PROCESAMIENTO DIGITAL DE IMÁGENES APLICADO AL ANÁLISIS DENDROCRONOLÓGICO EN ALGARROBO

Mónica Portal-Amaya

Piura, enero de 2019

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Mecánico-Eléctrica

Portal, M. (2019). *Procesamiento digital de imágenes aplicado al análisis dendrocronológico en Algarrobo* (Tesis para optar el título de Ingeniero Mecánico-Eléctrico). Universidad de Piura. Facultad de Ingeniería. Programa Académico de Ingeniería Mecánico-Eléctrica. Piura, Perú.

U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA

Procesamiento digital de imágenes aplicado al análisis dendrocronológico en Algarrobo

Tesis para optar el Título de
Ingeniero Mecánico Eléctrico

Mónica Betzabé Portal Amaya

Asesor: Dr. Rodolfo Rodríguez Arisméndiz, Dr. Cesar Chinguel Arrese

Piura, Enero 2019

Prólogo

En el presente trabajo, tesis de pregrado "Procesamiento digital de imágenes aplicado al análisis dendrocronológico en Algarrobo", se busca crear una herramienta que facilite la obtención de datos para los estudios dendrocronológicos en Algarrobo que lleva a cabo la Universidad de Piura en el área de climatología. Los motivos por los que se eligió este tema son el aprendizaje complementario en procesamiento de imágenes, el innovar al crear un programa para una aplicación específica y el aportar en la optimización de los métodos para los estudios dendrocronológicos en Algarrobo.

La importancia del presente trabajo reside en el avance de las técnicas de obtención de datos y resultados, y proseguir con los estudios de las relaciones planteadas entre la estructura de la madera y las variables ambientales. Al tener un software que automatice los procesos de identificación y se calcule los datos necesarios, se obtiene una ampliación del alcance de dichos estudios. Por lo tanto la realización de mi tesis contribuye en el mejoramiento de las técnicas aplicadas para los estudios y es un punto de partida para posteriores innovaciones.

Quisiera agradecer por el desarrollo de esta herramienta a mi asesor el Dr. Rodolfo Rodríguez Arisméndiz y a mi co-asesor Dr. Cesar Chinguel Arrese, por su continua guía y apoyo. Esta tesis se desarrolla como parte del proyecto "Estudio de indicadores biológicos en el algarrobo como un registro paleoclimático: fundamentos y uso de nuevos métodos de medición dendrocronológica", y al igual que la presente tesis; ha sido financiado por el Programa Nacional de Innovación para la Competitividad y Productividad (Innóvate Perú), de acuerdo al contrato 404-PNICP-PIBA-2014. Agradezco el apoyo y por confiar en los resultados positivos del presente trabajo.

Resumen

El análisis de imágenes en la dendrocronología se desarrolla como una herramienta para la obtención de datos a utilizar en los estudios de las relaciones entre las características anatómicas de la madera y las variables ambientales. Para desarrollar un asistente de obtención de información, se busca el mejor enfoque y método a aplicar. Por lo que se tiene en cuenta las aplicaciones generales que ofrece el software para crear un programa que ejecute la aplicación específica.

HDevelop es un software de *machine vision*, en el cual se aplican los métodos adecuados para lograr desarrollar el proceso deseado, junto con la librería HALCON. Los dos procesos principales son la identificación de vasos en las secciones de la madera y la identificación aproximada de los límites de anillo en Algarrobo.

El método utilizado para la identificación de vasos, es elegido debido a las pocas limitaciones en la ejecución del programa y así proseguir con la extracción de características, según sea los estudios dendrocronológicos en Algarrobo.

Índice

Introducción

El análisis de imágenes para estudios dendrocronológicos se viene desarrollando como una técnica para la obtención de datos como la distancia entre los anillos, la densidad de la madera, entre otras variables. Para mejorar la precisión y rapidez en el análisis de la imagen se utiliza un escáner de sobremesa(García, García, & Díaz, 1968) o por una cámara instalada en un estereoscopio. Sin embargo los softwares utilizados en estudios iniciales como *WinDENDRO*, el sistema de análisis de imagen de luz reflejada y el sistema *TREES* poseen limitaciones(Conner, 1999).

Como parte del desarrollo de una herramienta para el avance en los estudios dendrocronológicos en Algarrobo, la tesis se divide en tres capítulos: el primer capítulo desarrolla la descripción de la dendrocronología, sus aplicaciones, las características de interés de la madera para esta ciencia, y el desarrollo de esta en la especie especifica del Algarrobo; el segundo capítulo explica el procesamiento de imágenes y sus métodos para obtener la data deseada, también describe los softwares que se han utilizado y se utilizan para la dendrocronología; el tercer capítulo implica la descripción del software HDevelop para el desarrollo de la aplicación específica y los procesos que se llevan a cabo para la identificación de la características anatómicas en las secciones del Algarrobo como los vasos y los límites de anillo. Es decir que el enfoque se desarrolla desde el conocimiento básico de la dendrocronología en general y en la especie de interés, siguiendo por el conocimiento del análisis de imagen como herramienta en general y el desarrollo en la dendrocronología, hasta la explicación de la realización de la herramienta aplicado a dos procesos por medio del software HDevelop.

Los resultados se enfocan en la exactitud en la cual se puede realizar los dos procesos y proseguir con el cálculo de los datos que se requieren, según sea los estudios a realizarse en la dendrocronología. El alcance es lograr una herramienta factible, sin limitaciones en el ámbito de tiempo ejecución del programa, cantidad de imágenes para ser analizadas, y obtención de datos de manera automática.

Capítulo 1
Dendrocronología en Algarrobo

1.1. Definición de dendrocronología

La dendrocronología es la ciencia que se basa en la información estructural registrada de la planta para estimar los entornos probables del pasado. También conocida como la ciencia de la datación de los anillos de los árboles, ya que la coincidencia de patrones en anillos de estos se convirtió en un principio fundamental de esta ciencia, provee diversas técnicas para precisar esta información proveniente de dos tipos de estudios(Gartner, Aloni, Funada, Lichtfuss-Gautier, & Roig, 2002): Los estudios descriptivos, que describe patrones, que pueden ser a corta escala, larga escala, respuestas fenotípicas, respuestas genotípicas, respuestas de la comunidad, respuestas de población y respuestas de evolución; y los estudios mecánicos que describe también patrones y además busca las causas probables de estos.

1.2. Aplicaciones de la dendrocronología

La dendrocronología establece relaciones entre la data registrada en la planta y las variables ambientales tales como la temperatura y la precipitación. Dependiendo del estudio se elige los controles internos en las cuales la naturaleza y la robustez de estas relaciones puedan ser estudiadas.

1.2.1. Aplicaciones Climáticas o Paleo climáticas

El análisis de los anillos de los árboles es un enfoque alternativo para estimar retrospectivamente las tasas de crecimiento anual de especies en grandes paisajes relacionados directamente con los registros climáticos de su ubicación geográfica. Es decir, se establece la relación de crecimiento radial-clima, y se estandarizan las series de crecimiento para minimizar las tendencias atribuidas a la edad de los árboles individuales para finalmente convertirlas en series estacionarias y poder compararse entre sí. Por lo tanto, se obtiene una señal común o señal maestra, y de esta manera lograr la elaboración de cronologías.

1.2.2. Aplicaciones Ecológicas

Las aplicaciones ecológicas consisten en establecer relaciones entre la estructura de la madera o propiedad anatómica con factores que influyen en los ecosistemas como por ejemplo estudios la transición de las traqueidas a los vasos y fibras, el acortamiento y ensanchamiento de los vasos, el efecto del suelo fértil en el crecimiento, el efecto de la temperatura y precipitación en el ancho de la paredes celulares, tendencias ecológicas en la anatomía de la madera por clima y altitud para una variedad de especies en una cierta ubicación geográfica.

1.3. Anatomía de la madera

La madera es un material heterogéneo y anisotrópico, es decir que sus propiedades varían según la dirección en la cual son estudiadas, además de considerarse como la parte más sólida aquella debajo de la corteza. Su estudio permite dar apoyo a otras ramas como es el caso de la dendrocronología. La variación de su estructura se debe a factores intrínsecos o genéticos y factores extrínsecos o ambientales y el peso de cada uno difiere en cada especie y género. Para su estudio se tienen en cuenta diferentes cortes:

- Corte transversal: es el corte perpendicular al eje del árbol.
- Corte radial: paralelo a los radios o perpendicular a los anillos de crecimiento.
- Corte tangencial: tangencial a los anillos de crecimiento o perpendicular a los radios.

1.3.1. Estructura macroscópica del tronco

1.3.1.1.Corteza

La corteza está formada por una corteza interior llamada floema, compuesta por tejidos vivos especializados en la conducción de la savia elaborada. La corteza exterior o ritidoma o corteza muerta está formada por tejido que reviste el tronco y lo protege contra agentes externos como el desecamiento, ataques fúngicos o fuego.

La corteza tiene una gran importancia en la identificación de árboles y su estudio permite la diferenciación entre individuos semejantes. Otra capa que no es importante a nivel macroscópico es el cambium y tiene un ancho de una sola célula, su importancia en la anatomía de la madera reside en que sus células generan nuevos tejidos y de esta manera aumenta el diámetro del tronco.

1.3.1.2.Xilema

El xilema es la madera propiamente dicha conformada por dos partes diferenciadas a nivel fisiológico y en algunas especies a nivel macroscópico, la parte externa de color más claro es la albura y la parte interna y color oscuro es el duramen.

La albura es la parte activa, es decir representa el área donde se realiza la mayor cantidad de actividad fisiológica. Está conformada por células vivas y se

responsabiliza de la conducción ascendente de agua y sales, además de darle rigidez al tallo y ser reservorio de nutrientes.

El duramen se forma debido al proceso de maduración de las células en la albura en dirección a la medula, lo que implica la pérdida de la actividad fisiológica. Tiene una función de sostén y adquiere generalmente una coloración oscura por la transformación de las sustancias de reserva. Debido a esto es más compacto y resistente a ataques de hongos e insectos con una durabilidad superior a la albura.

1.3.1.3.Radios o líneas horizontales

Son células dispuestas en forma de bandas que se disponen de manera radial en dirección medula-corteza, además tienen un largo indeterminado y principalmente parénquimaticas. Una de sus funciones es almacenar sustancias nutritivas y también realizar el transporte horizontal de nutrientes.

1.3.1.4.Médula

Es un pequeño núcleo central cuya función es almacenar las sustancias nutritivas en los primeros periodos de crecimiento del árbol; y en comparación con los otros tejidos del tronco tiene menor densidad, además de presentar mayor susceptibilidad ante el ataque de hongos e insectos. También es una parte del tronco que ayuda a diferenciar e identificar especies a nivel macroscópico.

1.3.1.5.Anillos de crecimiento

Una capa de tejido leñoso formada en un periodo de crecimiento, constituye un anillo. En el caso de regiones con clima templado, con cuatro estaciones bien diferenciadas, se da la existencia de un periodo de crecimiento en un año, por lo que los anillos representan habitualmente un incremento anual en el árbol.

En el caso de climas tropicales, con periodos de lluvia y sequía en diferentes lapsos durante el año, puede provocar la aparición de dos o más anillos durante el año; por lo tanto la contabilización de los anillos puede no representar la edad exacta del árbol.

1.3.2. Tipos de madera

1.3.2.1.Madera de Coníferas

La madera de coníferas presente una constitución anatómica más simple, y está constituida por células homogéneas, del grupo de las traqueidas, las cuales realizan la doble función de sostén y conducción.

Los radios se visualizan como líneas delgadas en el plano transversal y son poco notorios en los demás cortes. En el caso de los anillos de crecimiento, presentan un leño temprano de coloración clara, debido a la presencia de traqueidas con paredes finas y lúmenes grandes; y un leño tardío con una coloración oscura, debido a la presencia de traqueidas con paredes gruesas y lúmenes estrechos.

El parénquima axial o longitudinal es difícil de ver en el corte transversal en el caso de las coníferas, debido a que las células del tejido son muy pequeñas y de paredes delgadas. Luego están los canales de resina definidos como espacios intercelulares especializados en la producción de resina. A nivel macroscópico se ven en el corte longitudinal y en el transversal; formando parte de los radios, por eso se les llama radios fusiformes.

1.3.2.2.Madera de latifoliadas

Las maderas pertenecientes a este grupo presentan una gran variedad de elementos anatómicos, lo cual resulta en una descripción más compleja. A escala macroscópica están los vasos o poros, que en el plano longitudinal son elementos o líneas vasculares, luego están el parénquima, el parénquima radial o radio y las fibras.

En el caso de los anillos de crecimiento, el difícil diferenciar en este tipo de madera el leño inicial y final en el interior del anillo; por lo tanto se describe la notoriedad de ellos por otras características y elementos. Hay dos grandes grupos en las maderas de latifoliadas con respecto a los anillos:

- Los anillos ausentes o indistintos: estos anillos no están, o se delimitan por cambios estructurales muy graduales por lo que es imposible o muy difícil de observar a nivel macroscópico.
- Los anillos distintos: estos anillos están presentes, es decir son fáciles de observar a simple vista o con aumento; ya que están delimitados por otras características estructurales.

Las características que permiten la diferenciación de los anillos en la madera son los siguientes:

- Parénquima marginal: Es una banda de células de parénquima en el límite del anillo, a nivel macroscópico se ve como una línea tenue de color más claro.
- Porosidad semicircular: Se observa una concentración de vasos al inicio del anillo de mayor diámetro, luego hay un cambio gradual en cantidad y tamaño al final del anillo.
- Porosidad circular: Se observa una concentración de vasos al inicio de anillo y de tamaño grande y luego hay un cambio abrupto en cantidad y tamaño al final.
- Mayor espesor de la pared de las fibras y disminución de su diámetro radial: Se observa en plano transversal como una banda de coloración oscura, parecido a la formación del leño final en las coníferas.
- Alteración en el espaciamiento entre bandas de parénquima axial: Se observa un cambio del ancho de manera abrupta entre las bandas que han seguido un patrón de ancho.

En el plano del corte transversal, como se ha mencionado antes, los elementos vasculares se llaman vasos o poros, y su distribución, cantidad, tamaño y agrupamiento son características importantes no solo para la identificación de

especies; sino también para iniciar otros estudios acerca del funcionamiento de la conectividad en el árbol. Existen varios tipos de agrupamiento, observados en el plano, y son los siguientes:

- Solitarios: son aquellos vasos que están rodeados de elementos no vasos en un porcentaje del 90% o más.
- Múltiples: son elementos que conforman la unión de dos poros, conectados por sus caras tangenciales y presentan un achatamiento; lo cual se observa como subdivisiones de un solo poro.
- Arracimados o agrupados: son elementos conformados por tres o más vasos, unidos por sus caras radiales o tangenciales.

La orientación de los poros o vasos puede variar; por lo que puede ser radial, paralela a los radios; también puede ser tangencial, paralela a los anillos de crecimiento y perpendicular a los radios, formando bandas cortas o largas; y tener orientación diagonal, un intermedio entre radial y tangencial con una pendiente. Estas orientaciones se presentan en la madera de manera combinada, como por ejemplo vasos orientados de manera tangencial o diagonal.

Otra característica de la madera de latifoliadas es la porosidad, definida por la distribución, tamaño y cantidad de poros. La porosidad difusa se da por la distribución y el tamaño uniforme de los vasos a los largo de la sección; también se presenta la porosidad semicircular y la porosidad circular, mencionadas anteriormente como características para identificación de límites de anillos de crecimiento.

1.4. Desarrollo de la dendrocronología en el Algarrobo

El género Prosopis implica 44 especies distribuidas en sur-oeste de Asia, África y predominantemente América, principalmente en regiones calurosas y secas del oeste(Pometti et al., 2009). Muchas de estas especies son usadas para la reforestación, debido a que cumplen un importante rol en contra de la desertificación de los suelos erosionados.

La especie Prosopis Pallida o comúnmente conocida como Algarrobo, ubicada principalmente en las zonas áridas y semiáridas de América Latina, tiene anillos anuales de crecimiento diferenciados, por lo cual son factibles los estudios dendrocronológicos. Al proseguir con estos estudios se establece que la cronología de los anillos de los arboles correlaciona correctamente los eventos de las fases de oscilación del fenómeno del niño, fenómeno que ocurre en la zona norte del Perú. La dendrocronología en el algarrobo discute las variaciones fisiológicas de la anatomía de su madera y el patrón cronológico que se construye con esta data.

Para que los anillos de los arboles sean útiles para la determinación de edades, las plantas deben producir distintos anillos de crecimiento anuales. Por lo tanto, si las técnicas de dendrocronología están para ser aplicadas, es necesario verificar la naturaleza del anillo en formación. En el caso de la Prosopis Pallida, los estudios realizados por la institución Universidad de Piura en el año 2002(López, Sabaté, Gracia, & Rodríguez, 2005), afirman que la formación de los anillos de esta especie se debe a la estacionalidad

marcada por las precipitaciones en el norte del Perú. Esto se verifica al demostrar una correlación entre los anillos de crecimiento anual y las variables ambientales como temperatura y precipitación.

1.4.1. Materiales y métodos

Para analizar la estructura de la madera del algarrobo, es decir analizar su anillado y otras características presentes, existen en la dendrocronología dos fases. La primera fase es la de campo y la segunda es la del laboratorio.

1.4.1.1.Fase de campo

En esta fase se realizan medidas dendrometrícas en troncos de árboles vivos y la obtención de muestras de testigos o secciones de troncos de árboles vivos o muertos.

Para realizar medidas dendrometrícas, se utiliza un instrumento mecánico que en esencia es un micrómetro y se instala sobre el tallo teniendo en cuenta un punto de medida; de esta manera se miden los incrementos radiales de forma periódica, y así determinar de esta forma la relación con los cambios del ambiente como el clima y las plagas.

En el caso de la obtención de muestras, se comienza con la selección del lugar. Para el caso de la Prosopis Pallida, el criterio para escoger el lugar es que la respuesta ante factores climáticos sea máxima, por lo tanto para estudiar la relación con las precipitaciones se buscan lugares secos o donde el agua sea un factor limitante. También se debe verificar la capacidad de réplica, para que los arboles muestreados tengan una señal similar. Además la distribución geográfica de los diferentes bosques secos es importante para identificar los lugares de mayor interés. Por lo tanto para la selección del lugar se considera lo siguiente:

- Información de los diferentes tipos de bosques secos en la región y sus especies predominantes.
- Densidad de cada tipo de especie, para identificar la mayor cantidad de individuos de la especie.
- Identificación de la antigüedad de las diferentes poblaciones por medio de una clasificación de diámetros.
- Referencia de lugares que hayan sido objeto de estudios previos.
- Referencia sobre factores cambiantes como la poda, contaminación o depredadores de la especie.

Luego de la selección del lugar, se eligen 10 ejemplares de cada localidad para posteriormente obtener las secciones. Los árboles se cortan a la altura de 1.30 m aproximadamente, y de ese tronco se cortan dos secciones de 5 cm de grosor. Finalmente las muestras se rotulan con la fecha, lugar, especie y autor del muestreo y de esta manera ser llevadas al laboratorio. (Palacios, 2018).

1.4.1.2.Fase de laboratorio

Un laboratorio de dendrocronología está equipado con un estereoscopio, un escáner de mesa, un computador, así como lijadoras, pulidoras e insumos como lijas de diferentes granulometría.

El trabajo en laboratorio consiste normalmente en la preparación de las muestras, la obtención de imágenes, su análisis para la obtención de las series de anillos y también la utilización de software para obtención de otras características anatómicas de la madera. Por último se realiza el procesamiento de las series para la obtención de series maestras y el procesamiento de otros datos de interés.

Para la preparación de las muestras, se requiere una preparación de superficie para que las características del tronco sean más visibles. Las muestras después de secarse se fijan en una base y se pulen con lijas de diferente granulometría, en el caso de las secciones se usan lijas con granulometría entre 24-2500.

En cada sección preparada se marcan tres radios en la dirección de parénquima radial desde la corteza hasta la medula. Luego con un estereoscopio se identifican los límites de anillos de crecimiento y se fechan. En las secciones de algarrobo se realizan dos procesos de análisis, uno es para la obtención de las series de anillos y otro es para el análisis de los vasos conductores entre los límites de los anillos de crecimiento.

En el caso de análisis de los vasos conductores entre los límites de anillos se presenta el esquema en la Figura 1.

En el caso del proceso de identificación de anillos, se realiza el conteo desde el anillo más joven hasta el anillo más antiguo. Este conteo se realiza en cada radio trazado de la misma sección para descartar los anillos falsos o ausentes. También se realiza una calibración que consiste en identificar un anillo de un ancho de 1mm con una lámina graduada. Este anillo seleccionado se emplea para la calibración posterior. En el esquema en la Figura 2 se explica parte del procesamiento digital para la obtención de las series de anillos.

La imagen mejorada se carga en el programa *CooRecorder 7.8.1-Cybis electronik & data AB*; en este programa se identifica las propiedades del archivo y se selecciona el DPI adecuado para la calibración. Al calibrar la imagen se selecciona el anillo de referencia y se establece una escala de milímetros por pixeles con la opción *calibrate.* Luego se mide el ancho de los anillos siguiendo la trayectoria de uno de los radios trazados y se registra las coordenadas de la ubicación del anillo con la herramienta *datos*. Al final la imagen y los datos se guardan en forma de coordenadas y luego en forma de grosores de anillos en el programa *CDendro 7.8.1-Cybis electronik & data AB*. Así se forma una serie individual de una sección o muestra.

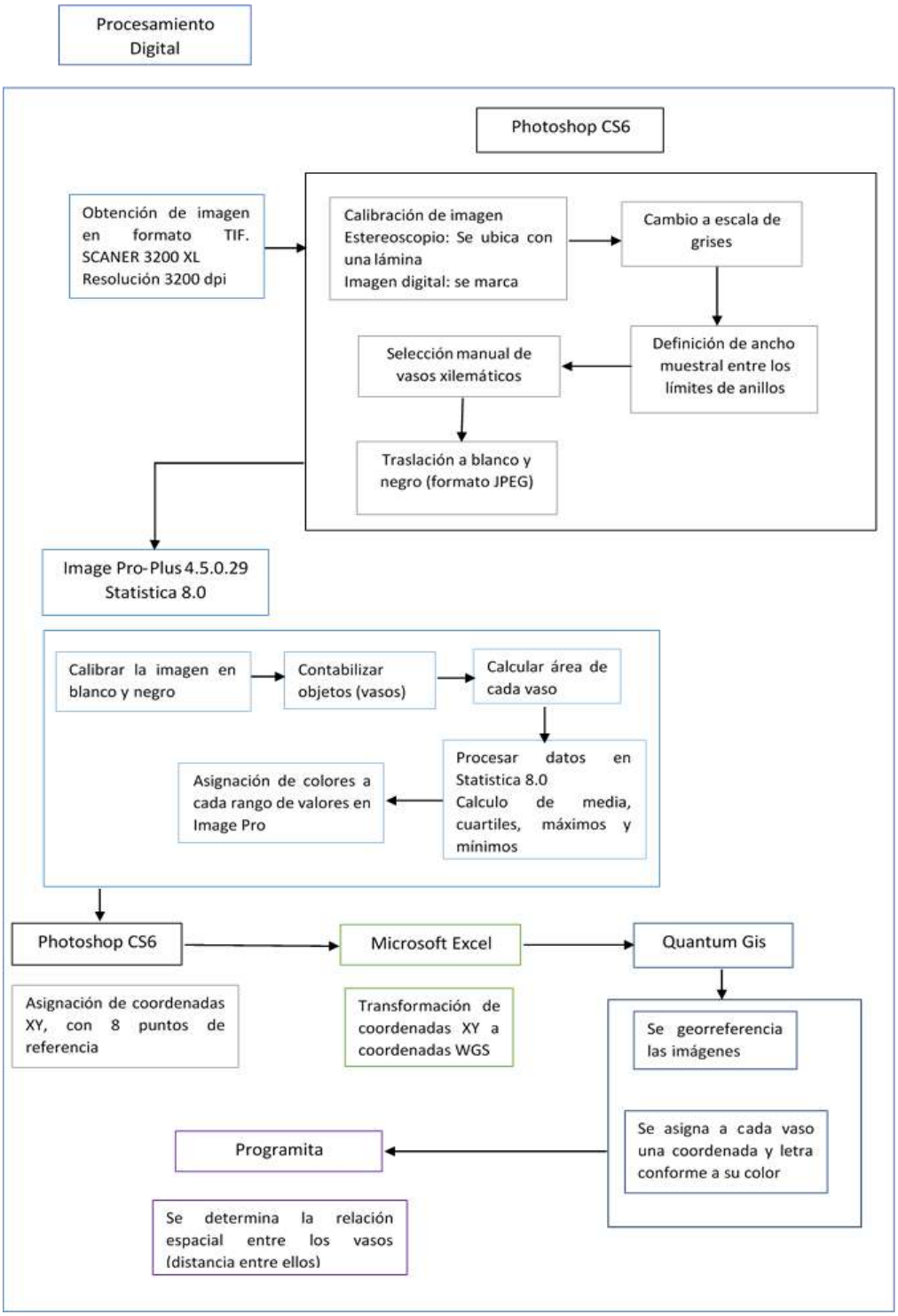


Figura 1. Esquema del procesamiento digital del análisis de vasos.
Fuente: Tesis "Relación entre la distribución intra-anual de los vasos xilemáticos en la madera de *Prosopis* sp. (Algarrobo) y la variabilidad climática durante un año de evento El Niño", Elva Palacios McCubbin, 2018. Elaboración: Elaboración Propia.

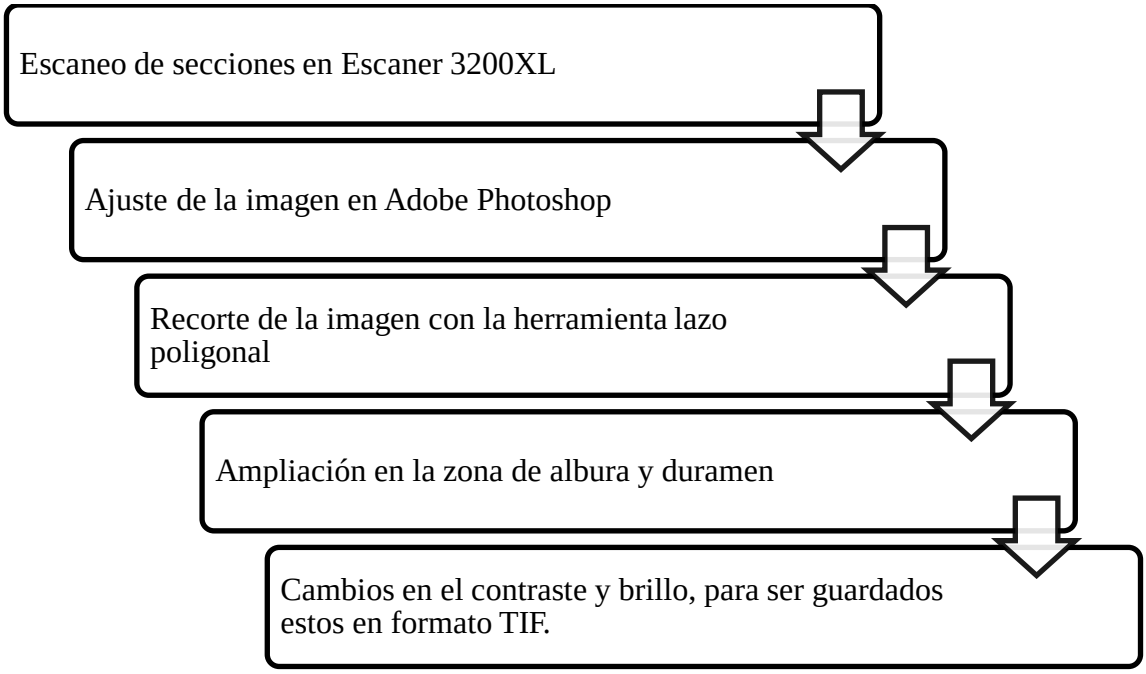


Figura 2: Esquema del procesamiento digital inicial para la obtención de series. Fuente: Tesis "Dendrocronología de *Prosopis sp.* en la región Piura", Edwin Ancajima More, 2017. Elaboración: Elaboración propia.

Las series individuales en forma de grosores de anillos se agrupan conforme a su localidad o población y se cofechan en el programa *CDendro 7.8.1*, es decir se cargan en el programa y se solapan estas series, considerando la alineación de los picos más pronunciados. Los parámetros a tener en cuenta en la alineación son el *offset*, se define como la distancia en años entre el último anillo de dos muestras, y el coeficiente de correlación. Este coeficiente se mejora al agregar o eliminar anillos de crecimiento según sea el seguimiento de la madera. También se descartan radios debido a que la correlación es muy baja con respecto a la colección en cada localidad. Esta colección al final se guarda en formato decanal para proseguir con el desarrollo de la cronología.

En el desarrollo de la cronología que es el cálculo del índice del ancho de los anillos, se estandarizan las series individuales pertenecientes a una colección poblacional. Esta estandarización se realiza con el paquete "Dendrochronology Program Library in R" (Dp1R) del programa R Studio. El proceso consiste en eliminar los factores no relacionados al cambio climático; para maximizar la variabilidad de alta frecuencia, comúnmente asociada a factores ambientales, y finalmente determinar la correlación final entre las series individuales.

Después los índices de crecimiento se calculan para cada año mediante la relación entre el valor medido y el valor de la serie de ajuste para el mismo año. Luego estos índices se pre-blanquean, eliminando la autocorrelación de la serie con modelos autorregresivos para mantener la varianza semejante y uniforme. Por último, se promedian los índices año por año para construir una cronología media; para ello se utiliza un método llamado "la media robusta", la cual minimiza la influencia de valores extremos en la derivación de la cronología.

Asimismo la calidad de las cronologías de cada localidad se determina calculando lo siguiente con la librería Dp1R:

- Promedio del ancho de anillos
- Sensibilidad media: compara la variación del ancho de los anillos de un año respecto a otro de los árboles, para saber si se vincula a variaciones del clima.
- Autocorrelación: es el grado de dependencia del crecimiento respecto al año anterior de una misma serie.
- Desviación estándar: mide la dispersión de los datos alrededor del promedio del ancho de los anillos.
- Correlación entre series: mide la asociación entre las series de crecimiento de distintos árboles.

Capítulo 2
Procesamiento digital de imágenes en dendrocronología

2.1. Procesamiento digital de imágenes

Los datos visuales son la información sensorial más compleja y más útil para los humanos. La percepción artificial visual se refiere a la interpretación artificial de la información visual, ya que hay muchas similitudes en las formas en que se perciben los datos sensoriales.

2.1.1. Representación digital de una imagen

Una imagen puede ser como una función de intensidad de luz en cada punto encima de la región del plano. Para operaciones digitales, se necesita la muestra de esta función en intervalos discretos y cuantificar la intensidad en niveles discretos. Los puntos a los cuales la imagen es muestreada son conocidos como elementos de visión o comúnmente nombrados como pixeles. La intensidad de cada pixel se representa por un entero, siendo 0 para negro y 255 para blanco, y es determinada a partir de una imagen continua promediando un pequeño vecindario alrededor de la locación del pixel.

La topología y geometría de un plano digital difiere del dominio continuo en varios aspectos importantes. Mientras que líneas horizontales y verticales se representan fácilmente en la red cuadrada, las líneas rectas a diferentes ángulos son aproximadas por un patrón escalonado.

2.1.2. Percepción de brillo y color

La percepción del brillo de una parte de una imagen está influenciada por los valores de iluminación de sus partes cercanas. Otro efecto importante es la relativa constancia de la luz y colores percibidos de una superficie bajo diferentes condiciones de iluminación. Por lo tanto la constancia de brillo y color son propiedades deseables para la percepción artificial.

2.1.2.1.Modelo del ojo humano

Un simple modelo del ojo consistiría en un dispositivo de lente que forma la imagen en la retina, en la cual la superficie de la retina es más cercano al plano esférico, y la apertura del lente es ajustada por el iris. Luego están los receptores que son los que miden la adaptabilidad ante los cambios de intensidad de luz como la noche y el día.

La retina tiene dos tipos de receptores, conos y bastones, los conos son responsables de la visión del color y los bastones son sensibles ante el cambio de luz. La mayoría de los conos se concentran en la región central, conocida como fóvea. Esta se enfoca en diferentes partes de las escenas aparentemente pseudo-aleatorias, pero se concentra en las partes de interés. Estas elecciones son basadas en un procesamiento más complejo. La resolución espacial de sistema humano es increíblemente alta y la sensibilidad a luz de los bastones es alta, ya que un fotón de luz puede activar un bastón.

2.1.2.2.Mediciones locales

La medición local del brillo de una pequeña área de una imagen que es hecha por un sensor es proporcional al promedio de la intensidad de luz en los alrededores. Es decir, el brillo percibido depende la intensidad de los pixeles cercanos, incluso en un contexto global.

El sistema humano es capaz de distinguir entre dos superficies homogéneas de diferentes colores con tres atributos diferentes: Intensidad, matiz y saturación. La intensidad es usada para imágenes monocromáticas, el matiz es el sentido del color y la saturación es la medida de la cantidad de blanco combinada con un matiz puro (Nevatia, 1982).

Para fuentes de luz monocromáticas, la percepción de matiz es directamente dependiente de la longitud de onda. Por lo que diferentes matices son obtenidos por la mezcla de colores. Es bien conocido que todos los colores pueden ser sintetizados por la mezcla de tres colores primarios que son el rojo, verde y azul. Pero no son la única opción de primarios. Estos colores primarios tienen un amplio espectro de intensidad de luz.

Los atributos de intensidad, matiz y saturación son inferidos de los valores de brillo de los tres primarios. Donde la intensidad es medida como:

$$I = c_1 R + c_2 G + c_3 B$$

Las constantes dependen de la opción de colores primarios que se elija y pueden ser determinados por las medidas hechas en una superficie blanca.

Para calcular los componentes de matiz y saturación, es conveniente introducir dos atributos adicionales T1 y T2:

$$T_1 = \frac{R}{R + G + B}$$

$$T_2 = \frac{G}{R + G + B}$$

El T1-T2 plano contiene toda la información cromática relacionada al no brillo. Los tres colores primarios son tres puntos en el plano T1-T2 y forman el conocido triangulo de color. El blanco es representado por el punto W (1/3, 1/3). Para el punto P en el triángulo de color, su distancia Y desde el punto W, determina la saturación y el ángulo Θ de la línea de P a W con el eje T1 determina el matiz.

2.1.2.3.Inhibición lateral y cálculo de luminosidad

El modelo más común de la interacción entre los receptores de la retina es la inhibición lateral; en el cual las salidas de un pequeño vecindario de receptores, conocido como campo receptivo, son la combinación de una suma ponderada de salidas individuales. Así los receptores en un vecindario excitatorio central contribuyen positivamente y aquellos en un vecindario inhibitorio circundante más grande tienen pesos negativos.

La inhibición lateral puede ser como el funcionamiento de una filtración 2D pasa banda de la señal de entrada, lo cual explica el fenómeno observado de bandas de Mach, un fenómeno de ilusión óptica.

La inhibición lateral es comúnmente usada para explicar el fenómeno de contraste simultáneo, lo cual es la dependencia del brillo percibido de los alrededores. Esta puede ser vista como el cálculo del ratio del brillo central con el brillo periférico, argumentos similares también se aplican para explicar la constancia del brillo, donde el brillo percibido se calcula de nuevo en relación con las otras superficies circundantes.

Una mejor explicación del contraste simultaneo y la constancia de brillo es dada por la teoría de retina. La intensidad de luz del receptor $p'(x)$, es un producto de la luz incidente $s'(x)$, y de la reflectividad de superficie $r'(x)$. Tomando los logaritmos de p', s' y r', tenemos:

$$p(x) = s(x) + r(x)$$

Donde $p(x) = \log(p'(x))$ y así sucesivamente. Después se toman las derivadas:

$$d(x) = D\big(p(x)\big) = D\big(s(x)\big) + D(r(x))$$

Si asumimos que la luz incidente $s(x)$ cambia suavemente a lo largo de la imagen, donde la reflectividad cambia abruptamente en los bordes de la imagen, entonces $D(s(x))$ es finito y $D(r(x))$ consiste en infinitos impulsos en los bordes del objeto. En el caso discreto, la derivada es aproximada a un diferencial, y la umbralización dada $D(r(x))$. La reflectividad ahora puede ser recuperada por integración del diferencial umbralizado, excepto por un término constante.

2.1.3. Detección de bordes y curvas

La detección de los bordes de los objetos es una parte importante del proceso de percepción, y las técnicas de detección desarrolladas ayudan a encontrar las discontinuidades en algún atributo de la imagen, como la intensidad y el color. Por eso es importante organizar los bordes locales en agregados que conducen a una segmentación de escenas.

2.1.3.1.Detección de bordes

Un borde ideal, en una dimensión, puede ser visto como un paso de cambio en intensidad; sin embargo en señales reales, el paso de cambio es como la combinación con 'ruido' causada por un sensor, superficie, o variaciones de iluminación. En el caso de dos dimensiones, el paso ideal ocurre a lo largo de una línea con cierta longitud, además los bordes de interés no son necesariamente limitados por pasos de bordes.

En imágenes reales, con ruido e imperfecciones en la superficie, se debe lograr un compromiso entre maximizar la detección de los bordes deseados y minimizar la detección de bordes no deseados y el ruido.

Si un operador realza los bordes en una imagen, significa que los valores de los puntos de los bordes deseados sean más altos que otros puntos, entonces la detección de bordes puede ser dada por un simple thresholding de la figura mejorada. El mejoramiento puede ser dado por cualquier operador, como los filtros lineales, sin embargo la dificultad con el uso de una técnica de filtro global reside en la opción de un adecuado realce de funciones.

La diferenciación puede ser vista como un operador de realce local de bordes. Entonces en aplicaciones de imágenes de dos dimensiones, una apropiada operación derivativa es la del gradiente.

Considere un vecindario de 3 por 3 de cierto pixel (i, j), con valores de intensidad de a_0 a a_7. Podemos definir la magnitud de la gradiente por:

$$S = \sqrt{S_x^2 + S_y^2}$$

Donde:

$$S_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$

$$S_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_7)$$

Siendo c una constante, c es escogido para ser 1 en un detector de bordes descrito por Prewitt y 2 por Sobel.

La dirección del borde Θ, es dado por la siguiente operación:

$$\theta = \tan^{-1}\left(\frac{S_x}{S_y}\right)$$

Otro enfoque para la detección de bordes es tener modelos de bordes ideales y determinar cuánto se acercan a estos modelos, dado un vecindario de una imagen. El operador Hueckel es usado para tener un modelo simplificado de un borde ideal, donde un borde con un ángulo Θ y una distancia r del centro separa dos regiones de brillo b y $b + h$. Entonces se desea calcular la función del paso ideal que armoniza mejor con la función de la imagen al variar la posición, orientación, e intensidad del paso ideal, esto es b, h, r, y Θ, comprimidos en un vector ξ. Se define la diferencia, N, entre un vecindario de una imagen y la del paso ideal por:

$$N^2 = \sum_{x \in R} (A(x) - S(x, \xi))^2$$

Donde $A(x)$ es la intensidad de la imagen en un punto x y $S(x, \xi)$ es la intensidad de una función del paso ideal por un vector escogido con parámetros ξ. La suma es calculada a través de todo el vecindario R del espacio ideal. El vector ξ es escogido de tal manera que N^2 es minimizado.

Para simplificar este cálculo de minimización, Hueckel sugiere aproximar las funciones $A(.)$ y $S(.)$ por series ortogonales $H(.)$. La serie escogida es una serie radial de Fourier donde solo las 9 primeras funciones son usadas.

$$N^2 = \sum_{i=0}^{8} (a_i - s_i(\xi))^2$$

Donde a_i y s_i son coeficientes de expansión de las siguientes series:

$$a_i = \sum_{x \in R} H_i(x) * A(x)$$

$$s_i = \sum_{x \in R} H_i(x) * S(x)$$

Una vez que ξ y N^2 son determinados, la presencia de un borde es determinada por la altura del paso requerido, h, para agrandar y N^2 para disminuirlo; y el vector ξ contiene dos parámetros adicionales, una para el valor de intensidad y otro para ancho del borde.

2.1.3.2.Detección de curvas y líneas

Un importante nivel en el procesamiento de bordes detectados en una imagen es agregar elementos de borde para formar los límites de objetos. La forma de estos bordes puede no ser conocidos a priori, pero en muchos casos pueden ser aproximados por segmentos de líneas. Sin embargo no es factible con solo arreglar un segmento de línea para todos los bordes en una imagen, primero es necesario

agregar bordes a lo largo de una sola línea u otra curva conocida. Por lo tanto, algunas técnicas son descritas:

- Transformada de Hough

Para describir esta técnica se aplica la detección de un conjunto de puntos a lo largo de una línea. La ecuación general de la línea es la siguiente:

$$xcos\theta + ysin\theta = r$$

Donde Θ es el ángulo hecho por una normal con respecto a la línea con una coordenada x y r es la longitud de esta normal, para algún punto dado (x_i, y_i) la ecuación es:

$$x_i cos\theta + y_i sin\theta = r$$

Considerando r y Θ como nuevas variables, la ecuación da la relación entre los parámetros de una línea a lo largo del punto (x_i, y_i), lo que corresponde a una curva sinusoidal en (r, θ) espacio. Esta relación entre el plano de la imagen y el plano del espacio (r, θ) es conocida como la transformada de Hough. Para un grupo de puntos colineales en el plano de imagen, las curvas son determinadas por la ecuación y deben ser intersectadas por un punto en común.

Para detectar un grupo de puntos colineales, podríamos construir las curvas transformadas en el espacio (r, θ) para cada punto y recoger los puntos correspondientes para coincidir tres o más de estas curvas. La construcción de una curva en (r, θ) requiere el incremento de contadores de células a lo largo de la curva. Para encontrar coincidencias, se escoge las células con el mayor número de contadores. Una robusta digitalización fallara en distinguir entre las líneas vecinas, y una exacta resolución que permitiría un pequeño error en colinearidad.

Este procedimiento puede ser más eficiente, si las direcciones son conocidas. Entonces, un solo punto, más que una sinusoidal, puede ser calculado en el espacio $((r, \theta)$ por un punto de borde dado.

La técnica de la transformada de Hough, puede ser generalizada para la detección de curvas arbitrarias, descritas por la función:

$$f(a_1, a_2, \dots, a_n, x, y) = 0$$

Donde $a_1, a_2, \dots, a_n$ son parámetros de la curva. Para cada punto (x_i, y_i), asociamos una superficie $f(a_1, a_2, \dots, a_n)$ en el espacio $a_1, a_2, \dots, a_n$. La complejidad del cálculo incrementa con la dimensionalidad de los parámetros de espacio.

- Técnicas graficas teóricas

Este método de detección de borde es por convolución con máscaras de forma de borde, seguido por un adelgazamiento en una dirección normal del borde. Tal detección de borde tiende a producir bordes que forman líneas

con algunos vacíos y algunas ramas conectadas. Si solo los pequeños vecindarios, por decir 8 adjuntos, son considerados como posible red, se hace una exhaustiva lista de reglas para las conexiones permitidas.

Generalmente, los puntos de borde en una imagen pueden ser vistos como los nodos de una gráfica y como una función de valor asociado con la conexión de dos puntos de borde. Entonces los límites deseados pueden ser definidos como el mínimo valor o el menor valor de trayectoria a través de la gráfica, y el valor de la conexión de dos puntos de borde es definido como la función de la distancia entre ellos, diferentes en sus direcciones, y similares en otras descripciones.

Las técnicas grafica teóricas pueden ser útiles cuando el grafico buscado puede ser restringido. Para aplicaciones especiales, curvas específicas se detectan exitosamente incluso en presencia de grandes ruidos.

2.1.4. Segmentación de regiones

Los dos enfoques de la segmentación de regiones son las técnicas de "región en crecimiento" y "región en división". En el enfoque de región en crecimiento, los pixeles son agrupados en regiones basadas en las similitudes de un atributo, y las regiones resultantes son probadas al unirlas con otras regiones vecinas con el criterio de propiedades promedio y relaciones espaciales. En el segundo enfoque, las regiones grandes son consecutivamente divididas en regiones más pequeñas, según distinciones sutiles entre las propiedades de los pixeles pertenecientes a estas.

2.1.4.1.Limitación y segmentación recursiva

La técnica más simple para la segmentación es *la limitación* o en ingles llamada *thresholding.* En forma general, las regiones se componen por pixeles que tienen los valores de los atributos en un cierto rango. Esta técnica es adecuada para escenas de objetos con regiones uniformes o superficies con fondos diferenciados, pero también es usada en imágenes más complejas.

Un modelo usual, implícito, es que los diferentes atributos del objeto son distribuidos en dos valores promedio. Las funciones de la distribución no son conocidas, sin embargo el número de pixeles que difiere más del valor promedio, son asumidos para disminuir la diferencia. Para dicho modelo, un histograma de los valores de intensidad se ve para posteriormente se pueda elegir valores razonables para *threshold.* Una mejor selección de umbrales se puede realizar si se conoce con anterioridad estos valores.

Por ejemplo en el segmentador de Ohlander-Price-Reddy (R. Ohlander, 1978), un número de los pixeles de la imagen conforman un histograma y este tiene un pico distintivo; por lo tanto se escoge un rango entre el pico más distintivo y otros picos presentes en el histograma. Este proceso es repetitivo hasta que no se encuentre nuevas regiones a segmentar, ya que algunas distinciones entre una sola región se vuelve más obvia si esta región es separada de otra región grande y

así permitir una segmentación más exacta. Esta técnica de repetición se llama segmentación recursiva. Una deficiencia en las técnicas basadas en histogramas se ve que en la pequeñas regiones en una imagen grande no producen un pico distintivo en el histograma, incluso siendo distintos de los alrededores.

Una generalización de la técnica de división de región al usar múltiples atributos es segmentar todos los atributos de la imagen simultáneamente, más que escoger un atributo en histogramas separados. Si los diferentes atributos son vistos para medir una determinada característica en un espacio multidimensional, entonces los pixeles en una región deberían agruparse en este espacio. Agrupar es difícil cuando el número de grupos es desconocido, lo cual es típico en la segmentación.

2.1.4.2.Región en crecimiento

En este enfoque un grupo de pixeles cuyos valores de sus atributos están en un rango predefinido, son agrupados para formar regiones “atómicas”. Estas son examinadas para unirlas basadas en sus propiedades y relaciones. Se sugiere que el procedimiento de crecimiento de la región sea de tal manera que las primeras agrupaciones de las regiones atómicas con propiedades promedio sea en un rango especifico de *threshold*. Este rango puede ser menos rígido y diferente que el usado para la formación de pequeñas regiones.

Otro enfoque para reducir la dependencia en la elección de umbrales es preservar un árbol completo de regiones producidas a varios niveles de la región en crecimiento, llamado procedimiento de división y agrupación. Se describe desde una raíz que sería la imagen completa, y las hojas o nodos serían las regiones atómicas; en el caso de los niveles intermedios, las regiones son formadas por la agrupación de los siguientes niveles. Los procesos de alto nivel pueden examinar varias alternativas de segmentación más que una sola única salida y de esta manera poder modificarlas. Un árbol de imagen es el árbol que describe las relaciones entre las regiones a diferentes niveles y el termino de cuadrángulo de árbol se usa en el caso de un pixel en un nivel se divide en cuatro pixeles en un nivel más bajo. Por lo tanto el procedimiento de agrupación y división se puede esperar como más tolerante que el simple procedimiento de la región en crecimiento en el caso de la formación de regiones atómicas.

2.1.5. Borde versus segmentación de regiones

Ambas técnicas tienen el objetivo de encontrar los límites de los objetos entre los atributos de la imagen. Si estuvieran disponibles definiciones matemáticas precisas, se podría esperar que los dos métodos tienen el mismo rendimiento en cuanto a resultados, sin embargo difieren en el esfuerzo de cálculo. Ante la ausencia de tales modelos, los dos métodos implementan similares pero diferentes nociones intuitivas de las discontinuidades deseadas.

La ausencia de adecuados modelos matemáticos para imágenes y límites deseados dificulta la comparación del funcionamiento de las diferentes técnicas de segmentación. Sin embargo, se puede realizar algunas observaciones cualitativas:

- La segmentación de región necesariamente es para límites cerrados, en cambio el enfoque basado en el borde resulta en partes para formar segmentos de los límites.
- La detección de borde es un proceso inherentemente local, es decir fallas locales pueden prevenirse con una segmentación completa. Por otro lado, la segmentación de regiones es más global, siendo un proceso menos sensitivo y tiende a perder límites de bajo contraste o pequeños objetos.
- La mejora en el funcionamiento por la adición de color es más dramática en la segmentación de regiones que en la segmentación basada en el borde, debido que el color es una propiedad global.
- La posición ante una detección de bordes no es de interés, lo cual sucede también ante un *threshold*. Sin embargo, la posición de los límites de la región es un poco sensible ante la elección del rango del atributo de segmentación.

2.2. Softwares de procesamiento de imágenes en dendrocronología

2.2.1. DENDRO

El sistema de análisis de imagen DENDRO, implementado para dendrocronología en la plataforma de Macintosh como MacDENDRO y para la plataforma Windows como WinDENDRO; adquiere la data a utilizar desde una carpeta de imágenes, obtenidas desde el escaneo de muestras de secciones de madera o por medio de una cámara digital. La detección de anillos se logra al analizar uno o más pasos radiales desde la medula hasta la corteza, y estos pasos se definen por el analista antes de la identificación de los límites de los anillos.

Al definirse los pasos, automáticamente los anillos de la sección se identifican por medio de uno de los dos algoritmos de identificación. Este primer algoritmo funciona al analizar las diferencias de intensidad de la escala de grises a lo largo de la línea escaneada, es decir que cualquier salto importante en intensidad es considerado para ser un límite de anillo. El segundo algoritmo usa una técnica de "enseñar y mostrar", lo cual requiere que el analista manualmente elija un límite de anillo validado de la imagen y así generar un modelo por medio de comparación de plantillas. Por lo tanto cualquier región a lo largo del paso del escáner que tenga similares características es considerada para ser seleccionada como un límite de anillo. Una vez identificados los limites, se procede con el cálculo del ancho de los anillos y en el caso que más de un paso de escáner se crea, se promedia el ancho calculado.

Este software es un sistema semiautomático, debido que al realizar la validación del anillo se realizan cambios a lo largo de la trayectoria del anillo, buscando anillos ausentes o falsos, sin cambiar la precisión. Sin embargo, tiene limitaciones como por ejemplo el paso del escáner es sensible ante una anomalía causada por un cambio en el ancho de los anillos.

2.2.2. CooRecorder-de Cybis

Es un programa utilizado para un análisis y registro de las coordenadas del ancho de los anillos de crecimiento desde imágenes. El procedimiento al usarlo se inicia con la apertura del archivo en un determinado formato, ya sea jpg, gif, bmp, png o tif; luego se prosigue con fijar el valor de DPI (datos por pulgada). Se puede controlar la vista de la imagen cargada con cinco botones los cuales son: dos de zoom, un botón de ajuste de pantalla, un botón de pixeles y un botón de herramienta para desplazamiento.

Para comenzar con la grabación de datos se hace clic en el botón *Datos*, y básicamente el programa tiene dos formas de grabación: un par de coordenadas para cada fila de datos y un grupo de pares de coordenadas para cada fila de datos. En el caso de la dendrocronología se usa la forma de un par de coordenadas para cada fila de datos que indica el ancho del anillo de crecimiento. Para la grabación de los puntos en muestras de madera siempre se inicia desde el anillo exterior hasta la parte interior de la sección. Luego se puede editar lo grabado, eliminando, reemplazando o deshabilitando puntos. Finalmente estos datos se guardan en archivo con un determinado formato de salida ya elegido, los cuales son: datos brutos, datos ordenador y Dendro con *seasonwood.*

Una herramienta de ayuda para la elección de puntos durante la grabación es *help line* y la extiendes hasta donde se requiere. Otra herramienta es *autoplace*, la cual permite solo elegir el punto de inicio y el punto final, para después generar una series de puntos que pertenecen a los límites de anillos; sin embargo esta herramienta es adecuada para la madera tipo conífera. También al elegir el formato de salida datos de ordenador, permite usar los datos en CDendro para generar las series individuales.

2.2.3. Lignovision

Es un programa cuyo objetivo es disminuir el tiempo de trabajo en la medición del ancho de los anillos, mayormente aplicable a maderas blandas como las coníferas que se caracterizan por tener suficiente contraste en la escala de grises. Para la obtención de imágenes se utiliza un escáner o una cámara con alta resolución. Al iniciar el programa y seleccionar la imagen, se requiere información de unidad y resolución, si esta información es desconocida se calibra la imagen al trazar una línea aleatoria en la imagen para luego introducir la distancia en el cuadro de dialogo.

El primer paso es la definición de la medición del ancho del anillo con el uso de las herramientas de segmento simple o segmento continuo. Después de la definición, el sistema reconoce el anillado mediante la diferencia en la escala de grises al seleccionar en el botón de detección de anillado. Si este reconocimiento automático no funciona, se utiliza la función de reconocimiento manual o se puede corregir el anillado. Luego de verificar si los anillos reconocidos son correctos se forma la serie de anillos con el botón *Split profile,* y aparece el grafico de los grosores. Finalmente se guardan los datos con *save trace analysis*.

Capítulo 3
Aplicación del procesamiento digital de imágenes a la dendrocronología en Algarrobo

3.1. Introducción al software HDevelop

El software HDevelop es una caja de herramientas para la construcción de aplicaciones de *machine perception* o *machine visión*. Ayuda a generar prototipos de manera rápida al ofrecer un entorno altamente interactivo. Basado en las librerías HALCON, es un paquete versátil de *machine perception* adecuado para el desarrollo de productos e investigación. En resumen existen cuatro maneras el desarrollo de aplicaciones en análisis de imágenes usando HALCON:

- Prototipado rápido en un ambiente interactivo, es decir que se usa para encontrar operadores óptimos y parámetros para resolver el análisis de imagen correspondiente, usando varios lenguajes de programación.
- Desarrollo de una aplicación que se ejecuta en HDevelop.
- Se puede ejecutar programas en HDevelop o procedimientos de una aplicación escrita en otro lenguaje que pueda ser integrado, usando el HDevEngine.
- Se puede exportar aplicaciones en HDevelop a otro código fuente como C, C++, NET, etc. Este programa se compila y se vincula con la biblioteca Halcon para ser ejecutado como una aplicación independiente.

3.1.1. Hechos sobre HDevelop

Para el desarrollo de una aplicación específica, el software da soporte de diferentes maneras:

- En la interfaz gráfica del usuario, los operadores y los objetos icónicos se pueden seleccionar, cambiar y analizar en un mismo entorno.
- Se sugieren operadores para tareas específicas, además de contar con un listado de ellos estructurado de manera temática.
- Admite funciones de programación estándar como procedimientos, bucles y condicionales, además se puede cambiar los parámetros mientras se está ejecutando.

- Permite probar operadores y parámetros, e inmediatamente ver el efecto en la pantalla. Además se obtiene una vista previa de los resultados sin cambiar el programa.
- Gracias a herramientas gráficas, se permite examinar datos icónicos y de control.
- Existen asistentes gráficos que proporcionan interfaces interactivas para tareas más complejas.

3.1.2. Dialogo de inicio e interfaz de usuario

El dialogo de inicio provee rápido acceso a los programas de HDevelop (1), material introductorio (2) y documentación (3). Para una primera introducción se prueba con los vínculos de "Como empezar". En la Figura 3 se muestra los componentes del dialogo de inicio.

(1) (2) (3)

Figura 3. Dialogo de inicio de HALCON.
Fuente: Guía de usuario de HDevelop (2017).

Cuando HDevelop se inicia por primera vez la ventana principal ofrece un menú (5), una barra de herramientas (6), para rápido acceso a funciones frecuentemente usadas. La barra de estado (7) en la parte inferior de la ventana muestra los mensajes y las propiedades de la imagen. Además, las siguientes ventanas están disponibles:

- La ventana gráfica (1): En esta ventana se muestra la data icónica que son las imágenes, regiones y XLDs (descripción de línea extendida). Proporciona su propia barra de herramientas para acercar y alejar la imagen mostrada, además de un menú contextual para adaptar la configuración de visualización. El software admite una cantidad arbitraria de ventanas gráficas.
- La ventana de operadores (2): Puedes seleccionar operadores HALCON y procedimientos HDevelop en esta ventana. También los parámetros pueden ser seleccionados, ejecutados e ingresados en el actual programa o ambos. Por ultimo puedes tener ayuda online para el operador seleccionado.
- La ventana de programa (4): Se muestra el programa actual, y proporciona un resaltado de sintaxis con colores definibles por el usuario. La columna de la

izquierda muestra el número de línea del programa; el pequeño triangulo negro es el cursor de inserción, conocido como IC, que es donde se agregarán nuevas líneas del programa; la pequeña flecha verde es el contador del programa, conocida como PC, que marca la siguiente línea para ser ejecutada. También se puede agregar o quitar puntos de interrupción en el programa en la columna de izquierda que permite al usuario detener la ejecución en lugares definidos para poder examinar los resultados intermedios. Al agregar nuevas líneas o modificar las líneas existentes, las funciones avanzadas de autocompletado aceleran el tipeo y ayudan a mantener el programa consistente y también las líneas del programa se pueden modificar haciendo doble clic en ellas y editándolas en la ventana del operador.

- La ventana de variables (3): Se muestran todas las variables de programa actual y sus valores actuales. Las variables icónicas se muestran como pequeñas ventanas, y las variables de control se muestran como texto. Para visualizar las variables icónicas, se hace doble clic y se muestran en la ventana grafica activa, además al hacer doble clic sobre las variables de control se muestra una ventana de inspección con un listado de los valores actuales y los datos estadísticos.

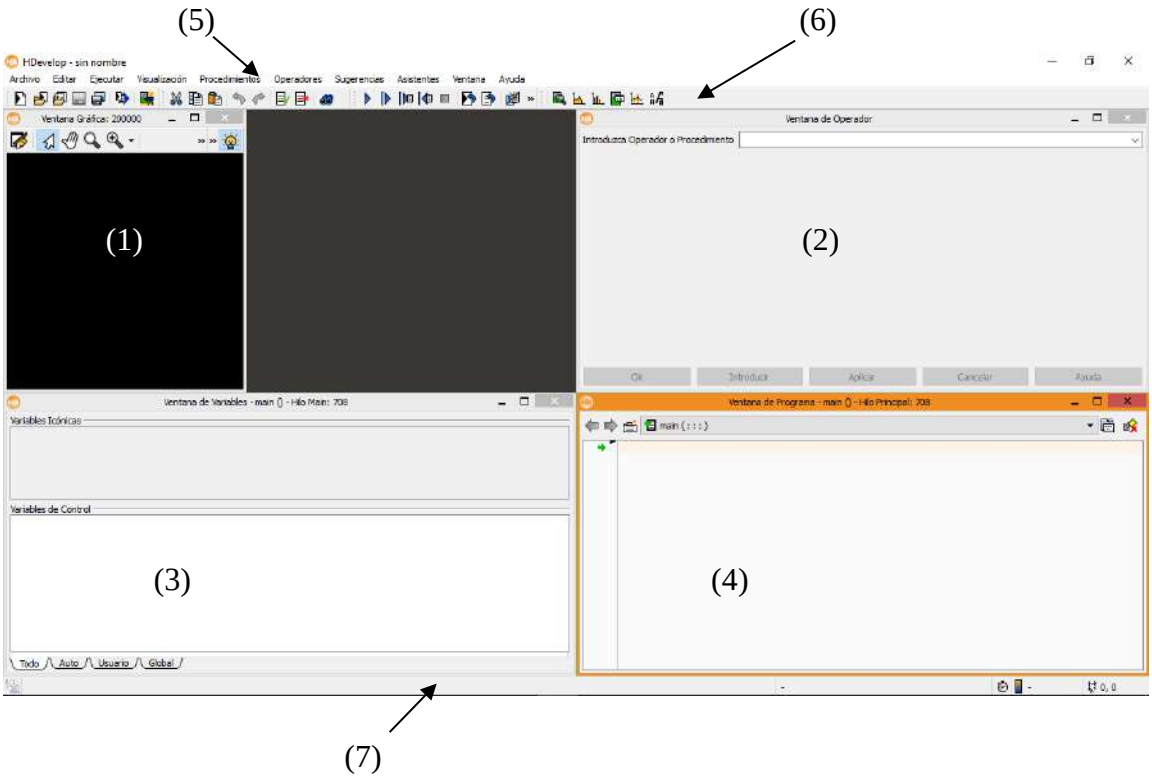


Figura 4. Componentes al iniciar el software HDevelop.

Fuente: Guía de usuario de HDevelop (2017).

3.2. Lenguaje HDevelop

3.2.1. Tipos básicos de parámetros

La librería HALCON distingue dos tipos de datos: los datos de control (números y cadenas) y los datos icónicos (imágenes, regiones, etc.). Al distinguir los parámetros de entrada y salida se obtienen cuatro tipos de parámetros y estos siempre aparecen en el orden siguiente:

Operador (entrada icónica: salida icónica: entrada de control: salida de control)

Como se observa, los objetos de entrada icónica siempre son primero, seguido por los objetos de salida icónica, luego viene la data de control. Algunos de estos tipos de parámetros pueden estar vacíos, como por ejemplo *sub_image*:

sub_image (*ImageMinuend*, *ImageSubtrahend* : *ImageSub* : *Mult*, *Add* :)

En el operador *sub_image* se tiene una entrada icónica, una salida icónica y una entrada de control. Los parámetros están separados por comas y los de control de entrada pueden ser variables, constantes o expresiones. Una expresión se evalúa antes de pasarse a un parámetro que recibe el resultado de la evaluación. En el caso de los parámetros icónicos, estos deben ser variables y los parámetros de salida de control también deben ser variables, ya que almacenan los resultados de una evaluación del operador.

3.2.2. Tipos de control y constantes

La data no icónica es representada por la llamada data de control en HDevelop, y el nombre deriva de las respectivas funciones en operadores de HALCON donde ellos controlan el comportamiento o el efecto del procesamiento de la imagen. Los parámetros de control contienen operaciones aritméticas o lógicas y un ítem de data de control puede ser cualquiera de los siguientes tipos de data:

- Entero: este tipo de data es usada bajo las mismas reglas de sintaxis que el lenguaje C. Los números enteros pueden ser ingresados en la notación decimal estándar, en la hexagesimal con el prefijo 0x, y en la octal con el prefijo 0 (cero). Ejemplos: 4711, 0xbeef (48879 en notación decimal), 073421 (30481 en notación decimal).
- Real: este tipo de data también sigue las mismas reglas de sintaxis que el lenguaje C. Ejemplo: 73.85, 0.324, 32E19, .56.
- Cadena: Una cadena es una secuencia de caracteres que está entre comillas simples ('). Los caracteres especiales, como el avance de línea, se representan en notación tipo C. En la siguiente Tabla 1 se observa este tipo de caracteres especiales:

Tabla 1. Sustitutos de caracteres especiales.

Meaning	Abbreviation	Notation
line feed	NL (LF)	\n
horizontal tabulator	HT	\t
vertical tabulator	VT	\v
backspace	BS	\b
carriage return	CR	\r
form feed	FF	\f
bell	BEL	\a
backslash	\	\\
single quote	'	\'
arbitrary character (hexadecimal)		\x*nn*
arbitrary character (octal)		\0*nnn*

Fuente: Guía de usuario de HDevelop (cap. 8, 2017)

- Booleano: las constantes *true* o *false* son data tipo booleana, el valor *true* es internamente representado por el número 1 y el valor *false* por el número 0. En general todo valor diferente de cero significa *true*. En el caso de los operadores de HALCON se espera más el uso de constantes cadena 'true' o 'false' que los valores booleano *true* o *false.*

En adición a estos tipos generales de data, hay otros tipos que son las constantes y las tuplas. Además que desde la versión HALCON 12.0 se considera otro tipo llamado vectores.

- Constantes: hay constantes para el valor de retorno o estado de resultado de un operador, las cuales representan el valor de retorno normal de un operador, los llamados mensajes. En la Tabla 2 se pueden encontrar todos los mensajes de devolución, adicionalmente hay constantes para el tipo de data de control que se ven en la Tabla 3.

Tabla 2. Retorno de valores para operadores.

Constant	Meaning	Value
H_MSG_TRUE	No error; for tests: (true)	2
H_MSG_FALSE	For tests: false	3
H_MSG_VOID	No result could be computed	4
H_MSG_FAIL	Operator did not succeed	5

Fuente: Guía de usuario de HDevelop (cap. 8, 2017)

Tabla 3. Retorno de valores para data de control.

Constant	Meaning	Value
H_TYPE_INT	integer value	1
H_TYPE_REAL	real value	2
H_TYPE_STRING	string value	4
H_TYPE_MIXED	mixed value	8

Fuente: Guía de usuario de HDevelop (cap. 8, 2017)

- Tuplas: Los tipos de control son usados de genérica como tupla. Es decir que la tupla consiste en varios ítems de data numérica de diferentes tipos de data de control. La representación de una tupla es el listado de sus elementos en corchetes, una tupla vacía se define como [] y una tupla con un solo elemento es considerada un caso especial. En la Figura 5 se ilustra su representación estándar.

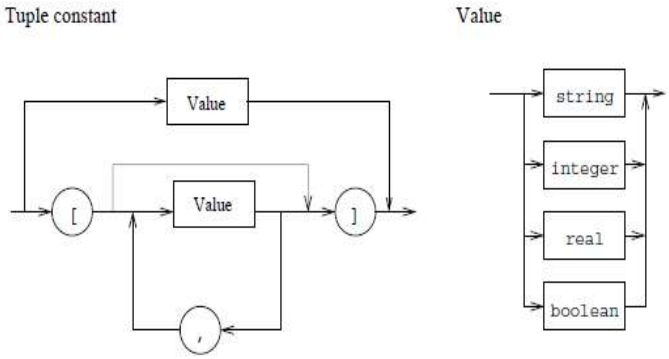


Figura 5. Sintaxis de constantes de tupla.

Fuente: Guía de usuario de HDevelop (cap. 8, 2017)

3.2.3. Variables

3.2.3.1. Tipos de variables

La primera tipificación de datos genera los objetos icónicos y la data de control; y el tipo de datos reales que son las cadenas, enteros, etc. no están definidos hasta que la variable no tiene un valor concreto. Por lo tanto es posible que el tipo de data de la nueva data ingresada difiera de la anterior. En HDevelop las variables se definen de tres maneras diferentes:

- De manera explícita: definiciones de parámetros de procedimiento y declaraciones de variables globales.
- De manera implícita: uso en el código.

De estas tres posibilidades las definiciones de parámetros de procedimiento son las más potentes, seguida por las declaraciones de variables globales. La menos

potente es uso en el código, en este caso el tipo de variable se define por líneas de código específicas y solo por esas líneas se escribe el valor de la variable y el nuevo valor es conocido a priori.

En el alcance de la variable, esta debe tener siempre el mismo tipo (icónica, control) y la misma dimensión, de otra manera causaría error en el procedimiento:

- Si la variable no es correctamente definida, entonces será un tipo no definido y las líneas que usan esta variable serán invalidas y la variable no se mostrara en la ventana de variables.
- Si la variable es definida de manera diferente dos o más veces, lo cual genera un tipo indefinido.
- Si el tipo de variable es definida apropiadamente, pero usada en un contexto erróneo, entonces solo la línea en la cual se ingresa mal es invalida.

3.2.3.2. Alcance de variables

Todas las variables son por defecto locales, es decir existen solo en su procedimiento. Por lo tanto una variable con el mismo nombre puede ser usada en otro procedimiento sin interferir una con la otra. En cambio, las variables globales acceden a todo el programa y deben ser enunciadas como tales con el operador *global*, por ejemplo:

La declaración *global tuple file*, declara como global a la variable de control *file*

Una vez que la variable global es declarada, se puede usar como variable local en el procedimiento que es usada. Si se quiere usar en otro procedimiento, se debe declarar la variable como global, sino se toma como una variable local.

3.2.4. Expresiones para parámetros de control de entrada

En HDevelop, el uso de expresiones como operaciones aritméticas u operaciones de cadena son limitadas para los parámetros de control de entrada.

Las operaciones se realizan asumiendo que son tuplas atómicas (tuplas de un solo elemento), si una tupla tiene más de un elemento, los operadores trabajan de la siguiente manera:

- Si una de las tuplas tiene un solo elemento, los elementos de la otra tupla son combinados con este único valor para la operación escogida.
- Si ambas tuplas tienen una longitud mayor que uno, ambas deben ser de la misma longitud. La operación seleccionada es aplicada a cada elemento con el mismo índice.
- Si una de las tupla es 0 [] entonces ocurre un error.

En la Tabla 4 se puede encontrar algunos ejemplos de operaciones aritméticas con tuplas y cadenas.

Tabla 4. Ejemplos de operaciones con tuplas y cadenas.

Input	Result
5 * 5	25
[5] * [5]	25
[1,2,3] * 2	[2,4,6]
[1,2,3] * 2.1 + 10	[12.1,14.2,16.3]
[1,2,3] * [1,2,3]	[1,4,9]
[1,2,3] * [1,2]	runtime error
'Text1' + 'Text2'	'Text1Text2'
17 + '3'	'173'
'Text ' + 3.1 * 2	'Text 6.2'
3.1 * (2 + ' Text')	runtime error
3.1 + 2 + ' Text'	'5.1 Text'
3.1 + (2 + ' Text')	'3.12 Text'

Fuente: Guía de usuario de HDevelop (cap. 8, 2017)

En el caso de la asignación de valores en HDevelop, este procedimiento funciona como operador assign (Input, Result). Sin embargo en el texto del programa se puede usar la sintaxis usual $Result \coloneqq Input$. En el siguiente ejemplo se muestra la diferencia:

Sintaxis en C:

$$u = \sin(x) + \cos(y)$$

En HDevelop como operador:

$$assign\ (\sin(x) + \cos(y)\,, u)$$

En la ventana de programa:

$$u \coloneqq \sin(x) + \cos(y)$$

Si el resultado de la operación no necesita almacenarse como una variable, entonces se puede usar como valor de entrada para cualquier operador. En cambio si el valor es para usarse varias veces o si la variable debe inicializarse como en el caso de un bucle, entonces es necesaria la asignación.

En la Tabla 5 se muestran algunas operaciones de tuplas con sus respectivos operadores.

Tabla 5. Operaciones Básicas en tuplas y sus respectivos operadores HALCON.

Operation	Meaning	HALCON operator
`t := [t1,t2]`	concatenate tuples	`tuple_concat`
`i := |t|`	get number of elements of tuple `t`	`tuple_length`
`v := t1[t2]`	select elements `t2` of tuple `t1`	`tuple_select`
`t := t[i1:i2]`	select from element `i1` to element `i2` of tuple `t`	`tuple_select_range`
`t := subset(t1,t2)`	select elements specified in `t2` from `t1`	`tuple_select`
`t := firstn(t,i)`	select first elements from `t` up to index `i`	`tuple_first_n`
`t := lastn(t,i)`	select elements from `t` from index `i` to the end	`tuple_last_n`
`t := select_mask(t1,t2)`	select all elements from `t1` where the corresponding mask value in `t2` is greater than 0	`tuple_select_mask`
`t := remove(t,i)`	remove elements specified in `i` from `t`	`tuple_remove`
`i := find(t1,t2)`	get indices of all occurrences of `t2` within `t1` (or `-1` if no match)	`tuple_find`
`i := replace(t1,t2,t3)`	replace elements	`tuple_replace`
`i := find_first(t1,t2)`	get indices of the first occurrences of `t2` within `t1` (or `-1` if no match)	`tuple_find_first`
`i := find_last(t1,t2)`	get indices of the last occurrences of `t2` within `t1` (or `-1` if no match)	`tuple_find_last`
`t := uniq(t)`	discard all but one of successive identical elements from `t`	`tuple_uniq`
`t := [i1:i2:i3]`	generate a sequence of values from `i1` to `i3` with an increment value of `i2`	`tuple_gen_sequence`
`t := [i1:i2]`	generate a sequence of values from `i1` to `i2` with an increment value of one	`tuple_gen_sequence`
`t := gen_tuple_const(i,t)`	generate a tuple with `i` values set to `t`	`tuple_gen_const`

Fuente: Guía de usuario de HDevelop (cap. 8, 2017)

3.3.Métodos HALCON

Los métodos HALCON listados implican áreas de aplicación común en *machine vision*, de esta manera con esta guía se puede desarrollar una aplicación específica(Halcon, 2014). En esta sección se describirá los métodos utilizados para el desarrollo de la aplicación específica a la dendrocronología en algarrobo.

Generalmente muchas aplicaciones usan los siguientes métodos:

- Adquisición de imágenes: usando HALCON se puede obtener imágenes desde diferentes tipos de dispositivos y configuraciones de cámaras en distintos modos de sincronización, también se permite el ingreso de imágenes desde una carpeta en diferentes formatos como TIFF, JPEG, PNG, etc.
- Visualización: En las ventanas de graficas del software se puede visualizar todo tipo de datos compatibles o utilizando operadores de pantalla específicos; no se

requiere mucho esfuerzo, ya que HALCON tiene una funcionalidad que esta optimizado en *machine vision*.

- Región de interés: el objetivo de este método es focalizar el proceso en una parte de la imagen, este enfoque combina la información de la región con la matriz de la imagen (se reduce el número de pixeles a ser procesados).

3.3.1. Adquisición de imágenes

Adquirir imágenes consiste simplemente en tres pasos, y es incluso más simple cuando se adquiere de una carpeta que sería un solo paso con el comando *read_image.* En la Figura 6 se describe el proceso para el método de adquisición de imágenes.

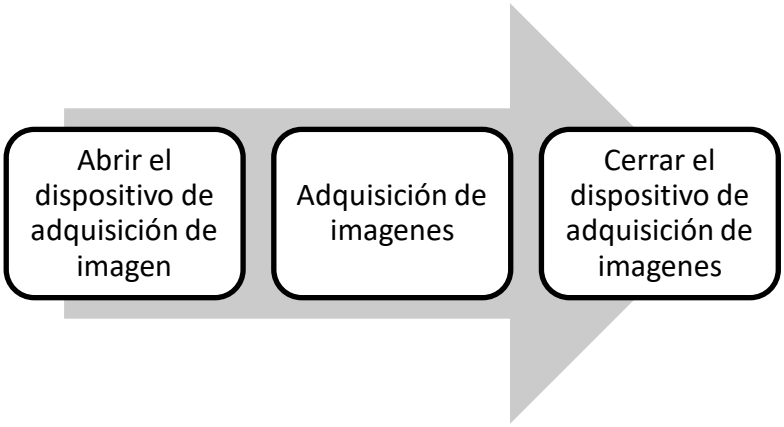


Figura 6. Proceso del método de adquisición de imágenes.

Fuente: Guía de soluciones I HDevelop (cap. 2, 2017). Elaboración: Elaboración propia.

Para abrir el dispositivo se usa el operador *open_framegrabber* especificando el nombre de la interfaz. También hay una interfaz de adquisición de imágenes "virtual" llamada File. Como su nombre lo sugiere, este "capturador de cuadros" lee imágenes de archivos y también de los denominados archivos de secuencias de imágenes. Por ultimo están los archivos específicos de HALCON, típicamente con una extensión .seq, que contienen una lista de nombres de carpetas, separado por una nueva línea. Si conectas tal secuencia, las llamadas subsiguientes a *grab_image* devuelven las imágenes en la secuencia especificada en el archivo. Otra alternativa es leer las imágenes de un directorio específico en lugar de crear un archivo de secuencia, y se especifica el nombre del directorio en el valor del parámetro 'Camara Type'. Si se quiere cargar una imagen desde el disco se usa el operador *read_image* y se busca la imagen en el directorio actual. Al final de la aplicación se cierra el dispositivo con el operador *close_framegrabber*.

En aplicaciones reales no es suficiente decirle al dispositivo para adquirir una imagen, en vez de eso puede ser importante que las imágenes sean adquiridas en el momento correcto. En HDevelop, se proporciona un asistente a través del elemento de menú Asistentes> Adquisición de imágenes que lo ayuda a seleccionar su fuente de imagen, ajustar los parámetros y generar el código adecuado. Por lo tanto el proceso

de adquisición de imagen se extiende a cuatro pasos, y se describe los operadores estándar y avanzados utilizados, el cual se describe en la Figura 7.

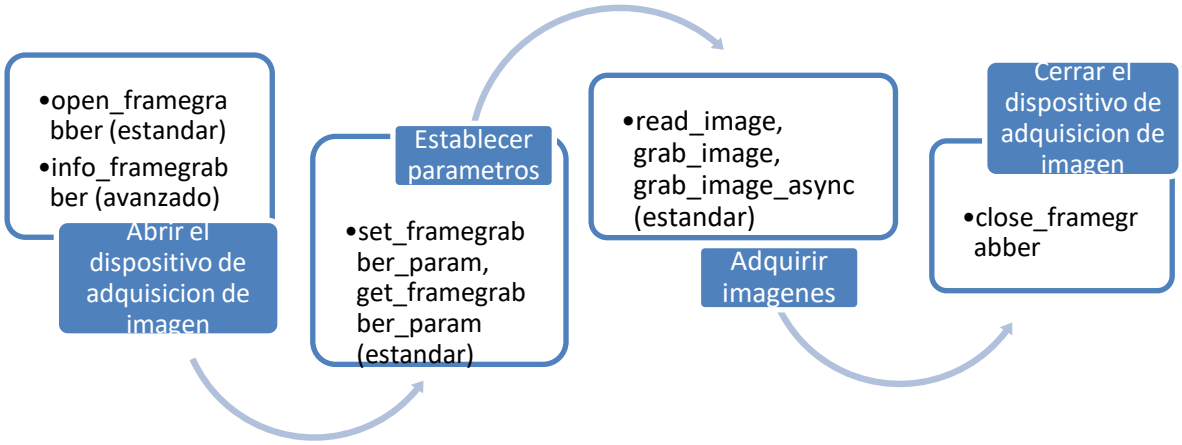


Figura 7. Proceso extendido de adquisición de imagen con sus respectivos operadores.

Fuente: Guía de soluciones I HDevelop (cap. 2, 2017). Elaboración: Elaboración propia.

3.3.2. Análisis de manchas

La idea de análisis de manchas es bastante sencilla, pues en una imagen los pixeles de los objetos relevantes o también llamados en primer plano se pueden identificar por su valor de gris. En muchas aplicaciones esta condición simple de pixeles oscuros y brillantes no se cumple, pero con un pre-procesamiento adicional o métodos alternativos se pueden obtener los mismos resultados. La ventaja de este método reside en la flexibilidad que viene con un gran número de operadores que se ofrecen en este contexto y que son combinados con otros tipos de tareas de *machine vision*.

Este proceso básicamente consta de tres pasos, que son la adquisición de la imagen, la segmentación y extracción de características. La segmentación consiste en seleccionar los pixeles de primer plano, típicamente referidos como manchas u objetos grandes binarios, se usa el tipo data llamada región. Para el paso final se extrae características como área, posición y orientación.

En muchas aplicaciones la segmentación es más compleja, debido a defectos como la variación de iluminación no homogénea y el desorden. Además se requiere procesos posteriores como la transformación de las características en unidades del mundo real y la visualización de resultados. En la Figura 8 se muestra un esquema del proceso extendido del método de análisis de manchas.

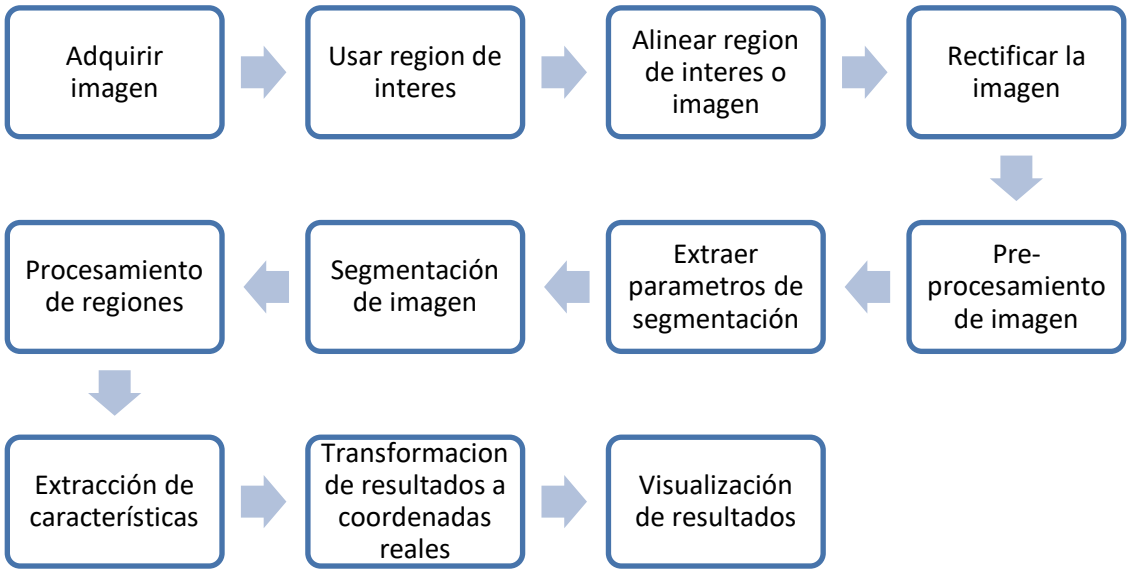


Figura 8. Esquema extendido del método de análisis de manchas.

Fuente: Guía de soluciones I HDevelop (cap. 4, 2017). Elaboración: Elaboración propia.

En el caso de la etapa de pre-procesamiento de imagen, los operadores como *mean_image* o *gauss_filter* pueden ser utilizados para eliminar ruido. Una alternativa para *gauss_filter* pero menos efectiva es el *binomial_filter,* el operador *mean_image* sirve para eliminar pequeñas manchas y líneas delgadas, el operador *anisotropic_diffusion* es útil para el suavizado preservando los bordes y por último el *fill_interlace* es usado para eliminar defectos causados por cámaras entrelazadas.

Para la etapa de extracción de parámetros de segmentación, en vez de usar valores de umbral fijos pueden ser extraídos dinámicamente para cada imagen. Para ello, los operadores como *gray_histo_abs* y *histo_to_thresh*. Como una alternativa avanzada puedes usar el operador *intensity* y una imagen de referencia que contiene el fondo.

Para la etapa de segmentación de imagen, el método más simple es con el operador *threshold*, con el cual uno o más valores de rangos de grises que pertenecen a los objetos de primer plano son especificados. Otro método común es con el operador *dyn_threshold*, donde una segunda imagen es usada como referencia y en vez de usar un umbral local usas un umbral global.

En la etapa de procesamiento de regiones, cuando ya son segmentadas es necesario modificarlos como por ejemplo suprimir pequeñas áreas, regiones con una orientación dada, o regiones cerradas. En este contexto, los operadores como *opening_circle* y *opening_rectangle1*son usualmente usados para suprimir ruido y los operadores *closing_circle* y *closing_rectangle1* para llenar vacíos. Por ultimo las manchas con determinadas características pueden ser seleccionadas con los operadores *select_shape*, *select_shape_std* y *select_shape_proto*.

3.3.3. Procesamiento de contornos

Una de las potentes herramientas de HALCON es el conjunto de herramientas para establecer los contornos precisos en pixeles. Los contornos son el tipo de datos XLD y son el resultado de algún tipo de procesamiento de imagen y representan por ejemplo los bordes de los objetos.

El procesamiento de contornos consiste en múltiples pasos que pueden ser combinados en un modo flexible mostrado en la Figura 9.

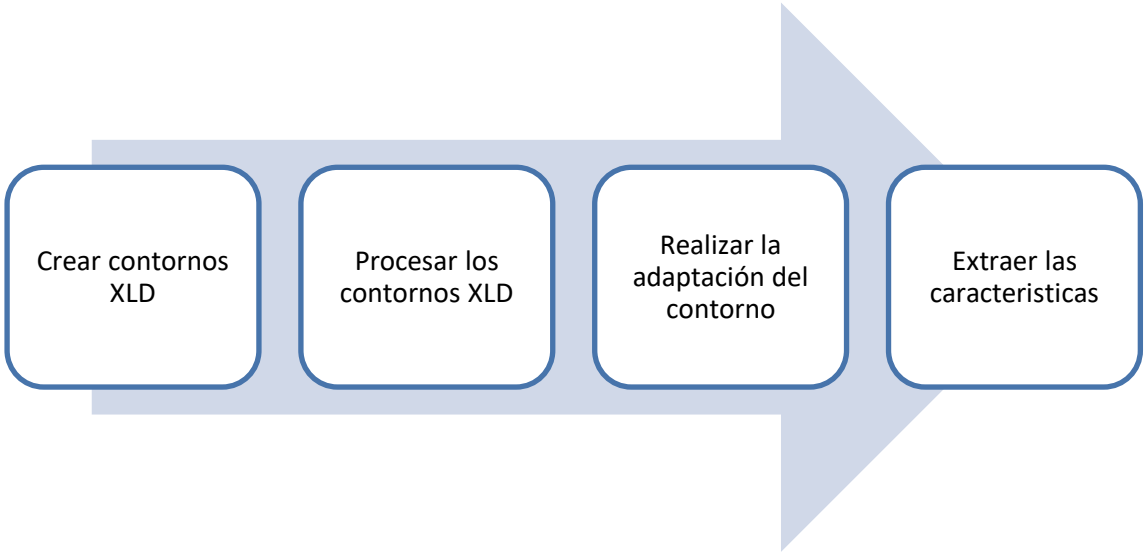


Figura 9. Esquema del procedimiento del procesamiento de contornos.

Fuente: Guía de soluciones I HDevelop (cap. 8, 2017). Elaboración: Elaboración propia.

3.3.3.1. Crear contornos XLD

Para extraer los contornos de una imagen el operador mas usado es *edge_sub_pix*, en el cual se puede seleccionar diferentes tipos de filtrado donde se especifica con el parametro Filter. Para aplicaciones comunes el valor usual es ‘canny’ (basado en la convolución gaussiana) o ‘lanser2’. En otros casos se puede ser una version rapida como ‘sobel_fast’ que es un filtro recomendado para imágenes que no tienen mucho ruido ni estan borrosas.

Otro operador usado es *zero_crossing_sub_pix* y se combina con el filtro *derivate_gauss* para utilizar el parametro ‘laplace’. Para extraer contornos en imágenes a color se utiliza *edge_color_sub_pix* que es similar *edge_sub_pix*.

El operador para extraer lineas es *lines_gauss*, comparado con *lines_facet* es mas robusto y provee mayor flexibilidad. El ancho de las lineas que deberian ser extraidas son especificadas por el parametro Sigma. Otro operador para la extracción de lineas de imágenes a color es *lines_color*.

Si la precisión del pixel es correcto, se puede usar un filtro de borde como *sobel_amp* o *edges_image*, o tambien un filtro de linea como *bandpass_image* seguido por una umbralización y reducción. Las regiones alargadas resultantes son convertidas en contornos XLD con el operador *gen_contours_skeleton_xld*.

Los contornos pueden ser sintetizados de diferentes fuentes como CAD data, usuario interactivo, medición; por lo tanto se tiene las coordenadas de control de tal fuente y con los operadores *gen_contour_polygon_xld* y *gen_contour_polygon_rounded_xld* se convierten en contornos. Tambien se puede convertir los bordes de las regiones en contornos con el operador *gen_contour_region_xld*.

3.3.3.2. Procesar los contornos XLD

Para procesar los contornos, estos se deben restringir a una region de interes. Sin embargo, al extraer los contornos extraidos no son los deseados. Esto puede suceder por el ruido, la textura, la existencia de vacios entre contornos debido al bajo contraste o intersección de otros contornos.

Para procesar contornos, primero se segmentan y el operador usual para este proceso es *segment_contours_xld*. Este operador ofrece varios modos de segmentar los contorno como segmentos de lineas, segmentos lineas circulares y segmentos de lineas circulares. El segmento de contorno puede ser escogido con *select_obj* y seguir con el siguiente paso. Si el segmento representa una linea, un arco circular o un arco eliptico, puede ser requerido con el atributo global de contorno 'cont_approx' usando el operador *get_contour_global_attrib_xld*.

Si solo los segmentos de linea son necesarios, se usa la combinación de los operadores *gen_polygons_xld* y *split_contours_xld*. Este procedimiento tiene el comportamiento similar al usar *segment_contours_xld*, sin embargo difiere porque este procedimiento es el paso inicial para agrupar los contornos y convertirlos en lineas paralelas.

Otro importante proceso en esta etapa es suprimir los contornos no deseados y se puede lograr con el operador *select_shape_xld*, que ofrece 30 tipos de características diferentes. Se especifica el valor mínimo y máximo de la característica seleccionada. Otra alternativa es el operador *select_contours_xld*, que ofrece características típicas de estructuras lineales.

Si hay vacios entre los contornos, se generan segmentos de línea gracias a los operadores *union_collinear_contours_xld* y *union_straight_contours_xld*. Adicionalmente se puede procesar contornos adjacentes con *union_adjacent_contours_xld*, que están en un mismo circulo (*union_cocircular_contours_xld*) y que sean cotangenciales (*union_cotangential_contours_xld*).

3.3.3.3.Realizar la adaptación del contorno

Al obtener segmentos de contorno que representan una línea, un arco elíptico o circular y un rectángulo se puede determinar los parámetros como las coordenadas de los puntos de inicio y fin de una línea. El objetivo es aproximar el contorno procesado a una línea, rectángulo, y arco elíptico o circular. Se utilizan operadores para obtener los parámetros del contorno aproximado y para su visualización, descritos en la Figura 10.

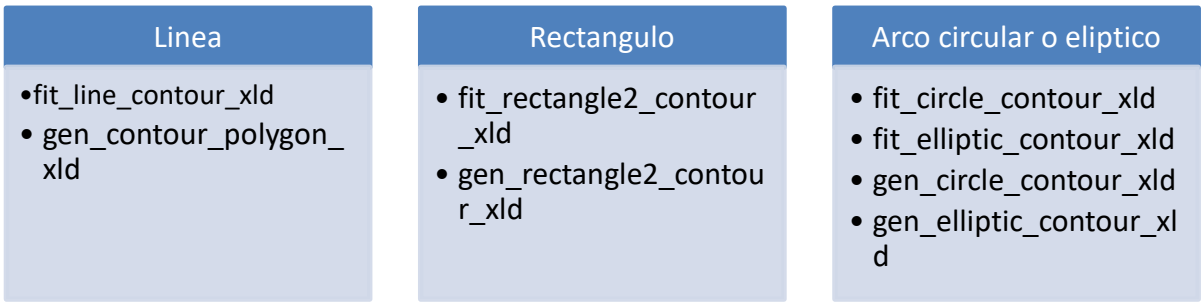


Figura 10. Esquema de operadores utilizados para obtener los parámetros del contorno.

Fuente: Guía de soluciones I HDevelop (cap. 8, 2017). Elaboración: Elaboración propia.

3.3.3.4. Extraer características

Los operadores comúnmente usados son: *área_center_xld*, *compactness_xld*, *convexity_xld*, *eccentricity_xld*, *diameter_xld* y *orientation_xld*. Los cascos de los contornos pueden ser determinados con los operadores *smallest_circle_xld* y *smallest_rectangle2_xld*.

3.4. Desarrollo de la aplicación específica para secciones de algarrobo

Para la extracción de datos en el caso de la secciones de algarrobo, se siguen dos procesos diferentes a partir de la imagen a color. Los objetivos de inicio de la programación para el desarrollo de la aplicación serían los siguientes:

- Identificar los vasos en la imagen, con el menor error posible.
- Identificar los contornos deseados que se aproximen a los límites de anillo.

Por lo tanto al cumplirse con estos objetivos, se continuaría con la extracción de datos en cada proceso. En el caso de la identificación de vasos se ha utilizado el método de análisis de manchas y en la identificación de límites de anillos se ha utilizado el método de procesamiento de contornos; agregando o variando en los pasos a seguir en los métodos descritos.

Las pruebas realizadas en el proceso de identificación de vasos difieren en cuanto a pre-procesamiento de la imagen y segmentación. En el proceso de identificación de

límites de anillos el pre-procesamiento y los parámetros cambian en el paso de procesar contornos.

3.4.1. Identificación de vasos

El proceso de identificación de vasos sigue los pasos descritos en la Figura 11.

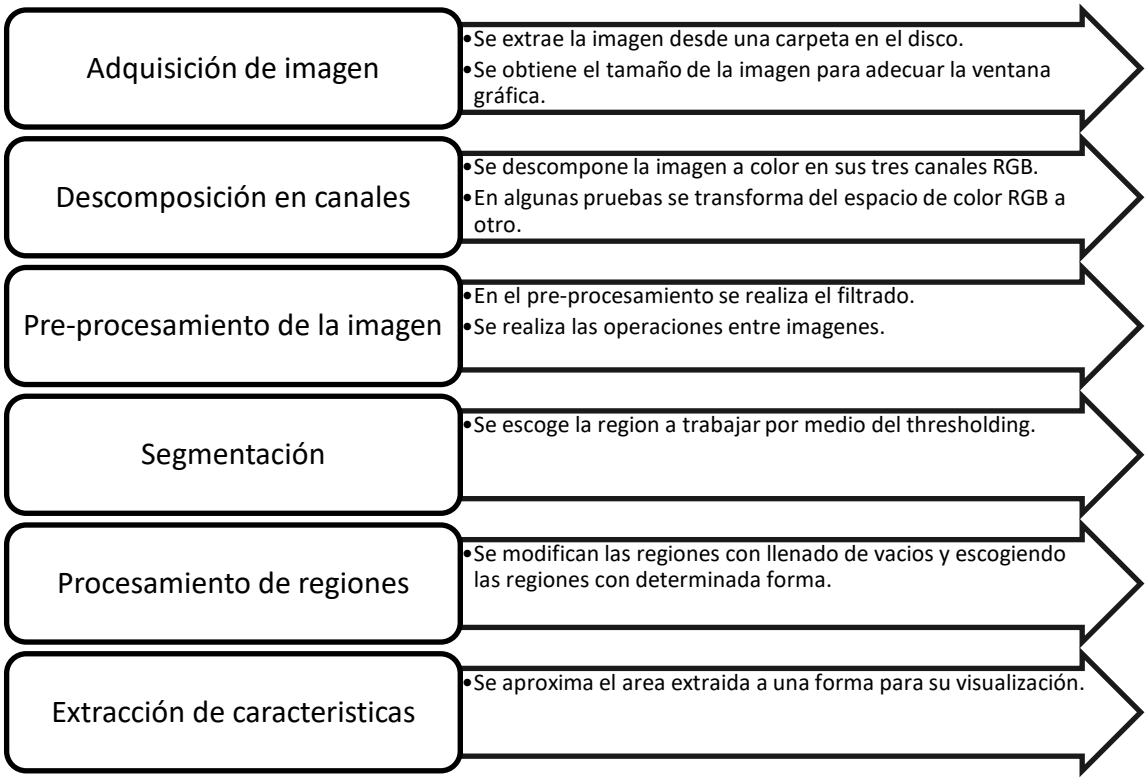


Figura 11. Esquema del proceso de identificación de vasos.

Fuente: Elaboración Propia.

Para las pruebas realizadas, difieren en los operadores utilizados y en sus respectivos parámetros. Esto provoca diferencias en los resultados, como por ejemplo en la cantidad de vasos identificada. Se escogió una imagen estándar cuyo número de vasos se aproximó para calcular el error.

3.4.2. Identificación aproximada de límites de anillos

En la Figura 12 se muestra el esquema de pasos generales a seguir para aproximar los límites de anillos.

Por las pruebas realizadas, aun no se han encontrado una integración para obtener los anillos en una sección. Es decir que al aplicar el mismo programa a otras imágenes no se obtiene la mayoría del contorno deseado como en la imagen probada inicialmente.

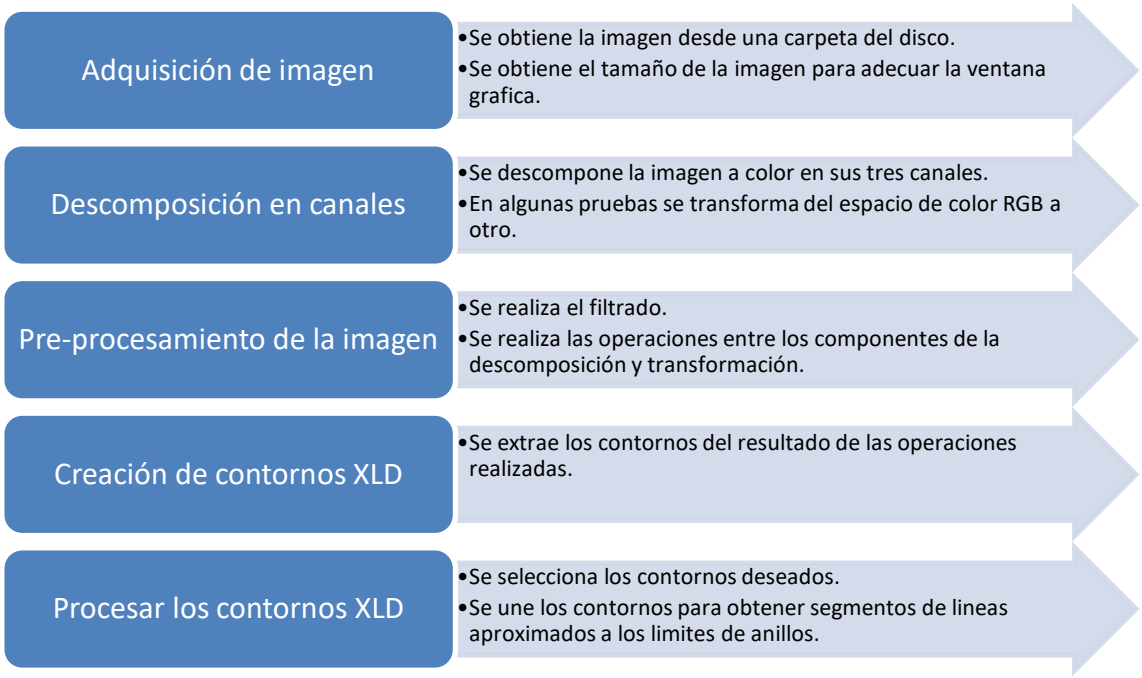


Figura 12. Esquema de pasos generales para obtención de los límites de anillos.

Fuente: Elaboración propia.

Capítulo 4
Resultados del procesamiento de imágenes al análisis dendrocronológico en Algarrobo

4.1.Pruebas de identificación de vasos

4.1.1. Prueba inicial

En la prueba inicial se busca un pre-procesamiento de imagen adecuado, logrando disminuir el dominio en la escala de grises de la variable icónica escogida; sin embargo, esto no influye mucho en el cambio de dominio para posteriormente realizar una elección más adecuada de la región. Después, se realiza un proceso de segmentación simple y luego se busca aproximar la región escogida a una forma conocida, en este caso un círculo, lo cual funciona debido a la forma de los vasos.

El primer paso en esta prueba, después de leer la imagen, es la descomposición de la imagen en tres canales o matrices, y se escoge la matriz Green para el pre-procesamiento.

Se usa el filtro *mean_image* con mascara de tamaño de matriz de 5x5, luego se escoge una región con *threshold* y se transforma esta región en una determinada forma con *shape_trans*, luego se reduce el dominio de la imagen.

Se segmenta la imagen reducida con un umbral y luego se llena los vacíos de la región con *fill_up_shape* para escoger las regiones de interés con *opening_circle*.

El código desarrollado en la prueba inicial*, especificado en la ventana de programa con sus respectivos comentarios, es el siguiente:

Adquisición de la imagen con el operador *read_image*
dev_close_window()
read_image(*Image, 'C:/Users/pc/Documents*/Tesis Elaboración/Imagenes/ImagenPoros.jpg')
get_image_size(Image, Width, Height)
dev_open_window(0, 0, Width, Height, 'black', WindowHandle)
Descomposición de la imagen en tres canales

decompose3(Image,Red, Green, Blue)
stop()
Pre-procesamiento de la imagen, el objetivo es reducir el dominio de la imagen y se obtiene un nuevo histograma de valores de grises.
mean_image(Green, ImageMean, 5, 5)
threshold(ImageMean, Region, 74, 232)
shape_trans(Region, RegionTrans, 'convex')
reduce_domain(Green, RegionTrans, ImageReduced)
dev_display(ImageReduced)
stop()
Se realiza una segmentación simple.
threshold(ImageReduced, Region1, 92, 251)
stop()
En el procesamiento de regiones se llena los vacios y se aproxima a una forma.
fill_up_shape(Region1, RegionFillUp, 'area', 1, 200)
stop()
opening_circle(RegionFillUp, RegionOpening, 9.5)
dev_display(ImageMean)
dev_display(RegionOpening)
stop()
connection(RegionOpening, ConnectedRegions)
dev_display(Green)
dev_display(ConnectedRegions)
stop()

El objetivo de la prueba inicial fue determinar la utilidad de los comandos en cada proceso y poder mejorar el programa, agregando más operadores y operaciones entre imágenes. En la Figura 13 se muestra el objeto icónico resultante, después de la ejecución del programa.

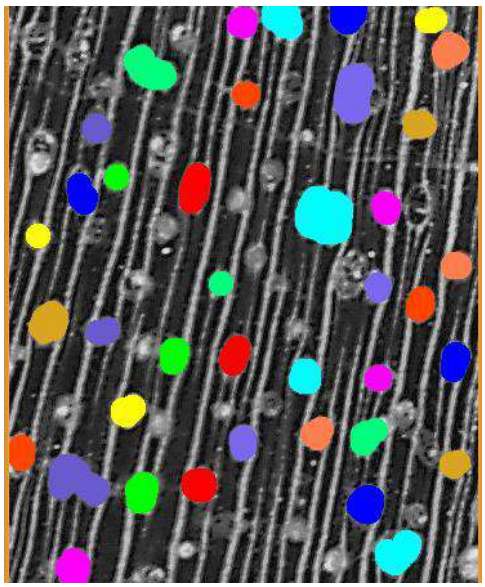


Figura 13. Objeto icónico resultante de la prueba inicial.

Fuente: Elaboración propia.

4.1.2. Primera prueba

La primera prueba se caracteriza por generar variables icónicas en otros espacios de color y cambiarlas a otro tipo de imagen, de esta manera los tres componentes resultantes se utilizan para operaciones que permitan llegar a la imagen binaria que se requiere. Es decir, la descomposición y el pre-procesamiento de la imagen son procesos más complejos y comprende la conversión y resta de imágenes.

Para comenzar con el pre-procesamiento, se descompone en tres canales RGB y se convierte a HLS, para convertir la imagen resultante tipo byte a int2. Después se realiza una diferencia de imagen entre la matriz L y S convertidas a int2. Se aplica un filtrado no recursivo gauss con un alpha de 1.5.

Se segmenta la imagen con un umbral binario, se llena los vacíos para luego eliminar el ruido con *opening_circle*.

Se separa las regiones por medio del comando *erosion_circle* y luego se agranda la región para obtener las regiones aproximadas de los vasos.

El código en la ventana de programa*, describiendo el proceso respectivo y sus parámetros correspondientes en cada operador, es el siguiente:

El enfoque es por medio de análisis de manchas o *blob analysis*
dev_close_window()
Adquisición de imagen
Se lee la imagen de color y se abre una ventana conforme a su tamaño dado
*read_image(Image, 'C:/Users/pc/Documents/*Tesis Elaboración/Imagenes/ImagenPoros.jpg')
get_image_size(Image, Width, Height)
dev_open_window(0, 0, Width, Height, 'black', WindowHandle)
Descomposición de la imagen
Se descompone la imagen de tres canales en tres imágenes de un canal con el mismo dominio de definición
decompose3(Image,Red, Green, Blue)
dev_display(Green)
stop()
Se transforma la imagen en espacio de color RGB a un espacio de color arbitrario, en este caso hls
trans_from_rgb(Red, Green, Blue, ImageResult1, ImageResult2, ImageResult3, 'hls')
stop()
La imagen resultante se convierte a un rango de valores de gris más bajo
convert_image_type(ImageResult2, I2, 'int2')
dev_display(I2)
stop()
La imagen resultante se convierte a un rango de valores de gris más bajo
convert_image_type(ImageResult1, I1, 'int2')
dev_display(I1)

stop()
La imagen resultante se convierte a un rango de valores de gris más bajo
convert_image_type(ImageResult3, I3, 'int2')
dev_display(I3)
stop()
Pre-procesamiento de la imagen escogida
sub_image(ImageResult2, ImageResult3, ImageSub, 1, 120)
dev_display(ImageSub)
stop()
smooth_image(ImageSub, ImageSmooth, 'gauss', 1.5)
dev_display(ImageSmooth)
stop()
Segmentación de la imagen
Se segmenta la imagen, escogiendo una región, en este caso los pixeles de rango de grises a 255(blanco)
binary_threshold(ImageSmooth, Region, 'max_separability', 'light', UsedThreshold)
dev_display(Region)
stop()
Se conecta los componentes de la región ingresada
connection(Region, ConnectedRegions)
dev_display(ConnectedRegions)
stop()
Procesamiento de regiones
Se rellena los agujeros de la región de entrada, dando como parámetro de forma 'area'
fill_up_shape(ConnectedRegions, RegionFillUp, 'area', 1,50)
dev_display(RegionFillUp)
stop()
Se eliminan regiones pequeñas al elemento estructurador circular con un radio dado
opening_circle(RegionFillUp, RegionOpening, 7.5)
dev_display(I2)
dev_set_color('red')
dev_set_draw('margin')
dev_set_line_width(2)
dev_display(RegionOpening)
stop()
connection(RegionOpening, ConnectedRegions1)
dev_display(I2)
dev_display(ConnectedRegions1)
stop()
erosion_circle(RegionOpening, RegionErosion, 8)
dev_display(Green)
dev_set_color('red')
dev_set_draw('fill')
dev_display(RegionErosion)

stop()
Se conecta y se aumenta las regiones conectadas para obtener las manchas
connection(RegionErosion, ConnectedRegions)
dev_display(Green)
dev_display(ConnectedRegions)
stop()
dilation_circle(ConnectedRegions, RegionDilation, 4.5)
dev_display(Green)
dev_set_color('red')
dev_set_draw('fill')
dev_display(RegionDilation)
stop()
Se clasifica las regiones con respecto a su posición relativa
sort_region(RegionDilation, SortedRegions, 'first_point', 'true', 'row')
dev_display(Green)
dev_display(SortedRegions)
stop()
Extracción de características
Se determina los círculos más pequeños de una región, para determinar su tamaño a pesar de no ser homogéneos las regiones.
smallest_circle(SortedRegions, Row, Column, Radius)
NumVasos:=|Radius|
*Diameter:=2*Radius*
meanDiameter:=mean(Diameter)
minDiameter:=min(Diameter)
dev_display(Green)
dev_set_draw('margin')
**se muestra los circulos en la ventana de salida*
disp_circle(WindowHandle, Row, Column, Radius)
dev_set_color('red')

En el paso extracción de características se calcula el número de vasos en la imagen resultante, la cual tiene como salida una data de control, también se tiene la tupla *Diameter* que tiene las medidas de los diámetros en pixeles. En la Tabla 6 se muestra la medida de algunos de los diámetros calculados con su respectivo índice.

Tabla 6. Cuadro de inspección de variables-Tupla *Diameter*.

	Diameter
87	37.6267
88	19.5274
89	26.2982
90	21.1361
91	29.4884
92	16.2315
93	11.2956
+	
Tipos	real
Dimensión	0

Fuente: Elaboración propia.

Las regiones encontradas para identificar los vasos, se aproximan a la forma de un círculo, entonces el objeto icónico resultante se muestra en la Figura 14 al ejecutarse el programa.

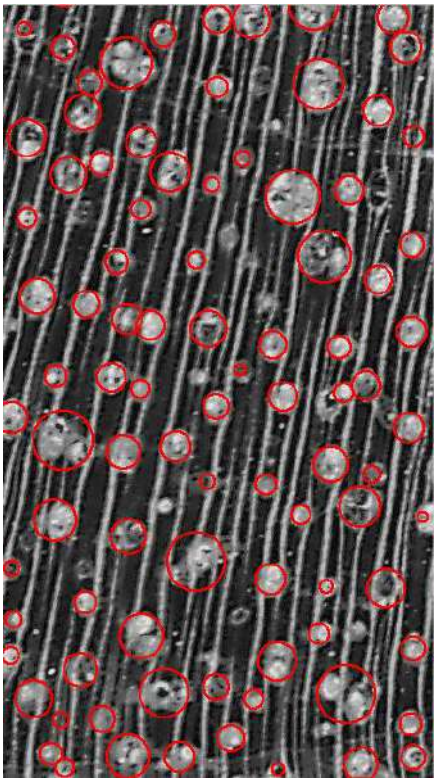


Figura 14. Objeto icónico resultante de la primera prueba

Fuente: Elaboración propia.

En la Figura 15 se muestra un esquema de los resultados con respecto a la exactitud de la identificación de vasos en la imagen pre-procesada.

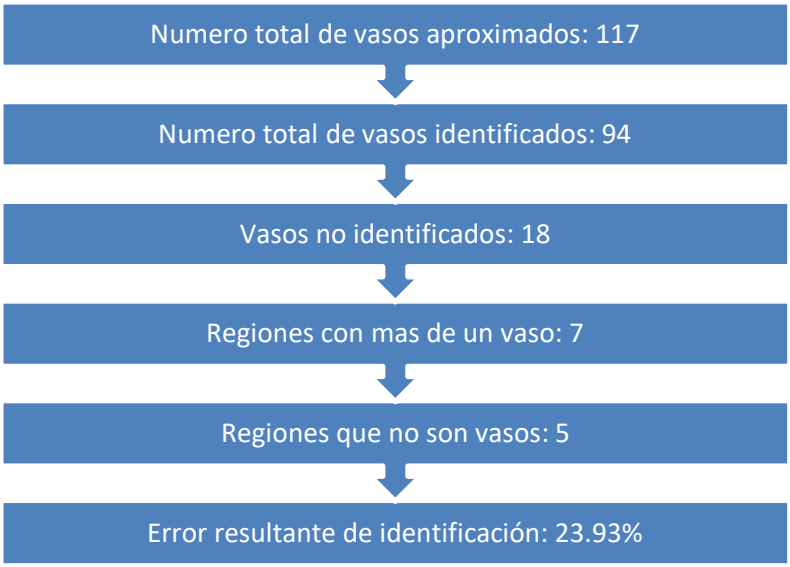


Figura 15: Esquema de cálculos de resultados de la primera prueba.

Fuente: Elaboración propia.

4.1.3. Segunda prueba

En la segunda prueba, se realiza una conversión a otro espacio diferente a la anterior prueba. Por lo tanto se analiza la diferencia, y las operaciones entre variables icónicas siguen un proceso similar al diagrama de flujo que explica la conversión de un espacio a otro. El objetivo es lograr la imagen a blanco y negro, después los pasos de procesamiento de regiones son similares a la anterior prueba, y se utiliza otros operadores para la extracción de características.

Al descomponerse la imagen RGB a HSV, el proceso no lineal que se sigue se describe en la Figura 16.

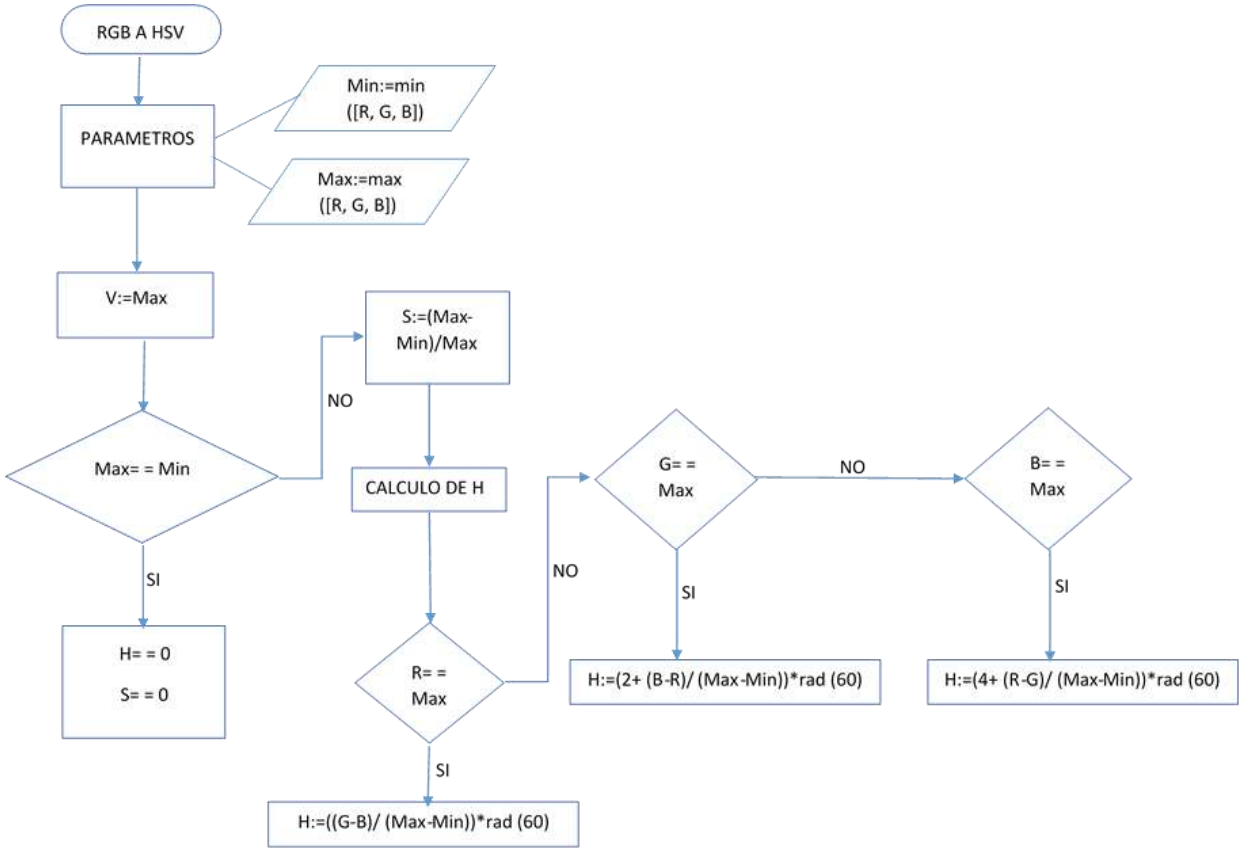


Figura 16. Diagrama de flujo de conversión de RGB a HSV.
Fuente: Referencia de operadores HALCON (Filtros, 2017). Elaboración: Elaboración propia.

Luego de esta conversión se resta la matriz Red y la matriz S, dando como resultado la variable icónica BW1.

Se realizan otras operaciones como la resta de imágenes de $(R - G) * 2 + 128$ y $(G - B) * 2 + 128$ con el comando *sub_image*, luego se realiza una división $\left(\frac{GB}{RG}\right) *$

$100 + 10$ con el comando *div_image*. La imagen resultante de esta operación se le resta a la variable BW1 y se obtiene la imagen BW.

A la imagen BW se le aplica un *threshold* de rango de 3 a 255, un llenado de vacíos con *fill_up_shape*, reducción de las regiones de interés con *opening_circle*, luego para separar las regiones de interés se usa *erosion_circle*, se conecta y se agranda las regiones disminuidas con *connection* y *dilation_circle*.

El código desarrollado en la segunda prueba*, con sus respectivos comentarios, es el siguiente:

Prueba para identificación de vasos
dev_close_window()
Adquisición de imagen
read_image(Image, 'C:/Users/pc/Documents/Tesis Elaboración/Imagenes/ImagenPoros.jpg')
get_image_size(Image, Width, Height)
dev_open_window(0, 0, Width, Height, 'black', WindowHandle)
decompose3(Image,Red, Green, Blue)
dev_display(Green)
stop()
Preprocesamiento de Imagen
trans_from_rgb(Red, Green, Blue, I1, I2, I3, 'hsv')
dev_display(I1)
stop()
Se realiza la resta (R-G)*2+128
sub_image(Red, Green, RG, 2, 128)
dev_display(RG)
stop()
Se realiza la resta de (G-B)*2+128
sub_image(Green, Blue, GB, 2, 128)
dev_display(GB)
stop()
Se realiza la división (GB/RG)*100+10
div_image(GB, RG, Hue, 100, 10)
dev_display(Hue)
stop()
Se obtiene una imagen casi totalmente blanco/negro
sub_image(Red, I2, BW1, 3, 0)
dev_display(BW1)
stop()
Se aumenta los pixeles blancos, resaltando los poros
sub_image(BW1, Hue, BW, 3, 0)
dev_display(BW)
stop()
Segmentación de Imagen
Se selecciona la región de interés, el histograma muestra pocos pixeles entre el 3 y 255.

threshold(BW, Region, 3, 255)
dev_display(Region)
stop()
Se conecta y se llena los vacíos con *fill_up_shape*
connection(Region, ConnectedRegions1)
Procesamiento de regiones
fill_up_shape(ConnectedRegions1, RegionFillUp, 'area', 1, 50)
dev_display(RegionFillUp)
stop()
Se elimina las regiones pequeñas o de poco interés al usar un elemento estructurador con un radio dado
opening_circle(RegionFillUp, RegionOpening, 6.5)
dev_display(Red)
dev_set_color('red')
dev_set_line_width(2)
dev_set_draw('margin')
dev_display(RegionOpening)
stop()
Se disminuye las regiones de interés, separando las regiones resultantes de *region_opening*
erosion_circle(RegionOpening, RegionErosion, 7.5)
dev_display(Red)
dev_set_color('red')
dev_set_draw('fill')
dev_display(RegionErosion)
stop()
Se conecta y se aumenta las regiones conectadas para obtener las manchas
connection(RegionErosion, ConnectedRegions)
dev_display(BW)
dev_display(ConnectedRegions)
stop()
dilation_circle(ConnectedRegions, RegionDilation, 4.5)
dev_display(Green)
dev_set_color('blue')
dev_set_draw('fill')
dev_display(RegionDilation)
stop()
Se clasifica las regiones con respecto a su posición relativa
sort_region(RegionDilation, SortedRegions, 'first_point', 'true', 'row')
dev_display(Green)
dev_display(SortedRegions)
stop()
Extracción de características
Se determina los círculos más pequeños de una región, para determinar su tamaño a pesar de no ser homogéneos las regiones.
smallest_circle(SortedRegions, Row, Column, Radius)
count_obj(SortedRegions, Number)

NumVasos:=|Radius|
*Diameter:=2*Radius*
meanDiameter:=mean(Diameter)
minDiameter:=min(Diameter)
maxDiameter:=max(Diameter)
dev_display(Green)
dev_set_draw('margin')
Se muestra los círculos en la ventana de salida
disp_circle(WindowHandle, Row, Column, Radius)
dev_set_color('red')

El objeto icónico resultante se muestra en la Figura 17, aunque no se denota diferencias obvias con respecto a la primera prueba, sin embargo el número de vasos identificados es mayor.

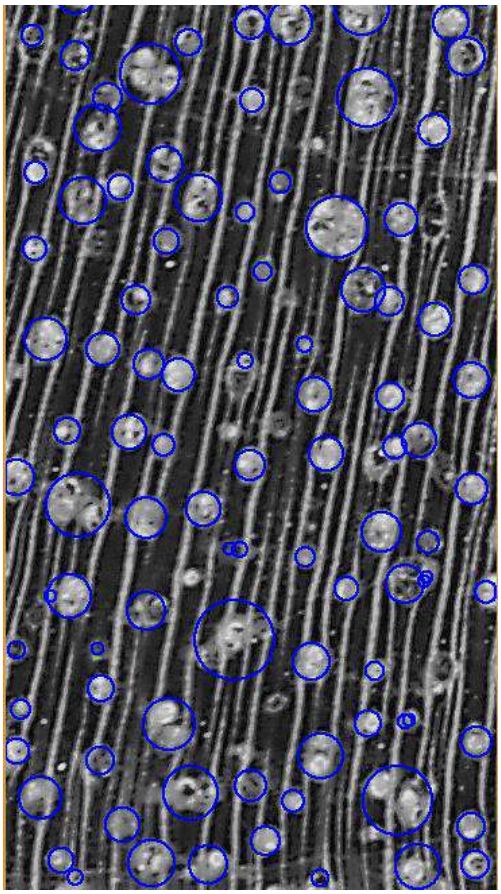


Figura 17. Objeto icónico resultante de la segunda prueba.

Fuente: Elaboración propia.

En la Figura 18 se muestra un esquema con los resultados con respecto a la exactitud de identificación de vasos, y su error es menor con respecto a la primera prueba.

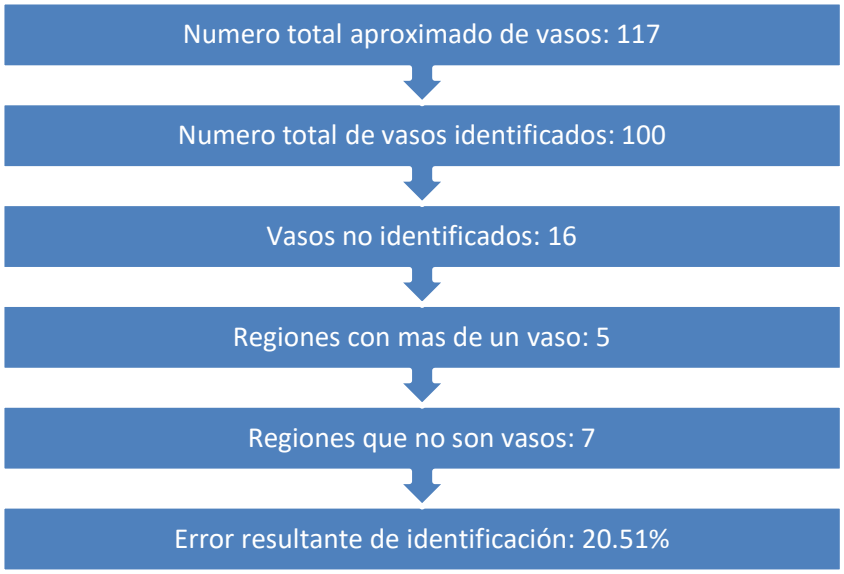


Figura 18. Esquema de cálculos de resultados de la segunda prueba.

Fuente: Elaboración propia.

4.2. Identificación de anillos

4.2.1. Primera prueba

En esta prueba se experimenta al generar seis matrices, convirtiendo las componentes RGB a otro espacio, y con las variables resultantes se realiza una resta de imágenes; para conseguir una imagen adecuada y convertirla en una imagen binaria. En este caso, la imagen se vuelve binaria al escoger una región y utilizar un nuevo operador; con el propósito de una identificación de contornos más exacto. El operador de obtención de contornos es sensible ante los cambios en sus parámetros y se debe escoger un rango adecuado. Luego sigue el proceso repetitivo de la selección de contornos y su unión para formar los anillos, hasta conseguir el segmento deseado.

El desarrollo del programa comienza con la adquisición de la imagen y adecuar la ventana grafica al tamaño de la imagen, luego se descompone en tres canales y transforma a dos espacios de color, uno es el espacio HSV y otro es el espacio Cielchuv; de esta manera se generan seis matrices H, S, V, X, Y y Z.

Se restan las matrices Red con S, siendo la operación $(R - S) * 2 + 50$; también se restan las matrices S y Y, siendo la operación $(S - Y) * 1 + 0$. Las imágenes resultantes de ambas operaciones se restan con la operación $(RS - SY) * 2 + 0$. Se obtiene la variable icónica *ImageSub*, y se escoge una región para luego convertirla en una imagen binaria *BinImage*.

A BinImage se le realiza una filtración con el filtro no recursivo gauss con un alpha de 1, por lo tanto resulta en la variables icónica *ImageSmooth*. Luego se empieza con la obtención de contornos con el operador *lines_gauss*, con un sigma de 1.5 y un rango de 3 a 3.4 y se escogen

La selección de contornos se da al establecer un rango de su dirección y luego de su circularidad, el asistente de histogramas de características ayuda en la elección de rangos. Luego se la unión de contornos y se prosigue con un proceso repetitivo en la selección de contornos según su longitud y la unión de ellos hasta conseguir el contorno deseado.

El código desarrollado para la primera prueba de identificación de anillos es el siguiente:

dev_close_window()
read_image(Image, 'C:/Users/pc/Documents/Tesis Elaboración/Imagenes/Imagen21.jpg')
get_image_size(Image, Width, Height)
dev_open_window(0, 0, Width, Height, 'black', WindowHandle)
Se descompone la imagen de tres canales en tres imágenes de un canal con el mismo dominio de definición
decompose3(Image,Red, Green, Blue)
dev_display(Green)
stop()
Se transforma la imagen en espacio de color RGB a un espacio de color arbitrario, en este caso cielchuv
trans_from_rgb(Red, Green, Blue, X, Y, Z, 'cielchuv')
dev_display(X)
stop()
Se transforma la imagen en espacio de color RGB a un espacio de color arbitrario, en este caso hsv
trans_from_rgb(Red, Green, Blue, H, S, V, 'hsv')
dev_display(H)
stop()
Se resta dos imágenes con valores de grises g1 y g2 siendo la ecuación según parámetros (g1-g2)*2+50
sub_image(Red, Y, RY, 2, 50)
dev_display(RY)
stop()
Se resta dos imágenes con valores de grises g1 y g2 siendo la ecuación según parámetros (g1-g2)*1+0
sub_image(S, Y, SY, 1,0)
dev_display(SY)
stop()
Se resta dos imágenes con valores de grises g1 y g2 siendo la ecuación según parámetros (g1-g2)*2+0
sub_image(RY, SY, ImageSub, 2, 0)
dev_display(ImageSub)
stop()
A partir de esta imagen, se escoge una región para convertirla a una imagen en binario
threshold(ImageSub, Region, 220, 255)
dev_display(Region)

stop()
Con el comando *region_to_bin*, se convierte la región seleccionada a una imagen binaria
region_to_bin(Region, BinImage, 255, 0, Width, Height)
dev_display(BinImage)
stop()
Se aplica el filtro no recursivo gauss con un alpha de 1
smooth_image(BinImage, ImageSmooth, 'gauss', 1)
dev_display(ImageSmooth)
stop()
Se comienza con el procesamiento de contornos
Se extrae las líneas con un *lines_gauss*, en este caso se extrae líneas más cercanas al blanco.
lines_gauss(ImageSmooth, Lines, 1.5, 3, 3.4, 'light', 'true', 'bar-shaped', 'true')
dev_display(ImageSmooth)
dev_set_color('red')
dev_set_line_width(2)
dev_display(Lines)
stop()
*se realiza la selección de líneas por medio de su dirección y circularidad
select_contours_xld(Lines, SelectedContours, 'direction', rad(-40), rad(40), -0.5, 0.5)
dev_display(ImageSmooth)
dev_set_color('red')
dev_set_line_width(2)
dev_display(SelectedContours)
stop()
select_shape_xld (SelectedContours, SelectedXLD1, 'circularity', 'and', 0, 0.10225)
dev_display(ImageSmooth)
dev_set_color('red')
dev_set_line_width(2)
dev_display(SelectedXLD1)
stop()
Se realiza una regresión y una unión de contornos resultantes
regress_contours_xld(SelectedXLD1, RegressContours, 'no', 1)
union_collinear_contours_xld(RegressContours, UnionContours, 40, 2, 2, 0.12, 'attr_keep')
dev_display(Red)
dev_set_color('red')
dev_set_line_width(2)
dev_display(UnionContours)
stop()
Se selecciona los contornos deseados al eliminar los contornos de poca longitud, el procedimiento se vuelve repetitivo
select_shape_xld (UnionContours, SelectedXLD, 'contlength', 'and', 7, 300)
dev_display(Red)

```
dev_set_color('red')
dev_set_line_width(2)
dev_display(SelectedXLD)
stop()
regress_contours_xld(SelectedXLD, RegressContours1, 'no', 1)
union_collinear_contours_xld(RegressContours1, UnionContours1, 70, 2.5, 3.5,
0.12, 'attr_keep')
dev_display(Red)
dev_set_color('red')
dev_set_line_width(2)
dev_display(UnionContours1)
stop()
select_contours_xld(UnionContours1, SelectedContours1, 'contour_length', 30,
600, -0.5, 0.5)
dev_display(Red)
dev_set_color('red')
dev_set_line_width(2)
dev_display(SelectedContours1)
stop()
regress_contours_xld(SelectedContours1, RegressContours2, 'no', 1)
union_collinear_contours_xld(RegressContours2, UnionContours2, 300, 6, 8.7,
0.15, 'attr_keep')
dev_display(Red)
dev_set_color('red')
dev_set_line_width(2)
dev_display(UnionContours2)
stop()
select_contours_xld(UnionContours2, SelectedContours2, 'contour_length', 180,
680, -0.5, 0.5)
dev_display(Red)
dev_set_color('red')
dev_set_line_width(2)
dev_display(SelectedContours2)
stop()
```

El objeto icónico resultante se muestra en la Figura 19 y se observan segmentos líneas aproximados a los límites de anillos; sin embargo al analizar otra sección o imagen, en el objeto icónico resultante mostrado en la Figura 20 no se observan todos los límites de anillo reconocidos o aproximados a segmentos de línea.

Los resultados varían debido a los parámetros de los operadores usados se deben variar de una imagen a otra para obtener los contornos deseados. Se debe encontrar una solución integral ante variaciones de selección de contornos, por lo que se busca la mejora del paso del pre-procesamiento de la imagen.

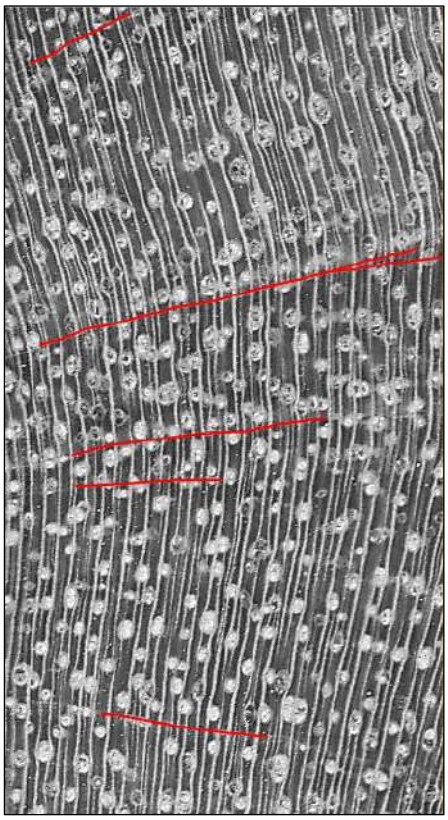


Figura 19. Objeto icónico resultante de sección 1.

Fuente: Elaboración propia.

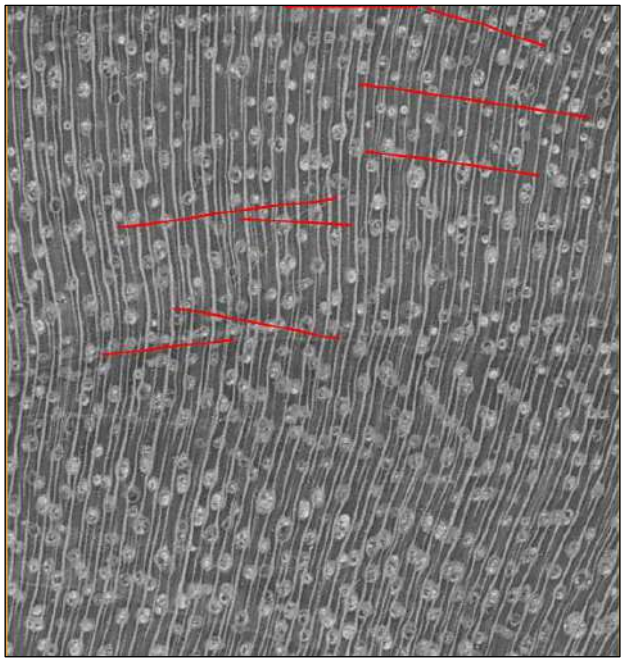


Figura 20. Objeto icónico resultante de sección 2.

Fuente: Elaboración propia.

Conclusiones

- Los métodos de HALCON son un punto partida para el desarrollo de programas de asistencia de obtención de información por medio del análisis de imágenes.
- El método de análisis de manchas ofrece el mejor enfoque para el desarrollo de un programa con el menor tiempo de ejecución y un error aceptable en el proceso de identificación de vasos, y de esta manera poder continuar con el avance de estudios dendrocronológicos.
- El proceso de identificación de vasos desarrollado en el software HDevelop tiene el potencial para convertirse en un asistente de obtención de información para la dendrocronología en Algarrobo.
- Para mejorar el proceso de identificación de poros en secciones de Algarrobo, en el paso de pre-procesamiento de la imagen, la variable icónica resultante debe ser binaria.

Bibliografía

Ancajima, E. M. (Abril de 2017). Dendrocronología de Prosopis sp. en la región Piura. Piura, Piura, Perú.

Gimenez, A. M., Moglia, J. G., Hernandez, P., & Gerez, R. (2000). *Anatomía de Madera.* Santiago del Estero, Argentina .

Nevatia, R. (1982). *Machine Perception.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Palacios, E. M. (Abril de 2018). Relación entre la distribución intra-anual de los vasos xilemáticos en la madera de Prosopis sp. (Algarrobo) y la variabilidad climática durante un año de evento El Niño. Piura, Piura, Perú.

R. Ohlander, K. P. (1978). *Picture Segmentation Using a Recursive Region Splittin Method.* Pittsburgh.

Rodríguez, R., & Fernández, R. (2009). *Dendrocronología Básica.* Piura: Santa Lucia .

Conner, W. S. (1999). A Computer Vision Based Tree Ring Analysis and Dating System by Signed : Approval by Thesis Director, 99 pages (Thesis).

García, I., García, L., & Díaz, E. (1968). Empleo de un escáner de sobremesa para la medición manual de anillos de crecimiento.

Gartner, B. L., Aloni, R., Funada, R., Lichtfuss-Gautier, a N., & Roig, F. a. (2002). Clues for dendrochronology from studies of wood structure and function. *Dendrochronologia*, *20*(1–2), 53–61. https://doi.org/10.1078/1125-7865-00007

Halcon. (2014). Solution Guide I Basics, 349.

López, B. C., Sabaté, S., Gracia, C. A., & Rodríguez, R. (2005). Wood anatomy, description of annual rings, and responses to ENSO events of Prosopis pallida H.B.K., a wide-spread woody plant of arid and semi-arid lands of Latin America. *Journal of Arid Environments*, *61*(4), 541–554. https://doi.org/10.1016/j.jaridenv.2004.10.008

MVTec Software GmbH. (2017). *HDevelop User ' s Guide* (14a ed.).

Pometti, C. L., Pizzo, B., Brunetti, M., Macchioni, N., Ewens, M., & Saidman, B. O. (2009). Argentinean native wood species: Physical and mechanical

characterization of some Prosopis species and Acacia aroma (Leguminosae; Mimosoideae). *Bioresource Technology*, *100*(6), 1999–2004. https://doi.org/10.1016/j.biortech.2008.09.061

Anexos

Anexo A: Operadores

Para la explicación de cada operador se explicará brevemente lo que hacen y se dará la forma en que debe ser escrito dentro del código. También se explicará cada parámetro que poseen los diferentes operadores.

A.1. Operadores principales

Este apartado contendrá operadores básicos y comunes que por lo general siempre se usan al comienzo de cualquier código orientado a procesamiento de imágenes.

- **read_image(Image, FileName):** Lee una imagen con diferentes formatos de archivos. Los formatos pueden ser: TIFF, GIF, BMP, JPEG, JPEG-2000, JPEG-XR, PNG, PCX, SUN-Raster, PGM, PPM, PBM y XWD. Si en FileName se ingresa más de un archivo, se genera una tupla de objetos de imágenes con un número de imágenes igual al número de archivos ingresados.
  - ✓ **Image:** Imagen leída. Puede salir de los siguientes tipos: image(-array) → object (byte / direction / cyclic / int1 / complex / int2 / uint2 / vector_field / int4 / int8 / real).
  - ✓ **FileName:** Lo que arroja el operador. Las siguientes extensiones son aceptadas: .hobj, .ima, .tif, .tiff, .gif, .bmp, .jpg, .jpeg, .jp2, .jxr, .png, .pcx, .ras, .xwd, .pbm, .pnm, .pgm, .ppm.

- **get_image_size(Image, Width, Height):** Retorna el ancho y alto de la imagen de entrada.
  - ✓ **Image:** Imagen de entrada que puede ser de los siguientes tipos: (multichannel-)image(-array) → object (byte / direction / cyclic / int1 / int2 / uint2 / int4 / int8 / real / complex / vector_field).
  - ✓ **Width:** Parámetro de salida. Ancho de la imagen. Es un número de tipo integer.
  - ✓ **Height:** Parámetro de salida. Alto de la imagen. Es un número de tipo integer.

- **threshold(Image, Region, MinGray, MaxGray):** Segmenta una imagen usando un threshold global. Selecciona los pixels cuyo valor de gris g cumple la siguiente condición: MinGray ≤ g ≤ MaxGray.
  - ✓ **Image:** Imagen de entrada. Puede ser de los siguientes tipos: singlechannelimage(-array) → object (byte / direction / cyclic / int1 / int2 / uint2 / int4 / int8 / real / vector_field).
  - ✓ **Region:** Region segmentada de salida. Puede ser de tipo: region(-array) → object.
  - ✓ **MinGray y MaxGray:** Valores inferior y superior del threshold. Puede ir de 0 a 255. MaxGray >= MinGray

- **decompose3(MultiChannelImage, Image1, Image2, Image3):** Convierte una imagen de tres canales en tres imágenes RGB.
  - ✓ **MultiChannelImage:** Imagen de 3 canales que puede ser de tipo: multichannel-image(-array) → object (byte / direction / cyclic / int1 / int2 / uint2 / int4 / int8 / real / complex / vector_field).

- ✓ **Image1, Image 2 e Image 3**: Imágenes de salida. Cada una corresponde a la capa Red, Green y Blue, respectivamente

- **trans_from_rgb(ImageRed, ImageGreen, ImageBlue, ImageResult1, ImageResult2, ImageResult3, ColorSpace):** Transforma una imagen del espacio de colores RGB a un espacio de colores arbitrario.
  - ✓ **ImageRed, ImageGreen e ImageBlue:** Son las imágenes de entrada y puede ser de tipo: singlechannelimage(-array) → object (byte / uint2 / int4 / real)
  - ✓ **ImageResult1,ImageResult2, ImageResult3:** Son las imágenes de salida y puede ser de tipo: singlechannelimage(-array) → object (byte / uint2 / int4 / real)
  - ✓ **ColorSpace**: Define el espacio de color de las imágenes de salida. Puede ser: 'argyb', 'cielab', 'cielchab', 'cielchuv', 'cieluv', 'ciexyz', 'ciexyz2', 'ciexyz3', 'ciexyz4', 'hls', 'hsi', 'hsv','i1i2i3', 'ihs', 'lms', 'yiq', 'yuv'.

A.2. Operadores dev_

Aquí se especificarán los operadores básicos que empiezas con dev_.

- **dev_open_window(Row, Column, Width, Height, Background, WindowHandle):** Abre una nueva ventana gráfica y esta se vuelve active aumáticamente. Esto implica que todos los dev_display y displays automáticos de los operadores se dirigen a esta ventana. La ventana soporta imágenes, regiones, líneas y texto.
  - ✓ **Row:** Índice de la fila de la esquina superior izquierda. Puede ser un número mayor que 0.
  - ✓ **Column:** Índice de la columna de la esquina superior izquierda. Puede ser un número mayor que 0.
  - ✓ **Width y Height:** Ancho y alto de la ventana gráfica. Puede ser cualquier número mayor que 0, por lo general es igual al ancho y alto de la imagen a proyectar.
  - ✓ **Background:** Color del fondo de la nueva ventana. El valor por defecto es: 'black'.
  - ✓ **WindowHandle:** Es una salida. Es un número lógico que arroja este comando y aparece en la barra de título. Puede ser de tipo: integer.

- **dev_display(Object):** Muestra los objetos de imágenes en la ventana gráfica activa. Pueden ser regiones, imágenes o XLD. Es equivalente a hacer doble click en la variable icónica del objeto.
  - ✓ **Object:** Objetos de imágenes a ser mostrados. Pueden ser del tipo: object(-array) → object

- **dev_set_line_width(LineWidth)):** Define el ancho de la línea para mostrar el contorno de una región.
  - ✓ **LineWidth:** Ancho de línea para mostrar una región en el modo contorno. El valor por defecto es 1 y tiene que ser mayor o igual a 1. El número debe ser de tipo integer.

- **dev_set_color(ColorName):** Define uno o más colores para mostrar regiones, XLD y otros objetos geométricos en la ventana gráfica.

- ✓ **ColorName:** Nombre del color de salida. El valor default es ‘white’. Y se sugieren los nombres: 'white', 'black', 'gray', 'red', 'green', 'blue', '#003075', '#e53019', '#ffb529'

- **dev_set_draw(DrawMode):** Define el tipo de llenado de la región. Si DrawMode está seteado a ‘fill’, se muestran regiones rellenas. Si se usa el modo ‘margin’ solo se muestran los contornos.
  - ✓ **DrawMode:** Tipo de relleno para la región de salida. La variable tiene que ser de tipo String.

- **Stop():** Detiene la ejecución de un programa. Equivale a presionar F9 y el programa puede ser reanudado por la acción Run,

A.3. Operadores de líneas

Los operadores aquí descritos tienen como objetivo encontrar, seleccionar y aislar elementos que se asemejen a líneas en cualquier dirección.

- **lines_gauss(Image, Lines, Sigma, Low, High, LightDark, ExtractWidth, LineModel, CompleteJunctions):** Detecta líneas y su ancho. Se puede usar para extraer líneas (estructuras curvilineas) desde la imagen *Image.*
  - ✓ **Lines:** Líneas extraídas**.** Tipo de variable: xld_cont-array → object
  - ✓ **Sigma:** Suavizado gaussiano a aplicar. Valor por defecto 1.5. Valores recomendados: 1, 1.2, 1.5, 1.8, 2, 2.5, 3, 4, 5. Rango de valores: 0.7..20. Incremento recomendado: 0.5. Restricción: LOW >=0
  - ✓ **High:** Threshold superior para la operación de threshold de histéresis. 8 como valor por defecto. Rango de valores: 0-35. Valores sugeridos: 0, 0.5, 1, 2, 3, 4, 5, 8, 10, 12, 15, 18, 20, 25. Incremento recomendado: 0.5. Restricción: High tiene que ser mayor que 0 y que Low.
  - ✓ **LightDark:** Extrae líneas claras o negras. Puede ser ‘light’ o ‘dark’.
  - ✓ **ExtractWidth:** ¿Debería extraerse el ancho de línea? Puede ser ‘false’ o ‘true’
  - ✓ **LineModel:** Modelo de línea usado para corregir la posición de la línea y su ancho. Puede ser 'bar-shaped', 'gaussian', 'none', 'parabolic'.
  - ✓ **CompleteJunctions:** ¿Debe añadirse uniones donde no se puedan extraer? Puede ser ‘false’ o ‘true’.

- **split_skeleton_lines(SkeletonRegion, MaxDistance : BeginRow, BeginCol, EndRow, EndCol):** Separa líneas representadas por regiones de un pixel de ancho y no ramificadas en líneas más cortas basadas en su curvatura. Una línea es separada si la máxima distancia de un punto de la línea al segmento de línea que conecta sus puntos finales es más larga que MaxDistance. Los puntos finales e iniciales de los segmentos de línea aproximados son devueltos en BeginRow, BeginCol,EndRow, y EndCol.
  - ✓ **SkeletonRegion:** Entrada. Líneas de entrada. Tienen que ser de tipo region-array → object.
  - ✓ **MaxDistance:** Entrada**.** Máxima distancia de los puntos de las líneas al segmento de línea que conecta ambos puntos finales.
  - ✓ **BeginRow:** Salida. Coordenadas de filas de los puntos iniciales de las líneas de salida.

- ✓ **BeginCol:** Salida. Coordenadas de columnas de los puntos iniciales de las líneas de salida.
- ✓ **EndRow:** Salida. Coordenadas de fila de los puntos finales de las líneas de salida.
- ✓ **EndCol:** Salida. Coordenadas de columnas de los puntos finales de las líneas de salida.

- **fit_line_contour_xld(Contours, Algorithm, MaxNumPoints, ClippingEndPoints, Iterations, ClippingFactor : RowBegin, ColBegin,Row End, ColEnd, Nr, Nc, Dist):** Aproxima Contours (contornos XLD) por segmentos de línea. No realiza una segmentación de los contornos de entrada. Es por eso, que uno tiene que asegurarse que cada contorno corresponde a uno y solo un segmento. El operador devuelve para cada contorno el punto de inicio (RowBegin, ColBegin), el punto final (RowEnd, ColEnd) y la línea de regresión del contorno que es dada por el vector normal (Nr, Nc) de la línea y su distancia Dist desde el origen. Por ejemplo, la ecuación de la línea es dada por: $r.Nr + c.Nc - Dist = 0$.

  El algoritmo usado para el ajuste de las líneas puede ser seleccionado mediante Algorithm**:**
  - ✓ **'regressión':** Ajuste de línea estándar por mínimos cuadrados.
  - ✓ **'huber':** Ajuste de mínimos cuadrados con pesos dónde el impacto de los valores atípicos es disminuido basado en la aproximación de Huber.
  - ✓ **'tukey':** Ajuste de mínimos cuadrados con pesos dónde el impacto de los valores atípicos es ignorado basado en el acercamiento de Tukey.
  - ✓ **'drop':** Ajuste de línea de mínimos cuadrados dónde el impacto de los valores atípicos es ignorado. En particular, todos los puentos del contorno

A.4. Operadores de contorno

En este apartado se expondrán operadores orientados a resaltar cambios bruscos de color (contornos) en las imágenes, las cuales tienen un proceso que consta de 6 pasos:

A.4.1. Crear contornos xld:

- Estándar:

1.- gen_contour_polygon_xld (: Contour : Row, Col :): Este operador sirve para generar un contorno XLD a partir de un polígono, contiene 3 parámetros:

- Contour: Este parámetro indica el contorno resultante, tiene como salida un XLD.

- Row: Este parámetro indica las coordenadas de fila del polígono.

- Col: Este parámetro indica las coordenadas de columna del polígono.

2.- gen_rectangle2_contour_xld (: Rectangle : Row, Column, Phi, Length1, Length2 :): Este operador sirve para crear un contorno XLD en forma de rectángulo, contiene 6 parámetros:

- Rectangle: Este parámetro da el contorno de un rectángulo, tiene como salida un XLD.

- Row: Este parámetro indica la coordenada de hilera del centro del rectángulo.

- Column: Este parámetro indica la coordenada de columna del centro del rectángulo.

- Phi: Este parámetro indica la orientación del eje principal del rectángulo [rad].

- Length1: Este parámetro indica el primer radio (mitad de longitud) del rectángulo.

- Length2: Este parámetro indica el segundo radio (mitad de ancho) del rectángulo.

- Avanzado:

1.-gen_contour_polygon_rounded_xld (: Contour : Row, Col, Radius, SamplingInterval :): Este operador sirve para generar un contorno XLD con esquinas redondeadas desde un polígono, contiene 5 parámetros:

- Contour: Este parámetro indica el contorno resultante, tiene como salida un XLD.

- Row: Este parámetro indica las coordenadas de fila del polígono.

- Col: Este parámetro indica las coordenadas de columna del polígono.

- Radius: Este parámetro indica los radios de las esquinas redondeadas.

- SamplingInterval: Este parámetro indica la distancia de las muestras.

A.4.2. Proceso de contornos xld:

- Estándar:

1.-distance_contours_xld (ContourFrom, ContourTo : ContourOut : Mode :): Este operador sirve para calcular la distancia puntual de un contorno a otro, tiene 4 parámetros:

- ContourFrom: Este parámetro indica los contornos para cuyos puntos se calculan las distancias, tiene como entrada un XLD.

- ContourTo: Este parámetro indica los contornos a los que se calculan las distancias, tiene como entrada una XLD.

- ContourOut: Este parámetro es la copia de ContourFrom (Primer parámetro de distance_contours_xld) que contiene las distancias como un atributo.

- Mode: Este parámetro calcula la distancia a puntos ('punto_al_punto') o a segmentos enteros ('punto_segmento_segmento').

2.- shape_trans_xld (XLD : XLDTrans : Type :): Este operador sirve para transformar la forma de los contornos o polígonos, contiene 3 parámetros:

- XLD: Este parámetro indica los contornos o polígonos a transformar, tiene como entrada un XLD.

- XLDTrans: Este parámetro indica los contornos transformados respectivamente polígonos, tiene como salida un XLD.

- Type: Este parámetro indica el tipo de transformación.

- Avanzado:

1.- symm_difference_closed_contours_xld (Contours1, Contours2 : ContoursDifference : :): Este operador sirve para calcular la diferencia simétrica de contornos cerrados, contiene 3 parámetros:

- Contours1: Este parámetro indica los contornos que encierran la primera región, tiene como entrada un XLD.

- Contours2: Este parámetro indica los contornos que encierran la segunda región, tiene como entrada un XLD.

- ContoursDifference: Este parámetro indica los contornos que encierran la diferencia simétrica, tiene como salida un XLD.

A.4.3. Realizar el montaje:

- Estándar:

1.-fit_circle_contour_xld (Contours : : Algorithm, MaxNumPoints, MaxClosureDist, ClippingEndPoints, Iterations, ClippingFactor : Row, Column, Radius, StartPhi, EndPhi, PointOrder): Este operador sirve para contornos aproximados de XLD por círculos 13 parámetros:

- Contours: Este parámetro indica los contornos de entrada, tiene como entrada una XLD.

- Algorithm: Este parámetro es un algoritmo para la instalación de círculos.

- MaxNumPoints: Este parámetro indica el número máximo de puntos de contorno utilizados para el cálculo (-1 para todos los puntos).

- MaxClosureDist: Este parámetro indica la distancia máxima entre los puntos finales de un contorno para ser considerado como "cerrado".

- ClippingEndPoints: Este parámetro indica el número de puntos al principio y al final de los contornos que deben ignorarse para el ajuste.

- Iterations: Este parámetro indica el número máximo de iteraciones para la conexión robusta y ponderada.

- ClippingFactor: Este parámetro indica el factor de recorte para la eliminación de valores atípicos (típico: 1.0 para Huber y 2.0 para Tukey).

- Row: Este parámetro indica la coordenada de hilera del centro del círculo.

- Column: Este parámetro indica la coordenada de columna del centro del círculo.

- Radius: Este parámetro indica el radio de círculo

- StartPhi: Este parámetro indica el ángulo del punto de inicio [rad].

- EndPhi: Este parámetro indica el ángulo del punto final [rad].

- PointOrder: Este parámetro indica el orden de puntos a lo largo del límite.

A.4.4.- Transformar resultados en coordenadas

- Estándar:

1.- contour_to_world_plane_xld (Contours : ContoursTrans : CameraParam, WorldPose, Scale :): Este operador sirve para transformar un contorno XLD en el plano z = 0 de un sistema de coordenadas mundial, contiene 5 parámetros:

- Contours: Este parámetro indica el ingreso de los contornos XLD para transformarlos en coordenadas de imagen, tiene como entrada un XLD.

- ContoursTrans: Este parámetro indica los contornos XLD transformados en coordenadas mundiales, tiene como salida un XLD.

- CameraParam: Este parámetro indica los datos internos de la cámara.

- WorldPose: Este parámetro indica el Pose 3D del sistema de coordenadas del mundo en coordenadas de la cámara.

- Scale: Este parámetro indica la escala o dimensión.

2.-change_radial_distortion_contours_xld (Contours : ContoursRectified : CamParamIn, CamParamOut :): Este operador sirve para cambiar la distorsión radial de los contornos, contiene 4 parámetros:

- Contours: Este parámetro indica los contornos originales, tiene como entrada un XLD.

- ContoursRectified: Este parámetro indica los contornos resultantes con distorsión radial modificada, tiene como salida un XLD.

-CamParamIn: Este parámetro indica el parámetro interno de la cámara para Contours (Primer parámetro de change_radial_distortion_contours_xld).

-CamParamOut: Este parámetro indica el parámetro interno de la cámara para ContoursRectified (Segundo parámetro de change_radial_distortion_contours_xld).

A.4.5.- Características del extracto:

- Estándar:

1.- smallest_circle_xld (XLD : : : Row, Column, Radius): Este operador sirve para conocer el círculo más pequeño de contornos o polígonos, contiene 4 parámetros:

- XLD: Este parámetro indica los contornos o polígonos para ser examinados, tiene como entrada un XLD.

- Row: Este parámetro indica la coordenada de hilera del centro del círculo circundante.

- Column: Este parámetro indica la coordenada de columna del centro del círculo circundante.

- Radius: Este parámetro indica el radio del círculo circundante.

2.- smallest_rectangle2_xld (XLD : : : Row, Column, Phi, Length1, Length2): Este operador sirve para conocer el rectángulo más pequeño que rodea con una orientación arbitraria de contornos o polígonos, contiene 6 parámetros:

- XLD: Este parámetro indica los contornos o polígonos para ser examinados, tiene como entrada un XLD.

- Row: Este parámetro indica la coordenada de hilera del punto central del rectángulo circundante.

- Column: Este parámetro indica la coordenada de columna del punto central del rectángulo circundante.

- Phi: Este parámetro indica la orientación del rectángulo circundante (medida de arco).

- Length1: Este parámetro indica el primer radio (mitad de longitud) del rectángulo circundante.

- Length2: Este parámetro indica el segundo radio (mitad de ancho) del rectángulo circundante.

3.- convexity_xld (XLD : : : Convexity): Este operador sirve para dar el factor de forma para la convexidad de contornos o polígonos, contiene 2 parámetros:

- XLD: Este parámetro indica los contornos o polígonos para ser examinados, tiene como entrada un XLD.

- Convexity: Este parámetro indica la convexidad de los contornos de entrada o polígonos.

A.4.6.- Convertir y acceder a los contours xld:

- Estándar:

1.- get_contour_xld (Contour : : : Row, Col): Este operador sirve para conocer las coordenadas de un contorno XLD, contiene 3 parámetros:

- Contour: Este parámetro indica el contorno de entrada XLD, tiene como entrada un XLD.

- Row: Este parámetro indica la coordenada de hilera de los puntos del contorno.

- Col: Este parámetro indica la coordenada de columna de los puntos del contorno.

2.- gen_region_contour_xld (Contour : Region : Mode :): Este operador sirve para crear una región a partir de un contorno XLD, contiene 3 parámetros:

- Contour: Este parámetro indica el (los) contorno(s) de entrada, tiene como entrada un XLD.

- Region: Este parámetro indica la (s) Región(es) creada(s), tiene como salida una Región.

- Mode: Este parámetro indica el modo de relleno de la(s) región(es).

- Avanzado:

1.- paint_xld (XLD, Image : ImageResult : Grayval :): Este operador sirve para pintar objetos XLD en una imagen, contiene 4 parametros:

- XLD: Este parámetro indica los objetos XLD para pintar en la imagen de entrada, tiene como primera entrada un XLD.

- Image: Este parámetro indica la imagen en la que se van a pintar los objetos xld, tiene como segunda entrada una Imagen.

- ImageResult: Este parámetro indica la imagen que contiene el resultado, tiene como salida una Imagen.

- Grayval: Este parámetro indica el valor de gris deseado del objeto xld.

- **gen_region_contour_xld(Contour(Input) : Region(output) : Mode :):** Crea una región a partir de un contorno XLD.
  - ✓ **Contour:** El contorno a convertir y puede ser de los siguientes tipos: xld_cont(-array) → object
  - ✓ **Region:** La región creada que es del tipo: region(-array) → object

- ✓ **Mode**: Determina el tipo de relleno de la región. Puede ser ‘filled’ o ‘margin’

A.5. Operadores de filtro

Estos operadores se utilizan para remarcar cambios de tonalidades de interés en las imágenes. Esto es muy útil porque permite resaltar uno u otro parámetro de la imagen que sea necesario.

- **gauss_filter(Image, ImageGauss , Size):** Sirve para realizar suavizados en la imagen con funciones Gausianas discretas. Las funciones gausianas discretas se aproximan a una función Gausiana verdadera. El efecto de suavizado aumenta con el tamaño del filtro y este tamaño depende del valor del sigma en la función gausiana.

- ✓ Image: Imagen a ser suavizada.
- ✓ ImageGauss: Imagen filtrada.
- ✓ Size: Tamaño del filtro requerido.

- ✓ **binomial_filter(Image, ImageBinomial , MaskWidth, MaskHeight):** Sirve para realizar el suavizado de una imagen utilizando un filtro binomial. Este filtro posee una máscara de un tamaño determinado por los parámetros de *MaskWidth* y *MaskHeight*. El filtro binomial es bastante aproximado a un filtro Gausiano. Los parámetros del operador son:

- ✓ Image: Imagen de entrada a ser filtrada.
- ✓ ImageBinomial: Imagen suavizada con un filtro binomial.
- ✓ MaskWidth: Ancho de la máscara del filtro.
- ✓ MaskHeight: Alto de la máscara del filtro.

- **bilateral_filter(Image, ImageJoint,ImageBilateral,SigmaSpatial,SigmaRange, GenParamName, GenParamValue)**: Sirve para realizar un filtro bilateral de una imagen a partir de una imagen conjunta. Cada píxel de la imagen se filtra con una máscara de filtro que depende de la imagen conjunta (ImageJoint). La máscara de filtro combina una función de proximidad gaussiana en función de SigmaSpatial y una función de similitud gaussiana que pondera las diferencias de valores grises según SigmaRange.
- ✓ Image: Imagen a ser filtrada.
- ✓ ImageJoint: Imagen conjunta para realizar el filtrado.
- ✓ SigmaSpatial: Define el tamaño de la máscara del filtro Gausiano.
- ✓ SigmaRange: Se usa para modificar la máscara del filtro dependiendo de la imagen conjunta.
- ✓ GenParamName y GenParamValue: Se pueden usar para controlar la compensación entre exactitud y velocidad

Si la imagen y la imagen conjunta son iguales se observa como los bordes de la imagen original se conservan sin suavizar.

Figura A1: Proceso de filtrado *bilateral_filter* con parámetros icónicos iguales.

Si la imagen y la imagen conjunta son diferentes, solo se conservan los bordes sin suavizar en la zona común entre la imagen y la imagen conjunta.

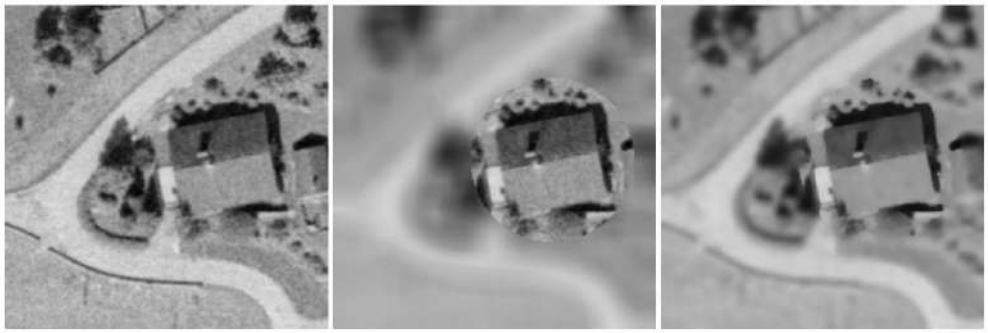


Figura A2: Proceso de filtrado *bilateral_filter* con parámetros icónicos diferentes.

Si la imagen y la imagen conjunta son distintas y la segunda es una constante, se realiza el suavizado Gausiano en toda la imagen

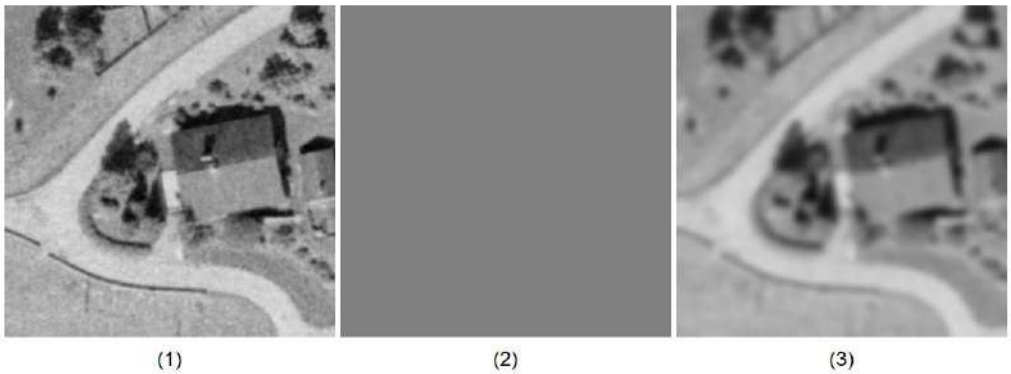
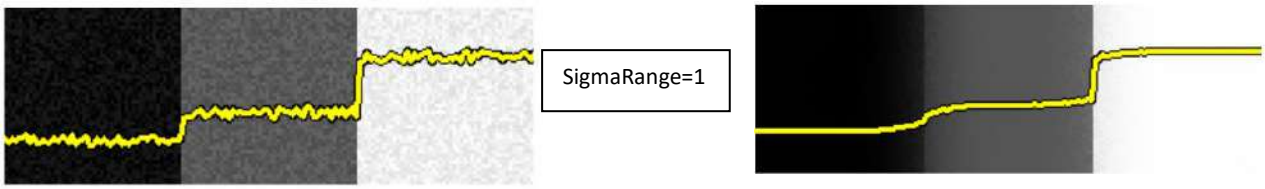


Figura A3: Proceso de filtrado *bilateral_filter* con parámetros icónicos diferentes siendo uno una constante.

Los efectos de la variación del *SigmaRange* en el suavizado es el siguiente:



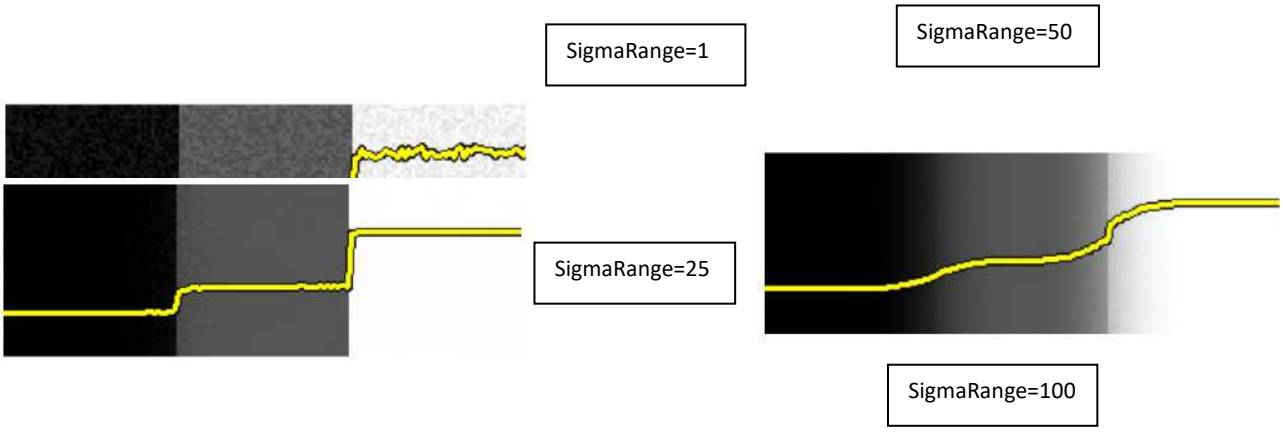


Figura A4: Efectos de la variación del parámetro *SigmaRange* en el proceso de filtrado bilateral.

- **shock_filter(Image,SharpenedImage,Theta, Iterations, Mode, Sigma):** Se utiliza para aplicar un filtro shock a una imagen para afilar los bordes contenidos.

✓ Image: Imagen a ser filtrada.
✓ SharpenedImage: Imagen filtrada.
✓ Theta: Tiempo de paso.
✓ Iterations: Numero de iteraciones.
✓ Mode: Tipo de detector de bordes.
✓ Sigma: Suavizado del detector de bordes.

- **mean_image(Image : ImageMean :MaskWidth, MaskHeight :):** Suavizado por promedio. Con este operador se realiza un filtrado con una matriz de tamaño asignado en los parámetros, se realiza una convolución y el resultado se divide entre el producto de las dimensiones de la matriz.

✓ Image: Imagen a ser suavizada.
✓ ImageMean: Imagen filtrada.
✓ MaskWidth: Largo de la máscara de filtrado.
✓ MaskHeight: Alto de la máscara de filtrado.

- **guided_filter(Image, ImageGuide, ImageGuided, Radius, Amplitude):** guided_filter filtra la imagen de entrada utilizando la imagen de guía mageGuide y devuelve el resultado en ImageGuided. Image e ImageGuide deben ser del mismo tamaño y tipo.

✓ Image: Imagen de entrada a la cual se le realizará el filtro.
✓ ImageGuide: Imagen
✓ ImageGuided:
✓ Radius: El radio es del tamaño de la máscara de filtro. Los valores más grandes aumentan el área de influencia del filtro y se conservan menos detalles. El valor de Radio no influye en el tiempo de ejecución del operador.

- ✓ Amplitude: La amplitud se usa para decidir qué es un borde y qué es un área homogénea. Los valores más grandes de amplitud conducen a bordes más fuertes que se suavizan. Como regla general, la Amplitud debe ser menor que el contraste de los bordes que se deben preservar. Tenga en cuenta que el contraste en uint2 o imágenes reales puede diferir significativamente de los valores predeterminados de Amplitud y ajustar el parámetro en consecuencia.

  Este filtro actúa bajo el mismo principio que el filtro bilateral visto anteriormente.

A.6. Operadores de regiones

Aquí se encontrarán operadores que sirven para operar, convertir y crear regiones.

- **region_to_label(Region, ImageLabel, Type, Width, Height):** Convierte una region o regiones en una imagen etiquetada.
  - ✓ **Region:** Región o regiones a ser convertidas. Pueden ser de tipo: region(-array) → object
  - ✓ **ImageLabel:** Imagen resultante y de dimensiones Width*Height que contiene las regiones convertidas.
  - ✓ **Type:** Determina el tipo de variable de la imagen de salida y puede ser: 'byte', 'int2', 'int4', 'int8'.
  - ✓ **Width:** El ancho de la imagen que se va a generar. Es un número que puede ir de 1 a 1024. Se recomienda un incremento de 16 y los valores sugeridos son: 64, 128, 256, 512, 1024.
  - ✓ **Height:** Alto de la imagen que se va a generar. Es un número que puede ir de 1 a 1024. Se recomienda un incremento de 16 y los valores sugeridos son: 64, 128, 256, 512, 1024.

- **region_to_bin(Region, BinImage, ForegroundGray, BackgroundGray, Width,Height):** Convierte una region en una imagen binaria tipo byte.
  - ✓ **Region:** Región a ser convertida. Puede ser del tipo region(-array) → object.
  - ✓ **BinImage:** Imagen resultante y de dimensiones Width*Height que contiene las regiones convertidas. Puede ser del tipo: image → object (byte)
  - ✓ **ForegroundGgray:** Valor del gris en el que las regiones son mostradas. Puede ser un número de 0 a 255. Se sugieren los siguientes valores: 0, 1, 50, 100, 128, 150, 200, 254, 255
  - ✓ **BackgroundGray:** Valor del gris en el que el fondo es mostrado. Puede ser un número de 0 a 255. Se sugieren los siguientes valores: 0, 1, 50, 100, 128, 150, 200, 254, 255.
  - ✓ **Width:** El ancho de la imagen que se va a generar. Es un número que puede ir de 1 a 1024. Se recomienda un incremento de 16 y los valores sugeridos son: 256, 512, 1024.
  - ✓ **Height:** Alto de la imagen que se va a generar. Es un número que puede ir de 1 a 1024. Se recomienda un incremento de 16 y los valores sugeridos son: 256, 512, 1024.

- **convert_image_type(Image, ImageConverted, NewType):** Convierte el tipo de una imagen.
  - ✓ **Image:** Imagen cuyo tipo es cambiado. Puede tener los siguientes formatos: (multichannel-)image(-array) → object (byte* / direction* / cyclic* / int1* / int2* / uint2* / int4* / int8 / real* / complex*) *allowed for compute devices.
  - ✓ **ImageConverted:** Imagen ya convertida.
  - ✓ **NewType:** Tipo de imagen deseado. Pueden ser: 'byte', 'complex', 'cyclic', 'direction', 'int1', 'int2', 'int4', 'int8', 'real', 'uint2'. Para dispositivos de cálculo pueden ser: 'int1', 'int2', 'uint2', 'int4', 'byte', 'real', 'direction', 'cyclic', 'complex'.

- **connection(Region, ConnectedRegions):** Calcula los componentes conectados de una región.
  - ✓ **Region:** Región de entrada. Pude ser de tipo: region(-array) → object
  - ✓ **ConnectedRegions:** Componentes conectados. Puede ser de tipo: ConnectedRegions.

- **select_shape(Regions, SelectedRegions, Features, Operation, Min, Max):** Escoge regiones con la ayuda de características. Para cada región de entrada (Regions) se calcula sus características (Features) indicadas. Si todas(Operations=’and’) o al menos una (Operations=’or’) de las características calculadas de la región de entrada está dentro de los límites Min y Max, se incluye en la región de salida. Si solo se usa una característica, el valor de Operation no importa.
  - ✓ **Regions (Input):** Regiones a ser examinadas. Puede ser de tipo: region-array → object.
  - ✓ **SelectedRegions (Output):** Regiones que cumplen la condición. Puede ser de tipo: region-array → object.
  - ✓ **Features:** Características a ser calculadas y comprobadas. Pueden ser: 'anisometry', 'area', 'area_holes', 'bulkiness', 'circularity', 'column', 'column1', 'column2','compactness', 'connect_num', 'contlength', 'convexity', 'dist_deviation', 'dist_mean', 'euler_number', 'height','holes_num', 'inner_height', 'inner_radius', 'inner_width', 'max_diameter', 'moments_i1', 'moments_i2','moments_i3', 'moments_i4', 'moments_ia', 'moments_ib', 'moments_m02', 'moments_m02_invar','moments_m03', 'moments_m03_invar', 'moments_m11', 'moments_m11_invar', 'moments_m12','moments_m12_invar', 'moments_m20', 'moments_m20_invar', 'moments_m21', 'moments_m21_invar','moments_m30', 'moments_m30_invar', 'moments_phi1', 'moments_phi2', 'moments_psi1', 'moments_psi2','moments_psi3', 'moments_psi4', 'num_sides', 'orientation', 'outer_radius', 'phi', 'ra', 'rb', 'rect2_len1', 'rect2_len2','rect2_phi', 'rectangularity', 'roundness', 'row', 'row1', 'row2', 'struct_factor', 'width'
  - ✓ **Operation:** Tipo de vínculo entre las características individuales. Puede ser: ‘and’ o ‘or’.
  - ✓ **Min y Max:** Límites mínimos y máximos de las características. Pueden ir de 0 a 99999. Se recomienda un increment de 1 y Max debe ser mayor a Min. Pueden ser de tipo: real(-array) → (real / integer / string)

- **Skeleton(Región (entrada), Skeleton(salida)):** Calcula el esqueleto de una región. El esqueleto es construido de tal forma que cada uno de sus puntos puede

ser visto como el centro de un círculo con el mayor radio que sea posible siempre y cuando todavía esté completamente contenido en la región.

- ✓ **Región:** Región que se va a afinar y puede ser del siguiente tipo: region(-array) → object
- ✓ **Skeleton:** Esqueleto resultante y puede ser de los siguientes tipos de variables: region(-array) → object.

Apéndices

Apéndice A: Aplicaciones desarrolladas en HDevelop HALCON

A.1. Identificación de vasos

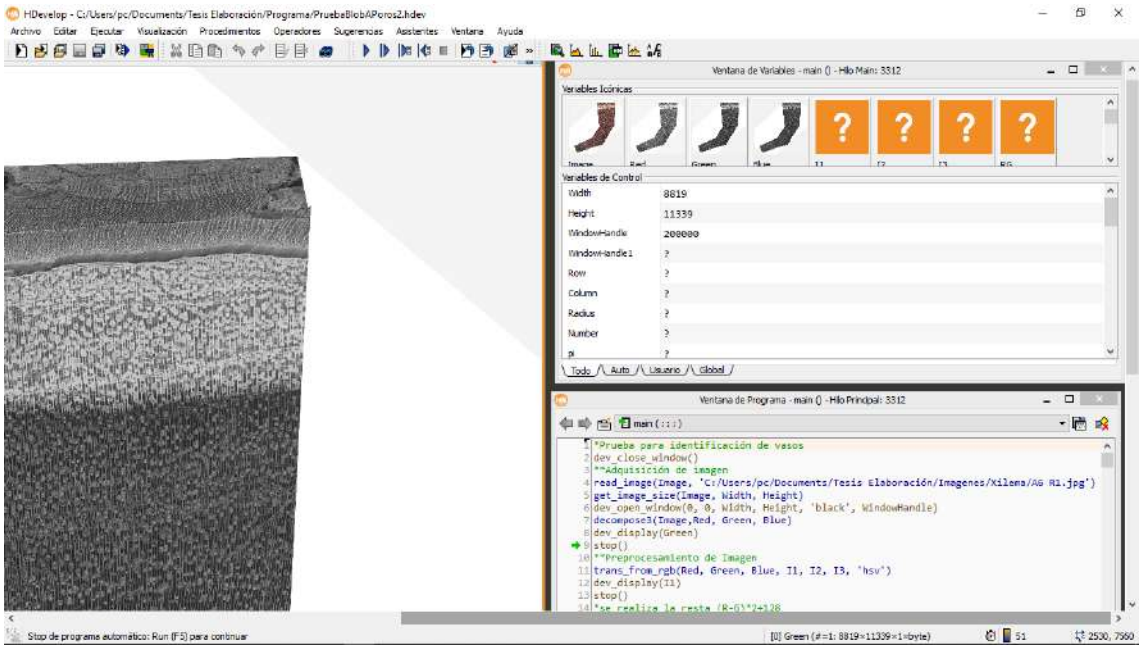


Figura A1. Procedimiento de adquisición de imagen

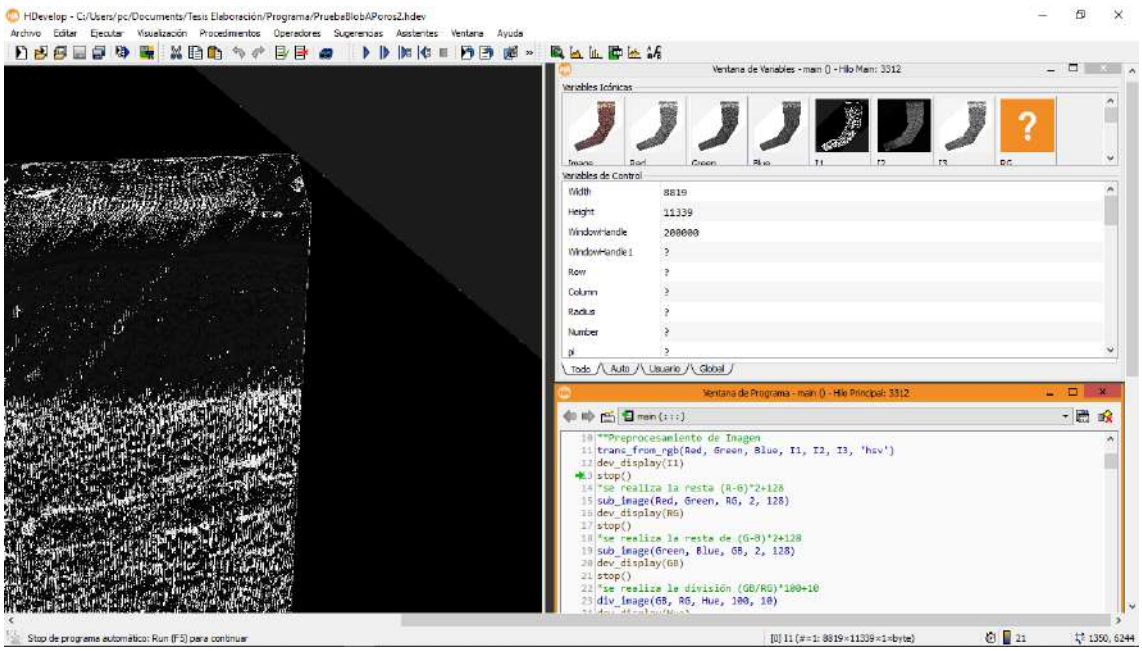


Figura A2. Proceso de transformación del índice RGB al HSV.

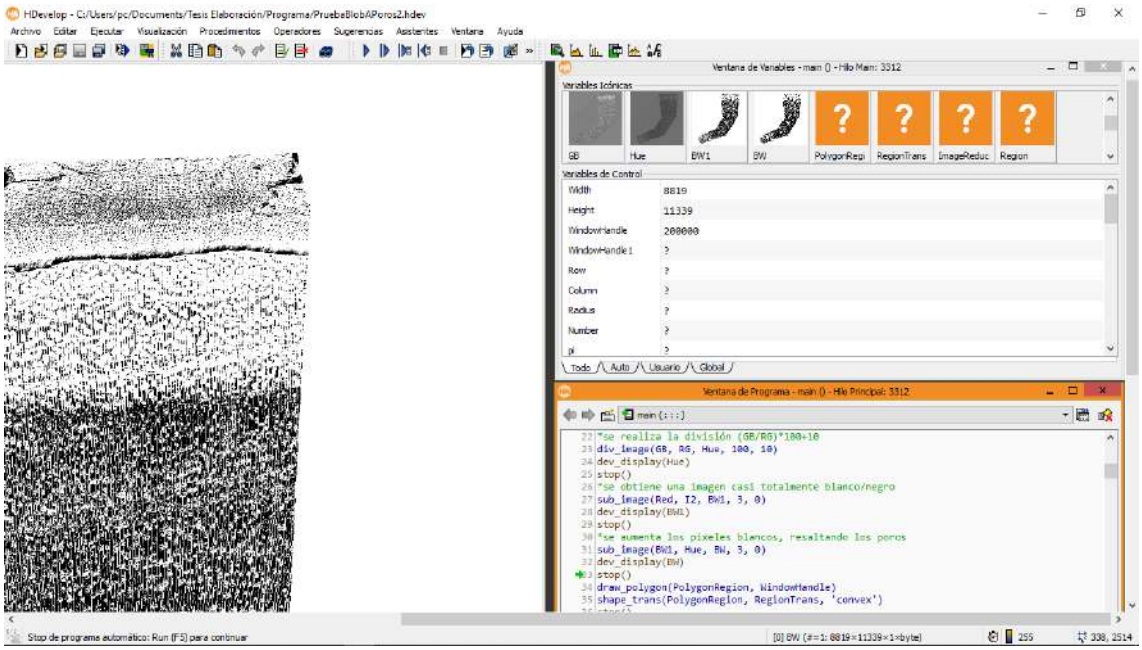


Figura A3. Finalización del pre-procesamiento de imagen.

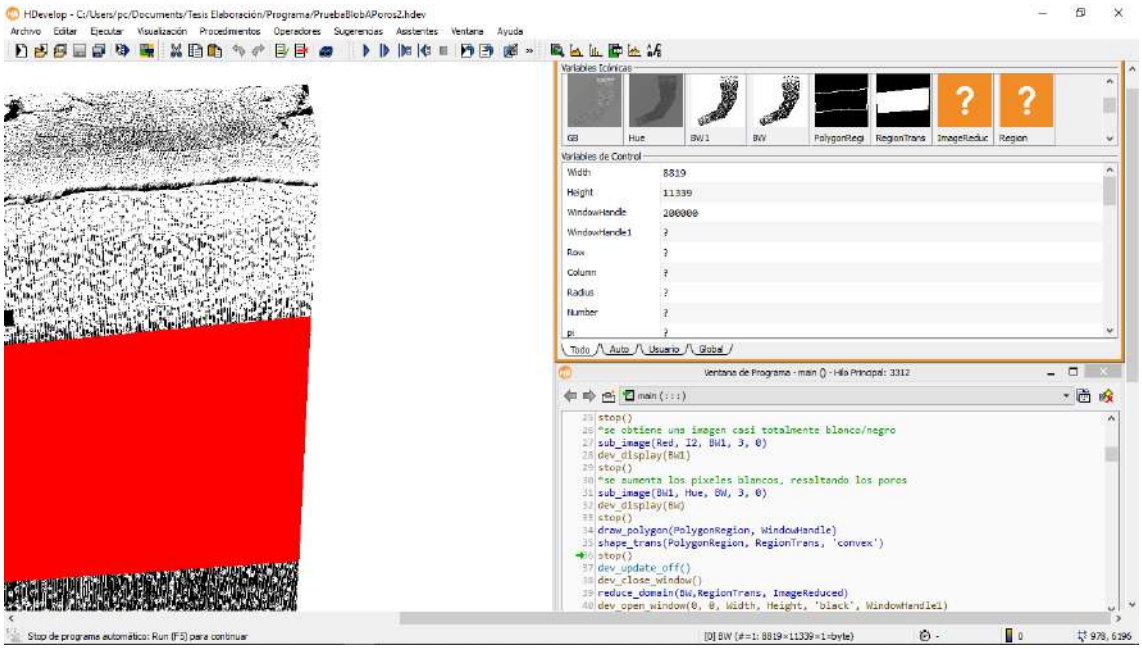


Figura A4: Selección de area para segmentación de imagen.

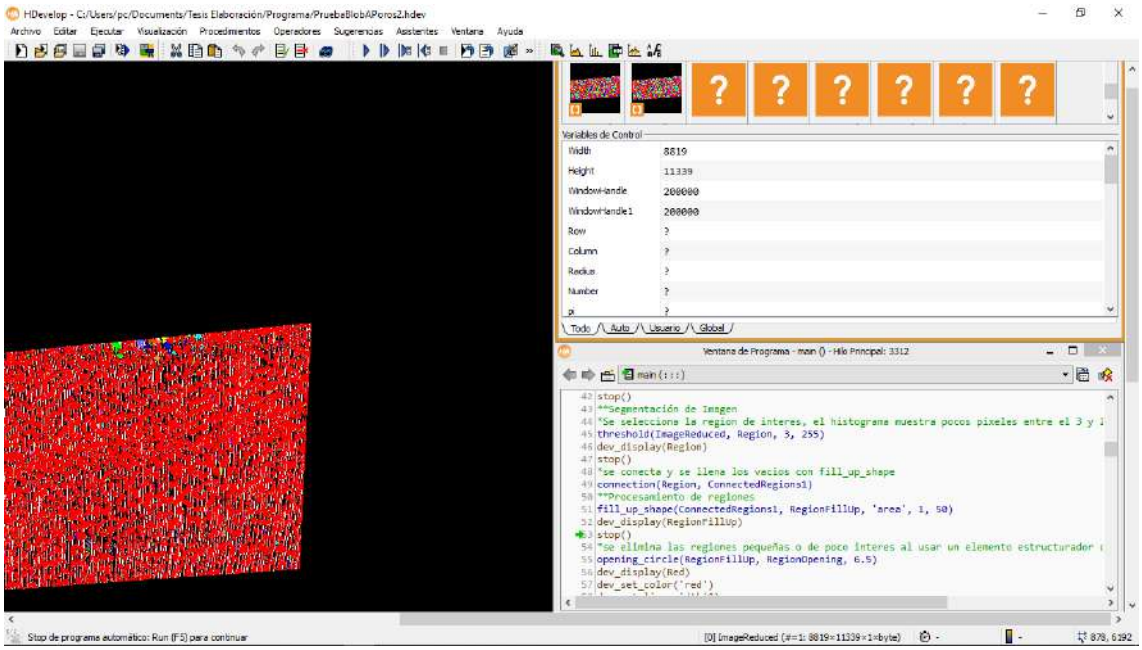


Figura A5. Segmentación de imagen desde selección de región hasta llenado de vacíos.

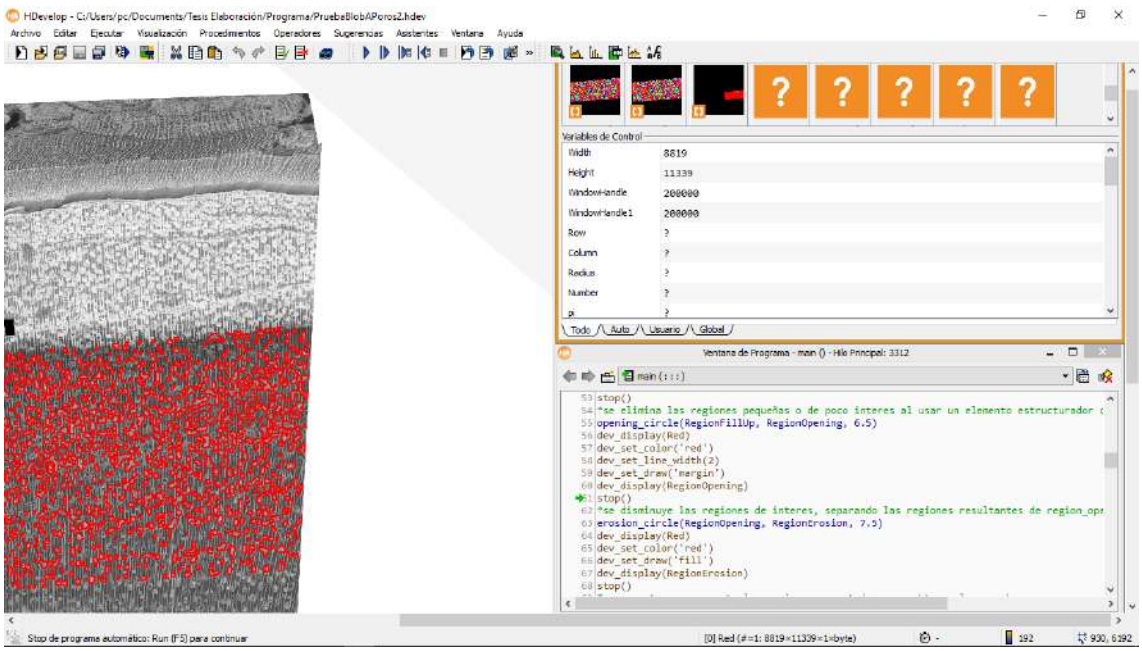


Figura A6. Proceso de elemento estructurador de regiones.

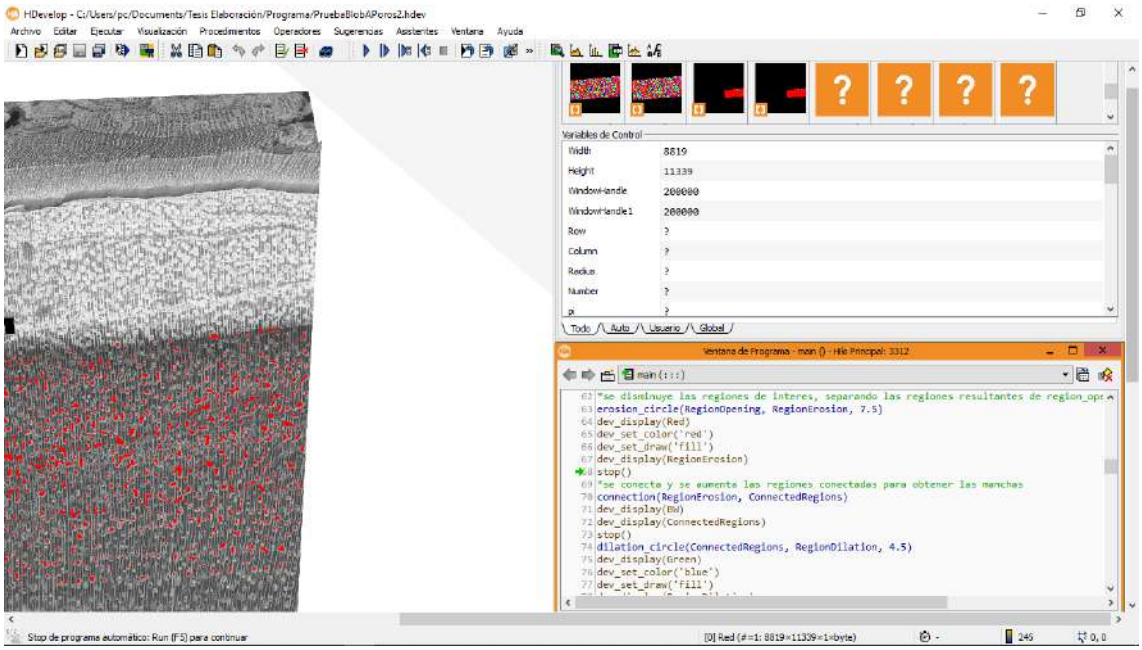


Figura A7. Proceso de elemento separador de regiones.

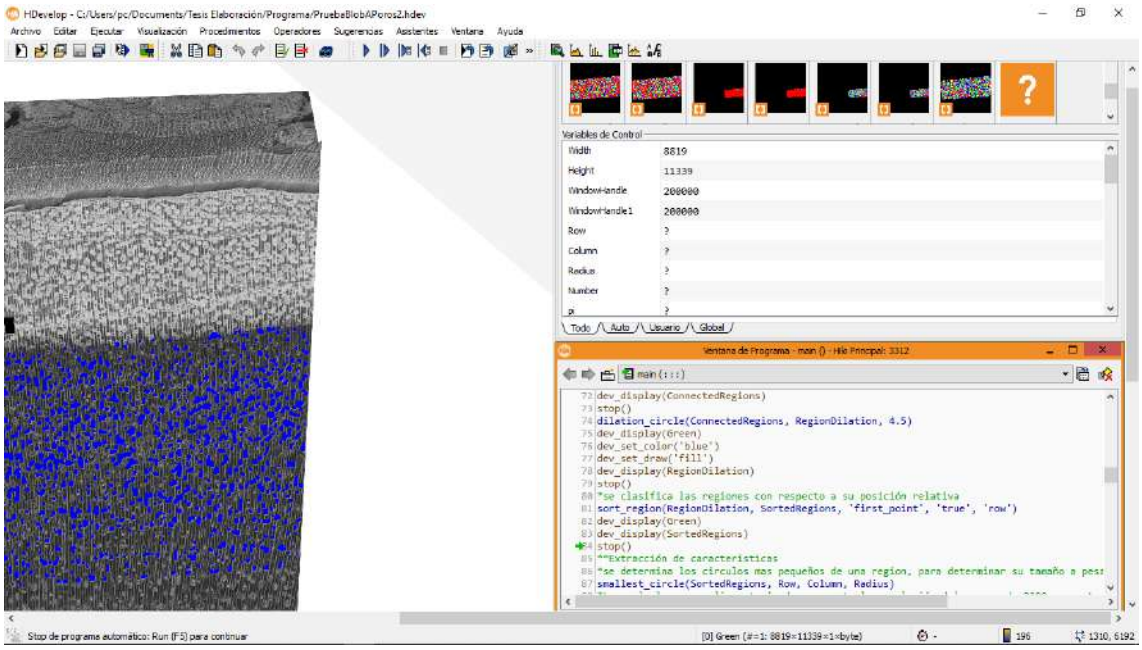


Figura A8. Finalización de la segmentación de imagen.

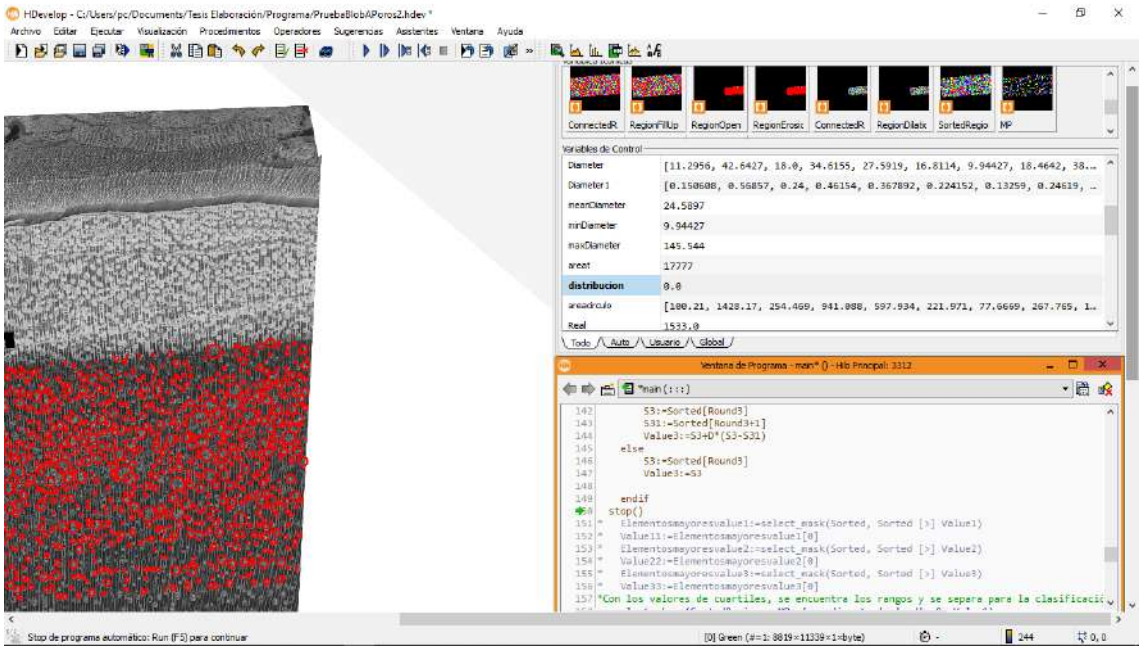


Figura A9. Proceso de extracción de características.

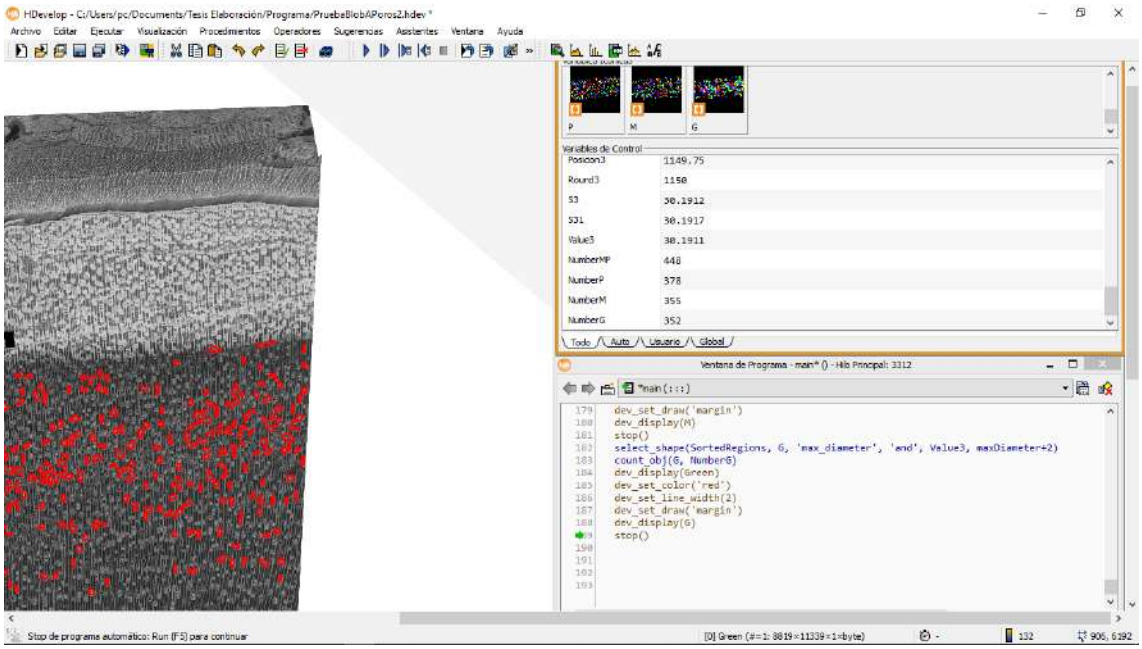


Figura A10. Proceso de separación por tamaño de las regiones identificadas.

A.2. Identificación de anillos de crecimiento anual

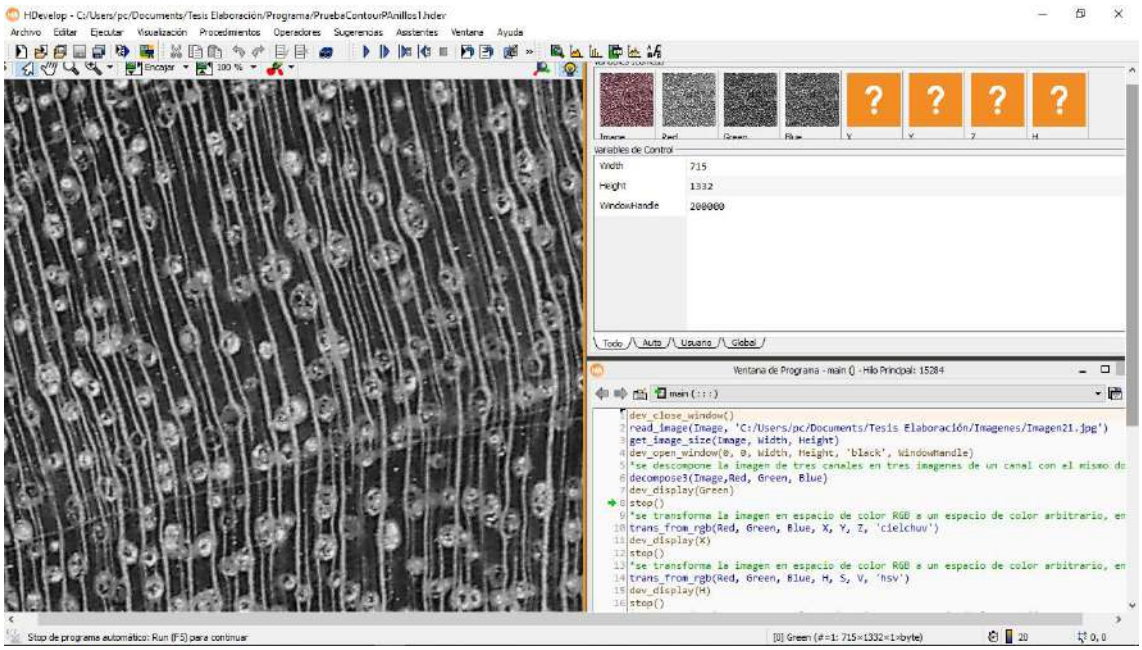


Figura A11. Proceso de adquisición de imagen y descomposición de componentes RGB.

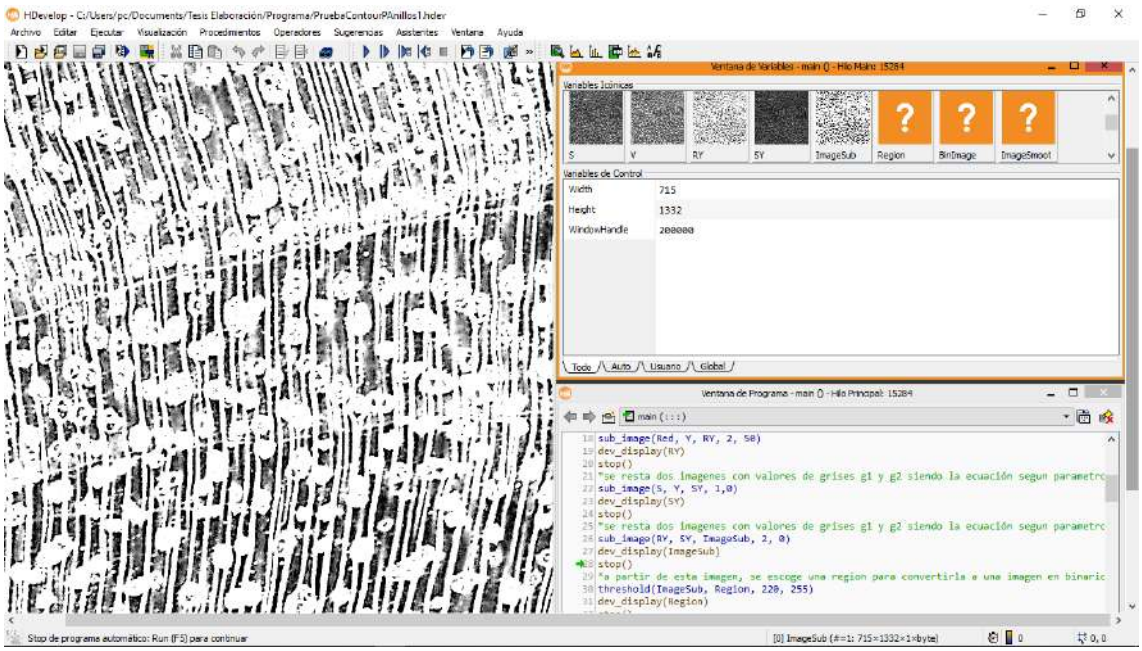


Figura A12. Finalización del pre-procesamiento de la imagen.

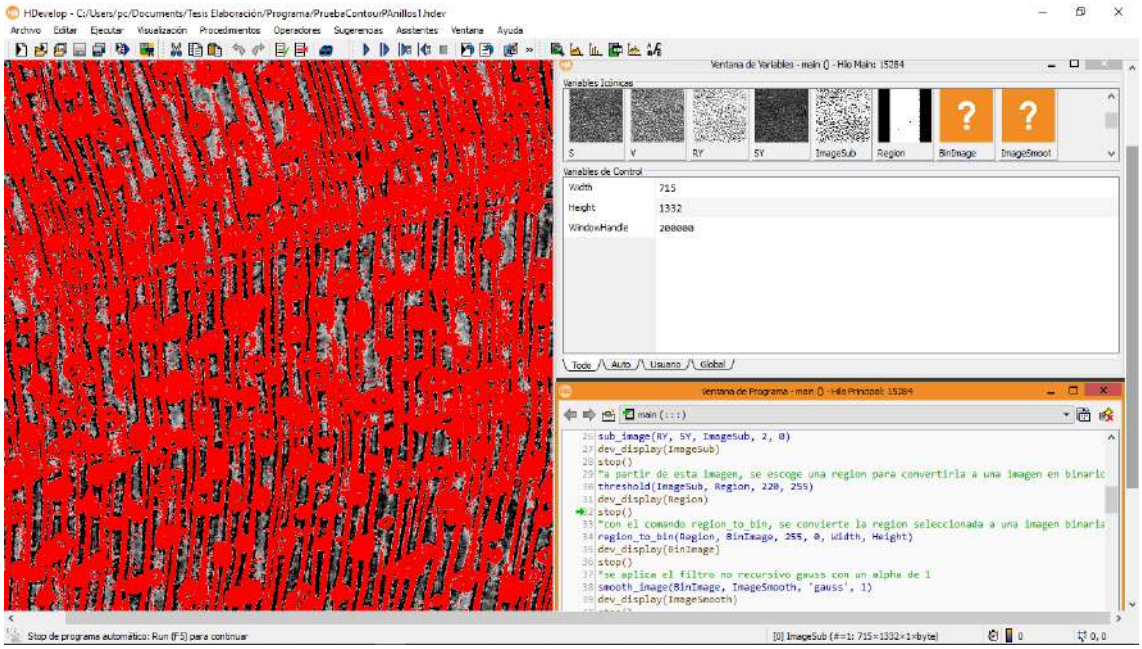


Figura A13. Proceso de selección de región.

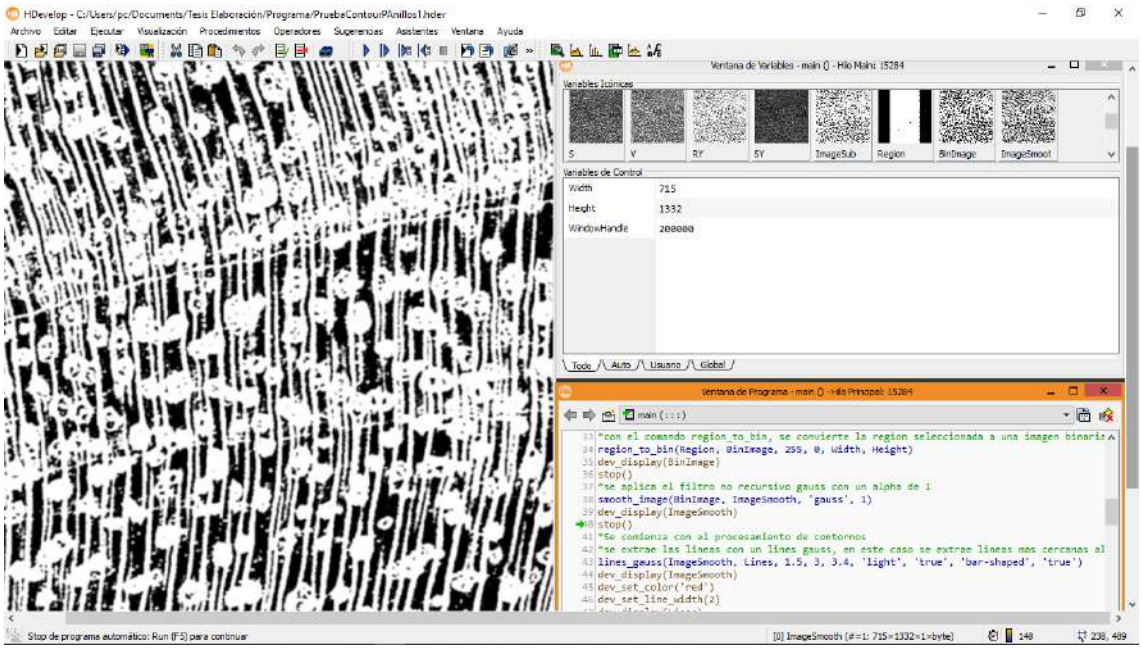


Figura A14. Proceso de conversión de una región a una imagen binaria y aplicación de filtrado.

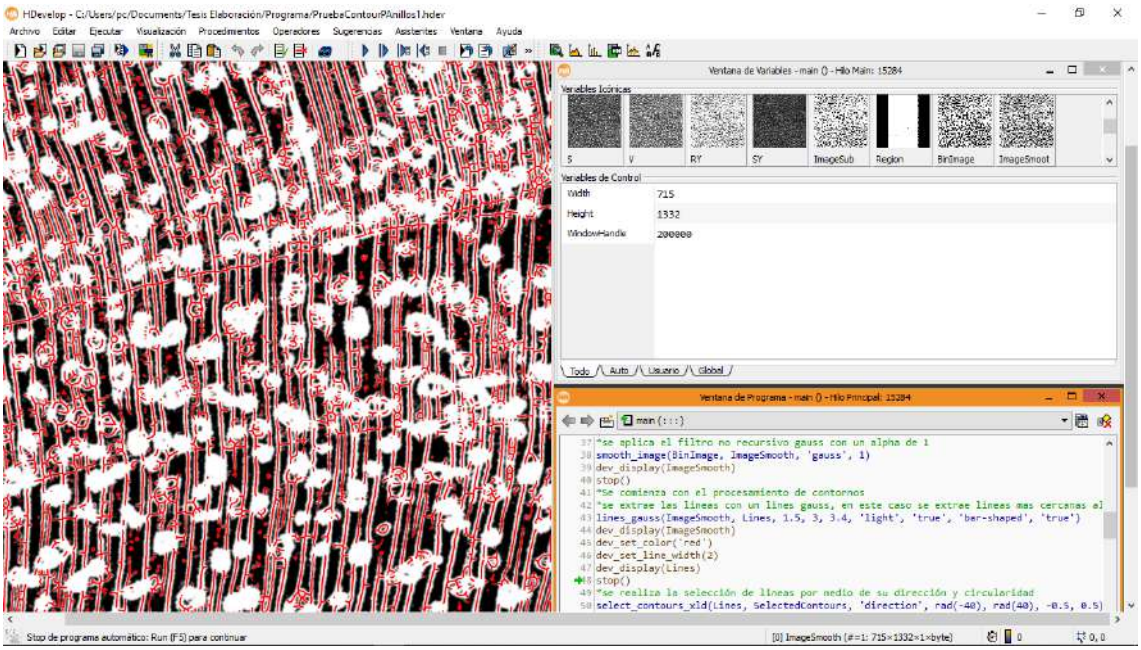


Figura A15. Extracción de líneas con el operador *line_gauss*.

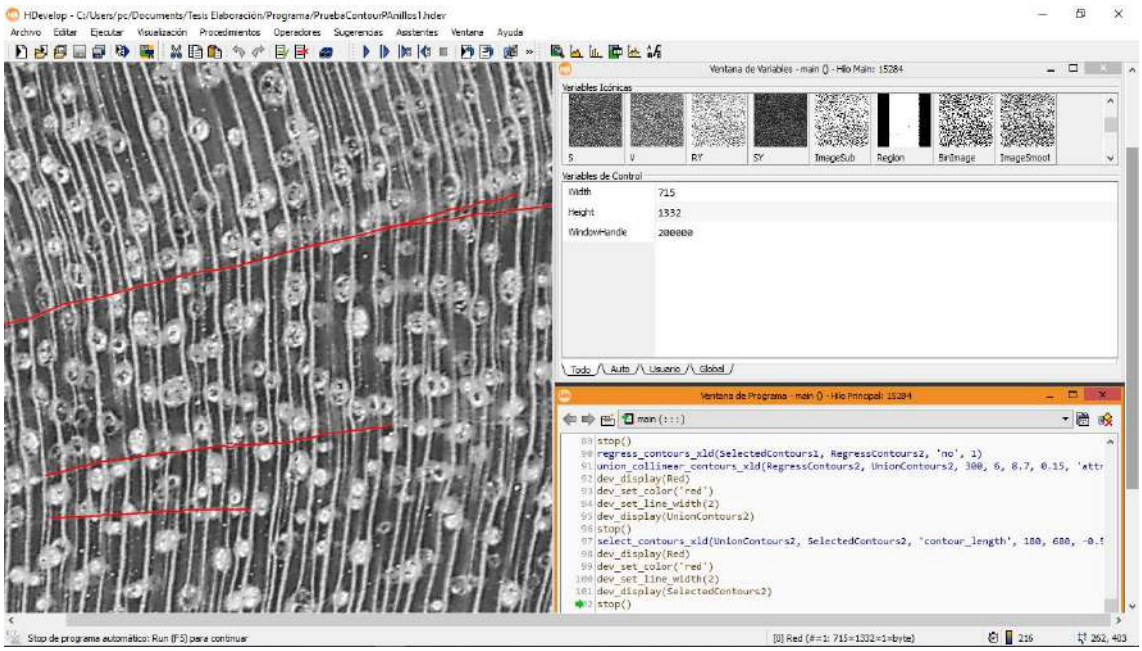


Figura A16. Selección y unión de contornos variando parámetros.