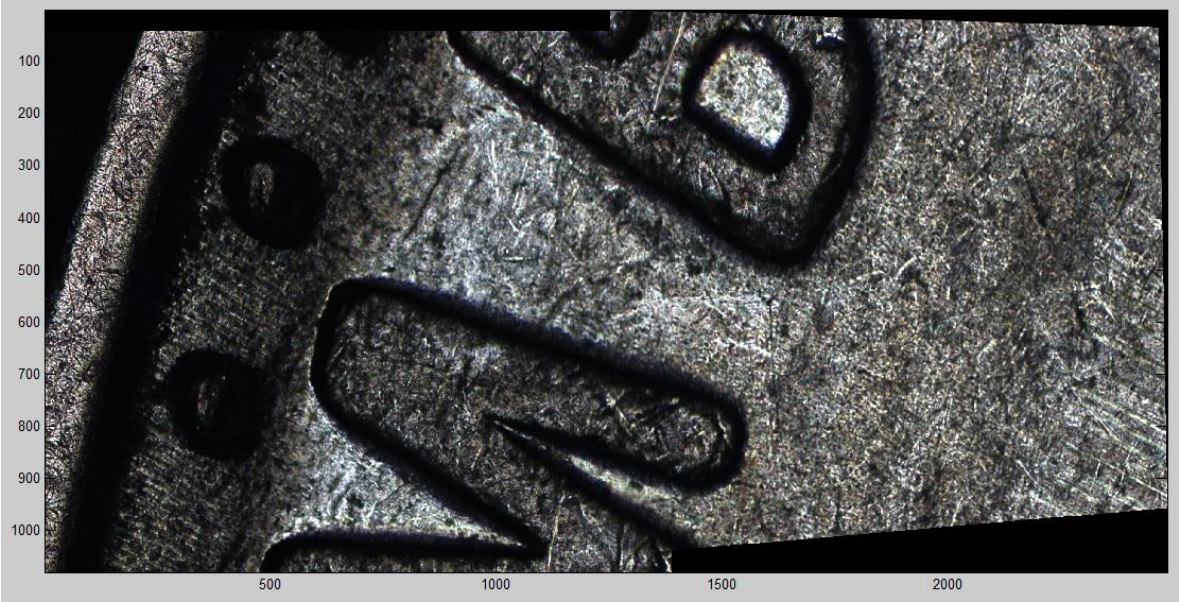
Las transformaciones necesarias para crear el mosaico entre las imágenes pueden ser del tipo corrimiento en el $eje\ x$ y/o $eje\ y$, escalamiento, rotaciones y proyecciones geométricas.

El cálculo de la transformación se realiza mediante el algoritmo RANSAC y se identifican los puntos de origen para el empalme entre las dos imágenes. Teniendo los puntos identificados y su localización relativa en cada imagen, se aplica la transformación necesaria para la fusión, las imágenes alineadas antes de realizar el mosaico y la imagen resultante se presentan en las figura 16 y 17 respectivamente.

Para asegurar la creación del mosaico también es necesario conocer el procedimiento de captura, el cual se lleva a cabo como un barrido que generalmente es de derecha a izquierda sobre la muestra en el área que se desea capturar, es vital conocer el recorrido realizado para después realizar el mosaico.

Figura 16. Imágenes alineadas antes del mosaico.

Figura 17. Mosaico resultado de 2 imágenes con zona de similitud.



Los recorridos pueden ser en zigzag comenzando por la primera fila de izquierda a derecha y luego saltando al principio de la siguiente fila como se ilustra en la figura 18.

Figura 18. Recorrido Zigzag.

El recorrido serpentín, ilustrado en la figura 19 consiste en ir de izquierda a derecha comenzando por la primera fila y continua por la última imagen de la segunda fila para devolverse de derecha a izquierda y volver a con la primera imagen de la tercera fila.

Las figuras 18 y 19 se muestran los recorridos para un conjunto de 64 imágenes tomadas con el objetivo de 5x, a una moneda de 50 pesos de Colombia.

Figura 19. Recorrido Serpentín.

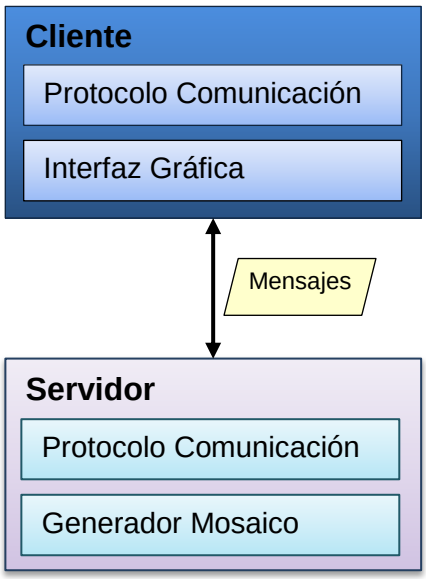
7.4 Arquitectura de la implementación

Para superar las limitantes por el uso de visual basic application y crear un módulo independiente al software del microscopio, se codifica el algoritmo en un lenguaje y entorno de desarrollo diferente.

Esta aproximación hace que el módulo sea independiente al software actual del microscopio, lo cual abre la posibilidad de ser usado en otro software o como un programa aparte, además de que futuro desarrollo y actualizaciones del módulo requerirá de muy pocos a ningún cambio en la interfaz del módulo en el software del microscopio.

Este decoplamiento entre el software del microscopio y el módulo donde estará implementado el algoritmo como tal, nos conlleva al uso de una arquitectura cliente-servidor. El cliente sería la interfaz gráfica en AxioVision, y el servidor será el módulo generador de mosaicos. Aunque se siga esta arquitectura, no implica que se ejecutarán los programas en computadores diferentes, sin embargo, el diseño deja que se decida en la implementación cómo será la ejecución del cliente y el servidor, si en un mismo proceso o en uno o más computadores.

Figura 20. Diagrama de Arquitectura.



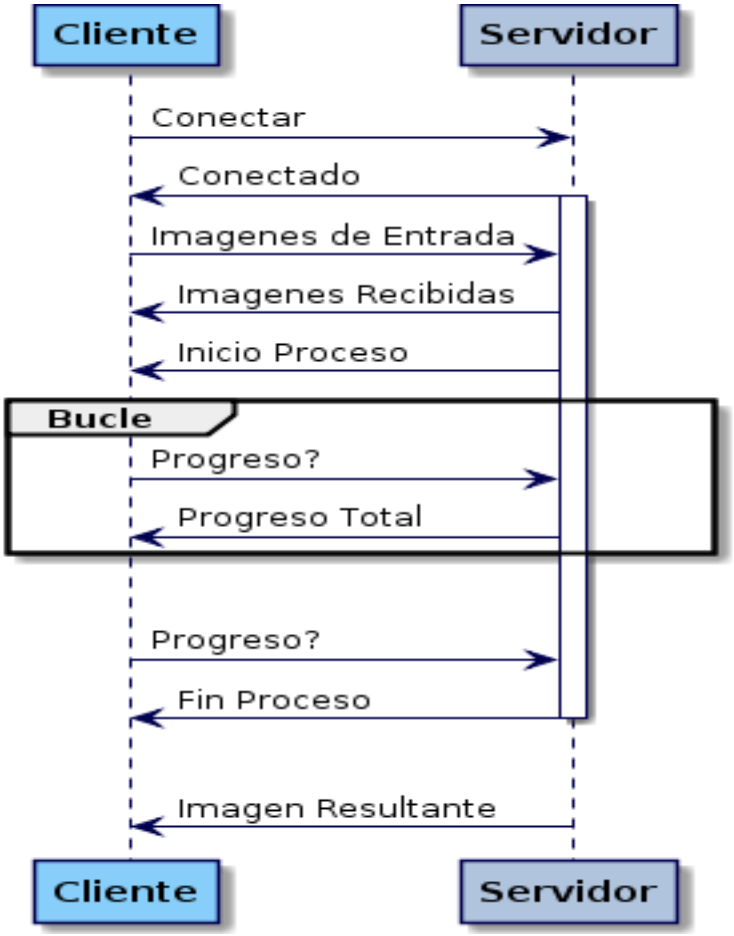
7.5 Mensajes entre el cliente y el servidor

Al usar una arquitectura cliente-servidor, es necesario definir cómo se comunican entre sí los programas y/o módulos. En este caso los mensajes vienen definidos por las entradas que requiere el algoritmo generador de mosaicos y las salidas de este. Además, son de importancia los mensajes que indican el progreso del algoritmo, es decir, aquellos mensajes que nos dicen que el servidor está haciendo su trabajo.

Los mensajes serían entonces: La ubicación de las imágenes de entrada o las imágenes de entrada en sí, progreso de la generación del algoritmo, y, la imagen resultante en sí o la ubicación de esta.

El cliente envía de mensajes al servidor las imágenes de entrada, y pregunta cómo va con el procesamiento. El servidor envía de respuesta que inicia el proceso al recibir las imágenes de entrada, cuánto lleva de progreso, y al terminar el proceso espera a que el cliente pregunte por el progreso de nuevo para informarle que ya terminó, y que le enviará la imagen resultante. La figura X (7.4) muestra cómo es el intercambio de mensajes, anteriormente descrito, entre el cliente y el servidor.

Figura 21. Mensajes entre cliente y servidor.



7.6 Diseño de la interfaz de usuario

Analizando el proceso de toma de imágenes y lo que espera el algoritmo como entrada, se establecen los siguientes requerimientos como los mínimos que debe cumplir la interfaz del módulo:

- Permite definir una grilla rectangular que cubre o bien toda la muestra o un área de esta. La grilla se define por dos dimensiones: número de cuadrantes de alto y de ancho.
- Permite seleccionar el lente de la cámara a usar.
- Permite seleccionar un módulo que esté disponible para las imágenes de profundidad.
- Permite definir la distancia mínima y máxima de la muestra al lente del microscopio y el delta de desplazamiento para adquirir la pila de imágenes necesaria para obtener la imagen de profundidad.
- Permite seleccionar entre crear la imagen de mosaico para la imagen de textura únicamente, o para la imagen de textura y profundidad.

7.7 Diseño de clase

El servidor básicamente consta de una única clase, cuyos atributos son una lista de las imágenes a unir, la imagen resultante, la cantidad de imágenes a procesar y el total de imágenes procesadas. Los métodos públicos son modelados a partir del intercambio de mensajes entre el cliente y el servidor, y estos son:

- Establecer imágenes de entrada: recibe de parámetros las imágenes en si o la ubicación de estas.
- Obtener la imagen de salida: devuelve la imagen en si o la ubicación de esta.
- Obtener el progreso: Informa en porcentaje el progreso de unión de imágenes. Este valor es calculado como el total de imágenes procesadas dividido por el

total de imágenes a procesar por cien. Es preferible el truncamiento a dos decimales.

Los demás métodos no han de ser visibles por el cliente y por ende se consideran privados. Estos métodos privados son los siguientes:

- Obtener imagen de entrada: devuelve una de las imágenes de entrada, tiene de parámetro la posición en la lista de la imagen.
- Establecer imagen de salida: recibe de parámetro una imagen, que por lo general es el progreso que lleva el algoritmo en determinado momento del proceso.
- Establecer el progreso: Recibe de parámetro la cantidad de imágenes que se han procesado.
- Unir imágenes: Recibe de parámetro dos imágenes, una del conjunto de entrada, y lo que se lleva de la imagen resultante.

La figura 22 muestra el diagrama de la clase principal del Servidor.

Figura 22. Diagrama de clase del servidor.

Otras clases o métodos que requiera el servidor ya son dependientes de la tecnología que se usará para la implementación.

7.8 Estrategias de implementación

El diseño de clase del servidor tiene pensado que se use en la implementación en un bucle ya sea en el hilo principal o en un hilo secundario.

Antes del bucle se establece la imagen resultante igual a la primera imagen del conjunto de entrada y la cantidad de imágenes procesadas se incrementa en 1.

Dentro del bucle se une la imagen resultante con una del conjunto de entrada indicada por la cantidad de imágenes de procesadas. Al terminar la unión se incrementa la cantidad de imágenes procesadas en uno nuevamente (notificando el nuevo progreso) y se prosigue con la siguiente iteración.

El bucle iterativo termina cuando se hayan terminado de procesar todas las imágenes, esto es, cuando el atributo de imágenes procesadas, antes de incrementarlo, sea igual al atributo total de imágenes de entrada.

7.8.1 Librerías de vínculo dinámico. En esta estrategia, la arquitectura de cliente-servidor no es tan evidente, y fácilmente puede ser considera bajo otra arquitectura, como por ejemplo modelo-vista-controlador, sin embargo, los principios expuestos de la arquitectura elegida en este trabajo aplican perfectamente a esta estrategia.

Esta estrategia restringe los posibles lenguajes de programación para implementar el algoritmo a aquellos que soporten un protocolo común para librerías de vínculo dinámico.

El requisito más común para crear estas librerías es que el lenguaje de programación genere código binario compatible con C, lo cual deja por fuera el uso de lenguajes como Java.

Usar una DLL directamente en visual basic application requiere definir con “Declare” todas las funciones que se vayan a usar de esta, además de tener que registrarlas y llamarlas usando “REGISTER” y “CALL” respectivamente. Sin embargo existe otra aproximación que no necesita de todos estos pasos, usar el estándar binario COM. Este estándar de Microsoft permite usar DLLs en C++, C# y Visual Basic (VBA incluido también) como si la librería fuese nativa al lenguaje usado.

El problema con esta estrategia de implementación es el uso de múltiples hilos. Como la librería es cargada y usada en el ámbito de visual basic application esto limita la cantidad de hilos posibles a uno sólo (para la versión que usa el software del laboratorio) ya que implementa el Modelo Apartamento de Hilo Único[12](Single-Threaded Aparment Model), lo que significa que VBA se encarga de manejar los objetos COM como si fuesen ventanas separadas y pone en una cola las diversas llamadas a los métodos.

Debido a que sólo se puede usar un único hilo, la implementación siguiendo esta ruta, inevitablemente hará que en algún momento la interfaz al módulo deje de responder por un tiempo hasta que complete la tarea que esté realizando.

7.8.2 Comunicación por TCP/IP. Comunicando procesos por TCP/IP es una implementación general para la arquitectura cliente-servidor. Se usan los sockets TCP porque es importante que se garantice que los datos enviados lleguen tal y como se enviaron, y no exista pérdida de datos.

[12] http://support.microsoft.com/kb/q150777

En esta estrategia el algoritmo se puede implementar en cualquier lenguaje que soporte conexiones por sockets TCP, además como el servidor correrá en un proceso aparte, permitirá el uso de múltiples hilos e incluso tecnologías más avanzadas como Nvidia CUDA, AMD FireStream u OpenCL para realizar el procesamiento de imágenes.

El código necesario en Visual Basic Application requerirá el uso de los sockets de Windows (winsockets) cargado desde DLLs. Esto requerirá de una clase para manejar conexiones de forma general y que la Interfaz implemente un bucle donde procese los mensajes específicos de la implementación.

Cabe recordar que sólo se podrá usar un hilo en Visual Basic Application, por lo que al iniciar la comunicación con el servidor, el bucle que se usa podría bloquear potencialmente el módulo mientras espera respuesta del servidor. Para superar esto se usa el comando “DoEvent”, lo que permitirá funcionar sin bloqueos el módulo, siempre y cuando no sea demorada la comunicación con el servidor.

Todo algoritmo está compuesto fundamentalmente de tres partes: una entrada, un proceso y una salida. El algoritmo para la generación de mosaico no es diferente en este aspecto y será definido bajo estos tres parámetros fundamentales.

7.9 Salida

Una vez que el algoritmo termina de procesar las imágenes de entrada deberá producir una única imagen, que es la combinación de todas las imágenes de entrada mostrando una panorámica de la muestra capturada.

Se reitera la importancia de que las imágenes de entrada cumplan con las dos principales características mencionadas anteriormente para obtener buenos resultados.

Figura 23. Imágenes de Entrada (Izquierda) vs Mosaico Resultante (Derecha). Escudo de 50 pesos.

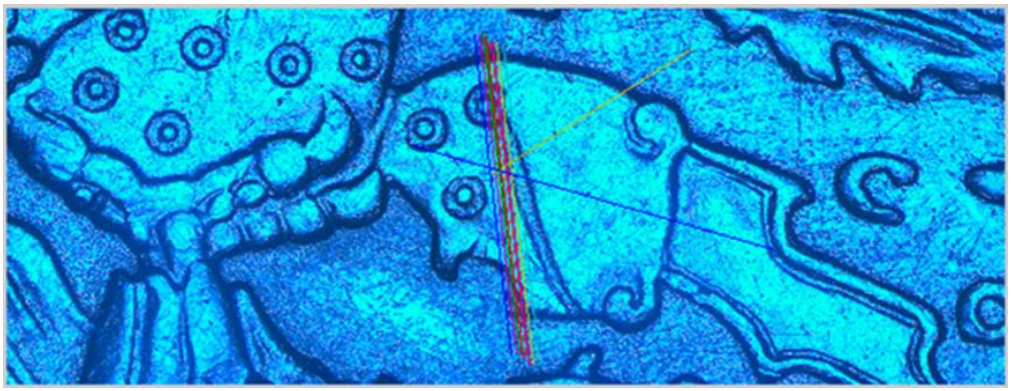
El mosaico resultante se puede apreciar mejor en la figura 24.

Figura 24. Mosaico Resultante 64 Imágenes. Escudo 50 Pesos.

7.10 Problemas

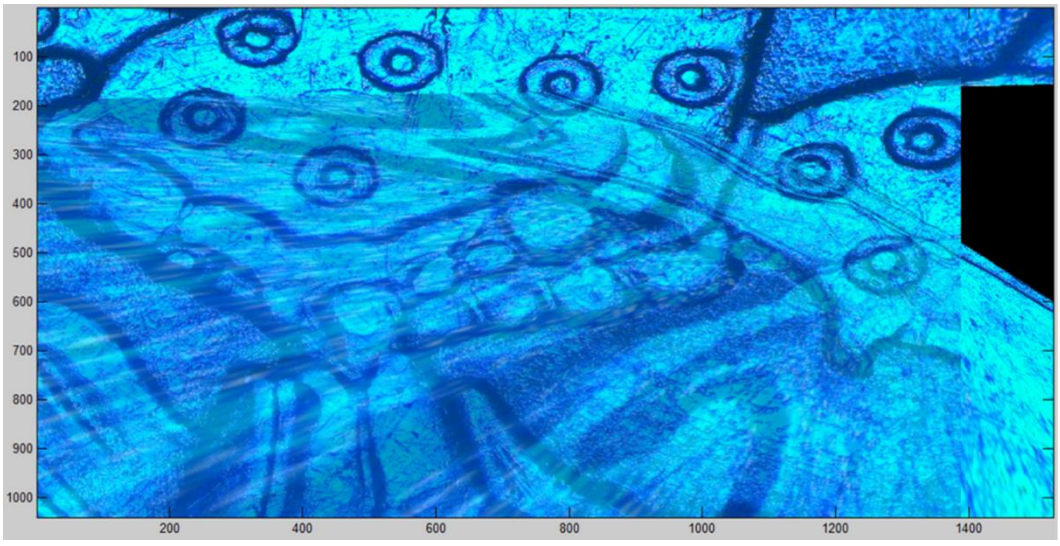
Existen problemas para realizar el mosaico cuando la transformación no es una buena representación homográfica y no se puede calcular la correcta transformación por problemas de convergencia como en imágenes con una zona de similitud muy pequeña o sin la tonalidad correcta, ausencia de luz o mal contraste.

Figura 25. Capturas aparentemente coincidentes.



Los resultados de problemas mencionados se muestran en la figura 26

Figura 26. Error en mosaico.



8. Mediciones Experimentales

8.1 Mediciones realizadas

Existen diferentes programas comerciales para la creación de panoramas a partir de secuencias de imágenes o videos, algunos de estos son pagos como por ejemplo Autopano[13], Serif PanoramaPlus X4[14], Calico[15], existen también versiones gratuitas o con licencia para uso no comercial como AutoStich[16] y Microsoft Image Composite Editor[17].

Se pondrá a prueba el Mosaico generado por el microscopio con el respectivo módulo incorporado y el generado por el algoritmo desarrollado en este trabajo contra los programas de referencias de AutoStitch y Microsoft Image Composite Editor. El algoritmo implementado en MATLAB (Anexo 2) utiliza la función sift.m provista por VLFeat como una función optimizada.

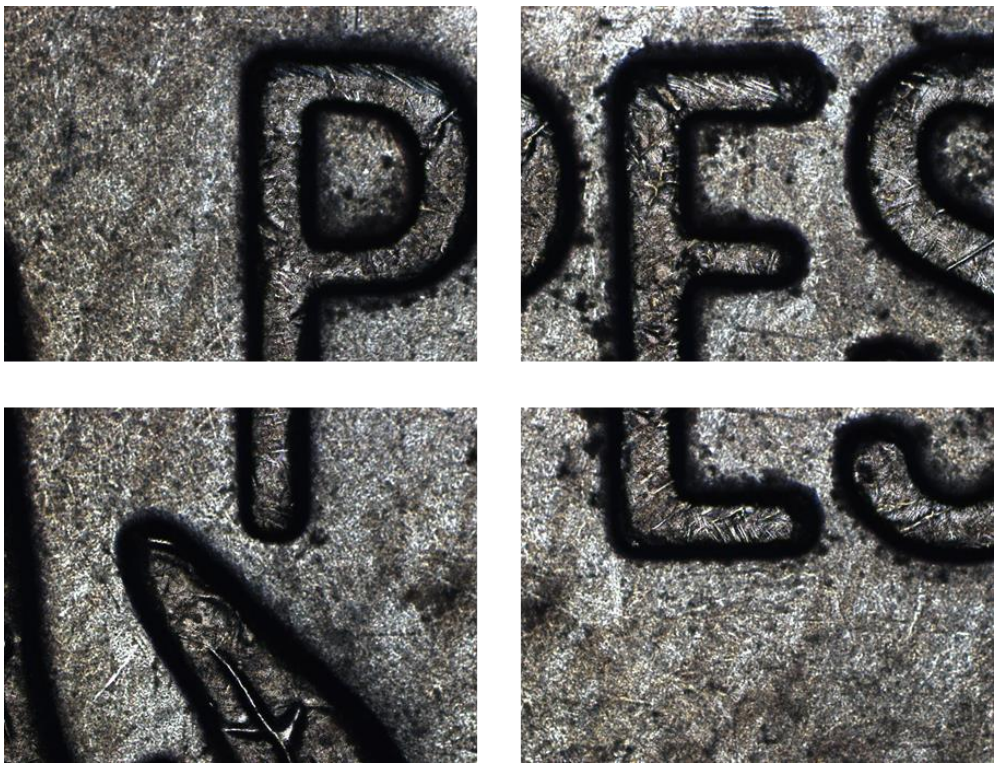
Las mediciones requeridas para comparar el algoritmo realizado con el programa de referencia poseen sólo dos variables: Número de imágenes procesadas y cantidad de tiempo requerido.

A continuación se presentan una serie de mosaicos generados por conjuntos de imágenes de 4, 9, 16 y 64 imágenes. Los tiempos aproximados para la generación del mosaico se calcularon midiendo manualmente ya que para Autostitch y Microsoft ICE no hay forma de implementar medida de tiempo directamente en el código. Las medidas de tiempo para el modulo del mosaico es una suma de lo que va tardando, ya que la unión de imágenes se produce a medida que se realizan las capturas. Los algoritmos utilizados se presentan en los anexos A y B.

[13] http://autopano.kolor.com/
[14] http://www.serif.com/panoramaplus/
[15] http://www.kekus.com/
[16] http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html
[17] http://research.microsoft.com/en-us/um/redmond/groups/ivm/ICE/

La figura 27 muestra el conjunto de imágenes de entrada para generar un mosaico de 2x2 imágenes.

Figura 27. Imágenes de entrada para mosaico 2x2.



Las figuras 28, 29 y 30 presentan el mosaico resultante usando el algoritmo implementado en MATLAB, AutoStitch y Microsoft ICE Respectivamente.

La figura 31 muestra el conjunto de imágenes de entrada para un mosaico de 3x3. Las figuras 32, 33 y 34 muestran los resultados obtenidos por el algoritmo, AutoStitch y Microsoft ICE.

Las imágenes correspondientes para el caso de imágenes de entrada de 4x4 no son incluidas dado a que el algoritmo en MATLAB toma demasiado tiempo a falta de optimizaciones. Sin embargo para el algoritmo de 8x8 se aplicó una optimización comprometiendo la robustez.

Figura 28. Imagen 2x2 generada con el algoritmo de MATLAB

Figura 29. Imagen 2x2 generada por AutoStitch

Figura 30. Imagen 2x2 generada por Microsoft ICE

Figura 31. Imágenes de entrada para el mosaico 3x3

Figura 32. Imagen 3x3 generada por por el algoritmo en Matlab

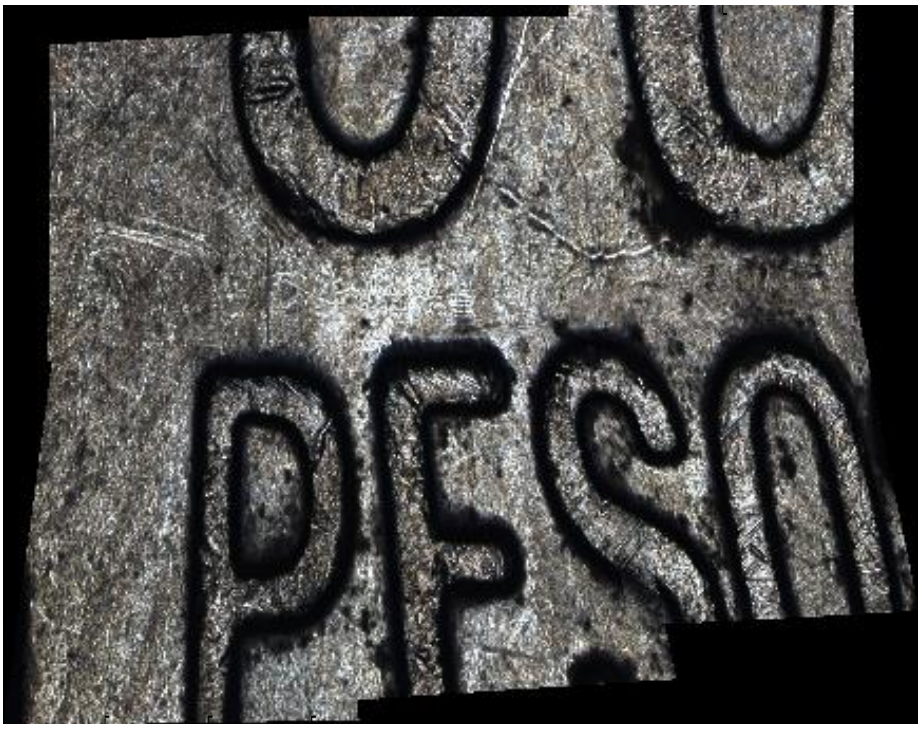


Figura 33. Imagen 3x3 generada por Autostitch

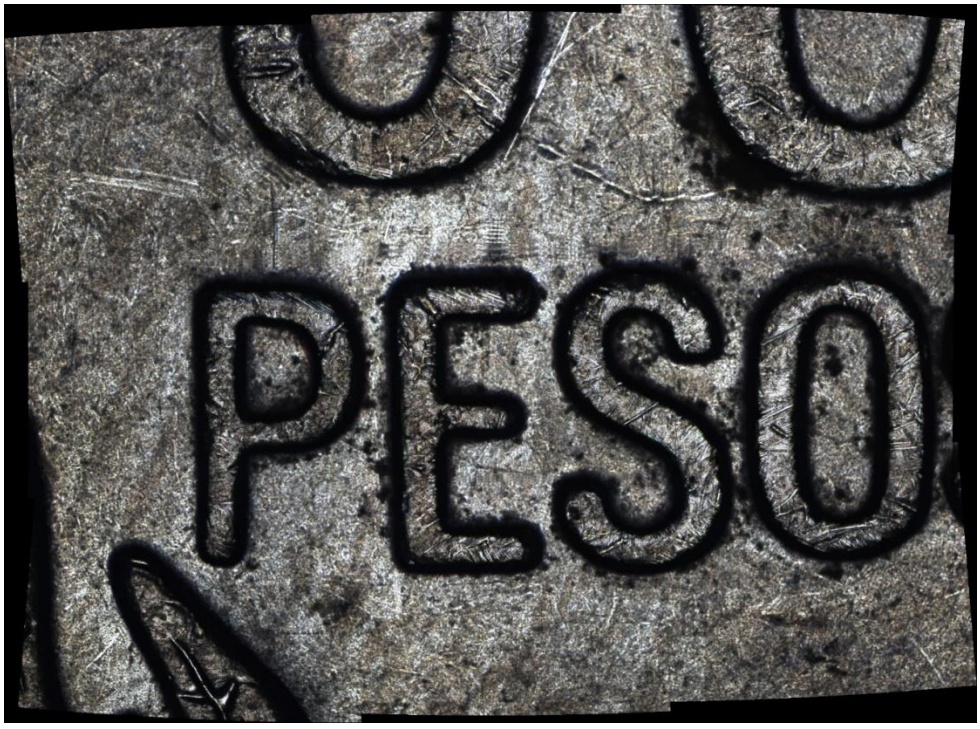
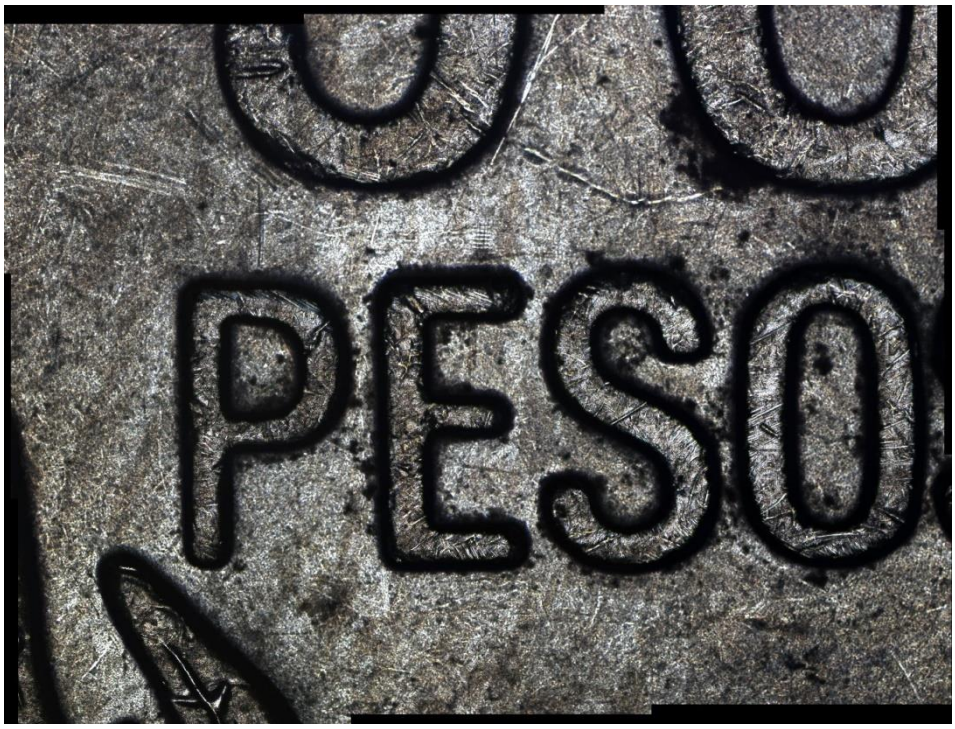


Figura 34. Imagen 3x3 generada por Microsoft ICE



Para el caso del conjunto de imágenes de entrada para un mosaico de 8x8, se creó un segundo algoritmo (Anexo B) que se basa en los recortes proporcionales, a la zona de similitud, que deben hacerse a las imágenes para fusionarlas. Este algoritmo es considerado una posible optimización a cambio de hacer el algoritmo menos robusto, esto es, específico para las imágenes tomadas con el microscopio Imager.Z1m y sin la garantía de poder obtener los resultados esperados usando imágenes tomadas de otros microscopios. El resultado obtenido es similar al de Microsoft ICE.

La figura 35 muestra el conjunto de imágenes de entrada utilizadas para esta prueba, y las figuras 36, 37 y 38 muestran los resultados obtenidos.

Figura 35. Conjunto de imágenes para un mosaico de 8x8

Figura 36. Imagen 8x8 generada por el algoritmo basado en recortes.

Figura 37. Imagen 8x8 generada por Autostitch

Figura 38. Imagen 8x8 generada por Microsoft ICE

Se recomienda a futuro, para aplicar esta técnica de optimización y para aumentar la robustez, implementar una rutina automática para la caracterización de los objetivos.

Los tiempos obtenidos se presentan en la tabla 2.

Tabla 2. Comparación Rendimiento de los Algoritmos.

Tamaño del Mosaico		2x2	3x3	4x4	8x8
Algoritmo Matlab	SIFT	3 min	8 min	25 min	-
	Recortes	-	-	-	10 seg
AutoStitch		15 seg	25 seg	37 seg	67 seg
Microsoft ICE		13 seg	18 seg	23 seg	35 seg

En la figura 39 se presenta los resultados experimentales, se debe notar que la curva con el algoritmo en Matlab esta escalada, los valores de esta corresponden a minutos.

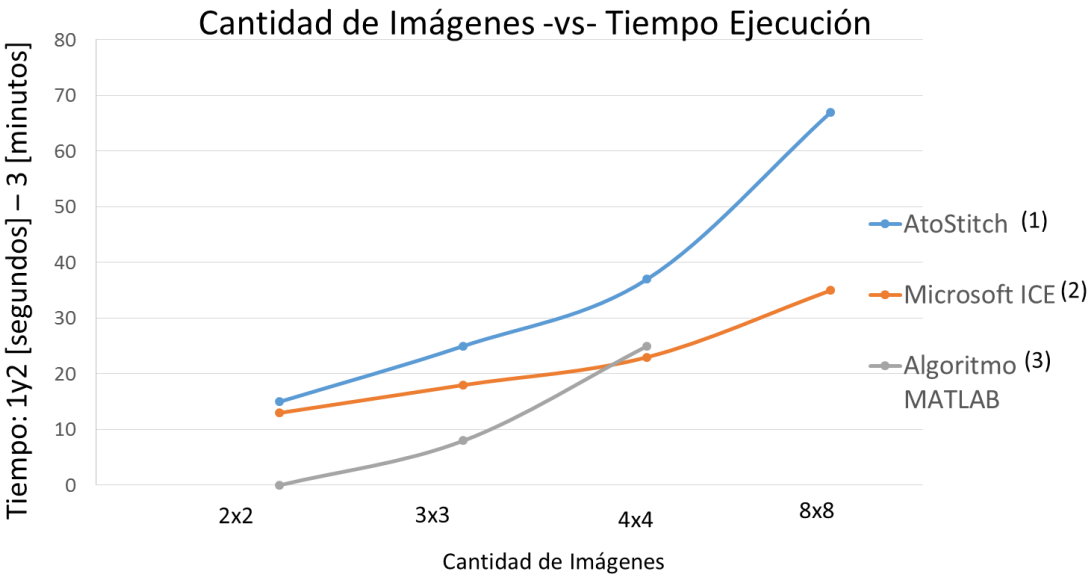
El rendimiento de programas como el de Microsoft ICE, se debe a la implementación de funciones de forma nativa, sin la interacción de interfaces gráficas y el uso de técnicas no distribuidas para procesamiento múltiple de imágenes.

Cada imagen de textura se encuentra comprimida en formato JPEG, con un tamaño de archivo aproximado de 1MB, resolución de 1388x1040 pixeles. Las características de las imágenes de mosaico resultantes presentan en la tabla 3.

Tabla 3. Características Mosaico Resultante.

Mosaico	2x2	3x3	4x4	8x8
Resolución [Pixeles]	2.668x2.017	3.952x2.998	5.233x3.981	10.464x8.189
Tamaño de Archivo [MB]	3.8	9.4	16.5	63,6

Figura 39. Resultados Experimentales Rendimiento.



9. Conclusiones

- Se desarrolló un algoritmo capaz de realizar los mosaicos de imágenes de rango y de textura, de campo amplio, resolviendo problemas de iluminación y haciendo corrección de contraste.

- Se encontró que los problemas en cuando a la cantidad de imágenes que se pueden procesar para la generación de mosaico, hacen referencia a la calidad de las mismas, dado que el algoritmo opera 2 imágenes en cada iteración, al agregarle una tercera imagen, se toma como entrada el mosaico generado en la iteración anterior lo cual exige recursos en memoria y vuelve ineficiente el sistema, tomándole mucho tiempo solucionar el mosaico.

- La implementación de un algoritmo basado en los recortes proporcionales a la zona de similitud entre las imágenes es una versión optimizada para la generación de mosaicos debido a que fue necesario caracterizar el sistema para conocer las rotaciones, translaciones y deformaciones presentes en las imágenes capturadas con un objetivo específico.

- Se implementó un módulo en el microscopio basado en componentes COM, y se dejó dos ejemplos, uno para cada una de las otras estrategias de implementación.

- Los módulos implementados con las estrategias desarrolladas, pueden ser mejorados por medio de la tarjeta de video existente en el laboratorio porque es posible usar Ati FireStream (el equivalente de Nvidia CUDA) lo cual podría ser útil para procesamiento de imágenes.

10. Recomendaciones

- Continuar con el desarrollo del algoritmo aprovechando las estrategias de implementación encontradas y desarrolladas en el laboratorio de óptica. La generación de módulos como componentes COM o servidores.

- Hacer un estudio de viabilidad para optimizar el algoritmo usando instrucciones de ensamblador de bajo nivel del procesador Intel Xeon que usa el equipo de cómputo del laboratorio.

- Hacer un estudio de viabilidad para implementar módulos que hacen uso de la estrategia de sockets TCP/IP que permitan usar una red de equipos para aumentar la capacidad de cómputo del mismo.

- Caracterizar los errores producidos por cada objetivo en el microscopio. Hecho esto puede utilizarse el algoritmo basado en recortes para generar las imágenes de rango y de textura, de campo amplio usando la menor cantidad de tiempo, con la limitación que el algoritmo solo funciona en el microscopio Imager.Z1m tratado en el presente proyecto.

- Automatizar la caracterización de los errores producidos al tomar un conjunto imágenes de entrada. De esta forma se recupera robustez al aplicar la optimización por recortes pero no será tan rápida la generación como en la recomendación anterior.

BIBLIOGRAFÍA

- FANT, Karl. A critical review of the notion of algorithm in computer science [online]. *Proceedings of the 1993 ACM conference on Computer science*, New York, NY, USA, 1993, pp. 1–6. <http://doi.acm.org/10.1145/170791.170794>

- GARZON REYES, Johnson. Metrología Óptica en Microscopía Convencional, Interferométrica Y Confocal Cromática. Trabajo de Doctor en Ciencias Naturales-Física. Bucaramanga: UNIVERSIDAD INDUSTRIAL DE SANTANDER. Facultad de Ciencias. Escuela de Física. 2005. 163 p.

- GU, Lingjia, An Automatic Target Image Mosaic Algorithm [online]. EUROCON, 2007. The International Conference on "Computer as a Tool", vol., no., pp.369, 374, 9-12 Sept. 2007. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4400290&isnumber=4400218>

- HABERMAN, Bruria, AVERBUCH, Haim and GINAT, David. Is it really an algorithm: the need for explicit discourse? [online]. *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, New York, NY, USA, 2005, pp. 74– 78. <http://doi.acm.org/10.1145/1067445.1067469>

- HERNANDEZ ARIZA, Liliana. Tratamiento de Imágenes usando el Método de Profundidad de Campo Extendida (Edf) en Arquitecturas Paralelas. Trabajo de grado Ingeniero de Sistemas. Bucaramanga: UNIVERSIDAD INDUSTRIAL DE SANTANDER. Facultad Físico-mecánicas. Escuela de Ingeniería de Sistemas e Informática. 2011. 109 p.

- JAMEEL QURESHI M, Rizwan and HUSSAIN, Asad. An adaptive software development process model [online]. *Advances in Engineering Software*, vol. 39, no. 8, pp. 654–658, Aug. 2008. <http://www.sciencedirect.com/science/article/pii/S0965997807001603>

- JIN-JIANG, Wang, MING, Xu and WEN-YAO, Liu. To Extend the Depth of Field Based on Image Processing [online]. *2nd International Congress on Image and Signal Processing, 2009. CISP '09*, Oct., pp. 1–3. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5304404>

- JING, Xu, XIAO HONG, Yang, XIANG XIN, Shao, *et al.* A new medical image mosaic algorithm base on regional features registration [online]. *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, vol.1, no., pp.173, 176, 25-27 May 2012. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6272573&isnumber=6272535>

- LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints [Online]. *Int. J. Comput. Vision*, nov 2004, p. 91-110. <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>

- OSABE, Taito, SHIRAI, Keiichiro, UTO, Toshiyuki, OKUDA, Masahiro, *et al.* A study on joint progressive coding of range data and texture images [online]. *Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on*, vol.2, no., pp.845, 848 vol.2, 26-29 Oct. 2004.

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1413836&isnumber=30638>

- YANG, Guo-Sheng and ZHANG, Huan-Long. Optimal Image Mosaic Wavelet Method Based on Fuzzy Integral [online]. *Machine Learning and Cybernetics, 2006 International Conference on* , vol., no., pp.3738,3744, 13-16 Aug. 2006 <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4028721&isnumber=4028022>

ANEXOS

Anexo A

- Algoritmo en MATLAB **Mosaico_recorte.m**

```
% % ==========================================
%
%          Mosaico_recorte
%         ===========
%
%    Mosaico basado en el recorte de las imágenes aprovechando el
% solapamiento entre las imágenes en el momento de la captura, desde el
% microscopio.
%
%   Se deben definir como valores de entrada el valor del porcentaje con el
%   cual fueron tomadas las imágenes y el tamaño del arreglo del mosaico en
%   filas y columnas, además se tiene implemento que el sistema tome las
%   capturas en zigzag comenzando en la esquina superior de la muestra.
%
%   Se puede guardar el mosaico resultante agregando un nombre al final de
% la ejecución de este script.
%
% % ==========================================
% % Borrar listado de imágenes actuales y variables
%
clear all ;
[FileName,PathName] = uigetfile('*.*','Seleccione las Imágenes para el Mosaico:',...
    'MultiSelect', 'on');
archivo = fullfile(PathName, FileName) ;
[m,n]=size(archivo);
if iscell(archivo)==0
    disp('Se necesita más de 1 imagen---');    return;
end
% % ==========================================
%
% % ==========================================
% % Iniciar el programa de Mosaico recortando los bordes de imágenes
%
clc ; disp(' --- Programa Vista Mosaico:   --- ');
porcen = 10;
%  Cargar la primera imagen:
```

```
primera=1; nombre1 = char(archivo(primera));
img1 = imread(nombre1);
% %  Método para quitar el porcentaje medio a cada lado
   tama = size(img1) ;  tama2 = tama*(100-porcen)/100 ;
   difer = uint32((tama-tama2)/2) ;
   image1=img1(difer(1):tama(1)-difer(1),difer(2):tama(2)-difer(2),:) ;
% % ============================================
%
% % ============================================
% % Definir el tamaño del mosaico filas y columnas, iniciar un contador:
close all force ;  filas = 8;  colum = 8;  tic;
for kk=1:filas
   for jj=1:colum
% % ============================================
%        Con base en el número de la fila definir la orientación del
%        sistema para posicionar  las imágenes dentro del mosaico.
      contador=filas*kk+jj-filas ;
%        Para mostrar una figura con cada imagen por separado des comentar
% % %        if (mod(kk,2)==0),   posicion=filas*kk+1-jj; else posicion=contador;
% % %        end
% % %        subplot(filas,filas,posicion),imshow(img1) ;
% % ============================================
%        Tomando el recorte de cada imagen.
      nombre1 = char(archivo(contador));  img1 = imread(nombre1);
      img1=img1(difer(1):tama(1)-difer(1),difer(2):tama(2)-difer(2),:);
% % ============================================
%        Composición horizontal:
      if (mod(kk,2)==0),  image0 = [img1 image1]; image1=image0;
      else image0 = [image1 img1]; image1= image0;
      end
      if jj==1
         primera=1+filas*(kk-1); nombre1 = char(archivo(primera));
         img1 = imread(nombre1);
%           Imagen Inicial en cada fila del mosaico
         image1=img1(difer(1):tama(1)-difer(1),difer(2):tama(2)-difer(2),:);
      end
   end
% % ============================================
%    Composición vertical
   if (kk<2),  image2=image1; else image02=[image2 ; image1]; image2 = image02
;
   end
end
```

```
%
% % ==========================================
toc; % % Mosaico Generado, fin del contador, mostrar la composición.
figure, imshow(image2)
% % Guardar Imagen:
nombre=input('Nombre para guardar imagen? (no = cancelar )  ','s');
if (strcmp(nombre,'no') == 0) , imwrite(image2,strcat([nombre,'.jpg']),'Quality',100) ;
end
%
% % ==========================================
```

Anexo B

- Algoritmo en MATLAB **Mosaico_SIFT.m**

```
% % ==========================================
%
%         Mosaico_SIFT
%         ===========
%
%         Este algoritmo está basado en un conjunto de VL_function, de
%         distribución gratuita bajo licencia BSD, dentro de ellas se
%         encuentra la función sift.m, no solo para MATLAB sino tambien
%         para Octave y otros lenguajes.
%
% % ==========================================
function img0=Mosaico_SIFT
close all force; clc;
[FileName,PathName] = uigetfile('*.*','Seleccione las Imagenes:','MultiSelect', 'on');
archivo = fullfile(PathName, FileName); [~,n]=size(archivo);
if (iscell(archivo)==0), disp('Se necesita más de 1 imagen---');    return; end
disp(' --- Programa Mosaico:   --- ')
nombre1 = char(archivo(1)); img1 = imread(nombre1);
% tic;
   if n>2
      for ii=2:n
         disp([' Procesando ',num2str(ii),' de ',num2str(n)])
         nombre = char(archivo(ii)); img2 = imread(nombre);
         img1=mosaico_2_imagenes(img1, img2);
      end
      img0 = img1;
   else
      nombre2 = char(archivo(2)); img2 = imread(nombre2);
      img0=mosaico_2_imagenes(img1, img2);
   end
% toc;
figure, imagesc(img0)
% % Guardar Imagen:
nombre=input('Nombre para guardar imagen? (no = cancelar )  ','s');
if (strcmp(nombre,'no') == 0) , imwrite(img0,strcat([nombre,'.jpg']),'Quality',100) ;
end
if nargout == 0, clear img0 ; end
% % ==========================================
end
```

```
% % ==========================================
function mosaic=mosaico_2_imagenes(im1, im2)

% tic;
disp('      - Convertir la imagen a presición Simple. ')
mn = 100 ; % Calidad para la Homografia, 10 es buen valor experimental
% Convertir la imagen a presición Simple.
im1 = im2single(im1) ;
im2 = im2single(im2) ;
% Convertir a Escala de Grises
if size(im1,3) > 1, im1g = rgb2gray(im1) ; else im1g = im1 ; end
if size(im2,3) > 1, im2g = rgb2gray(im2) ; else im2g = im2 ; end
% ---------------------------------------------------------------------
%                 - SIFT: características
% ---------------------------------------------------------------------
disp('      - SIFT: características ... ')
[f1,d1] = vl_sift(im1g) ;
[f2,d2] = vl_sift(im2g) ;
[matches, score] = vl_ubcmatch(d1,d2) ;
numMatches = size(matches,2) ;
X1 = f1(1:2,matches(1,:)) ;
   X1(3,:) = 1 ;
X2 = f2(1:2,matches(2,:)) ;
   X2(3,:) = 1 ;
% ---------------------------------------------------------------------
%                 - RANSAC: Modelo con hommografía
% ---------------------------------------------------------------------
disp('      - RANSAC: Modelo con hommografía... ')
clear H score ok ;
for t = 1:mn
  % estimate homograpyh
  subset = vl_colsubset(1:numMatches, 4) ;
  A = [] ;
  for i = subset
    A = cat(1, A, kron(X1(:,i)', vl_hat(X2(:,i)))) ;
  end
  [U,S,V] = svd(A) ;
  H{t} = reshape(V(:,9),3,3) ;
  % score homography
  X2_ = H{t} * X1 ;
  du = X2_(1,:)./X2_(3,:) - X2(1,:)./X2(3,:) ;
  dv = X2_(2,:)./X2_(3,:) - X2(2,:)./X2(3,:) ;
  ok{t} = (du.*du + dv.*dv) < 6*6 ;
```

```
  score(t) = sum(ok{t}) ;
end
[score, best] = max(score) ;
H = H{best} ;
ok = ok{best} ;
% ---------------------------------------------------------------------
%                    - Visualizacion las Caracteristicas
% ---------------------------------------------------------------------
disp('      - Visualizacion las Caracteristicas ... ')
dh1 = max(size(im2,1)-size(im1,1),0) ;
dh2 = max(size(im1,1)-size(im2,1),0) ;
% figure(1) ; clf ;
figure % Para visualizar el avance del proceso.
subplot(2,1,1) ;
   imagesc([padarray(im1,dh1,'post') padarray(im2,dh2,'post')]) ;
   o = size(im1,2) ;
   line([f1(1,matches(1,:)); f2(1,matches(2,:))+o],[f1(2,matches(1,:));
f2(2,matches(2,:))]) ;
   title(sprintf('%d Parejas Probables ', numMatches)) ; axis image off ;
subplot(2,1,2) ;
   imagesc([padarray(im1,dh1,'post') padarray(im2,dh2,'post')]) ;
   o = size(im1,2) ;

line([f1(1,matches(1,ok));f2(1,matches(2,ok))+o],[f1(2,matches(1,ok));f2(2,matches(
2,ok))]) ;
   title(sprintf('%d (%.2f%%) parejas alineadas de
%d',sum(ok),100*sum(ok)/numMatches,numMatches)) ;
   axis image off ;
   drawnow ;
% ---------------------------------------------------------------------
%                    - Creación del Mosaico
% ---------------------------------------------------------------------
disp('      - Creación del Mosaico ...  ')
box2 = [1  size(im2,2) size(im2,2)  1 ;
        1  1           size(im2,1)  size(im2,1) ;
        1  1           1            1 ] ;
box2_ = inv(H) * box2 ;
box2_(1,:) = box2_(1,:) ./ box2_(3,:) ;
box2_(2,:) = box2_(2,:) ./ box2_(3,:) ;
ur = min([1 box2_(1,:)]):max([size(im1,2) box2_(1,:)]) ;
vr = min([1 box2_(2,:)]):max([size(im1,1) box2_(2,:)]) ;

[u,v] = meshgrid(ur,vr) ;
```

```
im1_ = vl_imwbackward(im2double(im1),u,v) ;

z_ = H(3,1) * u + H(3,2) * v + H(3,3) ;
u_ = (H(1,1) * u + H(1,2) * v + H(1,3)) ./ z_ ;
v_ = (H(2,1) * u + H(2,2) * v + H(2,3)) ./ z_ ;
im2_ = vl_imwbackward(im2double(im2),u_,v_) ;

mass = ~isnan(im1_) + ~isnan(im2_) ;
im1_(isnan(im1_)) = 0 ;
im2_(isnan(im2_)) = 0 ;
mosaic = (im1_ + im2_) ./ mass ;
end
% % ==========================================
```