Desarrollo de aplicaciones en *software* libre para microscopía óptica avanzada.

Joaquim Soriano Felipe

Tesis Doctoral

Universidad de Castilla-La Mancha

Facultad de Medicina
Departamento de Ciencias Médicas

Desarrollo de aplicaciones en *software* libre para microscopía óptica avanzada.

Joaquim Soriano Felipe
Trabajo presentado para optar al Grado de Doctor

Directores de tesis:
Beatriz Domingo Moreno
Diego Megías Vázquez

Albacete, 2018

Beatriz Domingo Moreno, Profesor Contratado Doctor Interino del Departamento de Ciencias Médicas de la Universidad de Castilla-La Mancha y DiegoMegías Vázquez, Jefe de la Unidad de Microscopía Confocal del Centro Nacional de Investigaciones Oncológicas,

CERTIFICAN:

Que la presente tesis doctoral, titulada "Desarrollo de aplicaciones en software libre para microscopía óptica avanzada", constituye el trabajo de investigación realizado bajo nuestra dirección por Dº. Joaquim Soriano Felipe.

En este trabajo se desarrollan aplicaciones de procesado y análisis de imagen para la automatización de tareas complejas o rutinarias y para implementar diseños experimentales de captura guiada por análisis de imagen.

Esta tesis doctoral reúne la calidad y el rigor científico necesarios para ser defendida en la Universidad de Castilla-La Mancha, para que Joaquim Soriano Felipe opte al grado de Doctor.

Albacete, Julio de 2018

Fdo. Beatriz Domingo Moreno Fdo. Diego Megías Vázquez.

RESUMEN

Los microscopios actuales son equipos complejos compuestos por componentes mecánicos y electrónicos capaces de producir grandes cantidades de imágenes de origen muy diverso. El *software* que gobierna la interacción de estos componentes ocupa un papel central en la arquitectura de estos sistemas. Además, las imágenes generadas deben ser visualizadas, analizadas, procesadas y mantenidas por aplicaciones especializadas. El *software* propietario de equipos integrados es incapaz de adaptarse rápidamente a nuevas demandas experimentales, pero el desarrollo de aplicaciones en *software* libre puede salvar este problema.

En esta tesis presentamos el desarrollo de aplicaciones en *software* libre que han permitido extender las capacidades de procesado y captura de equipos de microscopía comerciales.

Palabras clave: microscopio, microscopía, desarrollo de *software*, *software* propietario, *software* libre, *software* de código abierto, análisis y procesado de la imagen digital.

ABSTRACT

Modern microscopes are complex instruments made of mechanical and electronic components that are able to produce large amounts of a wide diversity of images. The *software* orchestrating the interaction of such components plays a leading role in this equipment's architecture. Furthermore, captured images need to be visualized, analyzed, processed and stored by specialized applications. Integrated equipment's proprietary *software* is unable to rapidly adapt to new experimental demands, but free open source *software* development can overcome this problem.

In this thesis, we introduce free open source *software* based applications that are allowing us to extend the capturing and processing capabilities of commercially available microscopy equipment.

Key words: microscope, microscopy, *software* development, proprietary *software*, free *software*, open source *software*, digital image analysis and processing

Mi madre fue un día a hacerse una mamografía rutinaria y le dijeron aquello de: “Aquí parece que se quiere ver algo, pero no sé, … no será nada, … vuelva a hacerse la revisión el año que viene, …”.

Ahora está muerta.

Hay quien dice que la mataron unas células como las de las Figuras 11 o 33 de esta tesis, hay quien dice que los recortes … quiero pensar que esa consulta duró más de cinco minutos y que a la imagen que no se veía se le modificó brillo y contraste … no lo sé, porque no estuve ahí …

Dedico esta tesis a la memoria de mi madre y a la de todas las personas, sea cual sea su ocupación, que intentan hacer su trabajo responsable y profesionalmente, pensando en los demás, a pesar de los pesares, como hacía ella y como enseñó a hacer a sus hijos y alumnos.

Nunca pensé que escribiría unos agradecimientos. Siempre me han parecido una cursilería, un acto de vanidad, hipócrita y falso. Hartito terminé, en mi época de *predoc*, de escuchar aquello de "Menudo hijo de puta, allá revientes" y leer aquello de "Gracias a mi jefe, que todo me lo dio".

Luego está lo de:

"Voy a enumerar a todos mis amigos y familia, después escribiré un *biopic*, con tufillo a comedia romántica, y lo voy a adornar con citas de gente que no he leído, presocráticos, ¡que visten mucho una tesis!, o mejor: ¡Viva el empirismo inglés!, ¡¡y que no falten lomos de gigantes!!, ¡¡¡y el gato: a Flufly, por su cariño incondicional!!!".

Menuda grima …, evito leer los agradecimientos siempre que puedo …

Dice mi amigo Cesc que se creerán que soy un insociable o algo peor, si no los escribo, que es un convencionalismo, que todo el mundo odia a su director de tesis, …, pues yo no: mi director de tesis es un tío de puta madre, ¡si hasta le gusta Siniestro Total!; de mi directora podría escribir ríos de tinta, …, solo diré que tiene más cómics que yo y los he leído todos, …, ¡así no se pueden escribir unos agradecimientos en condiciones! La verdad es que a mis directores los elegí como son conscientemente, todo lo bueno (y lo malo) de estos años, es responsabilidad mía.

Al final, me he dado cuenta que los agradecimientos de la tesis será lo único que escribiré en mi carrera científica, …, ¡sin que nadie lo corrija! Visto así, tiene su *cosica*, …

Pues ala, al tema …

Aprendí a leer y a escribir en el sistema público de enseñanza. He tenido los mejores profesores. *Gratis*. Mi familia y amigos hicieron de mí una persona, mención especial a mi madre, no se lo puse fácil. Recibí una educación excepcional a un coste asequible en la Universitat de València. Aprendí lo que debería ser la ciencia en el "Màster per a la didáctica de les ciències experimentals i socials" de esta universidad, y lo que realmente es en el grupo de Neurobiología de la Audición de la Universidad de Castilla-la Mancha y en la Unidad de Microscopía Confocal del Centro Nacional de Investigaciones Oncológicas. A excepción de la escritura de la tesis, siempre tuve la suerte de cobrar por mi trabajo y siempre del erario público. Usé ése dinero con responsabilidad. Me siento un privilegiado por haber tenido la oportunidad de formarme allí dónde he estado y por haber podido ganarme la vida honestamente con lo que me gusta.

Mis directores de tesis han invertido trabajo y esfuerzo en este proyecto. Esto no es un agradecimiento, simplemente estoy constatando un hecho. Por hacerlo con ganas, optimismo y sentido del humor, sí que les estoy agradecido, y mucho. La ciencia es un trabajo en equipo, nadie escribe una tesis solo y acaba cuerdo.

Y hasta aquí el *biopic* cursi, ahora van los gigantes … Juan Llopis me enseñó lo que era un confocal, Diego Megías y el equipo de Leica (Paco Porto, Juan Luís Monteagudo y José Doncel "Joselito, el largo") cómo son los microscopios por dentro y a utilizarlos, de Pepe Pertusa, Sebastien Totsi, Julien Colombelli, Kota Miura … (¡casi ná!) aprendí el análisis de imagen, … y otros muchos que no vienen a cuento … (¡no me puedo dejar a Carlos Avendaño: menudo crack!).

Verónica Santamaría y Juan Carlos Alvarado son los mejores compañeros de trabajo que he tenido. Eckhard Friauf, el mejor jefe (Dieguito, no me llores, mejor solo puede haber uno, y este es alemán y te saca 20 años de experiencia …).

Llega ya el turno de mi familia y otros animales, …

Me fui a vivir con la loca de los gatos con la secreta esperanza de reformarla, …, ¡y me he vuelto uno de ellos! Así que, …, ¡a todos los gatos que han pasado por nuestra casa de acogida!, ¡cuánto más tarados, más los quiero! … ¡Manu, Rusito y Pam! … que entre los tres no hacen uno … se acabaron quedando porque nadie los quería … y porque nos iba a dar un mal si al final los adoptaban, … (Pam es de mi propiedad, me hace el mismo caso que el resto: ninguno).

Mis amigos de la infancia no van a leer esto, si lo hacen será para burlarse y que escriba o no una Tesis se la pela, y hacen bien, …, están para las cosas realmente importantes, como ya nos conocemos … pues ya está. Al resto: os quiero mucho, espero que os deis por aludidos, si no: preguntad.

Puc dir de forma completament objectiva, sense cap tipus de biaix, que tinc la millor familia del món i les nebodes més sabudes i reboniques,…, menció especial a la meua germana xicoteta i al pare Vicent, "Vicente Trincheras", "el trinchis", "el tigre del Maestrazgo", que entre tanta xapa de trinxeres, guerra civil, metereologia, elaboració del vi, …, em va inculcar l'afició als còmics, als esports, a Conxita Piquer, …, un pare friki!, això sí que mola!. De la mare no dic res, que m'entra la plorera …

Presocráticos y demás citas … al principio de cada apartado.

Índice

ABREVIATURAS **1**

INTRODUCCIÓN **3**

Evolución de la microscopía digital avanzada **5**

***Software* de captura, procesado y análisis de imagen en microscopía** **8**

Software propietario 8

Software libre y de código abierto 10

Desarrollo de herramientas basadas en *software* **17**

Generalidades 17

Metodología: modelos de desarrollo de *software* 19

1-Modelos pesados 19

1.1 Modelo en cascada 19

1.2 Modelo iterativo 20

2- Modelos ágiles 21

Recursos instrumentales 21

1-Programación 21

1.1 Lenguajes de programación de propósito general 22

1.2 Lenguajes de programación de propósito específico 22

1.3 Lenguajes de guion 22

1.4 Librerías y juegos de herramientas 23

2-Entornos de desarrollo específicos de programa 23

2.1 Grabado de comandos 23

2.2 Herramientas de ejecución en serie 23

2.3 Desarrollo visual de flujos de trabajo 24

Nuevas soluciones informáticas **25**

Contexto 25

iMSRC 26

ImageJ/FIJI 29

Identificación, análisis de problemas y justificación 31

Procesado de archivos de imágenes de videomicroscopía 31

Selección y visualización de imágenes procedentes de equipos HCS 35

Generación de mosaicos irregulares en serie 38

Conteo y caracterización de células cancerosas circulantes 41

Caracterización de un nuevo biomarcador de células madre cancerosas 42
Captura y visualización de TMAs 44
OBJETIVOS 47
Objetivos generales 49
Objetivos específicos 49
MATERIAL Y MÉTODOS 51
Desarrollo de aplicaciones 53
Modelo de desarrollo 53
Recursos instrumentales 53
Equipos de microscopía 55
DeltaVision RT Restoration Microscopy System 55
Opera LX High Content Screening System 57
Leica TCS SP5 X 58
Leica DMI6000 B 60
Objetivo 1. Desarrollar una aplicación que permita procesar pilas de imágenes procedentes de experimentos de videomicroscopía multicanal y multifocal 61
Objetivo 2. Desarrollar una aplicación que permita visualizar, procesar y seleccionar imágenes de HCS 62
Objetivo 3. Desarrollar una aplicación que permita fusionar mosaicos regulares e irregulares en serie 63
Objetivo 4. Captura inteligente de células cancerosas circulantes 64
Objetivo 5. Captura inteligente de células madre cancerosas 65
Objetivo 6. Captura inteligente de *tissue microarrays* 67
RESULTADOS 69
Objetivo 1. Desarrollar una aplicación que permita procesar pilas de imágenes procedentes de experimentos de videomicroscopía multicanal y multifocal 72
Objetivo 2. Desarrollar una aplicación que permita visualizar, procesar y seleccionar imágenes de HCS 79
Objetivo 3. Desarrollar una aplicación que permita fusionar mosaicos regulares e irregulares en serie 94
Bloque de código para cada experimento 97
Bucle para cada canal 99
Bucle para cada campo 100
Bucle para cada mosaico 101
Función createUncapturedFields 102

Objetivo 4. Captura inteligente de células cancerosas circulantes **104**
Configuración de los perfiles de primera y segunda captura 104
Análisis de imagen 106
Segunda captura 111
Procesado de imágenes 111
Objetivo 5. Captura inteligente de células madre cancerosas **112**
Configuración de los perfiles de primera y segunda captura 112
Análisis de imagen 113
Segunda captura 116
Procesado de imágenes 116
Objetivo 6. Captura inteligente de *tissue microarrays* **118**
Configuración de los perfiles de primera y segunda captura 118
Análisis de imagen (TMA Field Sampler) 119
Segunda captura 123
Procesado de imágenes (TMA Manager) 123
DISCUSIÓN **131**
DeltaVision Stack Manager **135**
Opera Exporter **137**
iMSRC Stitching Module **139**
Captura inteligente de células cancerosas circulantes **142**
Captura inteligente de células madre cancerosas **144**
Captura inteligente de *tissue microarrays* **144**
CONCLUSIONES **147**
BIBLIOGRAFÍA – WEBGRAFÍA **151**
GLOSARIO **165**

ABREVIATURAS

API: interfaz de programación de aplicaciones (*Application Programming Interface*).

BSD: Berkeley Software Distribution.

CCD: dispositivo de carga acoplada (*Charge-Coupled Device*).

CNIO: Fundación Centro Nacional de Investigaciones Oncológicas.

CCD: dispositivo de carga acoplada (*Charge-Coupled Device*).

DIC: contraste interferencial diferencial (*Differential Interference Contrast*).

EMCCD: dispositivo de carga acoplada con multiplicación de electrones (Electron Multiplying *Charge-Coupled Device*).

FIJI: FIJI is just ImageJ.

fps: capturas por segundo (*frames per second*).

FRET: Transferencia resonante de energía de fluorescencia o de Förster (*Förster Resonance Energy Transfer, Fluorescence Resonance Energy Transfer*).

GNU: GNU is not UNIX.

GPL: GNU General Public License.

GUI:interfaz gráfica de usuario (*Graphical User Interface*).

HCS: cribado de alto rendimiento (*Hight Content Screening*).

i. e. : esto es (*id est*).

IDE: entorno de desarrollointegrado (*Integral Development Environment*).

iMSRC: intelligent Matrix Screener Remote Control.

L-GPL: GNU Lesser General Public License.

LUT: paleta de falso color (*lookup table*).

MPL: Mozilla Public License.

mySQL: my Structured Query Language.

ROI: region de interés (*Region Of Interest*).

TMA: *tissue microarrays*.

INTRODUCCIÓN

<<El Ser es, el no-Ser no es.>>

Parménides de Elea.

EVOLUCIÓN DE LA MICROSCOPÍA DIGITAL AVANZADA

La complejidad de los microscopios ópticos ha aumentado considerablemente desde su aparición en el siglo XVII[1]. Los primeros utensilios se basaban en la absorción de la luz al transmitirse a través de una muestra. Actualmente, existen equipos que explotan prácticamente todos los fenómenos ópticos mediante multitud de técnicas microscópicas, como el contraste de fases, el contraste interferencial diferencial, la microscopía de fluorescencia convencional o confocal, FLIM, TIRF, FRET, STED, PALM, STORM, FRAP* etc., por citar las más conocidas[2, 3]. La historia del desarrollo de cada una de estas tecnologías es única y puede ser estudiada desde diferentes puntos de vista. En esta introducción revisaremos la evolución del registro, soporte, análisis y procesado de imágenes en microscopía, desde la perspectiva de las ciencias de la vida y tomando la microscopía óptica como ejemplo.

Los primeros microscopios eran utensilios sencillos en comparación a los equipamientos modernos. Estaban constituidos por componentes exclusivamente mecánicos, utilizaban luz ambiental, una única lente[4] y alcanzaban una resolución de hasta 1.35 micras[5]. El principal método de registro de las observaciones fue en forma de dibujos en papel hechos a mano hasta bien entrado el siglo XIX, lo cual permitía realizar estudios cualitativos[6].

Con la incorporación de la cámara fotográfica analógica a los microscopios[7], el soporte de las imágenes pasó del papel al film fotográfico, que era más duradero, ocupaba menos espacio y permitía obtener copias (aunque era altamente inflamable y se deterioraba con el tiempo y el duplicado). El registro de imágenes se volvió más rápido, reflejaba mejor los aspectos cualitativos y aportaba alguna información cuantitativa. Esto posibilitó el nacimiento del análisis de imagen en los años 60[8], el cual implementó los principios de la morfometría (utilizados en el campo de estudio de los materiales) a las ciencias biológicas[9]. En los primeros estudios morfométricos, las imágenes se capturaban manualmente una a una, se les superponía una serie de patrones, se analizaba la interacción de los mismos con los objetos de interés y se obtenía información de estos a partir de relaciones matemáticas[8–10]. Siguiendo este procedimiento se extraían datos sencillos, como la fracción de citoplasma ocupada por un orgánulo, pero también otros más complejos, como la superficie de intercambio gaseoso en los alveolos pulmonares[9] o el volumen ocupado por hueso trabecular y

* De sus siglas y acrónimos en inglés FLIM: *Fluorescence-lifetime imaging microscopy*, TIRF: *Total internal reflection*, FRET: *Fluorescence resonance energy transfer*, STED: *Stimulated emission depletion*, PALM: *Photo-activated localization microscopy*, STORM: *Stochastic optical reconstruction microscopy*, FRAP: *Fluorescence recovery after photobleaching*.

tejido adiposo en médula ósea[10]. En todo caso, las medidas obtenidas eran estimaciones de la magnitud real, el conteo se anotaba a mano o con la ayuda de contadores mecánicos y la preparación de las imágenes para su publicación y su procesado (creación de paneles y modificación de brillo y contraste) requería manipular reactivos químicos en cámaras oscuras[11].

Durante los años 80 (ligeramente antes en el laboratorio clínico), el abaratamiento de costes[2,12] populariza una serie de tecnologías (desarrolladas por la industria privada[13] y aeroespacial[14]) que revolucionarán el campo de la microscopía: los ordenadores, escáneres digitales, las cámaras de video de tubo electrónico, las cámaras de estado sólido y las técnicas de procesado y análisis de imágenes por ordenador. Como consecuencia:

1- Se produce la migración del soporte fotográfico al archivo informático[15], el cual permite visualizar, almacenar, recuperar, compartir, procesar y preparar las imágenes para su publicación mucho más rápidamente, de forma más segura y en mucho menos espacio físico[16,17].
2- Las imágenes se capturan y visualizan inmediatamente a través de pantallas de televisión.
3- Los defectos de iluminación de las imágenes se corrigen *in situ* [2,16].
4- Se pueden medir más magnitudes de forma rápida, precisa, reproducible[18,19] , absoluta y sin necesidad de realizar estimaciones.

La incorporación de estos y otros avances tecnológicos a los microscopios (como los sistemas de almacenamiento de datos, mejores fuentes de luz y la motorización de platinas, obturadores, revólveres y ruedas de filtros) posibilitaron la aparición de equipos integrados[2], los cuales permitían el control eficiente de los equipos a través de ordenadores personales. De esta forma, los procesos de captura, almacenado, visualización, procesado, análisis y preparación de imágenes para su publicación, pasaron a estar controlados por *software*, el cual ha acabado ocupando un papel central en la arquitectura de los microscopios[15].

Finalmente, el desarrollo de internet y la diversificación de los soportes de escritura digitales: cintas magnéticas, CD, DVD, Blu-Ray, discos duros, USB, redes de servidores con discos duros en RAID*, etc. facilitaron el intercambio y almacenamiento seguro de archivos[15,17].

El panorama de la microscopía actual se caracteriza por dos particularidades: la gran diversidad de equipos y técnicas, capaces de producir archivos que contienen imágenes de origen muy diverso[20,21], y el elevado grado de automatización que han adquirido los equipos, capaces de obtener una gran cantidad de imágenes en muy poco tiempo[6,20,21]. Esto supone un gran reto a la hora de visualizar, procesar, analizar y mantener las imágenes generadas[20–24], lo cual ha originado un gran interés en el

* De sus siglas en inglés: *redundant array of independent disks.*

desarrollo de herramientas informáticas[21,25–28], provocando que el centro de atención se esté desplazando del *hardware* al *software* y convirtiendo a la microscopía en un campo cada vez más multidisciplinar, en el que bioinformáticos[21,24], expertos en visión artificial y desarrolladores de *software*[22] se suman a físicos, médicos, biólogos, técnicos, etc.

La principal consecuencia es que hoy en día contamos con herramientas capaces de superar el mayor de los inconvenientes que tradicionalmente se le ha achacado a la microscopía clásica: la limitación de los estudios al análisis de un número relativamente bajo de objetos, seleccionados, a juicio del experimentador, como imágenes representativas de los mismos. Los equipos modernos permiten la captura de números estadísticamente significativos de muestras que pueden ser procesadas, analizadas y visualizadas de forma eficiente gracias al diseño de flujos de trabajo mediados por aplicaciones específicas[6,19,21,23]. El desafío actual consiste en conocer, saber utilizar y valorar, en cada circunstancia, qué herramienta informática es la más adecuada para cada procedimiento[15].

SOFTWARE DE CAPTURA, PROCESADO Y ANÁLISIS DE IMAGEN EN MICROSCOPÍA

Conviene tener presente que el *software* tiene un propietario (el autor intelectual o aquel a quién le cedió o vendió los derechos), un usuario (la persona que lo utiliza), unos desarrolladores (aquellos que escriben el código) y una licencia (documento legal que regula los derechos del propietario de un producto sobre el mismo, determinando sus condiciones de distribución y comercialización).

Los programas informáticos que permiten el desarrollo de flujos de trabajo en microscopía (captura, visualización, procesado, análisis de imágenes y tratamiento de datos), pueden ser de dos tipos atendiendo a su licencia: 1- *software* propietario, o 2- *software* libre o de código abierto. A continuación, expondremos las principales características de cada uno de ellos.

SOFTWARE PROPIETARIO

El modelo de desarrollo y comercialización clásico de *software* propietario en microscopía está basado en la colaboración entre universidades y centros de investigación (que crean prototipos y son responsables de la invención de algoritmos) y las empresas comerciales (que los desarrollan y comercializan)[20,23,29]. Para que este modelo sea sostenible, las compañías desarrolladoras deben poder recuperar la inversión económica que les supuso producir las aplicaciones. Existen tres formas (no excluyentes) de hacerlo: 1- liderar el mercado, 2- ampararse en la ley de propiedad intelectual (ejerciendo derechos de *copyright* sobre el código y patentando algoritmos y métodos) y 3- mantener el secretismo industrial (aplicando cláusulas de confidencialidad en los contratos con los trabajadores y distribuyendo sus productos en código máquina)[30,31].

En otras palabras, el *software* propietario lo producen desarrolladores contratados por una empresa, con la que adquieren el compromiso de no divulgar su contenido y a la que ceden los derechos de propiedad sobre el mismo. Esta, a su vez, cede a sus clientes el derecho a utilizarlo, a cambio de un beneficio económico, bajo unas condiciones definidas en una licencia[32] tipo *copyright*[30] (Figura 1: A). Dichas licencias limitan su distribución, comprometiendo al usuario a ejecutarlo en un número limitado de equipos informáticos y a no modificarlo, venderlo ni distribuirlo. A cambio, este recibe una copia de una aplicación en un formato que no le permite visualizar el código de forma comprensible y que tiene medios que le impiden duplicarla e instalarla ilimitadamente. El contenido de dicha copia es desconocido, puede contener código que implementa algoritmos públicos, sometidos a derechos de autor, patentados o secretos. Esto la convierte en una "caja negra", porque el usuario desconoce el proceso

de transformación de los datos en resultados, lo cual dificulta la reproducibilidad de los experimentos científicos [22,33].

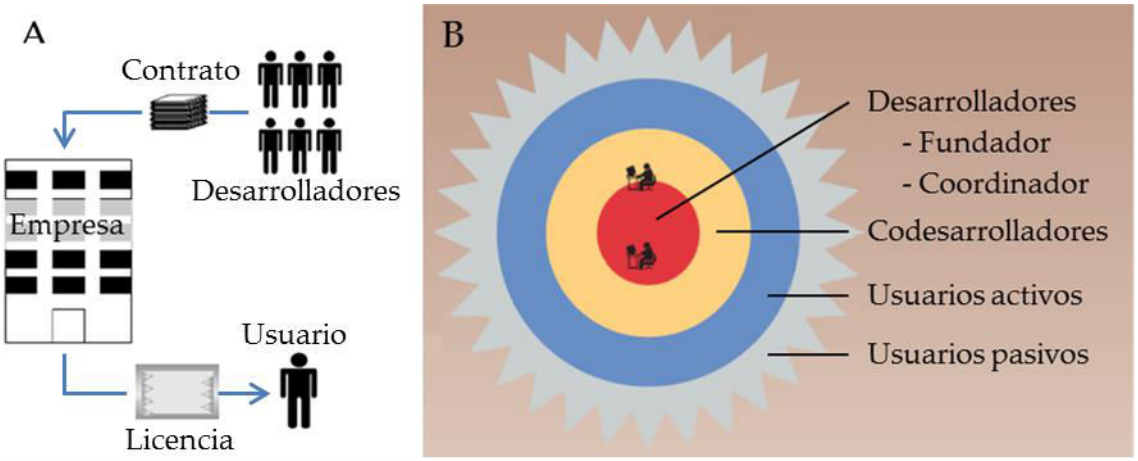


Figura 1. Modelos de desarrollo clásicos del *software* propietario (A) y libre (B). **A**: el *software* propietario es propiedad de una empresa a la que los desarrolladores ceden sus derechos a través de un contrato, esta cede los derechos de uso a los usuarios a través de una licencia que limita su divulgación. **B**: el *software* libre genera una estructura de usuarios activos alrededor de sus desarrolladores, su licencia garantiza su divulgación. Modificado de Lindman et al. (2010), y Crowston y Howison (2006), respectivamente.

La principal fuente de beneficio económico de las empresas desarrolladoras de programas informáticos fue la cesión del uso de los mismos mediante su venta, hasta la aparición del *software* libre y la popularización de internet en los 90. Actualmente, se imponen nuevos modelos de mercado que se centran en la prestación de servicios (soporte técnico, parches y actualizaciones continuas) a cambio de pagos de cuotas periódicas, pago por uso, visualización de publicidad, obtención de información[34,35] o la posibilidad de comprar una versión del producto con capacidades extendidas[36] .

Existe un amplio abanico de *software* propietario de captura, visualización, análisis, procesado de imágenes y tratamiento de datos, más o menos especializado en una fase del flujo de trabajo o en una técnica concreta. En general, este tipo de aplicaciones son fáciles de utilizar y cumplen un estándar de desarrollo y un soporte mínimos. En contraposición, suelen ser soluciones caras, que no son capaces de adaptarse a nuevas demandas experimentales (o a demandas experimentales sin interés comercial) o lo hacen muy lentamente[20,23,29,33,37,38].

En el campo de la microscopía, las principales compañías que venden equipos integrados (Leica, Zeiss, Nikon y Olympus) cuentan con *software* propietario capaz de aunar la captura, visualización, procesado y análisis de imagen en sus equipos*. Este presenta las ventajas de optimizar la interacción con el *hardware* y ser de fácil instalación. Además, produce las aplicaciones de uso más intuitivo, al incluir rutinas

* LAS (Leica Microsystems, Wetzlar, Alemania), ZEN (Carl Zeiss AG, Oberkochen, Alemania), NIS-Elements (Nikon Corporation, Tokio, Japón) y cellSens (Olympus Corporation, Tokio, Japón).

que hacen posible que personal no especializado pueda realizar flujos de trabajo completos. Por otra parte, este tipo de productos son, con diferencia, los menos versátiles, al no permitir la personalización de rutinas de trabajo o el control de componentes no relacionados con su casa comercial[15,23].

También existen compañías que distribuyen soluciones similares capaces de integrarse en equipos personalizados[23] (*i. e.* MetaMorph, Visilog*, etc.). Estos productos permiten diseñar y personalizar flujos de trabajo, controlar todos los microscopios de un mismo laboratorio bajo una única interfaz y comparar los datos obtenidos en equipos distintos de forma sencilla[15]. Sin embargo, pueden requerir conocimientos informáticos y/o de diseño de controladores para su instalación y uso.

Otra alternativa consiste en utilizar *software* distinto para la captura, visualización, análisis, procesado de imágenes y/o tratamiento de datos (i.e.: Definiens, Imaris, Huygens, Volocity, Pipeline Pilot**, etc.). Suelen ser herramientas especializadas que realizan tareas que los programas de equipos integrados no son capaces de abordar o lo hacen más lentamente. Además, permiten aumentar la disponibilidad del equipo de adquisición, si se instalan en un ordenador dedicado al análisis. En contraposición, suele disminuir la fluidez del proceso global, pueden requerir un desembolso económico mayor y obligan al usuario a familiarizarse con el manejo de varios tipos de aplicaciones.

Finalmente, por regla general (dependiendo de legislaciones regionales y estatales), el *software* propietario es el único que se puede utilizar cuando una persona puede sufrir un daño físico derivado directamente de su uso. En el campo de la microscopía, esto sucede en aplicaciones de análisis y visualización de imágenes médicas, cuando se utilizan para diagnóstico primario. En este caso, estas herramientas informáticas deben obtener una certificación que exige que haya una entidad legal, identificada como dueña del producto y vinculada contractualmente con el usuario, que garantice unos estándares de desarrollo y documentación[35] y notifique de la existencia de errores en el mismo. Esta certificación no garantiza que el programa funcione correctamente ni eficientemente, de lo que ninguna compañía se responsabiliza legalmente[37].

SOFTWARE LIBRE Y DE CÓDIGO ABIERTO

Software libre (del inglés, *free software*) y *software* de código abierto (del inglés, *open-source software*) tienen un origen común y polémico. Aunque significan casi la misma cosa, sus definiciones tienen cierto carácter ideológico y están influenciadas por

* Molecular Devices, CA, USA y FEI, Hillsboro, OR, USA, respectivamente.

** Definiens AG, Munich, Alemania; Bitplane AG, Zurich, Suiza; Scientific Volume Imaging, Hilversum, Holanda; Perkin Elmer , Waltham, MA, USA y Biovia, San Diego, CA, USA, respectivamente.

intereses económicos. Esto ha causado un poco de confusión en la comunidad científica, que tiende a usar estos términos arbitrariamente o de forma conjunta ("free open software" o "free/libre open source software"). En esta tesis haremos una breve revisión del campo y aportaremos una definición propia, que puede encontrarse al final de este apartado. A excepción del siguiente párrafo, en el resto de este trabajo usaremos ambos términos indistintamente (salvo que se especifique lo contrario), refiriéndonos a nuestra definición.

El término *software libre* es anterior al de *software de código abierto*. Fue acuñado por Richard Stallman a principio de los 80, para promover las libertades esenciales del usuario de *software*: la libertad de utilizarlo, ejecutarlo, estudiarlo y modificarlo, y de distribuir copias con o sin modificaciones[39]. El término *software de código abierto* se originó en 1989 como consecuencia de una escisión del movimiento de Stallman, que buscaba hacerlo más atractivo al mundo empresarial. Ambos movimientos han dado lugar a sendas fundaciones (Free Software Foundation y Open Source Initiative, respectivamente), en cuyas páginas web[40,41] los lectores interesados pueden encontrar una definición extensa de ambos conceptos, junto con críticas cruzadas.

El modelo de desarrollo clásico de este tipo de *software* empieza cuando un programador desarrolla una aplicación y comparte su código fuente. Si la aplicación despierta interés, este prototipo atrae a una comunidad de usuarios, algunos de los cuales participan en su desarrollo. Con el tiempo, se crea una estructura en capas en el centro de la cual está el fundador original (que dirige el proyecto), un núcleo de desarrolladores (que evalúan, validan, escriben código y lo incorporan al programa), una comunidad de usuarios activa (que propone modificaciones, detecta errores, aporta soluciones, escribe documentación y mantiene foros y *wiki*s) y una comunidad de usuarios pasiva (que utiliza la herramienta informática, beneficiándose de la estructura anterior)[42,43] (Figura 1: B). Finalmente, puede desarrollarse una fundación o brazo comercial alrededor del proyecto que asegure su supervivencia a largo plazo[29]. Actualmente, algunas empresas han incorporado el modelo de desarrollo de *software* libre a sus productos[36], por lo que el modelo clásico se ha vuelto un poco más complejo. Hoy en día, el *software* libre puede ser propiedad intelectual de una empresa (*i. e.* KNIME: KNIME AG, Zurich, Switzerland) o de una comunidad de personas al frente de la cual puede haber una persona (*i. e.* Linux: Linus Torvalds), un gobierno *(i. e.* ImageJ[44]: USA), una fundación (*i. e.* GNU: Free Software Foundation), una institución científica (*i. e.* FIJI[45]: Max Planck Institute of Molecular Cell Biology and Genetics)* o varios de ellos[22,29,36]. Además, la comunidad de desarrolladores puede ser altruista o tener una relación contractual con la empresa que produce la aplicación u otra empresa o fundación interesada en desarrollarla[29,46–48].

Sea como fuere, para que prospere uno de estos proyectos, tienen que darse dos principios básicos:

* KNIME: Konstanz Information Miner, GNU:GNU is not UNIX, FIJI: FIJI is Just ImageJ.

1- El programa debe ser fácil de conseguir (generalmente, a través de internet)[35].
2- El código debe poder ser utilizado, visualizado, modificado y, si se quiere, redistribuido.

Nótese que la libertad de distribución implica que no existe una limitación al número de equipos informáticos en los que se puede instalar este tipo de aplicaciones; por otra parte, aunque no es un requisito, el *software* libre suele ser gratuito. En cuanto a la libertad de modificación, en la mayoría de casos, el núcleo de desarrolladores controla las modificaciones que se implementan en su distribución, pero permite que existan otras. La modificación del código es muy variada, puede ser más o menos extensa, individual o colectiva, etc., pero es frecuente que la comunidad de usuarios se ponga en contacto con la de desarrolladores a través de listas de correo electrónico, *wiki*s o foros y sugieran cambios que pueden o no ser tenidos en cuenta. Suele ser un proceso público, consensuado, revisado, reconocido, auditado por uno o más desarrolladores a cargo del proyecto y registrado en un informe de actualizaciones[37,42,49].

La forma de garantizar la continuidad del modelo de desarrollo de *software* libre es:

1- Colgando los programas en repositorios públicos y páginas web*.
2- Facilitando el acceso a los ejecutables (código máquina) y al código fuente.
3- Mediante licencias tipo *copyleft*[30]. Estas garantizan los derechos de uso, visualización y modificación, y especifican las condiciones de la distribución. Hay una gran variedad de ellas[53,54], pero pueden clasificarse en[32,35,52]:
    a. Permisivas (o poco restrictivas): aquellas que permiten la explotación del *software* y sus derivados como *software* propietario (*i. e.* BSD**).
    b. Débiles (o restrictivas): aquellas que permiten que el *software* y sus derivados se puedan asociar a productos comerciales (*i. e.* L-GPL**).
    c. Fuertes (muy restrictivas): aquellas que exigen que el *software* y sus derivados hereden la misma licencia con la que se crearon (*i. e.* GPL**).
    d. Múltiples, pueden darse dos situaciones:

* La fuente más popular de código abierto es sourceforge.org[50,51], aunque muchos proyectos se alojan en sus propias páginas o las de sus creadores[52]
** BSD: Berkeley Software Distribution, L-GPL: GNU Lesser General Public License, GPL: GNU General Public License

i. Que un mismo *software* pueda tener dos licencias distintas sin que coexistan en la misma copia. Requiere que una única persona o compañía sea beneficiaria de todos los derechos de autor. Permite que un mismo *software* se distribuya como libre y propietario (*i. e.* MySQL[55], Osirix[56])[36,37,55,57].

ii. Que una única copia de un *software* tenga varias licencias simultáneamente. La licencia MPL*, por ejemplo, permite que, cuando se produzcan contribuciones significativas (en forma de archivos independientes) a un proyecto, la obra resultante herede dos licencias, la MPL para los archivos originales y otra distinta para los nuevos (ya sea permisiva, débil, fuerte o comercial). Permite la comercialización de módulos desarrollados para funcionar en *software* propietario. Suelen requerir la inclusión de excepciones cuando los archivos introducidos se basan en licencias tipo GPL[30].

e. Con excepciones: se basan en la modificación de las licencias anteriores, la Oracle's Free and Open Source Software License Exception[55], por ejemplo, es un licencia GPL modificada que evita que la inclusión de un *software* con esta licencia en un *software* con una licencia distinta lo transforme en GPL.

Es posible, aunque no habitual, que existan aplicaciones, accesibles en código máquina y fuente, que se distribuyan sin licencia (i.e: ImageJ). Estrictamente hablando, este *software* no es propietario ni libre ni de código abierto (aunque comparte muchas características con los dos últimos), sino *software* de dominio público.

En el campo de la microscopía, existe *software* libre para cada uno de los pasos del flujo de trabajo, desde la captura al tratamiento estadístico de los resultados (Tabla 1, página 16). Su amplia difusión en el entorno científico[23,29,33,58] y sanitario[35,37,52,59] obedece a varios motivos:

1- Comparten principios y metodología: reproducibilidad, evaluación por parte de una comunidad de expertos y difusión del conocimiento[22,33,37,42,49].

2- Bajo coste[35,37,38,59,60], aunque no necesariamente gratuito[33,48], rapidez de adquisición y ausencia de trabas burocráticas[50,61].

3- Gran flexibilidad: al distribuirse el código fuente, no hay límites a la modificación de las herramientas informáticas. Muchos desarrolladores generan soluciones, en forma de pequeñas piezas de código, que se

* MPL: Mozilla Public License.

integran en un programa mayor complementando su funcionalidad o adaptándola a las necesidades exactas de cada usuario[61]. Esto permite dar con una respuesta rápida a una necesidad que las empresas privadas no atienden debido a su escaso valor comercial[20,23,29,33,37,38,59].

4- Promueve la estandarización e interoperabilidad[29,33,37,50,60,62–64] (desarrollo de formatos de datos, imágenes, etc., que permiten la comunicación entre aplicaciones distintas), lo cual supone tres ventajas: 1- favorece la comunicación entre laboratorios que usan equipamiento de diferentes empresas, 2- abarata la adquisición de equipamiento (ya que el comprador no tiene que ceñirse a una única casa comercial)[35,59,61,65] y 3- protege de la discontinuidad del soporte[33,60].

Los principales inconvenientes que se le achacan al *software* libre son:

1- Escasa documentación y falta de soporte[29,35,59].
2- Baja eficiencia[66].
3- Problemas de responsabilidad legal[35,37,38,59,61].

Varios autores[20,35,50] recomiendan basar la selección de *software* libre en una serie de indicadores:

1- Calidad de la página web donde se aloja.
2- Calidad y cantidad de documentación.
3- Uso: número de veces que se ha descargado, frecuencia de actualización, número de usuarios registrados etc.
4- Facilidad de instalación.
5- Actividad de la comunidad: foros, *wikis*, listas de correos, congresos y conferencias.
6- Arquitectura modular que facilite la interoperabilidad.
7- Existencia de soporte profesional.

El modelo de comercialización se centra en la prestación de servicios. Una empresa puede hacer negocio con *software* libre:

1- Distribuyéndolo[30,51,67].
2- Adaptándolo a las necesidades exactas de un usuario[30,36,68].
3- Generando y comercializando soluciones basadas en él[36,69,70] .
4- Comercializando equipos integrados que lo incluyan[37,65].
5- Convirtiendo *software* de dominio público, con licencias permisivas o con excepciones en *software* propietario y explotándolo[36,57].
6- Desarrollando y vendiendo *software* comercial que se ejecute sobre *software* de dominio público, con licencias permisivas o con excepciones[30].

7- Ofreciendo servicios de soporte, mantenimiento, documentación y aprendizaje[30,36,71–73].

8- Desarrollando *software* libre bajo una licencia dual (libre y comercial): por ejemplo, adaptando *software* libre para que cumpla la certificación legal para su uso en un campo determinado (como el diagnóstico clínico) y explotándolo[37] (*i. e.* Osirix), o manteniendo una versión simplificada de una aplicación como *software* libre y comercializando otra con mayor funcionalidad (*i. e.* MySQL). Esta estrategia permite generar un producto superior de forma más rápida y barata que la estrategia de desarrollo comercial clásica, facilitando además la venta, el marketing, el desarrollo y el soporte del producto[36].

Además, las empresas tecnológicas se pueden posicionar en una situación de ventaja comercial si sus equipamientos se adhieren a los estándares del *software* libre y contribuyen en estos proyectos[26,29–31,50,74] (*i. e.* ampliando su mercado diseñando drivers para que sus dispositivos de *hardware* puedan funcionar en equipos basados en *software* libre).

Finalmente, el movimiento *software* libre ha derivado en varios movimientos con impacto en el panorama científico actual, como los movimientos *acceso abierto* y *ciencia abierta* (del inglés open access y open science, respectivamente). El primero promueve el acceso gratuito a los artículos científicos[75] y el segundo a los datos experimentales, protocolos y programas de procesado[49,63,75,76]. También se habla de la implementación de la metodología de diseño de código en el *software* libre en el proceso de revisión de los artículos científicos[49].

Resumidamente: el *software* libre *maduro* en microscopía lo producen desarrolladores profesionales contratados por empresas, gobiernos y fundaciones, y lo evalúa la comunidad de usuarios. Estos reciben una copia del mismo en dos formatos: uno que pueden ejecutar sin restricciones en tantos equipos como deseen (código binario ejecutable) y otro (código fuente) que pueden visualizar y modificar libremente. A cambio, si deciden distribuirlo, deben hacerlo bajo unas condiciones definidas en una licencia. A los efectos de esta tesis, entenderemos como *software* libre o *software* de código abierto a un programa, aplicación, módulo (*plug-in*, en inglés), macro o *script* informático, que aporta una solución a un problema y su código es accesible (puede ser leído, modificado y ejecutado), fácil de conseguir y no se pagan costes derivados de la licencia. La calidad del mismo puede ser variable y evaluable en función de los indicadores citados arriba.

Tabla 1. Listado del *software* libre más popular de captura, visualización, procesado, análisis y tratamiento de datos de imágenes biomédicas.

SOFTWARE	**DESCRIPCIÓN (sitio web)**	**TIPO DE LICENCIA***
ImageJ[44]	Procesado y análisis de imagen (ImageJ.nih.gov/ij/)	Dominio público
FIJI[45]	Procesado y análisis de imagen (fiji.sc)	GPL
ICY[77]	Procesado y análisis de imagen (icy.bioimageanalysis.org)	GPL
Osirix[56]	Procesado, análisis y visualización de imágenes médicas (osirix-viewer.com)	Dual (L-GPL y comercial)
CellProfiler[78]	Análisis de grandes volúmenes de imágenes (cellprofiler.org)	BSD
CellProfiler Analyst[79,80]	Análisis, representación y exploración de grandes volúmenes de datos. *Machine learning* (cellprofiler.org)	BSD
Ilastik[81]	Segmentación, análisis y *machine learning* (ilastik.org)	GPL con una excepción
KNIME[82]	Análisis, representación y exploración de grandes volúmenes de datos. (knime.org)	Dual (GPL con una excepción y comercial para extensiones)
R[83]	Análisis estadístico y creación de gráficos (r-project.org)	GPL
OME Bio-Formats[76]	Librería para la lectura y escritura de archivos de imágenes de biología (openmicroscopy.org)	Múltiple (GPL, BSD y comercial)
OMERO[84]	Repositorio web de imágenes de microscopía (openmicroscopy.org)	Dual (GPL y comercial)
OME TIFF Format[85]	Formato de imagen universal (openmicroscopy.org)	BSD
ITK[86]	Librería de algoritmos para el registro y segmentación de imágenes (itk.org)	Dependiendo de la versión: Apache, BSD o BSD modificada
VTK[87]	Librería de algoritmos para la visualización, análisis y procesado en 3D (vtk.org)	BSD
Micromanager[25,26]	Control de microscopios automatizados (micro-manager.org)	BSD y L-GPL

Modificado de: Carpenter et al. (2012), Hamilton (2012), Swedlow y Eliceiri (2009), Nagy (2007), y Schneider et al. (2012). * Ver página 12 y siguientes.

DESARROLLO DE HERRAMIENTAS BASADAS EN *SOFTWARE*

GENERALIDADES

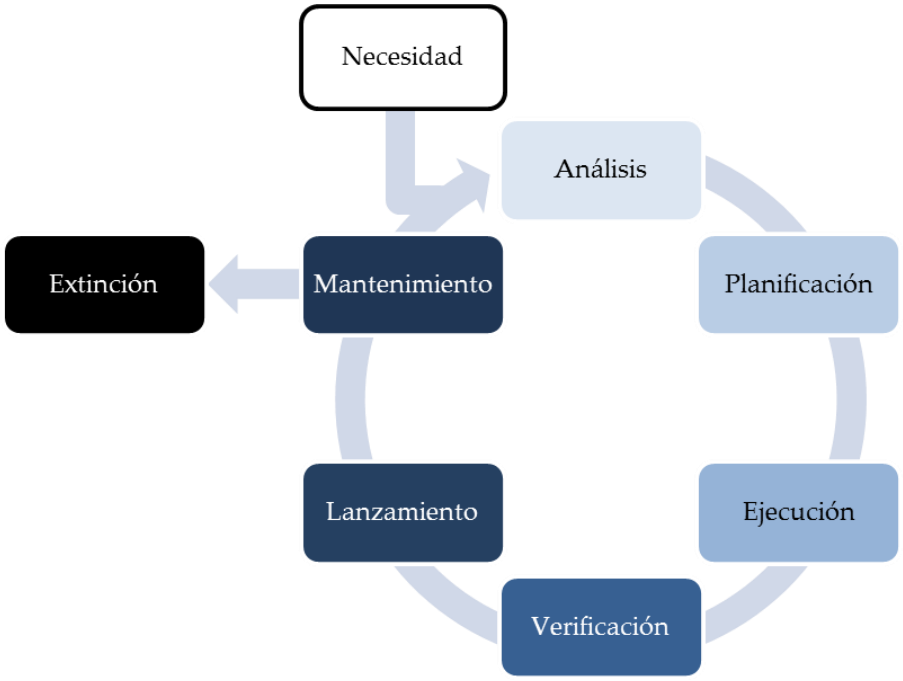


Figura 2. Ciclo de vida del *software* informático. Antes de escribir el código, debe analizarse el problema que se quiere solucionar y planificar una estrategia. El código escrito debe verificarse antes del lanzamiento del producto. Si el producto es mantenido, se generará documentación y soporte, y nuevas necesidades se incorporarán al mismo, repitiéndose el ciclo.

El desarrollo de una aplicación tiene como objetivo dar respuesta a una necesidad. Dicha necesidad debe ser analizada, debe planificarse un proyecto, llevarlo a cabo, depurarlo, lanzarlo, mantenerlo y, en último término, extinguirlo (Figura 2). Este proceso puede adquirir niveles de complejidad muy variable, desde un programador escribiendo unas pocas líneas de código para automatizar una tarea sencilla de uso propio, hasta equipos multidisciplinares de centenares de personas desarrollando un programa de millones de líneas capaz de solucionar tareas complejas y lanzar al éxito (o a la bancarrota) a la compañía que lo produce[88,89]. En cualquier caso, todos los proyectos de desarrollo de *software* tienen una serie de elementos comunes:

1- Un problema que se quiere solucionar.
2- Un contexto, en el cuál ocurre el problema.

3- Un ordenador, capaz de recibir datos (*input*), procesarlos siguiendo unas instrucciones (programa, aplicación, módulo, *script*, macro, etc.) y generar datos (*output*).
4- Una solución teórica (algoritmo) que se implementa a través de una serie de instrucciones a un ordenador.
5- Recursos instrumentales que permiten escribir instrucciones (lenguajes y herramientas de programación).
6- Recursos económicos (tiempo y dinero).
7- Recursos metodológicos (modelos que permiten planificar el proceso de desarrollo y la interacción de todas las figuras que participan en él).
8- Un programador, que usa los recursos disponibles para escribir instrucciones.
9- Errores de programación.
10- Un usuario.
11- Un ámbito de aplicación.

Niveles de complejidad mayores suelen implicar la aparición de otros elementos, como infraestructuras complejas (redes de ordenadores, equipos de IT, etc.) y equipos de personal especializado (desarrolladores de *software*, ingenieros informáticos, responsables, coordinadores, empresarios, accionistas, gestores, abogados, analistas de mercado, etc.).

Conviene diferenciar el contexto y el ámbito de aplicación, estos pueden ser coincidentes o no (al igual que el programador y el usuario final de la aplicación), ya que es habitual que un programa se desarrolle en un contexto determinado y se acabe expandiendo a un ámbito distinto con necesidades similares, siendo ejemplos frecuentes el paso del contexto militar al ámbito civil, o el paso de un laboratorio de investigación a otros laboratorios de investigación, al ámbito comercial o al clínico.

La gestión y subsanado de los errores de programación no se debe tomar a la ligera, dado que puede suponer el 50%-60% del coste económico total, el 30% del esfuerzo invertido en el proceso de desarrollo y el 40%-50% del total de recursos invertidos[90].

El usuario final al que va dirigido el programa determinará las características del mismo y la cuota de mercado.

Dos puntos críticos en el proceso de creación de aplicaciones son: 1- su planificación y 2- la selección de los recursos de programación que se va a utilizar, los cuales desarrollaremos en los puntos siguientes.

METODOLOGÍA: MODELOS DE DESARROLLO DE *SOFTWARE*

La planificación del proceso de desarrollo de *software* permite que este se desenvuelva de forma eficiente (mediante el control de los recursos dedicados al mismo) y eficaz (garantizando que el producto final resuelva el problema planteado, cumpliendo los requisitos impuestos por el usuario).

Existen distintos modelos de desarrollo, todos ellos con ventajas e inconvenientes. Ninguno funciona siempre y es habitual usar distintos modelos en distintas fases del proyecto, la clave consiste en adaptarlos a los requisitos específicos de cada proyecto con el fin de optimizar el proceso global[88]. Todos los modelos coinciden en la necesidad de que exista alguna forma de organización, un grado de planificación y documentación mínimo y mecanismos de comunicación entre las partes implicadas. Distintos modelos dan un grado diferente de importancia a estos elementos. Una posible clasificación de los modelos de desarrollo más relevantes es la siguiente:

1-MODELOS PESADOS

Los modelos pesados priorizan la planificación, documentación y organización.

1.1 MODELO EN CASCADA

El modelo en cascada considera cada una de las fases del ciclo de vida del producto como elementos independientes y secuenciales (Figura 3: A). Cada elemento se ejecuta concienzudamente y se documenta de forma extensa antes de pasar al siguiente. Idealmente, ninguna fase del ciclo es revisitada. Se suele utilizar en proyectos largos y cuando se conocen bien los requisitos del usuario (por ejemplo, en la implementación de una tecnología ya conocida)[88,89].

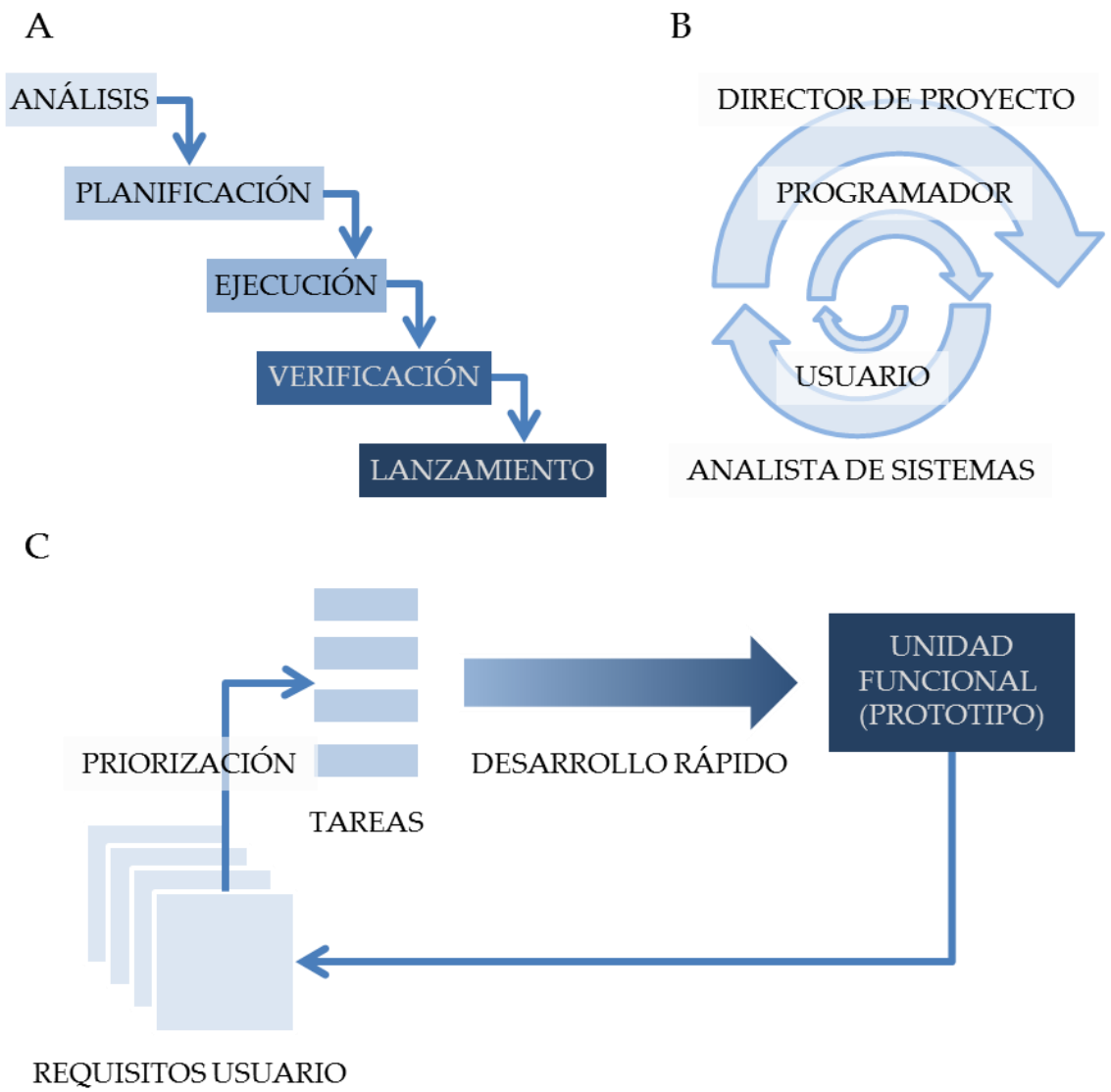


Figura 3. Principales modelos de desarrollo de *software*. **A**: modelo en cascada. **B**: modelo iterativo y **C**: modelo ágil. A y C muestran cada una de las fases del ciclo de vida del producto en un código de colores, B la fase de análisis. A y B modificado de Jonasson (2008), C modificado de Pitt-Francis et al. (2008).

1.2 MODELO ITERATIVO

El modelo iterativo es similar al anterior, pero cada una de las fases del ciclo se analiza de forma completa desde puntos de vista distintos (los del usuario, programador y gestor, por ejemplo), tantas veces como haga falta, hasta dar con una solución satisfactoria (Figura 3: B). Se suele utilizar en proyectos medios o grandes, cuando los requisitos no se conocen en detalle (por ejemplo en investigación y desarrollo de nuevas tecnologías)[88].

2- MODELOS ÁGILES

Los modelos ágiles priorizan el desarrollo rápido y continuado de soluciones, la comunicación directa e informal entre las personas del equipo y la implicación del usuario.

La idea es ordenar los requisitos del usuario según su importancia y desarrollarlos secuencialmente, completando el ciclo de vida del producto de forma tan rápida como sea posible, minimizando trabas burocráticas y lanzando una versión del programa al final de cada ciclo (Figura 3: C). El desarrollo de *software* es incremental, de las primeras versiones (más sencillas) a las más complejas, por lo que el usuario tiene la oportunidad de realizar aportaciones con cada nueva versión. Se suele utilizar en proyectos pequeños o proyectos grandes que pueden dividirse en partes, cuando el equipo de desarrollo es pequeño, el usuario es accesible y proactivo, o cuando los requisitos cambian rápidamente con el tiempo. Comprende un gran número de subtipos (programación extrema, melé, ágil, desarrollo rápido de aplicaciones, etc.) que utilizan varias técnicas, como el lanzamiento de prototipos, reuniones de pie o la programación dual[88,91]. El Manifiesto Ágil[92] resume sus características principales. Algunos autores[93] defienden que este modelo es el que mejor se ajusta al desarrollo de herramientas informáticas en el contexto científico y académico, en el que grupos pequeños de personas con contratos cortos y alta tasa de renovación, necesitan de soluciones rápidas y flexibles (capaces de adaptarse a cambios imposibles de anticipar impuestos por nuevos descubrimientos) que van a ser utilizadas por ellos mismos o una comunidad limitada de usuarios. El proceso de desarrollo de *software* libre, entendido como una metodología de programación, podría clasificarse dentro de esta categoría.

RECURSOS INSTRUMENTALES

Los ordenadores son capaces de resolver una necesidad implementando un algoritmo, la forma de hacerlo es ejecutando instrucciones secuencialmente, las cuales están registradas en el código de los programas. Existen varias formas de escribir código, la mayoría de ellas no son excluyentes, sino complementarias. En el contexto del análisis de imagen, destacan las siguientes posibilidades:

1-PROGRAMACIÓN

Programar consiste en dar instrucciones a un ordenador mediante un lenguaje de programación, que es un vocabulario de símbolos que se agrupan en sentencias[94] siguiendo una normativa o sintaxis.

Se pueden clasificar los lenguajes de programación atendiendo a diferentes criterios (nivel de abstracción, propósito, evolución histórica, manera de ejecutarse, de

abordar la tarea a realizar, paradigma de programación, lugar de ejecución, concurrencia, interactividad, realización visual, determinismo, productividad y modelo de distribución). No existe un lenguaje de programación ideal que se desenvuelva mejor que el resto en todos los contextos[95], por lo que los programadores profesionales suelen dominar varios de ellos para explotar sus ventajas y optimizar sus aplicaciones.

1.1 LENGUAJES DE PROGRAMACIÓN DE PROPÓSITO GENERAL

Los lenguajes de programación de propósito general están diseñados para instruir al ordenador en cualquier tipo de cometido, construyendo aplicaciones desde cero. Se caracterizan por su versatilidad y rendimiento. Se suele referir a ellos como lenguajes de tecleado fuerte, ya que exigen descripciones detalladas de la tarea a realizar por parte del ordenador[96]. Esto implica un mayor control sobre la ejecución (y por tanto mayor robustez, velocidad y gestión de recursos), pero una sintaxis compleja y mayor dificultad de aprendizaje. Tienen un grado de abstracción medio y suelen ser lenguajes compilados pertenecientes a los denominados lenguajes de tercera generación. Los más utilizados son: java, visual basic y C++[97]. Es habitual utilizarlos para desarrollar programas o módulos que extienden las capacidades de un programa[33].

1.2 LENGUAJES DE PROGRAMACIÓN DE PROPÓSITO ESPECÍFICO

Los lenguajes de programación de propósito específico se diseñan para desenvolverse en un ámbito determinado (el análisis de imagen, por ejemplo). Surgen para resolver limitaciones impuestas por los lenguajes de programación de dominio general, por lo que son más eficientes que estos[98,99]. Suelen tener una comunidad de usuarios más limitada, lo que puede dificultar el desarrollo y difusión de herramientas. El lenguaje Diderot, por ejemplo, incluye tipos de datos y operaciones matemáticas (que no aportan los lenguajes de propósito general) que le permiten implementar algoritmos de análisis y visualización de imágenes con un rendimiento muy alto[99].

1.3 LENGUAJES DE GUION

Los lenguajes de guion (del inglés, *scripting languages*) tienen un grado de abstracción mayor a los lenguajes de tercera generación. Son, por tanto, más fáciles de aprender[100] y permiten un desarrollo más rápido de aplicaciones que los anteriores, aunque pueden tener menos funcionalidad. Se suele referir a ellos como programas sin tecleado (del inglés, *typeless*), ya que son poco descriptivos[96]. Utilizan una sintaxis muy sencilla, dando lugar a programas más cortos[95] y menos eficientes que los lenguajes de tecleado fuerte. Suelen ser lenguajes interpretados, lo que facilita el desarrollo de aplicaciones, pero disminuye la velocidad de ejecución. Se les considera la alternativa a la programación de propósito general cuando la rapidez y gestión de recursos no es determinante o para desarrollar prototipos de forma rápida que pueden ser traducidos

luego a lenguajes compilados[96], por lo que se suelen utilizar en modelos de desarrollo ágiles[100]. En ocasiones se integran dentro de programas, en forma de lenguajes de guion propios. Los más conocidos son Perl, Python y Javascript.

1.4 LIBRERÍAS Y JUEGOS DE HERRAMIENTAS

Las librerías son una agrupación de implementaciones de algoritmos para ser utilizados por el lenguaje de programación en que se escribieron u otros lenguajes a los que se han adaptado. Existen multitud de ellas en el contexto del análisis de imagen, como ITK[86], VTK[87], etc. (ver Tabla 1).

Los juegos de herramientas (del inglés, *toolkits*) son colecciones de herramientas de programación con un objetivo común, suelen incluir varias librerías compatibles, documentación sobre su uso, IDEs (de sus siglas en inglés, *Integral Development Environment*, o entorno de desarrollo integrado), etc.[101]. En el campo de la bioinformática, son muy populares los juegos de herramientas basados en *software* libre, conocidos genéricamente como Bio* toolkits (BioPerl, Biojava, BioRuby, etc.), auspiciados por la Open Bioinformatics Foundation[102–104]. Representan un grado de abstracción mayor que el lenguaje de programación que los utiliza y permite el desarrollo rápido de aplicaciones mediante la reutilización de código[104].

2-ENTORNOS DE DESARROLLO ESPECÍFICOS DE PROGRAMA

Los entornos de desarrollo específicos de programa (del inglés, *program suites*), son herramientas que facilitan la automatización de rutinas dentro de algunas aplicaciones. No requieren conocimientos de programación, por lo que son muy fáciles de utilizar[101], pero su funcionalidad y rendimiento está limitada por el entorno. Los más populares son:

2.1 GRABADO DE COMANDOS

Los grabadores de comandos (del inglés, *command recorders*) anotan automáticamente las órdenes en una interfaz en la medida en que el usuario las ejecuta. El resultado es una macro (abreviado de macroinstrucción) o secuencia de comandos que el usuario puede desencadenar automáticamente[33]. Se suelen utilizar para automatizar tareas sencillas y muy repetitivas o para personalizar el programa en que se ejecutan (asignándoles un botón o atajo por teclado). También es habitual integrar macros en *scripts* o herramientas de ejecución en serie.

2.2 HERRAMIENTAS DE EJECUCIÓN EN SERIE

Las herramientas de ejecución en serie (del inglés, *batch processing tools*), permiten ejecutar un comando, secuencia de comandos o macro, sobre una agrupación de archivos o estructura de archivos y guardar el resultado. Son más eficientes que los

anteriores porque suelen omitir la generación de interfaces de usuario y el visualizado de imágenes intermedias entre comandos.

2.3 DESARROLLO VISUAL DE FLUJOS DE TRABAJO

Algunos programas (en inglés: *GUI suites*, *visual workflow builders* o *workflow management software*) permiten la automatización de tareas mediante la implementación de interfaces gráficas de usuario que suelen incluir bloques con menús multiopción, casillas de verificación y campos numéricos rellenables[33,58,101,105]. Los bloques agrupan algoritmos distintos para el mismo problema (puede existir un bloque de segmentación de imágenes, por ejemplo) y los menús multiopción etc., permiten elegir el algoritmo a utilizar y ajustar las variables para su implementación de forma sencilla por parte del usuario. La combinación secuencial de bloques (un bloque de procesado de imágenes, seguido de otro de segmentación, análisis y filtrado de objetos, por ejemplo) permite generar flujos de trabajo que se pueden implementar en serie, de forma automática, a un conjunto de archivos. El uso de interfaces gráficas sencillas permite que algoritmos complejos puedan ser implementados por usuarios sin conocimientos informáticos, pero su funcionalidad está limitada a las herramientas disponibles. Son muy útiles en experimentos de cribado de alto rendimiento (*High Content Screening*, en adelante HCS) para el procesado, análisis, visualización y tratamiento estadístico de grandes volúmenes de imágenes[58]. Algunos ejemplos de programas que los utilizan son Cell Profiler[78], KNIME[58], ICY[77] y Pipeline Pilot .

NUEVAS SOLUCIONES INFORMÁTICAS

CONTEXTO

El contexto de este estudio determina su justificación, su contenido, la metodología con que se ha llevado a cabo y los recursos disponibles.

La parte experimental de esta tesis se realizó en la Unidad de Microscopía Confocal de la Fundación Centro Nacional de Investigaciones Oncológicas del Instituto de Salud Carlos III (en adelante, CNIO).

El CNIO es un centro de excelencia de reconocido prestigio internacional en el que trabajaban, a fecha de 2015, unos 401 investigadores, de los cuales, el 17% eran extranjeros. Su objetivo es la investigación en el cáncer y su aplicación en forma de nuevos y más eficaces métodos de diagnóstico y tratamiento. Su organización es sencilla: el personal de administración y servicios del centro provee a los grupos de investigación de las necesidades básicas y les aligeran de trámites y cargas burocráticas, y las unidades de apoyo a la investigación les proporcionan conocimiento especializado en la instrumentación y de procedimientos para poder llevar a cabo su tarea[106].

La Unidad de Microscopia Confocal es una unidad de apoyo a la investigación que centraliza todos los recursos de microscopía óptica avanzada de fluorescencia, siendo la referencia en el centro a la hora de diseñar y llevar a cabo experimentos que impliquen dicha tecnología. Entra dentro de sus funciones calibrar y mantener el equipamiento del que dispone, enseñar y supervisar el uso del mismo y sus técnicas relacionadas (inmunofluorescencia, hibridación *in situ*, métodos de aclarado, etc.), mantener la ingente cantidad de imágenes obtenidas, implementar, diseñar y dar apoyo en técnicas de análisis y procesado de imagen y, finalmente, testar y desarrollar nuevas tecnologías relacionadas con su campo[28,107]. Esta unidad cuenta con cuatro microscopios de barrido láser confocal, dos microscopios fluorescentes de campo ancho, un sistema de captura HCS (todos ellos automatizados y equipados para trabajar *in vivo*) y tres estaciones de análisis de imagen (los detalles de equipamiento relevante para esta tesis pueden encontrarse en la sección de material y métodos). La dotación de personal estable era de dos técnicos de microscopía, uno de ellos especializado en análisis de imagen, y un jefe de unidad. De los 401 trabajadores del CNIO, 136 estaban suscritos a la lista de distribución de la unidad, lo que significa que utilizaban sus instalaciones. La elevada ratio usuario-equipamiento/personal de servicio, la ingente cantidad de datos que se deriva del uso de las tecnologías mencionadas (esta unidad genera un volumen mensual de datos aproximado de 430 gigabyte), junto con la naturaleza del trabajo científico, hace que las exigencias de eficiencia sean muy altas y en consecuencia, la automatización de procesos es una prioridad.

Esta tesis tiene como objetivo dotar a los técnicos y usuarios de la Unidad de Microscopía Confocal del CNIO de herramientas que les permitan satisfacer necesidades concretas. Todas ellas están relacionadas con la automatización de tareas complejas o engorrosas, el manejo de grandes volúmenes de datos y la superación de limitaciones impuestas por el *software* propietario, y se basan en el desarrollo de rutinas de análisis y procesado de imágenes en *software* libre. Cinco de las siete aplicaciones* desarrolladas están relacionadas con intelligent Matrix Screener Remote Control (iMSRC), una herramienta publicada y desarrollada por nuestro grupo[28].

El usuario medio de la Unidad de Microscopía Confocal no tiene conocimientos en informática, análisis y procesado de imágenes ni en programación, por lo que las aplicaciones diseñadas para su uso son sencillas y presentan interfaces autoexplicativas. Los técnicos de la Unidad poseen conocimientos medios-avanzados en estos campos, por lo que algunas de las aplicaciones desarrolladas para su uso carecen de interfaces gráficas, permitiendo una mayor flexibilidad a través de la interacción directa con el código.

iMSRC

La formación de mosaicos es una técnica de microscopía que permite observar objetos enteros de tamaño mayor al campo de visión de un objetivo. Consiste en capturar una serie de fotografías parcialmente solapantes de un objeto y fusionarlas posteriormente en una única imagen. El resultado de la fusión se denomina mosaico o imagen mosaico, cada una de las imágenes individuales que lo componen se denomina campo. La fusión de los mismos se realiza localizando las mismas estructuras en zonas de solapamiento de campos adyacentes[108,109]. Es posible capturar campos y fusionarlos a mano, pero la realización de mosaicos se suele efectuar en microscopios con platinas motorizadas y programas informáticos de fusión de imágenes[110–112].

La aplicación Matrix Screener (Leica Microsystems, Wetzlar, Alemania) es una extensión del *software* de captura propietario de Leica (LAS AF) que permite automatizar: 1- la captura de imágenes a distancias regulares y 2- la captura y fusión de campos en mosaicos (Figura 4: A y B, respectivamente). Las condiciones de captura de un experimento (el formato de la imagen, la potencia del láser, la localización espacial de los campos, etc.) se pueden guardar en archivos de captura para reutilizarlas.

* iMSRC Stitching Module , Circulating Cancer Cells, Single Cell Well Detector, TMA Field Sampler y TMA Manager

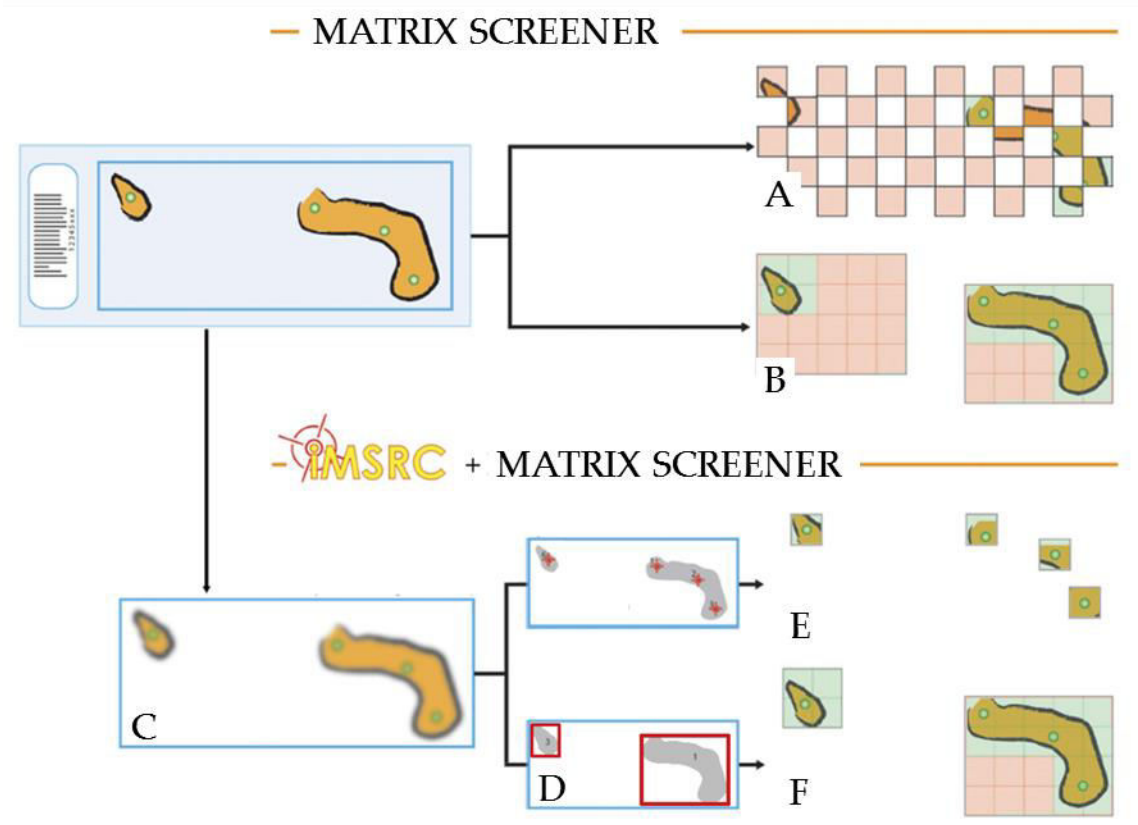


Figura 4. Comparativa de las capacidades de un microscopio equipado con Matrix Screener (A y B) y con Matrix Screener e iMSRC (E y F). Matrix Screener captura automáticamente imágenes a intervalos regulares (A) y mosaicos (B). Cuando se quiere capturar varios mosaicos de forma automática, Matrix Screener ajusta el tamaño del mosaico más grande a todos los mosaicos, la posición de cada mosaico la determina el usuario manualmente (B). iMSRC permite a Matrix Screener capturar objetos (E) y mosaicos (F) de forma automática, siguiendo una rutina en tres pasos: 1- capturar un mosaico que comprenda los objetos de interés (C), 2- ejecutar una rutina de análisis de imagen para obtener sus coordenadas (D) y 3- usar estas para realizar una segunda captura (E y F). iMSRC aumenta la eficiencia de Matrix Screener porque evita capturar imágenes sin interés (cuadros rojos en A, B y F). Cuando se quiere capturar varios mosaicos de forma automática, iMSRC permite determinar su posición automáticamente y ajustar el tamaño de cada mosaico a cada objeto (F). Modificado de Carro et al. (2015).

iMSRC es una aplicación desarrollada en *software* libre por las unidades de Microscopía Confocal y de Bioinformática del CNIO que ha sido publicada[28], siendo el autor de esta tesis uno de los primeros firmantes del trabajo. iMSRC transforma un microscopio equipado con Matrix Screener en un sistema de captura automático guiado por análisis de imagen (Figura 4). Esta tecnología permite detectar y capturar automáticamente los objetos de una muestra. Para hacerlo, se sigue una rutina en tres pasos:

1- Primera captura: adquirir un mosaico (denominado mosaico de búsqueda) que comprenda toda el área donde se encuentran los objetos (Figura 4: C y Figura 5: A).

2- Análisis de imagen: ejecutar una rutina de análisis en el mosaico de búsqueda que permita detectar objetos de interés y extraer sus coordenadas (Figura 4: D

y Figura 5: B). Si el objeto a capturar requiere la formación de un mosaico, se extraen las coordenadas del rectángulo que lo contiene*.

3- Segunda captura: adquirir automáticamente los objetos detectados en 2 (Figura 4: E y F y Figura 5: C).

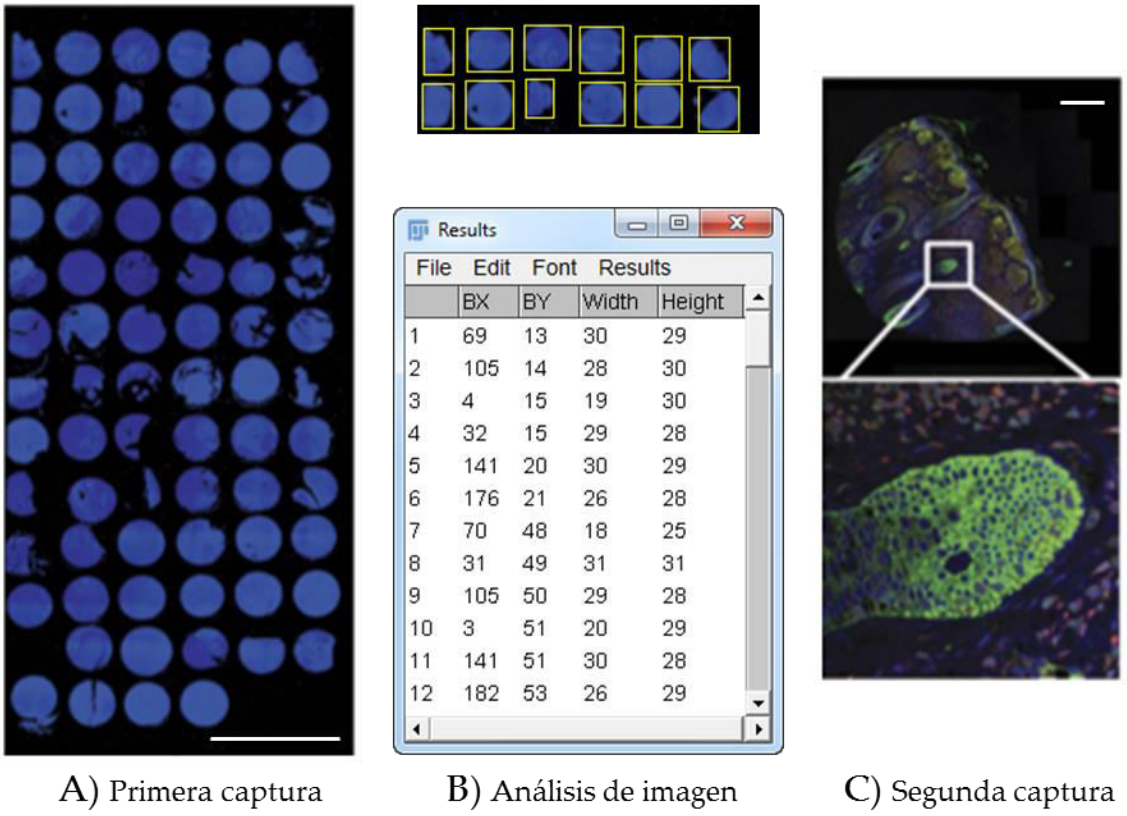


Figura 5. Ejemplo de captura de múltiples tejidos usando iMSRC. Se sigue una rutina en tres pasos: **A**- capturar un mosaico de búsqueda, **B**- extraer las coordenadas de los objetos de interés ejecutando una rutina de análisis de imagen (se muestran los 12 primeros objetos detectados, el rectángulo que los circunscribe y sus coordenadas) y **C**- capturar a gran aumento en las coordenadas detectadas en 2. El resultado de la segunda captura fueron 81 imágenes mosaico (se muestra una a modo de ejemplo, el detalle muestra el tamaño de un campo). iMSRC incrementa la eficiencia de captura evitando al usuario tener que definir manualmente el principio y final de un mosaico para cada una de las 81 piezas de tejido y permitiendo la captura de mosaicos de distinto tamaño. Modificado de Carro et al. (2015). Las barras de escala en 1 y 3 miden 5 mm y 250 micras, respectivamente.

iMSRC genera archivos de captura de Matrix Screener modificando archivos de captura previos. Ambas aplicaciones son independientes:

- Matrix Screener realiza la primera y segunda capturas (pasos 1 y 3, respectivamente), proporciona un archivo de segunda captura preliminar (que

* iMSRC incluye una pequeña rutina de análisis de imagen muy sencilla. Esta permite extraer coordenadas de objetos en un mosaico de búsqueda poco complejo, basándose en tres parámetros (rango de tamaño, brillo mínimo y circularidad). En casos más complicados, iMSRC puede implementar fácilmente rutinas de procesado y segmentación diseñadas en ImageJ/FIJI. Alternativamente, se puede binarizar la imagen en cualquier software, sustituirla por la original y analizarla en iMSRC.

contiene las condiciones de captura, pero no las posiciones a capturar) y fusiona los campos capturados en una imagen mosaico.

- iMSRC obtiene las coordenadas de los objetos a capturar a partir del análisis del mosaico de búsqueda y las añade al archivo de segunda captura preliminar. El resultado es un archivo de segunda captura que Matrix Screener no puede crear, pero sí ejecutar y que permite capturar automáticamente los objetos detectados en el paso 2.

iMSRC incrementa la eficiencia de captura de Matrix Screener superando varias de sus limitaciones:

1- Matrix Screener necesita que el usuario introduzca manualmente la posición de objetos y mosaicos. iMSRC detecta automáticamente la ubicación de los objetos de interés (Figura 4: E) y los puntos de inicio y final de los mosaicos (Figura 4: F y Figura 5).

2- Matrix Screener solo puede capturar varios mosaicos de forma automática cuando estos tienen el mismo tamaño. Si se quiere capturar automáticamente varios objetos de tamaño distinto, Matrix Screener obliga a capturar mosaicos del tamaño del objeto más grande (Figura 4: B). iMSRC permite a Matrix Screener capturar automáticamente mosaicos de tamaño distinto (Figura 4: F y Figura 5: B).

En ambos casos, iMSRC evita la captura de campos vacíos o sin interés (cuadros rojos en Figura 4: A, B y F).

Si el número de objetos o mosaicos a capturar es muy alto, se genera gran cantidad de datos que suelen necesitar ser organizados y procesados de forma automática. Cuatro de las aplicaciones desarrolladas en esta tesis* están encaminadas al desarrollo de rutinas de detección, procesado y visualización de imágenes procedentes de la aplicación iMSRC. Otra aplicación** permite incrementar todavía más el rendimiento de Matrix Screener e iMSRC (como se verá más adelante).

En el apartado de material y métodos se pueden encontrar ejemplos de perfiles de primera y segunda captura utilizados en esta tesis.

IMAGEJ/FIJI

ImageJ es un producto de dominio público, programado en java, financiado por el National Institute of Health y continuado desde hace más de 25 años[44]. Actualmente, debido a su éxito, ha recibido un notable impulso con el incremento de la plantilla de programadores dedicados a actualizar su arquitectura a los estándares actuales de ingeniería informática.

* TMA Field Sampler y TMA Manager, Circulating Cancer Cells, y Single Cell Well Detector.
** iMSRC Stitching Module.

ImageJ puede encontrarse en su página web oficial[113], en otras localizaciones[114], en links de sitios afines[115] y en webs que alojan *software*[51,67]. Se puede descargar el código fuente y versiones ejecutables para Linux, MacOS y Windows. Su instalación es muy sencilla: solo requiere descargar, descomprimir y ejecutar un archivo.

La página web oficial de ImageJ es sencilla pero ordenada, rigurosa y con una gran cantidad de contenido, tanto para usuarios como para desarrolladores. Aloja documentación muy variada: introducción, conceptos básicos, guías de instalación y de usuario, ejemplos y tutoriales, preguntas frecuentes, varios manuales y libros, una API* de un lenguaje de guion propio, una lista de distribución de correo y una gran cantidad de enlaces externos a sitios con más documentación, una *wiki*[116], programas compatibles con ImageJ y aplicaciones basadas en ImageJ.

La comunidad de usuarios es extremadamente activa, con más de 1700 usuarios y desarrolladores inscritos en la lista de distribución de correo y más de 14 000 000 de visitas a la página web. Existen más de 300 macros y 500 módulos accesibles a la comunidad, la mayoría de ellos con ejemplos de uso y documentación asociada. El programa y el lenguaje de guion se actualizan con frecuencia, los cambios son registrados y las versiones anteriores son accesibles en un repositorio en github[117]. Además, existen cursos presenciales de carácter nacional[118] e internacional[119] y una conferencia bianual internacional propia[120].

La aceptación en la comunidad científica es muy buena, a fecha de 29 de mayo de 2018, una búsqueda en Medline del término "ImageJ" devuelve 1691 resultados, incluyendo documentación[33,74,121], pero artículos científicos en su amplia mayoría.

ImageJ tiene una interfaz sencilla, que proporciona una funcionalidad básica que puede ser extendida mediante módulos, macros y *scripts*. Puede trabajar con imágenes o pilas de imágenes de 8, 16, 32-bit y RGB color y virtualmente con cualquier formato de imágenes (libre y propietario) gracias a su completa compatibilidad con el OME Project[76,122].

Su diseño favorece la interoperabilidad. La conexión con otros tipos de *software* comercial (Matlab, Imaris**, etc.), *software* libre (OMERO, OME Bio-Formats, OME-XML, KNIME, Cell Profiler, Micro-Manager) y librerías (ITK, VTK, etc.) es directa, además permite ejecutar módulos y *scripts* escritos en varios lenguajes como java, JavaScript y Python.

ImageJ tiene un lenguaje de guion propio muy bien documentado que permite automatizar tareas de análisis y procesado de imágenes sencillas, pero que puede llegar a adquirir grandes niveles de sofisticación con conocimientos de programación.

---

* de sus siglas en inglés, *Application Programming Interface,* o interfaz de programación de aplicaciones

** Matlab (MathWorks, Natick, MA, USA), Imaris (Bitplane AG, Zurich, Suiza)

Para una lectura exploratoria de las características generales de ImageJ, recomendamos visitar su página web[113] y consultar las siguientes referencias: Collins (2007), Schneider et al. (2012) y Hartig (2013).

FIJI[45] (de sus siglas en inglés: *FIJI is Just ImageJ*) es una distribución en *software* libre de ImageJ que se distribuye bajo una licencia GPL. Consta de una colección de módulos orientados a las ciencias de la vida y algunos cambios que lo hacen más atractivo para desarrolladores. FIJI incluye: 1- una versión actualizada de ImageJ con modificaciones en la arquitectura básica, 2- una colección integrada de *plugins*, 3- ImgLib: una librería de algoritmos enfocados al análisis, procesado, visualización y escritura de imágenes para el desarrollo eficiente de algoritmos complejos, 4- la posibilidad de ejecutar una mayor variedad de idiomas de *scripting*, 5- una herramienta de actualización automática y 6- un entorno de desarrollo integrado (ver Schindelin et al. [2012] para un análisis exhaustivo). FIJI tiene su propia página web[123], que incluye un foro y una *wiki*[124], pero mantiene tal grado de colaboración con ImageJ y la compatibilidad es tal que la comunidad de usuarios y desarrolladores se considera la misma, de manera que todos aquellos recursos disponibles para ImageJ (manuales, documentación, lista de distribución de correos, etc.) son válidos, en su amplia mayoría, para FIJI.

IDENTIFICACIÓN, ANÁLISIS DE PROBLEMAS Y JUSTIFICACIÓN

Esta tesis aborda problemas concretos, relativamente independientes, que han llevado al desarrollo de una serie de aplicaciones para resolverlos. Los siguientes apartados de la tesis estarán divididos por aplicaciones, expuestas en orden de complejidad creciente. Es por eso que, a partir de aquí, se recomienda la lectura ordenada por bloques de aplicaciones a una lectura convencional, es decir, se recomienda la lectura de todos los apartados relacionados con el "Procesado de archivos de imágenes de videomicroscopía" (Análisis, objetivos, material y métodos, resultados y discusión), antes de pasar a leer los relacionados con la "Selección y visualización de imágenes procedentes de equipos HCS", etc.

Al final de la tesis, el glosario de términos ofrece una descripción contextualizada de conceptos técnicos que se dan por conocidos.

A continuación se presenta el análisis del estado de varias técnicas, como justificación al desarrollo de cada una de las aplicaciones.

PROCESADO DE ARCHIVOS DE IMÁGENES DE VIDEOMICROSCOPÍA

DeltaVision RT Restoration Microscopy System (Applied Precision, Issaquah, WA, USA, en adelante DeltaVision) es un equipo integrado de microscopía de campo claro y fluorescente equipado para la realización de experimentos in vivo. En la

actualidad, la marca que lo regenta ha discontinuado este modelo, junto con los servicios de soporte, en favor de otros modelos más avanzados. Se puede encontrar una descripción detallada de este equipo en la sección de material y métodos, junto con los perfiles de captura del experimento concreto que se desarrolla como ejemplo en esta tesis.

El experimento más frecuente en DeltaVision, en nuestra unidad, consiste en la captura automatizada de cursos temporales de células vivas con un número variable de marcadores fluorescentes, en diferentes posiciones espaciales. Suele ser necesario, además, la captura de varios planos por posición, bien debido a que el foco de las imágenes de campo claro no coincide con las de fluorescencia o porque las células quiescentes y en división tienden a situarse en planos distintos (las células en mitosis pierden adherencia al sustrato y se elevan sobre el resto).

El número de marcadores fluorescentes, posiciones espaciales, planos focales, frecuencia y tiempo total de captura, y el orden de adquisición de canales y planos focales, varía entre experimentos. Un experimento habitual, como el que se muestra en material y métodos, por ejemplo, consiste en la captura de:

- 31 posiciones espaciales.
- 3 canales: uno para un marcador fluorescente rojo, otro para un marcador fluorescente verde y otro para campo claro (capturados en ese orden).
- 2 planos: uno coincidiendo con el plano focal de la marca fluorescente y otro con el del campo claro (capturados en ese orden).
- 361 ciclos (una captura cada 8 minutos durante 48 horas).

El resultado de este experimento es un archivo en formato propietario (.dv) por cada posición (Figura 6), consistente en una pila de 2166 imágenes ordenadas de la siguiente manera:

1- Canal rojo en foco.
2- Canal verde en foco.
3- Canal de campo claro desenfocado.
4- Canal rojo desenfocado.
5- Canal verde desenfocado.
6- Canal de campo claro enfocado.

(Todo esto repetido 361 veces, Figura 6: A-G).

Nótese que las imágenes se capturan a 16 bit, en escala de grises, y que la mitad de las imágenes están desenfocadas.

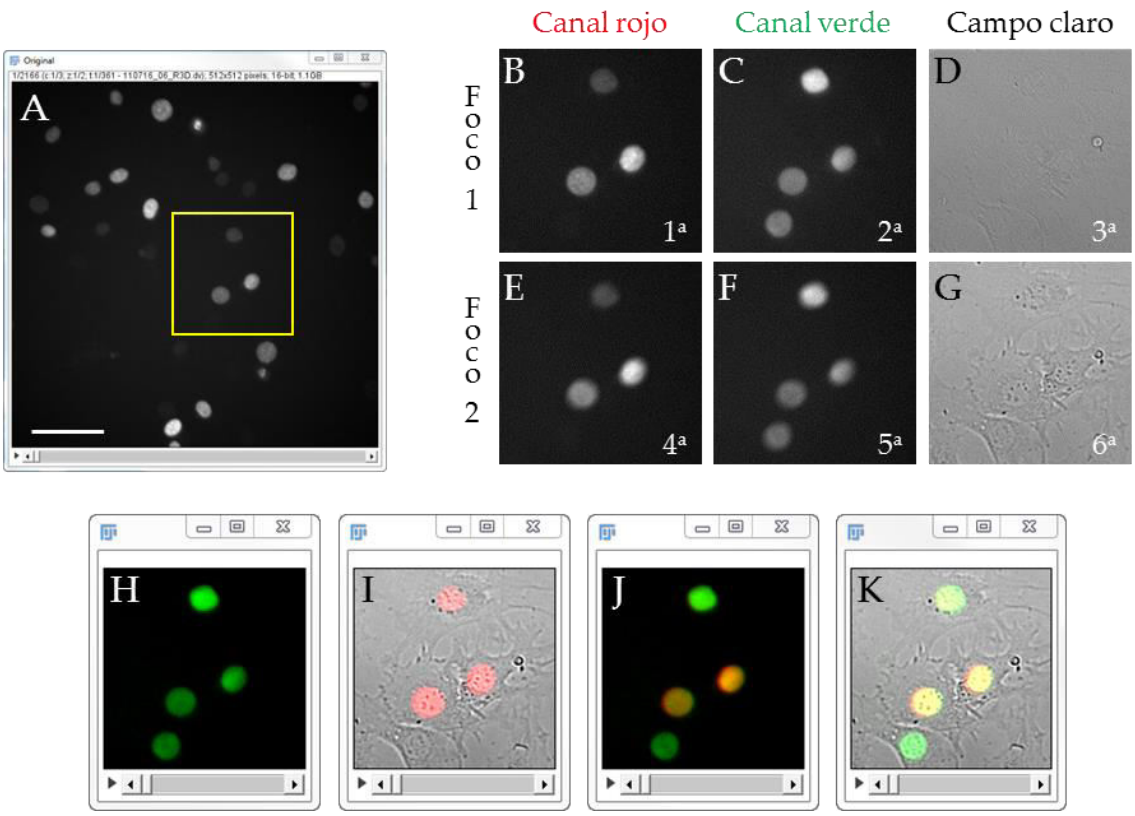


Figura 6. Procesado de archivos .dv. La pila de imágenes original (A) contiene imágenes de diferentes planos y canales, en un orden que se repite para cada ciclo de captura de un curso temporal. B, C, D, E, F y G son las imágenes del primer ciclo de captura; el ordinal en la esquina inferior derecha indica su posición en A, el título superior indica el canal que codifican y el título a la izquierda el plano focal del que proceden (Foco 1: plano focal de los canales rojo y verde, Foco 2: plano focal del canal de campo claro). Nótese que D, E y F están desenfocadas. El resultado informativo son pilas de imágenes procesadas y en foco, que contienen todos los puntos temporales para cada canal (H: canal verde) y/o composiciones de canales (I: canales rojo y campo claro, J: canales rojo y verde, K: canales rojo, verde y campo claro.). Las imágenes de B a K muestran un detalle del inserto en A. La figura ilustra el ejemplo mostrado en el cuerpo del texto. Barra de escala: 50 micras.

Obviamente, el resultado más informativo es una sucesión temporal de imágenes coloreadas en foco, organizadas en pilas de la siguiente manera:

- Una pila de 361 imágenes del canal rojo en foco.
- Una pila de 361 imágenes del canal verde en foco (Figura 6: H).
- Una pila de 361 imágenes de campo claro en foco.
- Diversas combinaciones de las pilas anteriores (una pila de 361 imágenes con los dos canales fluorescentes en foco, etc.) (Figura 6: I, J y K).

Un experimento de tres canales (rojo, verde y campo claro) y cuatro planos de captura (plano focal para los canales rojo y verde, plano focal para el campo claro, plano focal para las células en reposo y plano focal para las células en división), daría lugar a una pila de imágenes todavía más compleja, con una sucesión de imágenes que podría seguir el siguiente orden:

1- Campo claro en foco sobre células en reposo.
2- Canal verde desenfocado.
3- Canal rojo desenfocado.
4- Campo claro desenfocado.
5- Canal verde enfocado sobre células en reposo.
6- Canal rojo enfocado sobre células en reposo.
7- Campo claro enfocado sobre células en división.
8- Canal verde desenfocado.
9- Canal rojo desenfocado.
10- Campo claro desenfocado.
11- Canal verde enfocado sobre células en división.
12- Canal rojo enfocado sobre células en división.

Siendo en este caso el resultado más informativo:

- Una pila de imágenes de campo claro con las células en reposo enfocadas.
- Una pila de imágenes del canal verde con las células en reposo enfocadas.
- Una pila de imágenes del canal rojo con las células en reposo enfocadas.
- Una pila de imágenes de campo claro con las células en división enfocadas.
- Una pila de imágenes del canal verde con las células en división enfocadas.
- Una pila de imágenes del canal rojo con las células en división enfocadas.
- Una pila de imágenes de campo claro, verde y rojo con las células en reposo enfocadas.
- Una pila de imágenes de campo claro y verde con las células en reposo enfocadas.
- Etc. (cualquiera de las combinaciones posibles a gusto del usuario y dependiendo del diseño experimental).

Queda claro que el *software* de captura del equipo DeltaVision impone limitaciones al análisis de datos en este tipo de experimentos, al generar una pila de imágenes que contiene fotografías fuera de foco (Figura 6: D, E y F). Otros programas de captura, como el LAS AF de Leica, evitan tomar fotografías desenfocadas utilizando herramientas que corrigen las diferencias de foco durante la adquisición. En ausencia de estas herramientas, una buena alternativa sería el procesado de los archivos .dv mediante programas de procesado y análisis de imagen.

El *software* de visualización, procesado y análisis de imágenes de DeltaVision (softWoRx, Applied Precission), permite generar pilas informativas manualmente a partir de la pila original. Hacerlo supone visualizar dicha pila, extraer los canales y

planos de foco de interés, modificar su brillo y contraste (si es necesario) y fusionarlos posteriormente en imágenes multicanal. El proceso es largo y tedioso porque este *software* no permite automatizarlo, es poco intuitivo, requiere introducir rutas manualmente, es lento (debido a que este y el ordenador que lo ejecuta están obsoletos) y no se puede realizar cuando el equipo está capturando.

Alternativamente, los archivos .dv pueden ser exportados a una estación de análisis y procesados en ImageJ, permitiendo que el equipo pueda capturar durante el análisis y viceversa. Sin embargo, este procedimiento sigue requiriendo tiempo y, por otra parte, conocimientos de procesado de imágenes. En una unidad de soporte, esto significa tener un técnico especializado dedicando parte de su jornada a procesar imágenes y/o enseñando a procesarlas.

El desarrollo de una aplicación que permitiera automatizar el procesado de pilas de imágenes procedentes de experimentos de videomicroscopía multifocal y/o multicanal, permitiría aumentar el rendimiento del equipo y de la unidad. Si, además, la aplicación fuera flexible y su uso intuitivo, dicho procesado podría llevarse a cabo por personal sin conocimientos especializados.

SELECCIÓN Y VISUALIZACIÓN DE IMÁGENES PROCEDENTES DE EQUIPOS HCS

El equipo Opera LX High content screening system (Perkin Elmer, Waltham, MA, USA, en adelante Opera) está diseñado para la captura y análisis automatizado de grandes volúmenes de imágenes en placas multipocillo. Las características detalladas del mismo pueden encontrarse en la sección de material y métodos, junto con los perfiles de captura del experimento que se desarrolla como ejemplo en este apartado y el de resultados.

Opera se usa mayoritariamente en experimentos de cribado de alto rendimiento de fármacos, viabilidad celular y curvas de crecimiento. Un experimento habitual en este dispositivo, consiste en la captura de fotografías en los 60 pocillos centrales de una placa de 96, a razón de 48 campos por pocillo y dos canales por campo. Como el equipo Opera genera un archivo de imágenes en formato propietario (.flex) por campo, el experimento anterior generaría 2880 archivos .flex, que contendrían 5760 fotografías y ocuparían 3.67 GB de datos (Figura 7).

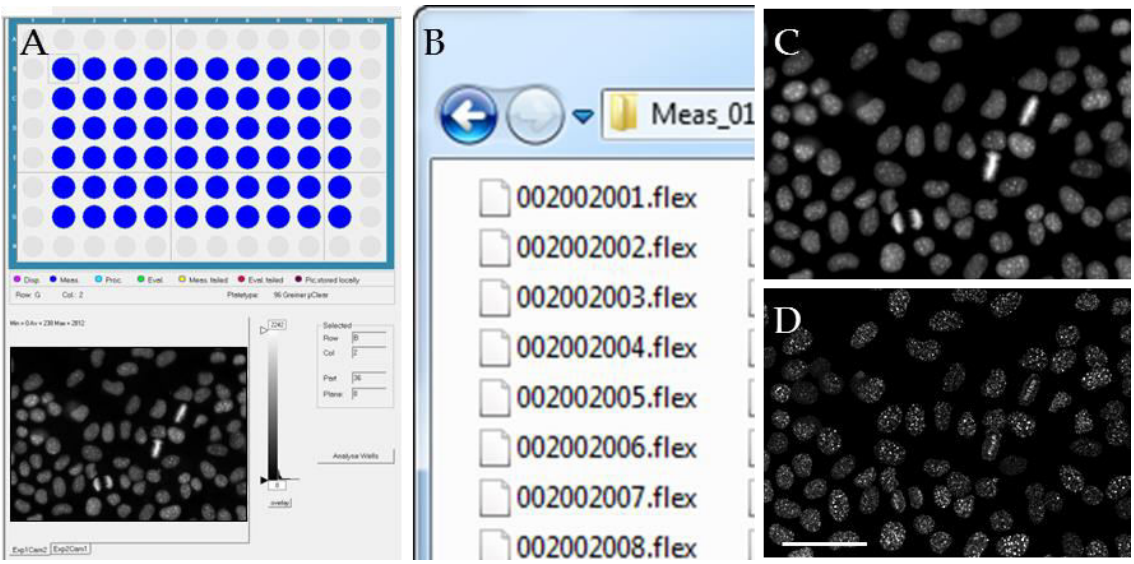


Figura 7. Visualización y codificación de archivos en el equipo Opera. **A**: detalle de la interfaz del *software* de captura de Opera, puede observarse un esquema representativo de una placa de 96 pocillos (en azul los pocillos capturados) y un detalle del visor que permite visualizar una única imagen. **B**: detalle de los archivos generados en el experimento capturado en A. El archivo 002002001.flex contiene las imágenes del primer campo del pocillo de la segunda fila y segunda columna de la placa. Dicho archivo puede codificar imágenes de varios canales (**C** y **D**). Barra de escala: 50 micras.

Una vez capturadas, procesadas y analizadas las imágenes, la mayoría de usuarios necesitan una manera eficiente de contrastar los resultados obtenidos con las imágenes originales, así como de obtener fotografías representativas de los experimentos para su registro y publicación.

Obviamente, el equipo Opera está dotado de *software* para la visualización de imágenes (Acapella, PerkinElemer, Waltham, MA, USA), sin embargo, este tiene tres limitaciones importantes:

1- No permite comparar diferentes imágenes simultáneamente (Figura 7: A).

2- El acceso al programa de visualización depende del número de usuarios y de licencias adquiridas. El equipo Opera consta de dos ordenadores, un ordenador de trabajo y control instrumental (aloja el software de adquisición de imágenes) y otro de análisis (almacena las imágenes y aloja el software para su análisis y visualización). El ordenador de análisis consta de un procesador de 4 núcleos accesible por licencias: cada licencia permite el acceso de un usuario a un procesador a través de un ordenador conectado en red al equipo de análisis. Un sistema con una, dos o cuatro licencias, permite visualizar imágenes hasta a un máximo de uno, dos o cuatro usuarios, respectivamente. Si algún usuario está capturando o analizando un experimento, bloqueará una licencia, de este modo, por ejemplo, en un sistema con una única licencia, no se pueden visualizar imágenes si el equipo está capturando o analizando; en un sistema con dos licencias se podrán realizar dos de las tres tareas simultáneamente (nunca podrá haber más de un

usuario capturando, obviamente, pero sí puede haber varios usuarios analizando o visualizando imágenes a la vez); etc.

3- El rendimiento de la visualización depende del uso de los procesadores. En un sistema de dos licencias, por ejemplo, es habitual que la visualización de imágenes sea lenta, si el equipo está capturando.

El primero de los inconvenientes se puede solucionar adquiriendo una aplicación (High Content Profiler, TIBCO Software Inc., Palo Alto, CA, USA) para la visualización de imágenes, perfectamente adaptada al entorno Opera y con mucha funcionalidad. Los otros dos problemas se pueden resolver adquiriendo tantas licencias como se considere oportuno. Normalmente, la adquisición de licencias suele venir acompañada de la adquisición de ordenadores, que se dedicarán al análisis o visualización de datos.

Alternativamente, se puede usar el equipo Opera para exportar las imágenes generadas en formato multi-TIFF o bien abrirlas directamente usando el módulo OME Bio-Formats de ImageJ (instalado por defecto en FIJI). Esto permite visualizar las imágenes simultáneamente a un número virtualmente ilimitado de usuarios, usando un amplio abanico de programas, pero obliga a abrir las imágenes a comparar manualmente, generalmente con sus canales separados, y de forma poco intuitiva (sin ayuda de interfaces gráficas y siguiendo un patrón de nombres en los archivos de imágenes que no es el habitual para el usuario*). Esto convierte la exploración de datos y obtención de imágenes representativas en un proceso tremendamente lento y farragoso, especialmente cuando el volumen de imágenes es tan elevado y no se tiene conocimientos de procesado de canales.

En una unidad de soporte, toda esta disyuntiva se traduce en asumir una o más de las siguientes consecuencias: 1- la pérdida de productividad que supone tener un equipo de captura dedicado al análisis y procesado de imágenes, 2- una inversión económica en *software* propietario de visualización, licencias para habilitar núcleos y ordenadores extra, 3- colas de usuarios esperando para el uso de la estación de análisis o 4- tiempo de personal dedicado a la visualización y extracción de imágenes en *software* libre y formación de usuarios.

El desarrollo de una aplicación en *software* libre que automatizara la visualización, selección, procesado y exportado de imágenes procedentes del equipo Opera, aumentaría la eficiencia de este proceso, sin coste económico ni limitación de acceso por licencias y permitiendo que el equipo pasara más tiempo dedicado a la captura y análisis de imágenes. Si el uso de dicha aplicación fuera intuitivo, esta

* La forma habitual de codificar la posición en las placas multipocillo sigue el patrón alfanumérico: "filas nombradas alfabéticamente, ordinal para las columnas" (por ejemplo, B3: pocillo que ocupa la segunda fila y tercera columna de la placa). Los archivos .flex siguen un patrón numérico: "tres primeros dígitos para la fila, tres siguientes para la columna y tres últimos para el campo" (por ejemplo, 002003005.flex: archivo que contiene las fotografías del quinto campo de la segunda fila y tercera columna de la placa, ver Figura 7: B).

permitiría, además, disminuir el tiempo dedicado por el personal de apoyo a la enseñanza del mismo.

GENERACIÓN DE MOSAICOS IRREGULARES EN SERIE

La aplicación iMSRC permite aumentar el rendimiento de Matrix Screener, al automatizar el posicionamiento de los mosaicos y permitir la captura de varios mosaicos de tamaño distinto (Figura 4 y Figura 5). Sin embargo, la productividad de los equipos dedicados a la captura de mosaicos se podría incrementar todavía más:

1- Ajustando el mosaico a la forma real del objeto.

2- Fusionando los campos en un equipo distinto al de captura.

En el primer caso se produce un ahorro de tiempo y recursos al evitarse la captura de campos sin información (Figura 8), en el segundo se gana tiempo de equipo dedicado a la captura y se ahorra recursos al evitar tener encendidos un láser y una lámpara que no se están utilizando durante la fusión de mosaicos.

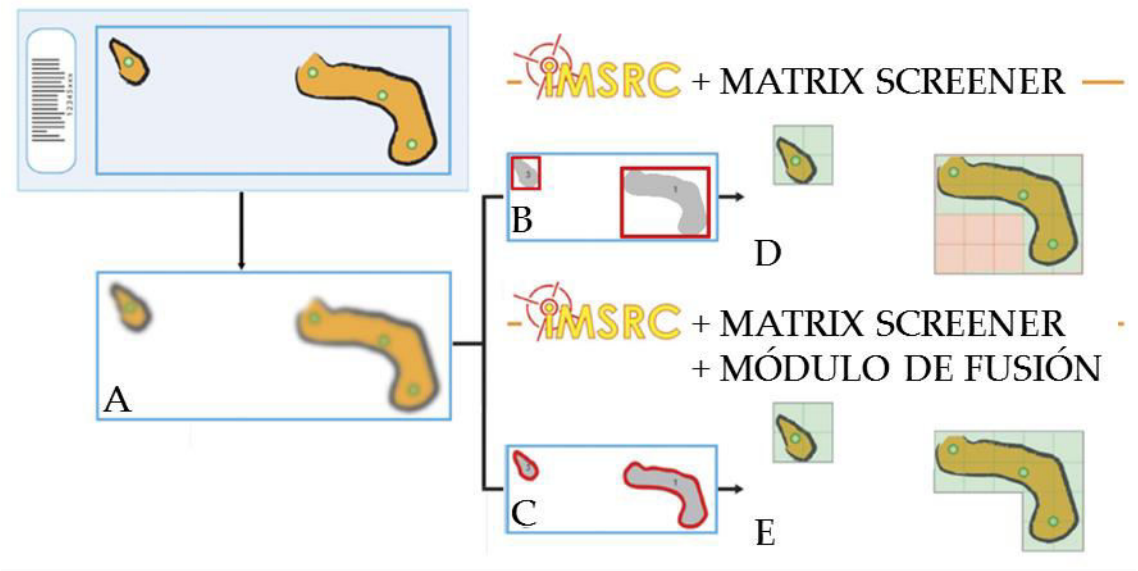


Figura 8. Comparativa de las capacidades de iMSRC y Matrix Screener con y sin un hipotético módulo de fusión de campos que permitiera generar mosaicos irregulares en serie. Sobre el mosaico de búsqueda (A) se ejecuta una rutina de análisis que extrae las coordenadas del rectángulo que circunscribe los objetos (B) o las coordenadas de los campos que contienen parte del objeto (C). El resultado son mosaicos rectangulares que pueden contener campos vacíos (cuadros rojos en D) o mosaicos que se ajustan a la forma del objeto (E). Matrix Screener no puede generar mosaicos irregulares, esto obliga a tener que utilizar un módulo de fusión de campos alternativo en E. Modificado de Carro et al. (2015).

El primero de los objetivos se puede cumplir adaptando iMSRC para que devuelva las coordenadas de los campos que contienen objetos mediante el análisis de un mosaico de búsqueda (Figura 8: C). Esto permite a Matrix Screener capturar todos los campos de un mosaico irregular (aquel cuya forma se ajusta a la del objeto y no al cuadrado que lo contiene, Figura 8: E, derecha), pero tiene un inconveniente: Matrix Screener no puede realizar la fusión de este tipo de campos en una imagen mosaico.

El segundo de los objetivos se puede cumplir: comprando tantos ordenadores y licencias de LAS AF y Matrix Screener como se desee o fusionando las imágenes en *software* libre. En ambos casos, sería deseable que se pudiera fusionar varios mosaicos en serie (uno detrás de otro, sin la intervención de un usuario), cosa que Matrix Screener no puede hacer.

Grid/Collection Stitching[125] es un plugin de ImageJ para la fusión de campos en mosaicos que se podría utilizar en vez de Matrix Screener, pero presenta los siguientes inconvenientes:

1- No puede fusionar archivos .lif (formato propietario de captura de Leica, en el que Matrix Screener captura los campos). Una posible alternativa consiste en utilizar la función de autoexportado de Matrix Screener para generar una colección de campos en archivos .tif.

2- Asume que todos los campos de un mosaico se localizan en el mismo directorio. La función de autoexportado de Matrix Screener genera una estructura de carpetas compleja (Figura 9), cuyas imágenes deben ser procesadas antes de que Grid/Collection Stitching pueda fusionarlas en un mosaico. Automatizar dicho procesado no es intuitivo, por lo que suele realizarse a mano. Esto obliga al usuario a repetir una serie de operaciones para cada campo: fusionar las imágenes de distintos canales, proyectar los distintos planos focales, corregir la rotación del escáner si la imagen se ha capturado en un microscopio confocal, renombrar la imagen resultante con las coordenadas del campo en el mosaico, y mover todos los campos a un único directorio, lo cual puede ser muy farragoso (o inviable) dependiendo del número de campos.

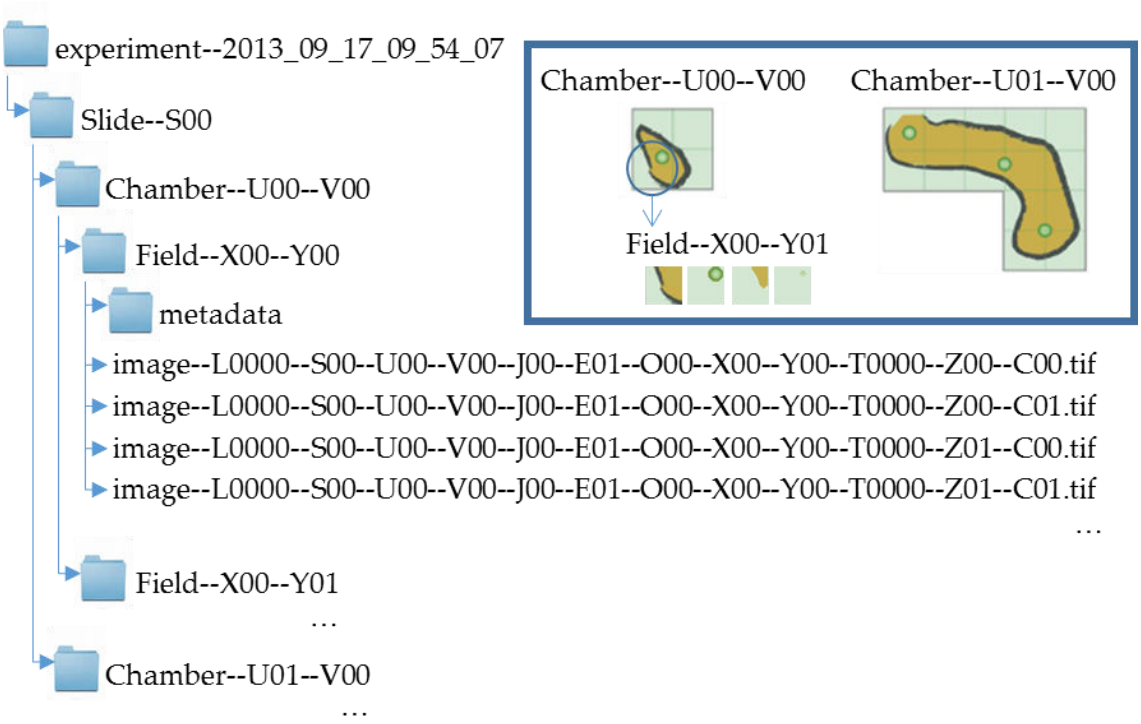


Figura 9. Estructura de archivos generada por el módulo de autoexportado de Matrix Screener de Leica. Los puntos suspensivos indican elementos que se repiten un número indefinido de veces. Cada una de las carpetas Chamber contiene todas las imágenes individuales ("image …") de un mosaico, organizadas por campos (carpetas Field). Los dos dígitos que siguen a U en Chamber son un ordinal para cada mosaico, los que siguen a X e Y en Field, representan la posición de cada campo en el mosaico, en un sistema cartesiano con el origen en el punto superior izquierdo, T, Z y C en el nombre de la imagen, codifican el orden en un curso temporal y el número de planos y canales, respectivamente*. Para que esta colección de imágenes pueda combinarse en un mosaico utilizando la herramienta Grid/Collection Stitching, las imágenes de cada campo deben proyectarse en un único plano, fusionarse los canales, rotarse (si se capturaron en un microscopio confocal) y copiarse a una carpeta con el nombre de las coordenadas en el mosaico. El inserto ilustra la nomenclatura de mosaicos, campos e imágenes.

3- El algoritmo de fusión de mosaicos irregulares es ineficiente y suele causar errores de solapamiento en mosaicos grandes. Grid/Collection Stitching funciona de dos maneras distintas. 1- En mosaicos regulares, cuando la posición relativa de los campos es conocida, compara el área de solapamiento entre campos adyacentes. 2- En mosaicos irregulares, o cuando la posición relativa de los campos es desconocida, compara todas las posibles combinaciones de solapamiento entre todos los posibles campos (Figura 10).

* Por ejemplo: image--…T0001--Z01--C01.tif en Field--X01--Y02 en Chamber--U03--V00, contiene la segunda imagen del curso temporal, de la segunda sección y segundo canal del campo en la segunda fila y tercera columna del cuarto mosaico capturado en un experimento.

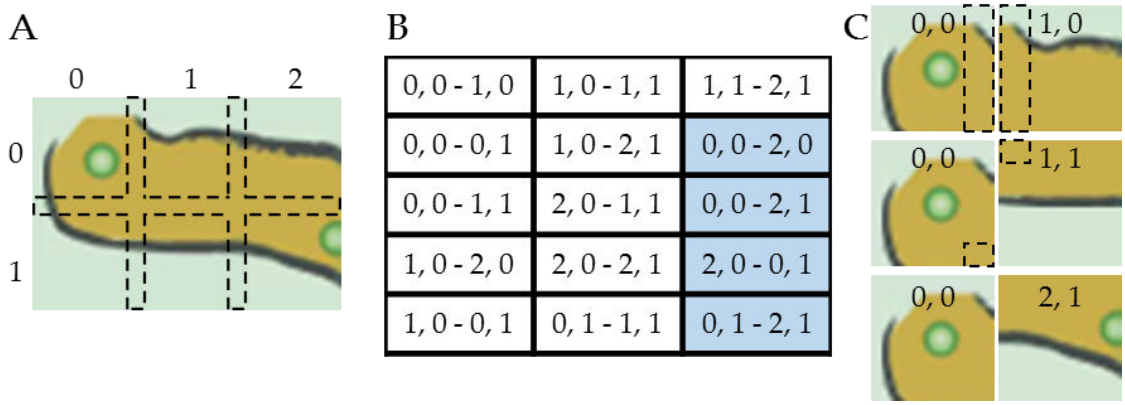


Figura 10. Comparativa de los algoritmo de fusión de campos en mosaicos, con o sin la información de la posición relativa de los mismos. **A**: objeto a capturar, muestra las zonas de solapamiento entre campos (delimitadas por líneas discontinuas) y las coordenadas que permiten identificarlos. **B**: posibles combinaciones de campos dos a dos del objeto en A. **C**: detalles de campos de A. Para hacer un mosaico de un objeto como el de A, sin conocer la posición relativa de los campos, se tienen que calcular los índices de regresión para cada una de las posibles traslaciones entre todas las posibles combinaciones de las imágenes de los campos tomadas de 2 en 2 (B). Cuando se conoce la posición relativa de los campos, solo se calculan los índices de regresión para las posibles traslaciones en el área de solapamiento, entre las imágenes de los campos contiguos (en fondo blanco en B). Grid/Collection Stitching es capaz de determinar la posición relativa de los campos de un mosaico a partir del nombre de sus imágenes. Los números en B y C representan coordenadas del eje en A.

4- No puede fusionar mosaicos en serie.

Resumidamente: el desarrollo de una aplicación en *software* libre capaz de fusionar mosaicos regulares e irregulares en serie, conservando la información de la posición relativa de los campos, en colecciones de imágenes autoexportadas de Matrix Screener, supondría salvar todas las limitaciones planteadas. Esto incrementaría enormemente la eficiencia del proceso de generación de mosaicos, sin coste monetario, liberando al personal dedicado a estas funciones, aumentando la disponibilidad de los equipos y ahorrando recursos.

CONTEO Y CARACTERIZACIÓN DE CÉLULAS CANCEROSAS CIRCULANTES

Los procesos cancerosos tienen peor pronóstico cuando el cáncer primario deja de estar localizado y pasa al torrente sanguíneo. Determinar la existencia, origen y grado del proceso metastásico es crucial para definir la terapia a seguir por el paciente. Existe actualmente equipamiento y metodología para la extracción de células cancerosas de muestras de sangre y su posterior caracterización. El grupo del Doctor Miguel Martín Jiménez (del Laboratorio de Oncología Traslacional del Hospital General Universitario Gregorio Marañón), con el que colaboramos, está interesado en implementar el conteo de células cancerosas circulantes de origen epitelial (cáncer de mama, de próstata, pulmón y colorrectal) como método de diagnóstico médico. Para ello cuenta con muestras de pacientes diagnosticados con diferentes tipos de cáncer, muestras de pacientes sanos y muestras control (en las que se insertan un número conocido de células cancerosas en muestras de pacientes sanos).

Las células cancerosas de origen epitelial se identifican mediante doble inmunofluorescencia (para un marcador verde y otro rojo), en células fijadas, con los núcleos teñidos con Hoechst 33342 y depositadas en un portaobjetos de 8 pocillos.

Desde el punto de vista del investigador encargado del recuento de objetos, el diseño experimental expuesto supone visitar manualmente un gran número de campos* con el microscopio de fluorescencia, alternando entre tres canales (azul, verde y rojo), en busca de posibles candidatos a célula verde negativa para rojo, lo que se traduce en una cantidad ingente de horas de trabajo tedioso por muestra.

La aplicación iMSRC podría proporcionar los medios necesarios para automatizar la captura de este tipo de muestras, incrementando enormemente la eficiencia del proceso. Esto requeriría: 1- configurar unos perfiles de primera y segunda captura, 2- desarrollar una rutina de análisis capaz de localizar candidatos a células positivas en un mosaico de bajos aumentos y 3- desarrollar una rutina capaz de procesar las imágenes de la captura a altos aumentos para su visualización.

CARACTERIZACIÓN DE UN NUEVO BIOMARCADOR DE CÉLULAS MADRE CANCEROSAS

Se piensa que las células madre cancerosas son responsables del crecimiento tumoral, la malignidad y la resistencia a terapia[126].

El doctor Cristopher Heeschen, lidera un grupo del CNIO interesado, entre otras cosas, en el desarrollo de sistemas de detección de células madre cancerosas. Un buen candidato sería un patrón de autofluorescencia característico de una subpoblación de células aisladas de varios cánceres primarios (Figura 11: A, B, C y E). Como parte de la caracterización *ex vivo* de este posible biomarcador, resultaría interesante seguir su transmisión entre divisiones celulares, en un número elevado de células. Con esta finalidad, se planteó el siguiente experimento: 1- sintetizar chips de micropocillos (Figura 11: D), 2- sembrar células sobre ellos, 3- seleccionar micropocillos con una única célula (Figura 11: E) y 4- capturar imágenes de campo claro y de la marca autofluorescente de forma periódica.

* Nominalmente, 1649 campos (relación entre el campo de visión del objetivo que permite identificar un objeto como positivo y la superficie de búsqueda).

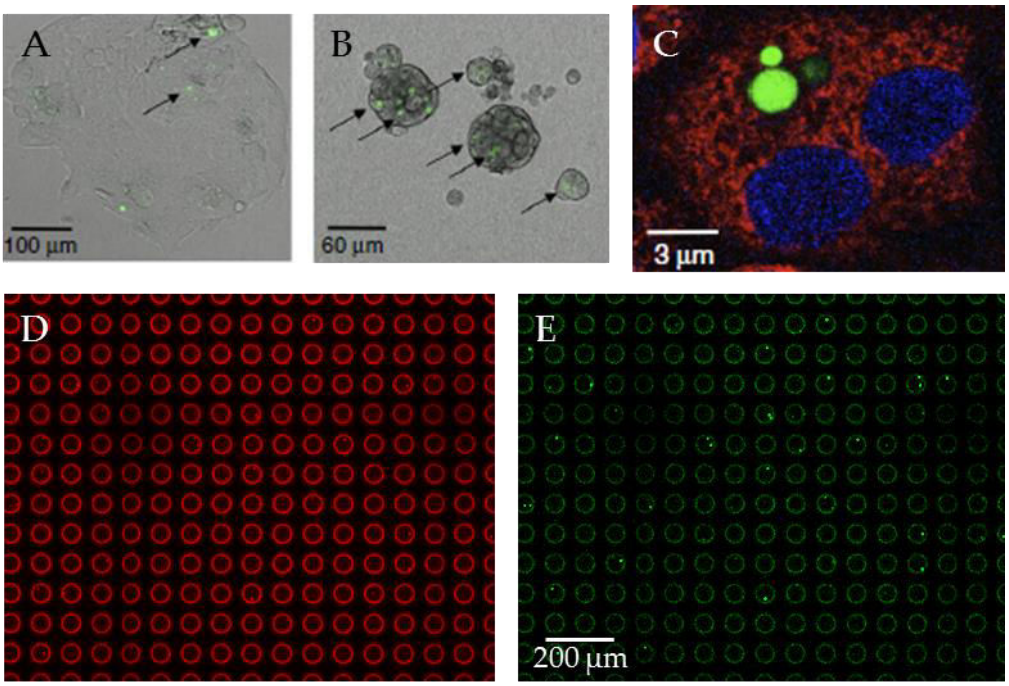


Figura 11. Patrón de autofluorescencia característico de un cultivo primario de células de un adenocarcinoma. **A** y **B**: imágenes de campo claro y fluorescencia de células cultivadas en condiciones de adherencia o como esferas, respectivamente. Las flechas señalan células positivas para el biomarcador. **C**: imagen de microscopía confocal de una célula marcada con Hoechst (azul) y Mito Tracker (rojo). **D** y **E**: fragmento de un chip de micropocillos que muestra los micropocillos (D) y las células (E). El patrón de autofluorescencia se muestra en verde en todos los casos. A, B y C modificados de Miranda et al. (2014).

Llevar a cabo este experimento de forma manual supone diseñar los perfiles de captura de cada canal y una rutina de autofoco, y localizar y marcar el centro de tantos micropocillos que contengan una única célula autofluorescente como sea posible. Todos estos pasos deben realizarse rápidamente, ya que el número de micropocillos con una única célula tiende a disminuir en la medida en que estas se dividen. Esto no es trivial porque, aunque el diseño experimental *a priori* parece sencillo, su ejecución se puede alargar debido a pequeños detalles técnicos imprevisibles, relacionados, generalmente, con la rutina de autofoco y la especificación fina de los perfiles de captura. Todos los pasos pueden dejarse grabados con anterioridad, excepto el de encontrar y marcar el centro de masas de los pocillos que contienen una única célula verde, lo cual lo convierte en el principal candidato susceptible de optimización del proceso global.

La aplicación iMSRC podría usarse para automatizar la captura de este tipo de muestras, incrementando la eficiencia global y la tasa de éxito de este experimento. Esto requeriría desarrollar unos perfiles de primera y segunda captura y dos rutinas informáticas: una para localizar las coordenadas de los micropocillos que contengan una única célula autofluorescente y otra para procesar las imágenes resultantes de la segunda captura.

CAPTURA Y VISUALIZACIÓN DE TMAs

Los *tissue microarrays* (TMAs) son pequeños cortes de tejidos dispuestos sobre un mismo portaobjetos siguiendo un patrón regular (Figura 5: A). Los TMAs son útiles para comparar tejidos distintos (sano/enfermo, hepático/pulmonar, normóxico/hipóxico, etc.) sometidos a las mismas condiciones experimentales (hibridación *in situ* fluorescente, inmunofluorescencia, etc.), ahorrando muestras y reactivos.

Un TMA estándar contiene decenas de pequeñas piezas de tejido que sintetizan una gran cantidad de información. Desde el punto de vista de su captura, el operador debe tomar varias fotografías sobre cada una de ellas, teniendo la precaución de registrar cuidadosamente su procedencia. Este es un trabajo tremendamente tedioso, por ejemplo, un TMA de 80 piezas de tejido, en el cual se quieren capturar 5 campos por pieza, requiere capturar 400 imágenes, lo que para un usuario medio y dependiendo de las condiciones de captura (número de canales, planos, etc.) puede requerir un día de trabajo. Es por eso que se han desarrollado múltiples equipos comerciales, denominados escáneres digitales (SensoScope Fluorescence-inv, Aperio FL, Ariol, Axio Scan.Z1, etc.*), para la captura automática de este tipo de muestras.

Los escáneres digitales son microscopios robotizados, más o menos modificados, equipados con platinas motorizadas, fuentes de iluminación, juegos de objetivos, cámaras y sistemas de autofoco. Todos estos elementos se controlan a través de un programa de captura específico, que permite a un usuario definir (generalmente a través de una imagen de baja resolución) las áreas que se capturarán automáticamente. Estos sistemas también disponen de *software* para la visualización de los resultados obtenidos, el cual incluye rutinas de procesado (para fusionar canales, proyectar planos, etc.) y proporciona medios para localizar las imágenes capturadas dentro de cada pieza de tejido[127].

La Unidad de Microscopía Confocal del CNIO no dispone de escáneres digitales, pese a disponer de varios microscopios equipados con todo el *hardware* necesario, por lo que la captura de TMAs y el procesado de las imágenes resultantes, se realiza manualmente. Determinar el origen de las imágenes depende de las notas que haya tomado el usuario.

Como se ha visto anteriormente, la aplicación iMSRC podría proporcionar los medios necesarios para automatizar la captura de TMAs. Esto requeriría: 1- desarrollar una macro capaz de extraer las coordenadas del rectángulo que contiene cada uno de los tejidos (si se van a capturar enteros) o de un número definido de campos por tejido y 2- desarrollar una herramienta para la visualización y procesado de las imágenes resultantes. De conseguirlo, se habría dotado a la unidad de la funcionalidad de un

* Sensovation AG, Radolfzell, Alemania; Leica Microsystems, Wetzlar, Alemania; Carl Zeiss AG, Oberkochen, Alemania; respectivamente.

equipamiento de decenas de miles de euros a coste cero (de realizarse sobre un microscopio fluorescente convencional) o de un equipamiento completamente puntero (no tenemos conocimiento, hasta la fecha, de un escáner de tejidos de barrido láser confocal). Todo esto, por supuesto, permitiría incrementar enormemente la eficiencia de la captura manual de TMAs.

OBJETIVOS

<<Every good work of software starts by scratching a developer's personal itch.>>

Eric S. Raymond

OBJETIVOS GENERALES

Desarrollar herramientas informáticas en el *software* libre FIJI que permitan automatizar tareas complejas en microscopía de fluorescencia convencional o confocal, superando las limitaciones impuestas por el *software* de captura y análisis propietario. Dichas tareas comprenden: 1- la automatización del procesado y visualización de grandes volúmenes de imágenes para que puedan ser llevadas a cabo eficientemente por usuarios sin conocimientos especializados, y 2- implementar diseños experimentales de captura guiada por análisis de imagen usando la aplicación iMSRC.

OBJETIVOS ESPECÍFICOS

En el área de la gestión de grandes volúmenes de imágenes, se desarrollarán aplicaciones fáciles de usar que permitan:

1- Extraer automáticamente pilas de imágenes informativas de pilas procedentes de experimentos de videomicroscopía multicanal y/o multifocal del equipo DeltaVision.

2- Visualizar, procesar y seleccionar, de forma rápida y sencilla, imágenes procedentes del sistema de HCS Opera.

3- Fusionar mosaicos regulares e irregulares en serie, a partir de colecciones de imágenes procedentes de las aplicaciones Matrix Screener de Leica e iMSRC.

En el área de la captura guiada por análisis de imagen basado en iMSRC, se desarrollarán macros de detección y aplicaciones de procesado de imágenes, para la captura automatizada de imágenes de:

4- Células cancerosas circulantes.

5- Células aisladas en micropocillos, positivas para un posible biomarcador de células madre cancerosas.

6- *Tissue microarrays.*

MATERIAL Y MÉTODOS

<<The computer allows you to make mistakes faster than any other invention, with the possible exception of handguns and tequila.>>

Mitch Ratcliffe.

DESARROLLO DE APLICACIONES

MODELO DE DESARROLLO

La metodología utilizada es similar en todas las aplicaciones.

Se utilizó un modelo de desarrollo de *software* ágil. Se diferenciaron 6 fases secuenciales: análisis, planificación, ejecución, testado, lanzamiento y soporte de la aplicación. El análisis se llevó a cabo mediante reuniones del equipo de desarrollo con el usuario, en las cuales se determinaron y priorizaron los requisitos de las aplicaciones. La planificación siguió el mismo sistema de reuniones, pero solo con el equipo de desarrollo, en ellas se establecieron las bases estructurales del *software*, hitos de desarrollo y plazos de entrega a cumplir durante la fase de ejecución. El testado lo llevó a cabo el equipo de desarrollo primero, el resto del equipo de la Unidad de Microscopía Confocal después y, finalmente, el usuario responsable del proyecto. Todas las mejoras se introdujeron en el código y, una vez obtenida una versión estable, se colgó en los ordenadores de análisis compartidos de la unidad y se comunicó al resto de usuarios de la misma. Cualquier tipo de incidencia con las aplicaciones a posteriori se resolvió e incorporó al código durante la fase de soporte. La identificación de nuevos requisitos y adquisición de funcionalidad implicó, en mayor o menor medida, la repetición del procedimiento anteriormente expuesto.

La documentación generada en este proceso se recogió en los comentarios del código fuente de estas aplicaciones y en la presente memoria de tesis.

Se eligió este modelo de trabajo por las siguientes razones:

1- Equipo de desarrollo muy pequeño: compuesto por un programador y un coordinador.
2- Proyectos pequeños o divisibles en funciones pequeñas, con requisitos variables o poco conocidos.
3- Usuarios accesibles y proactivos.
4- Tiempos de entrega cortos, impuestos por la necesidad urgente de adquisición de funcionalidad básica, expandible posteriormente.

RECURSOS INSTRUMENTALES

Las aplicaciones generadas en esta tesis se han desarrollado usando macros y un lenguaje de guion específico del *software* de dominio público ImageJ[44]. Su programación se realizó en su distribución en *software* libre FIJI[45], usando la herramienta de grabado de comandos de este programa y su entorno de desarrollo integrado.

Se decidió usar un lenguaje de guion porque proporciona una buena relación entre el aprendizaje del mismo y la eficiencia de ejecución y escritura del código.

Se decidió usar *software* libre porque las características revisadas en la introducción de esta tesis (fácil accesibilidad, código fuente visualizable y modificable, bajo precio, flexibilidad e interoperabilidad) hacen que sea la opción que mejor se ajusta a las buenas prácticas científicas de obtener, reproducir y compartir el conocimiento científico.

Finalmente, se decidió usar ImageJ/FIJI por ser, sin duda alguna, el *software* libre dedicado al análisis y procesado de imagen de mayor difusión y reconocimiento entre la comunidad científica[22,23]. Se descartaron otras opciones, como ICY[77] o Diderot[99], por estar mucho menos difundidas.

Las aplicaciones de esta tesis se han desarrollado en FIJI ejecutándose sobre ImageJ (v.1.50b) en una plataforma con las siguientes características: sistema operativo de 64 bit, Windows Server 2008 R2 Standard (Service Pack 1, Microsoft, Redmond, WA, USA), 16 GB de memoria RAM principal, dos procesadores Intel (Santa Clara, CA, USA) Xeon CPU ES620, de cuatro núcleos a 2.4 GHz y 2 TB de memoria secundaria. Alternativamente, en casos de gran demanda de recursos de procesado (*i. e.* mosaicos de gran tamaño), iMSRC Stitching Module se ejecutó en una plataforma con un sistema operativo de 64 bit, Windows Server 2012 R2 Standard (Service Pack 1, Microsoft), 60 GB de memoria RAM principal, 2 procesadores Intel Xeon CPU X5660, de seis núcleos a 2.8 GHz y 70 GB de memoria secundaria.

FIJI incluye los módulos OME Bio-Formats[76], Grid/Collection Stitching[125], ROI Manager y Auto Local Threshold por defecto. Deltavision Opener es una contribución de Sebastien Huart y Fabrice Cordeliers que requiere ser instalada (puede encontrarse en: https://imagej.nih.gov/ij/plugins/index.html). La operación de reconstrucción (BinaryFilterReconstruct_), forma parte de una colección de operadores morfológicos contribuidos por Gabriel Landini[128] que pueden instalarse desde: http://www.mecourse.com/landinig/software/software.html.

Las aplicaciones desarrolladas deberían funcionar en ImageJ, siempre y cuando se instalen los módulos no incluidos en esta distribución que requiera cada una de ellas (ver párrafo anterior). Podría ser necesario realizar adaptaciones menores del código.

Todas las aplicaciones desarrolladas en esta tesis pueden encontrarse en el apéndice en formato .ijm. Si este archivo se arrastra a la interfaz principal de FIJI, la aplicación se abrirá en su editor de código. Esto permite visualizarlas y ejecutarlas (pulsando sobre el botón Run). Alternativamente, los archivos .ijm se pueden copiar a la carpeta de plugins de ImageJ/FIJI. Al hacerlo e inicializar este programa, las aplicaciones aparecerán en el menú plugins, y se podrán ejecutar haciendo clic sobre su nombre.

EQUIPOS DE MICROSCOPÍA

DELTAVISION RT RESTORATION MICROSCOPY SYSTEM

DeltaVision RT Restoration Microscopy System es un equipo integrado de microscopía de fluorescencia y campo claro desarrollado por la Universidad de California (San Francisco, CA, USA) en colaboración con la compañía Applied Precision (Issaquah, WA, USA) entre 1983 y 1994.

La unidad de microscopia confocal del CNIO adquirió uno de estos equipamientos en 2005. En la actualidad, la compañía proveedora de servicios y tecnología biomédica GE Healthcare Life Sciences (Chicago, IL, USA) regenta la marca DeltaVision. El modelo DeltaVision RT ha sido discontinuado, junto con los servicios de soporte, en favor de otros modelos más avanzados.

DeltaVision RT se diseñó para la captura y análisis de imágenes microscópicas tridimensionales durante periodos largos de tiempo y múltiples muestras vivas o fijadas, así como para la realización de imágenes mosaico. Está dotado para el uso de microscopía de fluorescencia, campo claro, contraste de fases y contraste interferencial diferencial (en adelante DIC, de sus siglas en inglés: *differential interference contrast*). Una descripción detallada del equipo puede encontrarse a continuación:

Microscopio invertido Olympus IX71 (Olympus Corporation, Tokio, Japón), equipado con:

- Fuente de luz halógena para luz transmitida.
- Lámpara de arco de mercurio para microscopía de fluorescencia.
- Polarizador, analizador, prismas de Wollaston y anillos para DIC y contraste de fases.
- Revólver con el siguiente juego de objetivos de Olympus:
    - 20X, 0.75 AN, UPlanSApo.
    - 40X, 1.30 AN, oil, UPlan FL N.
    - 60X 1.42 AN, oil, PlanApo N.
    - 100X 1.40, oil, UPlanSApo.
- Rueda automática de filtros de densidad neutra situada a la salida de la lámpara.
- Rueda automática de filtros de excitación de paso de banda situada a la salida de la lámpara (Tabla 2).
- Rueda automática de filtros de emisión de paso de banda situada a la entrada de la cámara (Tabla 2).
- Rueda manual de filtros situada en la entrada de los oculares.
- Obturador automático a la salida de la lámpara.

- Platina galvanométrica motorizada.
- Cámara EMCCD* Cascade II: 512 (Photometrics, Tucson, AZ, USA). Cámara EMCCD, monocroma, 16 bit de rango dinámico, refrigerada a -65ºC, sensor retroiluminado de 512x512 píxeles de 16x16 micras cada uno. Con una frecuencia máxima de captura de al menos 26.2 fps**.
- Mesa antivibratoria.
- Cámara termostatizada y suministro de aire comprimido y CO_2 para la realización de experimentos in vivo.
- Ordenador de control instrumental: coordina la actividad de todo el *hardware* (motores de la platina, rueda de filtros, obturador, cámara, etc.). Sistema Operativo híbrido 16-32 bit Windows 98 second edition (v. 4.10.2222A, Microsoft, Redmond, WA, USA), 512MB de memoria RAM principal, procesador Intel x86.
- Ordenador de trabajo: interfaz principal de control del equipo. Contiene el *software* de captura y análisis, y almacena las imágenes capturadas. Sistema Operativo 32 bit, CentOS Linux 4.8, 1GB de memoria RAM principal, procesador Intel Pentium 4 a 2.8 GHZ y. 300 GB de memoria secundaria. El *software* de captura softWoRx 3.5.1 (Applied Precision, Issaquah, WA, USA), permite controlar la adquisición y visualización de imágenes, así como operaciones de procesado sencillas (cuantificación de niveles de gris, modificación de brillo y contraste, etc.) y otras más complejas como deconvolución.

Tabla 2. Características de los filtros de excitación y emisión del equipo DeltaVision: combinaciones que permiten visualizar los fluoróforos más comunes.

FLUORÓFORO	***EXCITACIÓN (nm)***	***EMISIÓN (nm)***
DAPI, Hoechst	360 /40	457/50
Fluoresceína, GFP, CY2, Alexa488	490/20	528/38
Rodamina, Texas Red, CY3, Alexa555	555/28	617/73
CY5, Alexa647	640/20	685/40
CFP	436/10	465/30
YFP	492/18	535/30
DsRed Express, DsRed2	580/20	630/60
EGFP	470/40	525/50

* De sus siglas en inglés: electron-multiplying charged-coupled device.
** De sus siglas en inglés: *frames per second.*

OPERA LX HIGH CONTENT SCREENING SYSTEM

El equipo Opera LX High Content Screening System (Perkin Elmer, Waltham, MA, USA) es un lector automatizado de placas multipocillo, confocal de disco de Nipkow, diseñado para experimentos de cribado de alto rendimiento en células fijadas o en cultivo. El equipo consta de un microscopio confocal de disco de Nipkow Andrómeda (Till photonics, Gräfelfing, Alemania, desde 2016: FEI Munich GmbH-Thermo Fisher Scientific, Waltham, MA, USA), equipado con:

- Tres láseres de estado sólido: 488 (75mW), 561 (40mW) y 640 (40mW).
- Triple filtro de banda y dicroicos para el juego de láseres.
- Cámara CCD* pixelfly (pco.imaging, Kelheim, Alemania). Cámara monocroma, 12 bit de rango dinámico, 1392X1024 píxeles de 6.45x6.45 micras cada uno. Con una frecuencia máxima de captura de 23 fps.
- Cámara CCD sensicam (pco.imaging, Kelheim, Alemania). Cámara CCD monocroma, 12 bit de rango dinámico refrigerada a -15ºC, 1376X1040 píxeles de 6.45x6.45 micras cada uno. Con una frecuencia máxima de captura de 19.8 fps.
- Lámpara de Xenon Polychrome V, de 150W, con monocromador para la emisión a 360-410 nm.
- Juego de filtro de banda de emisión y dicroicos de excitación y emisión para la fuente de luz UV.
- Platina motorizada.
- Sistema de autofoco por *hardware* guiado por láser.
- Revólver automatizado con el siguiente juego de objetivos de Olympus:
    - 10X, 0.4 AN, UPLAAPO.
    - 20X, 0.45 AN, LCPLFL.
    - 20X, 0.7 AN, *water*, UAPO.
    - 40X, 0.9 AN, *water*, WLSM.
- Cámara termostatizada y suministro de aire comprimido y CO2 para la realización de experimentos *in vivo.*
- Bomba y circuito cerrado de suministro de agua a los objetivos.
- Ordenador de control instrumental y trabajo: coordina la actividad de todo el *hardware* (motores de la platina, rueda de filtros, obturador, cámara, etc.) y contiene la interfaz del *software* de captura y análisis (Evoshell y Acapella, respectivamente, PerkinElemer, v.2.0.0.12017), es capaz de realizar rutinas de corrección de luz no homogénea (*shading,* en inglés) y alineamiento de las cámaras. Sistema operativo de 32 bit,

* De sus siglas en inglés: *charged-coupled device.*

Windows 7 Professional (Service Pack 1, Microsoft), 3 GB de memoria RAM principal, procesador Intel Core 2 Duo P8400, de dos núcleos a 2.26 GHz. 500 GB de memoria secundaria.

- Ordenador de análisis: realiza las labores de análisis asignadas por el ordenador de control y almacena las imágenes capturadas. Contiene el *software* de análisis Acapella v.2.6 (Perkin Elmer), que contiene un lenguaje de guión propio y herramientas de programación visual para el análisis automatizado de imágenes. Sistema Operativo de 64 bit, Windows 7 Professional (Service Pack 1, Microsoft), 16 GB de memoria RAM principal, procesador Intel Xeon E3-1220 v2 de 4 núcleos a 3.10 GHz y. 2 TB de memoria secundaria. Núcleos del procesador accesibles por licencias. Se dispone de dos licencias, por lo que la capacidad operativa del equipo es de dos núcleos, uno accesible a través del ordenador de control instrumental y trabajo, y otro a través de una estación de análisis independiente en red.

LEICA TCS SP5 X

El equipo Leica TCS SP5 X (Leica Microsystems, Wetzlar, Alemania) es un microscopio de barrido láser confocal diseñado para la captura y cuantificación de muestras fluorescentes en el área de la investigación de las ciencias de la vida. Permite la captura automatizada y el análisis de imágenes microscópicas tridimensionales durante periodos largos de tiempo y múltiples muestras vivas o fijadas, así como la realización de imágenes mosaico. Una descripción detallada de los elementos del equipo relevantes para esta tesis puede encontrarse a continuación:

microscopio invertido Leica DMI6000 B equipado con:

- Fuente de luz halógena para luz transmitida.
- Lámpara de mercurio para microscopía de fluorescencia.
- Revólver automatizado con el siguiente juego de objetivos de Leica:
    - 10X, 0.4 AN, HC PLA APO CS.
    - 20X, 0.5 AN, HC PL FLUOTAR PH2.
    - 20X, 0.7 AN, imm, HC PL APO.
    - 40X, 1.25 AN, *oil*, HCX PLA APO CS.
    - 63X, 1.3 AN, *glycerol*, HCX PLA APO.
    - 63x, 1.4 AN, *oil*, HCX PL APO CS PH3.
- Rueda de cubos de filtros automatizada para la visualización en campo ancho de los fluoróforos más comunes.
- Platina galvanométrica motorizada.

- Sistema de autofoco por *software.*
- Mesa antivibratoria.
- Juego de láseres para la excitación de fluoróforos:
    - Diodo 405: longitud de onda de excitación de 405 nm, estado sólido, emisión continua, 90mW.
    - Ar: longitudes de onda de excitación de 458, 476, 488, 496, 514 nm, estado gaseoso, emisión continua, 200 mW.
    - DPSS 561: longitud de onda de excitación de 561 nm, estado sólido, emisión continua, 100 mW.
    - HeNe: longitud de onda de excitación de 633 nm, estado gaseoso, emisión continua, 15mW.
    - Láser blanco: longitud de onda de excitación ajustable entre 470-670 nm, estado sólido, emisión pulsante, 500 mW.
- Escáner convencional: capaz de rastrear a una velocidad máxima de 1400 Hz (2800 Hz en modo bidireccional).
- Escáner resonante: capaz de rastrear a una velocidad máxima de 8000 Hz (16 000 Hz en modo bidireccional).
- Microscopía espectral por AOTF y AOBS.
- Un detector híbrido, tres fotomultiplicadores para fluorescencia y un fotomultiplicador para luz transmitida.
- Sistema de *time gating.*
- Cámara termostatizada y suministro de aire comprimido y CO_2 para la realización de experimentos *in vivo.*
- Ordenador de control instrumental, trabajo y análisis: coordina la actividad de todo el *hardware* (motores de la platina, rueda de filtros, obturador, etc.), contiene la interfaz del *software* de captura y análisis (LAS AF v.2.7.3.9723, incluye el módulo Matrix Screener para la captura automatizada de mosaicos) y almacena las imágenes capturadas. Sistema Operativo de 32 bit, Windows 7 Professional (Service Pack 1, Microsoft), 4 GB de memoria RAM principal, procesador Intel Xeon X5550 de ocho núcleos a 2.66 GHz. 1 TB de memoria secundaria.

LEICA DMI6000 B

El equipo Leica DMI6000 B (Leica Microsystems, Wetzlar, Alemania) es un equipo integrado de microscopía diseñado para su uso en el contexto biomédico y la investigación. Está equipado para el examen de tejidos y cultivos celulares, así como para la micromanipulación y microinyección. Permite la captura automatizada y el análisis de imágenes microscópicas tridimensionales durante periodos largos de tiempo y múltiples muestras, vivas o fijadas, así como la realización de imágenes mosaico. Este equipo permite el uso de microscopía de fluorescencia, campo claro, contraste de fases y DIC. Una descripción detallada de los elementos del equipo relevantes para esta tesis puede encontrarse a continuación:

estativo invertido equipado con:

- Fuente de luz halógena para luz transmitida.
- Lámpara de mercurio para microscopía de fluorescencia.
- Polarizador, analizador y prismas de Wollaston para DIC y contraste de fases.
- Revólver automatizado con el siguiente juego de objetivos de Leica:
    - 5X, 0.15 AN, HCX PL FLUOTAR.
    - 10X, 0.4 AN, HC PLA APO CS.
    - 20X, 0.5 AN, HC PL FLUOTAR PH2.
    - 40X, 1.25 AN, *oil*, HCX PLA APO CS.
    - 63X, 1.3 AN, *glycerol*, HCX PLA APO.
    - 63x, 1.4 AN, *oil*, HCX PL APO PH3.
- Rueda de cubos de filtros automatizada para la visualización de los fluoróforos más comunes, resumidos en la Tabla 3.
- Filtros externos de excitación y emisión automáticos para ensayos de FRET Cyan-Yellow.
- Platina motorizada.
- Sistemas de autofoco por *software* y por *hardware* guiado por láser.
- Cámara CCD ORCA-ER C4742 (HAMAMATSU, Hamamatsu, Japón). Cámara monocroma, 12 bit de rango dinámico, 1344X1024 píxeles de 6.45x6.45 micras cada uno. Con una frecuencia máxima de captura de 31.3 fps.
- Mesa antivibratoria.
- Cámara termostatizada y suministro de aire comprimido y CO_2 para la realización de experimentos *in vivo.*
- Ordenador de control instrumental, trabajo y análisis: coordina la actividad de todo el *hardware* (motores de la platina, rueda de filtros,

obturador, cámara, etc.), contiene la interfaz del *software* de captura y análisis (LAS AF, Leica Microsystems, Wetzlar, Alemania, v.2.7.3.9723, incluye el módulo Matrix Screener para la captura automatizada de mosaicos) y almacena las imágenes capturadas. Sistema Operativo de 64 bit, Windows 7 Professional (Service Pack 1, Microsoft), 16 GB de memoria RAM principal, procesador Intel Xeon E5-2620 de dos núcleos a 2.4 GHz. 2 TB de memoria secundaria.

Tabla 3. Juego de cubos de filtros del equipo Leica DMI6000 B.

NOMBRE	***FILTRO DE EXCITACIÓN***	***DICROICO***	***FILTRO DE EMISIÓN***
A4	360/40	400	470/40
L5	480/40	505	527/30
N2.1	515-560	580	LP 590
TX2	560/40	595	645/75
Y5	620/60	660	700/75

Todos los filtros de excitación y emisión son de paso de banda excepto el filtro de emisión N2.1, que es de paso largo (LP, de sus siglas en inglés: *Long Pass*).

OBJETIVO 1. DESARROLLAR UNA APLICACIÓN QUE PERMITA PROCESAR PILAS DE IMÁGENES PROCEDENTES DE EXPERIMENTOS DE VIDEOMICROSCOPÍA MULTICANAL Y MULTIFOCAL.

La figura que ilustra el procesado de pilas de imágenes de videomicroscopía (Figura 6), se obtuvo de un experimento perteneciente a estudios en curso no publicados, por lo que se omite toda información sensible relativa a su contenido. Las imágenes muestran células adherentes en un cultivo en fase de crecimiento, algunas de las cuales expresan GFP y/o RFP en una localización nuclear.

La captura se realizó en el equipo DeltaVision configurado de la siguiente manera: objetivo 20X, 0.75 AN, UPlanSApo. Formato de captura 512x512, sin binning, ganancia 4X. Tamaño de pixel: 0.50016 micras. Se capturaron 2 planos focales y tres canales en 31 posiciones, cada 8 minutos durante 48 horas. El resultado fueron 31 archivos .dv de 1.120 GB cada uno (34.72 GB en total).

La configuración de canales utilizada es la siguiente:

1- Filtros de excitación y emisión para rodamina, filtro de densidad neutra de 10% de transmitancia, 0.6 segundos exposición.

2- Filtros de excitación y emisión para fluoresceína, filtro de densidad neutra de 32% de transmitancia, 0.5 segundos de exposición.
3- Campo claro, 0.1 segundos de exposición.

Los canales fluorescentes (1 y 2) están en foco en el primer plano focal, y el campo claro (3) en el segundo plano focal. El archivo resultante (.dv) contiene una secuencia de imágenes que se repiten para cada intervalo de tiempo siguiendo el orden: canal 1 en foco, canal 2 en foco, canal 3 desenfocado, canal 1 desenfocado, canal 2 desenfocado y canal 3 en foco (Figura 6).

OBJETIVO 2. DESARROLLAR UNA APLICACIÓN QUE PERMITA VISUALIZAR, PROCESAR Y SELECCIONAR IMÁGENES DE HCS.

Las figuras que ilustran la selección y visualización de imágenes procedentes de equipos HCS (Figura 7, Figura 21, Figura 23, Figura 24 y Figura 25) se obtuvieron de un experimento perteneciente a estudios en curso no publicados, por lo que se omite toda información sensible relativa a su contenido. Las imágenes muestran células adherentes fijadas, con los núcleos teñidos con Hoechst 33342 (Thermo Fisher Scientific, Waltham, MA) en el canal 1 y una proteína de unión al ADN revelada por inmunofluorescencia indirecta con Cy3 (Jackson ImmunoResearch, West Grove, PA) en el canal 2.

La captura se realizó en el equipo Opera configurado de la siguiente manera: objetivo 40X, 0.9 AN, *water*, WLSM. Formato de captura 678x499, binning 2x2. Tamaño de pixel: 0.32251 micras. Se capturaron dos canales en 48 posiciones de cada uno de los 60 pocillos centrales de una placa de 96. El resultado fueron 2880 archivos .flex de 1.337 MB cada uno (un total de 3.67 GB).

Los dos canales se capturaron secuencialmente utilizando el filtro de triple banda, el filtro de UV y un binning de 2x2, el resto de la configuración de canales utilizada es la siguiente:

1- Canal 1: excitación con lámpara de Xenon, captura en cámara CCD sensicam, 0.04 segundos de exposición.
2- Canal 2: excitación con láser 562 a máxima potencia, captura en cámara pixelfly, 2 segundos de exposición.

OBJETIVO 3. DESARROLLAR UNA APLICACIÓN QUE PERMITA FUSIONAR MOSAICOS REGULARES E IRREGULARES EN SERIE.

La figura que ilustra la generación de mosaicos irregulares en serie (Figura 26), se obtuvo de un experimento perteneciente a estudios en curso no publicados, por lo que se omite toda información sensible relativa a su contenido. Las imágenes muestran tejidos de ratón fijados, con los núcleos teñidos con Hoechst 33342 (canal 1) y una proteína de unión al ADN marcada por inmunofluorescencia indirecta utilizando un anticuerpo secundario marcado conAlexa Fuor 488 (Thermo Fisher Scientific, canal 2).

La captura se realizó usando la aplicación iMSRC en el equipo Leica TCS SP5 X configurado de la siguiente manera:

- Primera captura: objetivo 10X, 0.4 HC PLA APO CS. Formato de captura 256x256, *zoom* 1. Tamaño de píxel 6.08 micras. 600 Hz bidireccional. 195 campos con un 10% de solapamiento, un único plano, *pinhole* completamente abierto (8.83 Airy- sección óptica de 7.5 micras). Autofoco predictivo y convencional asistidos por *software*. Un único canal con la siguiente configuración: excitación con láser 405 al 13 %. Detección con fotomultiplicador entre 409-477 nm, con la ganancia en 522 V y offset en -0.09 %.
- Segunda captura: objetivo 40X, 1.25 AN, *oil*, HCX PLA APO CS. Formato de captura 1024x1024, *zoom* 2. Tamaño de píxel: 189.39 nm. 600 Hz bidirectional. Autofoco asistido por *software*. 577 campos con un 10% de solapamiento, 7 planos separados por una micra, *pinhole* 1 Airy (sección óptica de 0.97 micras), 2 canales capturados secuencialmente por pila, cada uno con la siguiente configuración:
    - Canal 1: excitación con láser 405 al 72%. Detección con fotomultiplicador entre 409-477 nm, con la ganancia en 617 V y offset en -0.71%.
    - Canal 2: excitación con láser 488 al 35%. Detección con fotomultiplicador entre 498-468 nm con la ganancia en 733 V y offset en -4.7%. Se promediaron dos imágenes.

Las coordenadas de la segunda captura se obtuvieron de la siguiente manera: 1- un patólogo determinó las áreas a capturar utilizando mosaicos de campo claro de cortes de tejidos adyacentes, 2- estas áreas se seleccionaron (utilizando ImageJ) sobre el mosaico de búsqueda, 3- el resto de la imagen se pintó de negro, y 4- la imagen resultante se analizó con la rutina de análisis de iMSRC, configurada para generar mosaicos irregulares, segmentando los píxeles entre los valores 1 y 255, sin límite de tamaño ni circularidad (ver nota al pie en la página 28).

OBJETIVO 4. CAPTURA INTELIGENTE DE CÉLULAS CANCEROSAS CIRCULANTES.

La extracción de células cancerosas de muestras de sangre se realiza con un sistema IsoFlux (Fluxion Biosciences Inc, South San Francisco, CA, USA) y la caracterización del origen epitelial se realiza por inmunofluorescencia, mediante un ensayo (Circulating Tumor Cell Enumeration Kit , de la misma compañía) basado en detectar células nucleadas positivas para citoqueratina (CK) y negativas para CD-45[129].

Los protocolos detallados pueden encontrarse en Harb et al. (2013). Resumidamente, se extraen las células de una muestra de sangre y se incuban con una solución de esferas magnéticas recubiertas de un polímero que contiene anticuerpos contra EpCAM (molécula sobreexpresada en carcinomas). A continuación, se deja fluir la muestra a través de un microcapilar que separa las células atraídas por un campo magnético. Finalmente, el resultado se incuba con anticuerpos contra CD-45 (revelado con el secundario Cy3) y CK (directamente conjugada a FITC), se tiñe para Hoechst 33342 y se deposita en uno de los pocillos de un portaobjetos de un total de 8 (μ-Slide 8 Well, ibiTreat: #1.5 polymer coverslip, tissue culture treated, sterilized; ibidi GmbH, Martinsried, Alemania). CK es un filamento característico de células epiteliales y CD-45 un marcador de la línea hematopoyética nucleada, por lo que el recuento de células CK positivas y CD-45 negativas se considera un buen indicador de células cancerosas de origen epitelial en estas muestras. En los controles se aíslan 10-15 células EpCAM positivas y se mezclan con unas 4000 células procedentes de muestras de sangre de pacientes sanos.

La figura que ilustra la captura automatizada de imágenes de células cancerosas circulantes (Figura 33), se obtuvo de un experimento perteneciente a estudios en curso no publicados, por lo que se omite toda información sensible relativa a su contenido. Se capturó una muestra (procesada tal y como se ha expuesto en el párrafo anterior) cada vez, usando la aplicación iMSRC en el equipo Leica TCS SP5 X, configurado de la siguiente manera:

- Primera captura: objetivo 20X, 0.7 AN, imm, HC PL APO, usando aceite como medio de inmersión. Formato de captura: 512x512, *zoom* 1. Tamaño de píxel 1.52 micras. 600 Hz bidireccional. Autofoco predictivo asistido por software. 240 campos con un 10% de solapamiento, 3 planos en el volumen total, *pinhole* 3 AU (sección óptica de 8.2 micras), 2 canales capturados secuencialmente por pila, cada uno con la siguiente configuración:
    - Canal 1: excitación con láser 405 al 5% y 561 al 50%. Detección con un fotomultiplicador entre 423-475 nm, con la ganancia en 774 V y

offset en -0.54 %, y otro fotomultiplicador entre 588-633 nm, con la ganancia en 656 V y offset en -1.61 %.

  - Canal 2: excitación con láser 488 al 35%. Detección con fotomultiplicador entre 503-549 nm, con la ganancia en 705 V y offset en -0.47%.

- Segunda captura: objetivo 63x, 1.4 AN, oil, HCX PL APO CS PH3. Formato de captura: 1024x1024, *zoom* 2. Tamaño de píxel: 120.25 nm. 400 Hz bidireccional. Tantos planos como fuera necesario a *pinhole* 1 AU (sección óptica de 0.78 micras), 3 canales capturados secuencialmente por pila, cada uno con la siguiente configuración:
  - Canal 1: excitación con láser 405 al 28%. Detección con fotomultiplicador entre 410-481 nm, con la ganancia en 703 V y offset en -1.1%. Se promediaron dos imágenes.
  - Canal 2: excitación con láser 488 al 40%. Detección con detector híbrido entre 505-533 nm con la ganancia en 92%. Se promediaron dos imágenes.
  - Canal 3: excitación con láser 561 al 30%. Detección con fotomultiplicador entre 571-610 nm con la ganancia en 830 V y offset en -2.59%. Se promediaron dos imágenes.

Los perfiles de captura se optimizaron mediante la realización de espectros de excitación y emisión de cada uno de los fluoróforos sobre células positivas para solo uno de ellos. Se comprobó la ausencia de cruce entre los fluoróforos Hoechst 33342 y Cy3 en el primer canal de los perfiles de primera captura. Se incluyó un control de tinción con cada experimento. En este se realizó la tinción con Hoechst y la inmunofluorescencia, en paralelo a la muestra experimental, utilizando los mismos reactivos, sobre células MDA MB 231. Se descartaron las muestras experimentales cuyo control presentaba problemas de fijación o un patrón de tinción anómalo.

OBJETIVO 5. CAPTURA INTELIGENTE DE CÉLULAS MADRE CANCEROSAS.

Las figuras que ilustran la captura automatizada de cursos temporales de células aisladas en micropocillos (Figura 11, Figura 37 y Figura 38) se obtuvieron de muestras pertenecientes a estudios ya publicados[130]. El lector interesado en los detalles de los procedimientos experimentales no especificados en esta tesis (síntesis de microchips, aislamiento de células madre cancerosas, caracterización de la señal autofluorescente, etc.) podrá encontrarlos en el artículo original (Miranda-Lorenzo, I. *et al*.[2014]).

Brevemente, las células se sembraron sobre un chip de 4600 micropocillos de 80 micrómetros de diámetro cada uno, alojado en una placa de 12 pocillos.

La señal del biomarcador se caracterizó mediante la realización de espectros de excitación y emisión de células positivas y negativas para la misma.

Las imágenes de la Figura 11: D y E, la Figura 37 y la Figura 38 se capturaron usando la aplicación iMSRC en el equipo Leica TCS SP5 X, configurado de la siguiente manera:

- Primera captura: objetivo 10X, 0.4 AN, HC PLA APO CS. Formato de captura: 512x512, *zoom* 1. Tamaño de píxel: 3.03 micras. 400Hz bidireccional. Autofoco predictivo asistido por *software*. 35 campos con un 2% de solapamiento, 1 plano, *pinhole* 3 AU (sección óptica de 16.469 micras), 2 canales capturados simultáneamente, cada uno con la siguiente configuración:
    - Canal 1: orientado a la captura del contorno de los pocillos. Excitación con láser 561 al 10%. Detección con fotomultiplicador entre 590-700 nm con la ganancia en 862 V y offset en 0%.
    - Canal 2: orientado a la captura del biomarcador. Excitación con láser 488 al 10%. Detección con detector híbrido entre 500-550 nm con la ganancia en 120 %.

  Se comprobó la ausencia de cruce de señal entre ambos canales en muestras de células sin micropocillos y micropocillos sin células.
- Segunda captura: objetivo 63x, 1.4 AN, *oil*, HCX PL APO CS PH3. Formato de captura: 512x512, *zoom* 1.2. Tamaño de píxel: 390.63 nm. 400 Hz bidireccional. Se capturaron 100 campos cada 10 minutos durante 72 horas. Autofoco asistido por *software*. 1 plano, *pinhole* 2 Airy (sección óptica de 1.659 micras), 2 canales capturados simultáneamente, cada uno con la siguiente configuración:
    - Canal 1: excitación con láser 488 al 10%. Detección con fotomultiplicador entre 496-547 nm con la ganancia en 1146 V y offset en -24.7%, y detector de luz transmitida con la ganancia a 10 V y offset en -6.6 %.
    - Canal 2: excitación con láser 561 al 10%. Detección con fotomultiplicador entre 572-694 nm con la ganancia en 1128 V y offset en -1.49%.

OBJETIVO 6. CAPTURA INTELIGENTE DE *TISSUE MICROARRAYS*.

Las figuras que ilustran la captura automatizada de TMAs se obtuvieron de experimentos pertenecientes a estudios publicados (Figura 5) y no publicados (Figura 40 y Figura 45).

En el primer caso, se remite a los lectores interesados en los detalles experimentales a las publicaciones originales (ver Alonso-Curbelo et al.[2014] y Carro et al.[2015]). Brevemente, una mezcla de tejidos sanos y de diferentes tipos y estadios de cáncer se obtuvieron de modelos injertados en ratones, se sometieron a un protocolo de doble inmunofluorescencia para una diversidad de anticuerpos primarios, se revelaron con Alexa Fluor 488 y Alexa Fluor 555 (Thermo Fisher Scientific) y se tiñeron con DAPI (Thermo Fisher Scientific). La totalidad de las piezas de tejido se capturaron automáticamente usando el equipo Leica TCS SP5 X e iMSRC. Para hacerlo, se utilizaron unos perfiles de primera captura de bajos aumentos (objetivo 10X seco, 1 canal y una sección) y unos perfiles de segunda captura de altos aumentos (objetivo 40X aceite, 3 canales y 10 secciones).

En el segundo caso, se omite toda información sensible relativa al contenido de las figuras. Las imágenes muestran tejidos fijados y teñidos con Hoechst 33342 y doble hibridación fluorescente *in situ* utilizando SpectrumGreen (canal 2) y SpectrumOrange (canal 3) como fluoróforos (ambos de Abbott Laboratories, Abbott Park, IL, USA). La captura se realizó usando la aplicación iMSRC configurada para obtener un máximo de 5 imágenes en cada pieza de tejido, en el equipo Leica DMI6000B, configurado de la siguiente manera:

- Primera captura: objetivo 10X, 0.4 AN, HC PL APO CS. Formato de captura: 336x256 (binning 4), intensidad de iluminación máxima, tamaño de pixel: 3.7 micras. Autofoco predictivo y convencional asistidos por *software*. Se capturaron 391 campos con un 10% de solapamiento, un único canal y plano por campo: filtro L5, 225 ms de exposición, ganancia: 255, paleta de falso color (en adelante, LUT) azul.
- Segunda captura: objetivo 63x, 1.4 AN, *oil*, HCX PL APO PH3. Formato de captura: 1344x1024 (sin binning), intensidad de iluminación máxima, tamaño de pixel: 0.146 micras. Autofoco predictivo y convencional asistidos por *software*. Se capturaron 5 planos focales y tres canales en un total de 396 posiciones distribuidas en 80 piezas de tejido. El resultado fueron 5940 archivos .tif de 2.6 MB cada uno (15.4 GB en total). La configuración de canales fue la siguiente:
    - Canal 1: filtro A4, 4.30 ms de exposición, sin ganancia, LUT azul.

- Canal 2: filtro L5, 82.70 ms de exposición, ganancia: 27.3, LUT verde.
- Canal 3: filtro N21, 30.46 ms de exposición, ganancia: 31, LUT rojo.

RESULTADOS

<<In theory, there is no difference between theory and practice. But, in practice, there is.>>

Jan L.A. van de Snepscheut.

Se incluye en los apéndices el código completo de cada una de las rutinas diseñadas. Se ha intentado minimizar la referencia al código a lo largo de este trabajo, prefiriendo un abordaje funcional y priorizando el uso de diagramas de flujo y figuras en su lugar. La Figura 12 muestra la simbología utilizada en los diagramas de flujo de las diversas aplicaciones.

Figura 12. Simbología utilizada en los diagramas de flujo de las distintas aplicaciones. Las flechas de flujo continuas simbolizan el orden de ejecución de los distintos pasos de un algoritmo. Las flechas discontinuas, hacia o desde un directorio, indican escritura o lectura de datos, respectivamente; y hacia o desde un archivo de datos indican creación/modificación o uso del mismo, respectivamente. El cuerpo del archivo de datos indica su contenido, y su cabecera el tipo de archivo (.tif, .txt, ROI Manager o ventana de resultados). El cuerpo del directorio indica el nombre del mismo; un directorio temporal se genera durante la ejecución de una aplicación y se elimina al finalizar la misma. El cuerpo de la interfaz gráfica de usuario indica su nombre y orden de aparición.

OBJETIVO 1. DESARROLLAR UNA APLICACIÓN QUE PERMITA PROCESAR PILAS DE IMÁGENES PROCEDENTES DE EXPERIMENTOS DE VIDEOMICROSCOPÍA MULTICANAL Y MULTIFOCAL.

Se desarrolló una única aplicación, denominada DeltaVision Stack Manager, para llevar a cabo los objetivos propuestos. Esta se diseñó para usuarios sin conocimientos especializados en procesado de imágenes, permitiéndoles procesar archivos de experimentos de videomicroscopía del equipo DeltaVision.

DeltaVision Stack Manager convierte, de forma automática, una pila con imágenes dentro y fuera de foco, de varios canales, a 16-bit y en escala de grises (Figura 6: A-G), en un conjunto de pilas de imágenes enfocadas, con la combinación de canales, colores, contraste y profundidad, a gusto del usuario (Figura 6: H-K).

La Figura 13 muestra el diagrama de flujo de la aplicación. El código completo puede encontrarse en el apéndice.

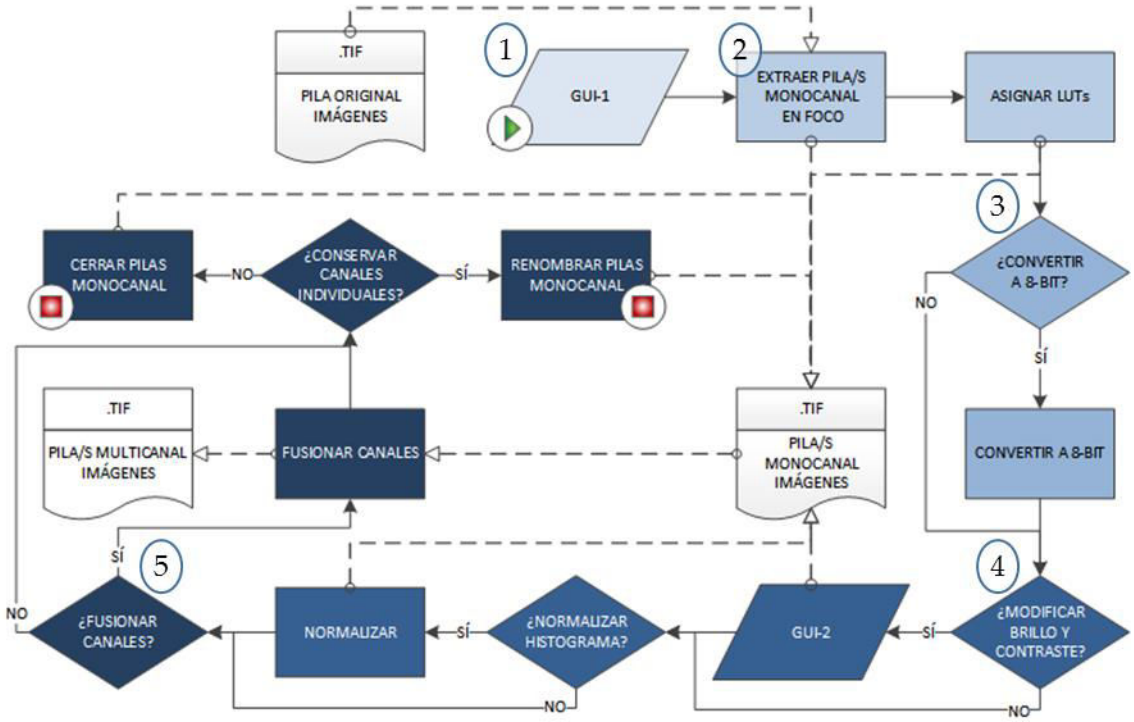


Figura 13. Diagrama de flujo de DeltaVision Stack Manager. Esta aplicación procesa automáticamente una pila de imágenes procedente de un experimento de videomicroscopía. Permite extraer focos y canales, modificar su profundidad, brillo y contraste, asignar colores y fusionar los canales a gusto del usuario. La pila de imágenes original deber ser abierta manualmente por el usuario. **1**: GUI-1 permite que la aplicación pueda adaptarse a experimentos de varios canales, planos focales, etc. **2**: extracción de pilas monocanal y asignación de LUTs. **3**: reducción de la profundidad de las imágenes. **4**: Modificación de brillo y contraste. **5**: Fusión de canales.

La aplicación DeltaVision Stack Manager debe ejecutarse sobre una pila de imágenes procedente de un archivo .dv, abierta previamente de forma manual. Para hacerlo, se pueden usar los módulos DeltaVision Opener o OME Bio-Formats. La secuencia de imágenes de la pila, una vez abierta, es poco práctica, pero conocida. Cada una de las imágenes de cada canal, plano focal y secuencia temporal, ocupan una posición determinada, que se repite periódicamente (Figura 6: A-G). La aplicación se basa en un algoritmo, como se verá más adelante, que permite extraer pilas informativas a partir de la información introducida por un usuario.

DeltaVision Stack Manager es autoexplicativa. Al ejecutarse, se genera una interfaz gráfica de usuario (en adelante GUI, de sus siglas en inglés: *Graphical User Interface*, Figura 13: 1 y Figura 14) que contiene texto con las instrucciones de uso, casillas de verificación, campos numéricos rellenables, menús desplegables, un botón *OK* y otro *Cancel*.

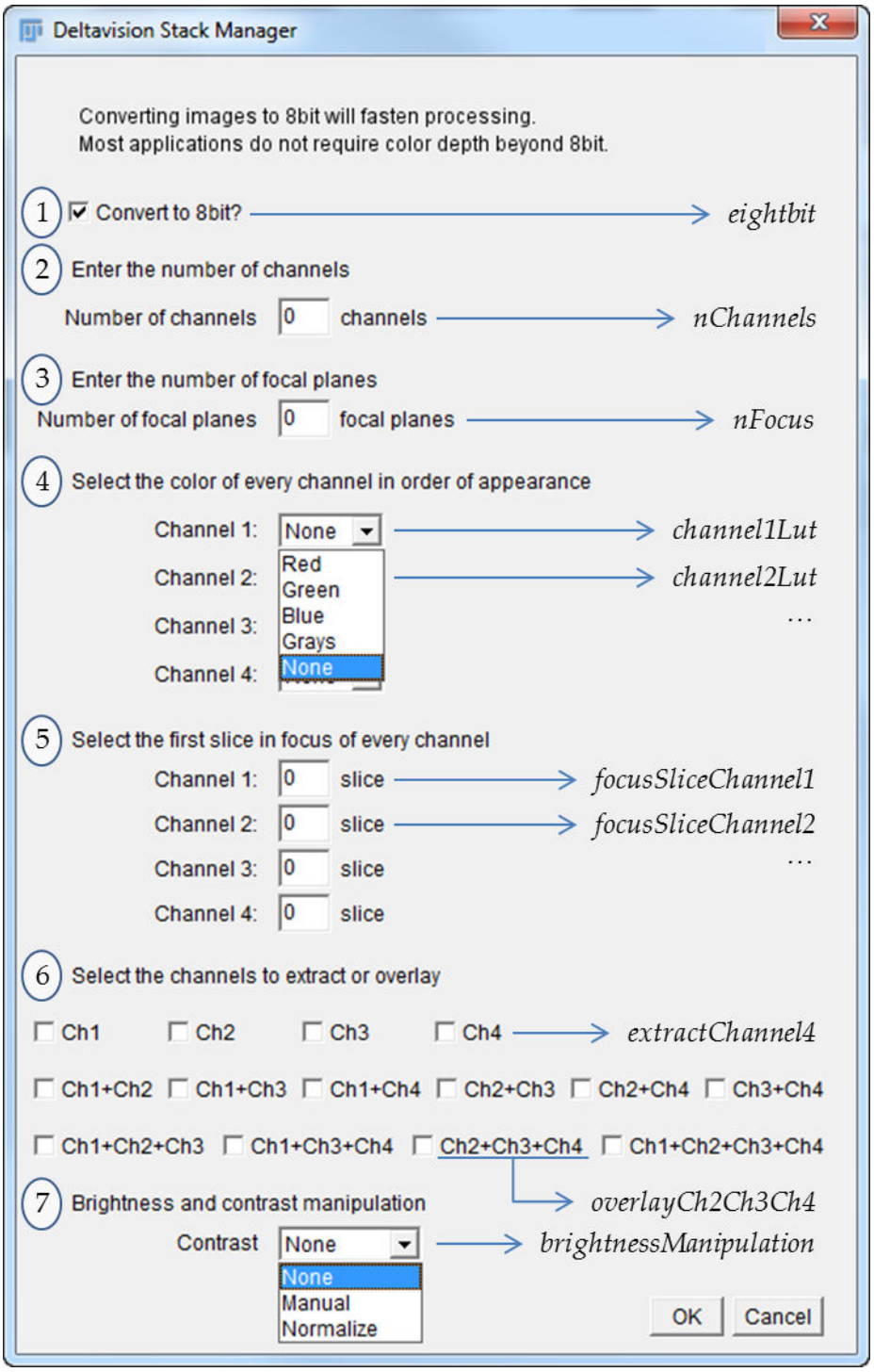


Figura 14. Primera interfaz gráfica de usuario de DeltaVision Stack Manager. La aplicación utilizará los valores asignados por el usuario a las casillas de verificación (**1** y **6**), los campos numéricos rellenables (**2**, **3** y **5**) y los menús desplegables (**4** y **7**), para procesar automáticamente la pila de imágenes procedente de un experimento del equipo Deltavision. Las flechas apuntan a las variables a las que se asignarán los valores. Los puntos suspensivos indican elementos que se repiten, en 6 se muestran dos ejemplos del total de 14 variables. Los menús multiopción se muestran desplegados

El usuario debe rellenar los campos necesarios, seleccionar las opciones y casillas adecuadas en función de su diseño experimental y presionar *OK* para continuar la ejecución.

Todos los campos de la interfaz gráfica de usuario se asignan a variables: cuando se selecciona una casilla de verificación (Figura 14: 1 y 6), se asigna el valor 1 (*true*) a una variable lógica, cuando se introduce un número entero en un campo numérico (Figura 14: 2, 3 y 5), este se asigna a una variable numérica, y cuando se selecciona una opción de un menú desplegable (Figura 14: 4 y 7), esta se asigna a una variable tipo cadena de caracteres.

El usuario debe dejar seleccionada la primera casilla de verificación (Figura 14: 1), si desea reducir la profundidad de las imágenes a 8 bit (Figura 13: 3); el valor de la selección se asignará a la variable *eightbit*. En los campos numéricos (Figura 14: 2, 3 y 5), el usuario debe introducir el número de canales, planos focales y la primera imagen de la pila en que un determinado canal aparece por primera vez en foco; la aplicación asignará los valores introducidos a las variables *nChannels*, *nFocus* y *focusSliceChannel1, 2, 3* y *4*, respectivamente. En los menús desplegables (Figura 14: 4 y 7), se debe seleccionar la paleta de falso color que se desea asignar a cada canal (Figura 13: 2) y un método de manipulación de brillo y contraste, respectivamente (Figura 13: 4); las opciones escogidas se guardarán en las variables *channel(1-4)Lut* y *brightnessManipulation*. El último grupo de casillas de verificación (Figura 14: 6) permiten al usuario seleccionar las pilas de imágenes monocanal a conservar, así como las combinaciones de canales a fusionar en una única pila multicanal (Figura 13: 5); la selección realizada por el usuario se asigna a las variables *extractChannel1-4* y *overlayCh1Ch2*, *overlayCh1Ch3*, *overlayCh1Ch2Ch3*, etc.

Al pulsar el botón *OK*, se extraen todas las pilas monocanal en foco de la pila original (Figura 13: 2). Para hacerlo, se alimenta el comando Make Substack con las variables *nChannels*, *nFocus* y *focusSliceChannel1, 2, 3* y *4*. Siendo *numSlices* una variable que codifica el número de imágenes de la pila original (a la cual se le asigna el valor de la función nSlices) y *periodicity* una variable que codifica el resultado de multiplicar *nChannels* por *nFocus*, entonces, la sentencia:

run("Make Substack...", "slices="+*focusSliceChannel1*+"-"+*numSlices*+"-"+*periodicity*+"") (Figura 15: A),

genera una pila de imágenes con las imágenes en foco del canal 1 (Figura 15: B). Posteriormente (Figura 13, 2: asignar LUT), a esta se le asigna la paleta de falso color codificado en la variable *channel1Lut*. Este procedimiento se repite para cada canal. La condición *focusSliceChanne1* > 0) garantiza que el canal (en este caso el 1), existe. Las variables *channel(1-4)Lut* se inicializan con el valor de las paletas de falso color, una vez aplicadas estas, se les reasigna un valor distinto para su uso posterior (el nuevo valor será el mismo que el primero en minúsculas, excepto para el valor "Grays", que se corresponde con "gray", Figura 15: A). A continuación, si la primera de las casillas de verificación de la interfaz (Figura 14: 1) se dejó seleccionada (valor por defecto), se ejecuta el comando 8-bit sobre la pila de imágenes monocanal (Figura 13: 3 y Figura 15:

A). Al reducir la profundidad de la pila de imágenes de 16 a 8 bit, se genera una pila mucho más rápida de procesar y que ocupa menos espacio al guardarse.

A

```
if(focusSliceChannel1>0)
{
    selectImage(id1);
    run("Make Substack...", "slices="+focusSliceChannel1+"-"+numSlices+"-"+periodicity+"");
    run(""+channel1Lut+"");
    if (channel1Lut == "Grays"){channel1Lut = "gray";}
    else {channel1Lut = toLowerCase(channel1Lut);}
    rename(Title+"-"+channel1Lut);
    run("Brightness/Contrast...");
    resetMinAndMax();
    if (eightbit == 1){run("8-bit");}
}
```

B

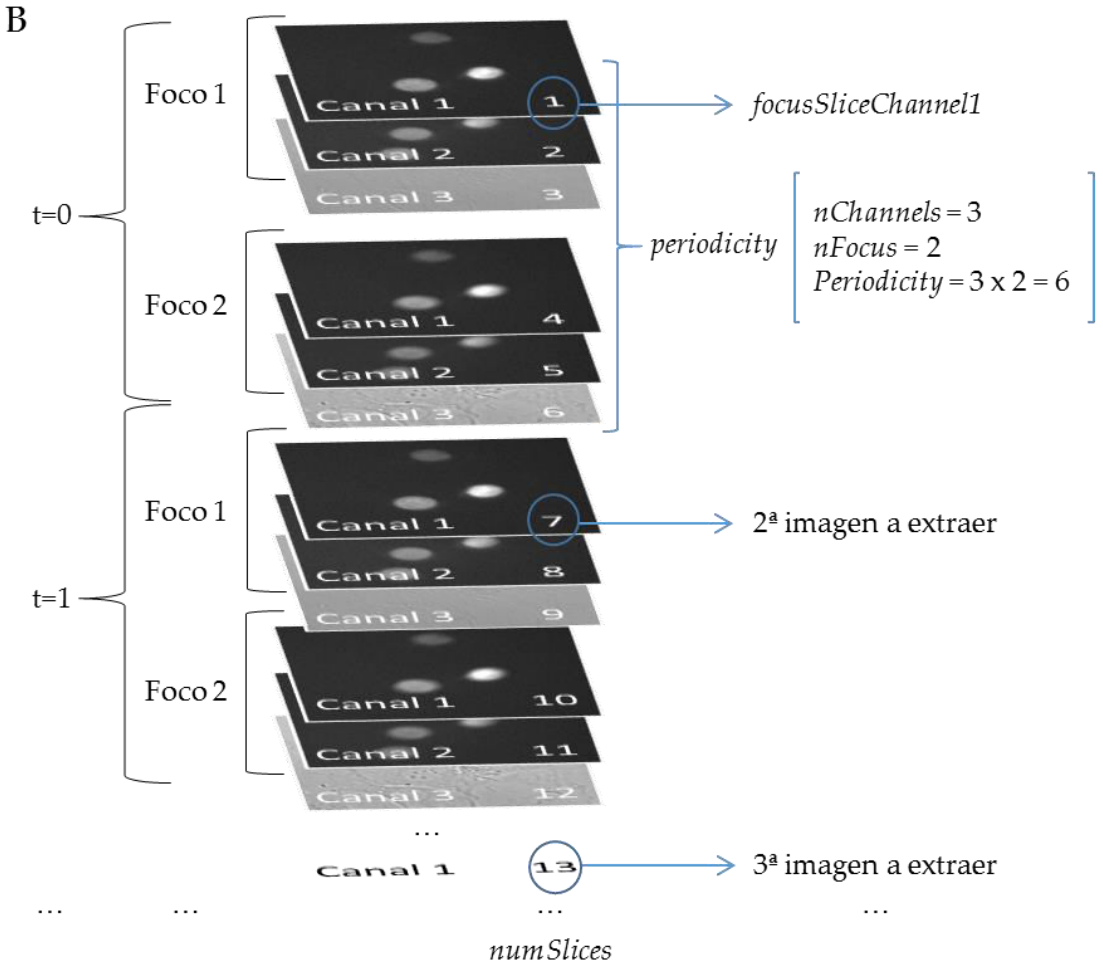


Figura 15. Algoritmo para la extracción de las pilas de imágenes en foco de cada canal, a partir de la pila original de imágenes desordenadas (id1). **A**: código para focusSliceChannel1. El mismo código se repite para focusSliceChannel2, 3 y 4, en estos casos, channel1Lut, pasa a ser channel2Lut, channel3Lut o channel4Lut, respectivamente. **B**: pila original de imágenes. Para cada canal seleccionado por el usuario (*focusSliceChannel1-4*>0), se genera una pila de imágenes que es el resultado de seleccionar la primera imagen en foco en la pila original (focusSliceChannel1-4) y las siguientes con un incremento (*periodicity*). *numSlices*: número de imágenes en la pila original. t=0 y t=1: imágenes capturadas en el primer y segundo ciclos de captura, respectivamente. Foco 1: plano focal en que las imágenes fluorescentes están en foco. Foco 2: plano focal en que las imágenes de campo claro están en foco. El número en la parte inferior derecha de las imágenes indica la posición en la pila original. Canal 1, 2 y 3: canales para un marcador

fluorescente rojo, verde, y campo claro, respectivamente. Los puntos suspensivos indican elementos que se repiten.

Seguidamente, se realiza la manipulación de brillo y contraste de las pilas monocanal en foco (Figura 13: 4). Si el valor de la variable *brightnessManipulation* es *Manual*, se deshabilita el comando batch mode de ImageJ, aparece una ventana emergente con el texto "Adjust Brightness and Contrast if necessary, then click OK" y se pausa la ejecución de la aplicación, permitiendo al usuario interaccionar libremente con el entorno (Figura 13, 4: GUI-2). La aplicación continúa una vez el usuario pulsa el botón *OK*. Si el valor de la variable *brightnessManipulation* es *Normalize*, se seleccionan secuencialmente cada una de las pilas monocanal en foco y se ejecuta el comando Enhance Contrast de ImageJ, produciéndose la normalización del histograma de cada pila de imágenes. Si el valor de la variable *brightnessManipulation* es *None*, no se modifica el brillo y contraste de las pilas de imágenes.

Una vez extraídas las pilas monocanal en foco con los niveles de gris modificados, se fusionan los canales utilizando el comando Merge Channels de ImageJ (Figura 13: 5). Para cada una de las variables *overlayCh1Ch2*, etc. (Figura 14, 6: Ch1+Ch2, Ch1+Ch3, etc.), una cadena de sustituciones que utilizan la función *replace* y las variables *channel(1-4)Lut*, generarán la sentencia adecuada para la correcta fusión de canales (Figura 16).

```
if(overlayCh1Ch2 == 1)
{
    string0 = "red=*None* green=*None* blue=*None* gray=*None*";
    string1 = replace(string0, channel1Lut+"=\\*None\\*", channel1Lut+"="+Title+"-"+channel1Lut);
    string2 = replace(string1, channel2Lut+"=\\*None\\*", channel2Lut+"="+Title+"-"+channel2Lut);
    run("Merge Channels...", ""+string2+" create keep");
    run("Stack to RGB", "slices");
    setBatchMode("exit and display");
    selectWindow("Composite");
    rename("Ch1+Ch2");
}
```

Figura 16. Código responsable de la fusión de los canales 1 y 2. La variable *overlayCh1Ch2* (a la que se asigna el valor de la casilla de verificación seleccionada por el usuario en la primera interfaz gráfica, Figura 14: 6) cumple la condición == 1, desencadenando una serie de sustituciones que alimentan la función Merge Channels. Nótese que el valor original de las variables *channel(1-4)Lut*, se cambió para facilitar la fusión de canales en este paso (ver texto y Figura 15: A).

Finalmente (Figura 13: 5), si el usuario selecciona algún canal sencillo en la interfaz gráfica (Figura 14, 6: Ch1, Ch2, Ch3 y Ch4), la pila que lo contiene se renombra "Ch1-4". En caso contrario, las pilas no seleccionadas se eliminan.

Al finalizar la aplicación, el usuario dispone de un conjunto de pilas de imágenes en foco, coloreadas y contrastadas a su gusto, que contienen su selección personal de canales independientes o combinaciones de canales (Figura 6: H-K) y puede acceder al entorno de FIJI para su posterior análisis.

El funcionamiento de la aplicación se validó comparando sus resultados con los obtenidos mediante el procesado manual de varias pilas de imágenes. Para ello, se utilizaron pilas procedentes de experimentos reales y pilas generadas *ex profeso*, con la intención de abarcar todas las posibles combinaciones de variables (planos focales, número de canales, orden de canales, fusión de canales, etc.).

OBJETIVO 2. DESARROLLAR UNA APLICACIÓN QUE PERMITA VISUALIZAR, PROCESAR Y SELECCIONAR IMÁGENES DE HCS.

Se desarrolló una única aplicación, denominada Opera Exporter, para llevar a cabo los objetivos propuestos. Opera Exporter permite visualizar, simultáneamente, un conjunto de imágenes procedentes del equipo Opera, con sus canales fusionados, y escoger aquellas de interés. La selección realizada por el usuario se guarda en un directorio de su elección, tanto en formato nativo (.flex) como en formato .tif (16-bit o RGB, según si se capturó uno o más de un canal, respectivamente). El proceso es intuitivo, está automatizado y no requiere conocimientos especializados en análisis y procesado de imágenes. La aplicación se basa en el uso de interfaces gráficas y en la nomenclatura propia de las placas multipocillo (ver nota al pie de la página 37).

El código completo de la aplicación puede encontrarse en el apéndice. Las especificaciones técnicas del equipo Opera y las condiciones de captura del ejemplo mostrado en esta sección, pueden encontrarse en el apartado de material y métodos.

La Figura 17 muestra el diagrama de flujo de la aplicación.

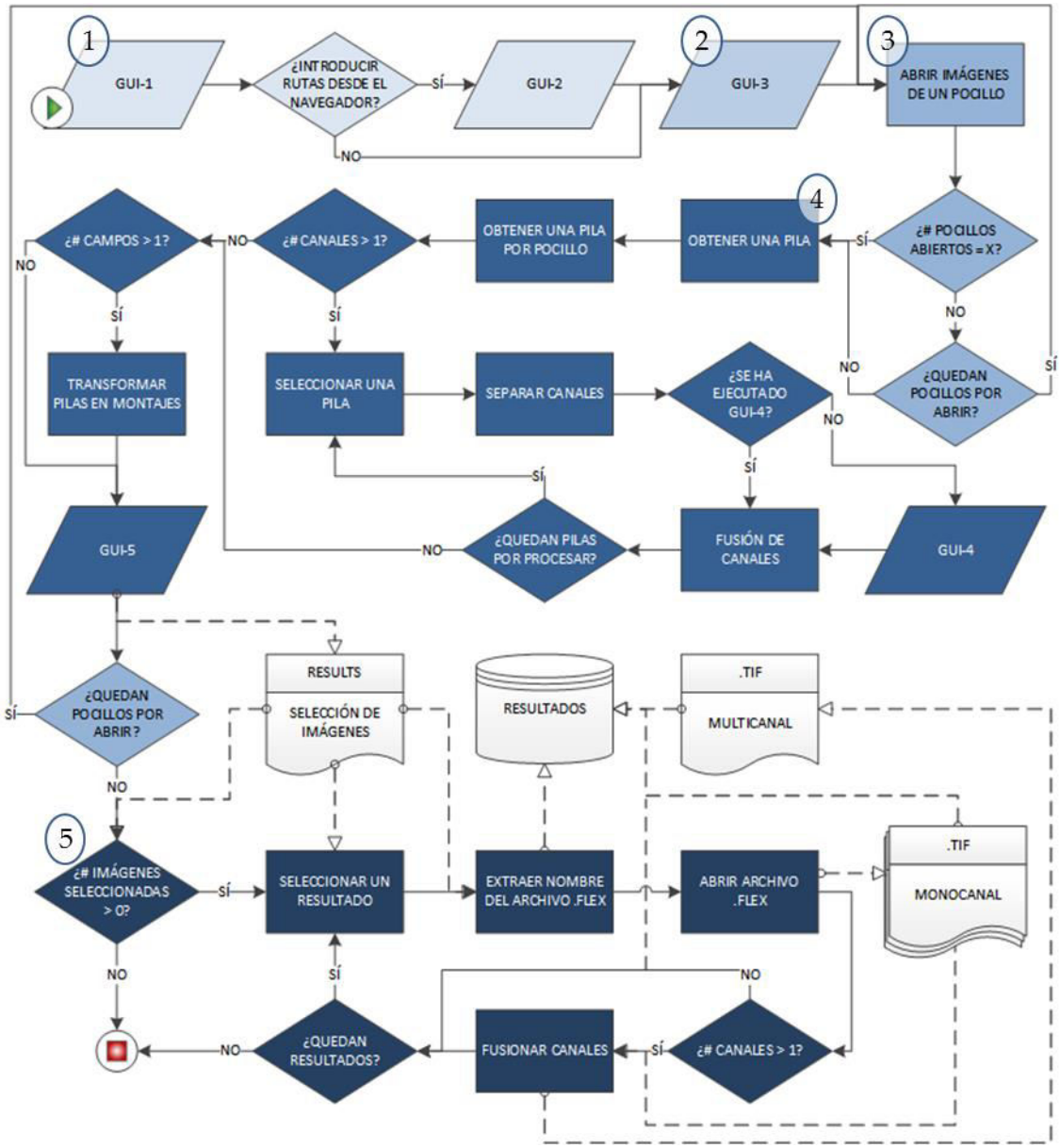


Figura 17. Diagrama de flujo de la aplicación Opera Exporter. Diferentes interfaces gráficas guían al usuario en la visualización y selección de imágenes procedentes del equipo de HCS Opera. Las imágenes seleccionadas se guardan automáticamente, tanto en formato nativo (.flex) como .tif, en el último caso, si se capturó más de un canal, estos se fusionan automáticamente. **1**: Asignación del directorio que contiene las imágenes a visualizar y del directorio donde se desea guardar las imágenes seleccionadas (Resultados). **2**: Selección de los pocillos y campos cuyas imágenes se quieren visualizar. **3**: Bucle de control: permite abrir las imágenes seleccionadas en 2 secuencialmente, en bloques de un máximo de 3 pocillos. "X": número máximo de pocillos a visualizar simultáneamente; vale tres por defecto, pero su valor se puede cambiar modificando el código. **4**: Función *ImagePicker*: encargada de procesar las imágenes abiertas en 3, para su visualización y selección de imágenes de interés. **5**: Código de exportado, procesado y guardado de las imágenes originales. La aplicación corre en batch, al usuario solo se le muestran las interfaces gráficas.

Opera Exporter está diseñada para trabajar sobre colecciones de imágenes .flex, capturadas en placas de 96 pocillos con el equipo Opera (Figura 7: B, C y D). Es autoexplicativa: al ejecutarse se generan una serie de interfaces gráficas que guían al usuario durante todo el proceso. Dichas interfaces se muestran en la Figura 18, Figura 19, Figura 23 y Figura 24.

GUI-1 y GUI-2 (Figura 17: 1) permiten al usuario introducir el directorio que contiene las imágenes y el directorio donde se guardarán las imágenes seleccionadas, GUI-3 (Figura 17: 2) permite especificar el número de campos a visualizar de cada pocillo y GUI-4 (Figura 17: 4) permite determinar los colores para la fusión de canales. GUI-5 (Figura 17: 4) hace uso de toda la información recogida en las interfaces anteriores. Su propósito es mostrar al usuario las imágenes de los campos y pocillos anteriormente seleccionados, en forma de montajes con los canales fusionados, y permitirle seleccionar las que sean de su interés. GUI-5 muestra un máximo de 3 montajes simultáneamente cada vez (uno por pocillo seleccionado), por lo que se crea tantas veces como sea necesario, dependiendo del número de pocillos seleccionados por el usuario (Figura 17: 3). GUI-1, 3 y 5 se crearán siempre, GUI-2 solo se creará si el usuario desea introducir rutas usando un navegador y GUI-4 solo se creará una vez, si existe más de un canal por campo. Finalmente, las imágenes seleccionadas se copian al directorio de salida (en formato propietario) y luego se abren, se fusionan sus canales y se guardan en formato .tif (Figura 17: 5). La aplicación corre en modo batch para aumentar su rendimiento, las imágenes mostradas en las figuras de este apartado (a excepción de las GUIs) son invisibles al usuario y se incluyen con objetivos didácticos.

Al ejecutarse la aplicación, se genera una interfaz gráfica que contiene dos campos de texto y dos casillas de verificación, junto con instrucciones (Figura 18: A y GUI-1 en Figura 17: 1). Los cuadros de texto permiten al usuario introducir el directorio que contiene el conjunto de imágenes a procesar y el directorio donde se desea guardar las imágenes seleccionadas (Resultados en Figura 17: 5). Si se selecciona alguna de las casillas de verificación, se generará una interfaz alternativa en la que el usuario podrá seleccionar dichos directorios mediante un navegador (Figura 18: B y GUI2 en Figura 17: 1). El usuario debe pulsar el botón *OK* para continuar, al hacerlo, los directorios de los archivos .flex y de guardado se asignan a las variables *flexDir* y *outputDir*, respectivamente. Si no se ha asignado alguno de los dos directorios, la aplicación se interrumpe y se muestra un mensaje de error: "No Meas' Directory has been selected, please run the macro again", o "No output's Directory has been selected, please run the macro again"*.

* Este código se reutilizará, con pequeñas modificaciones, para la selección de directorios en las aplicaciones iMSRC Stitching Module y TMA Manager, que se verán más adelante.

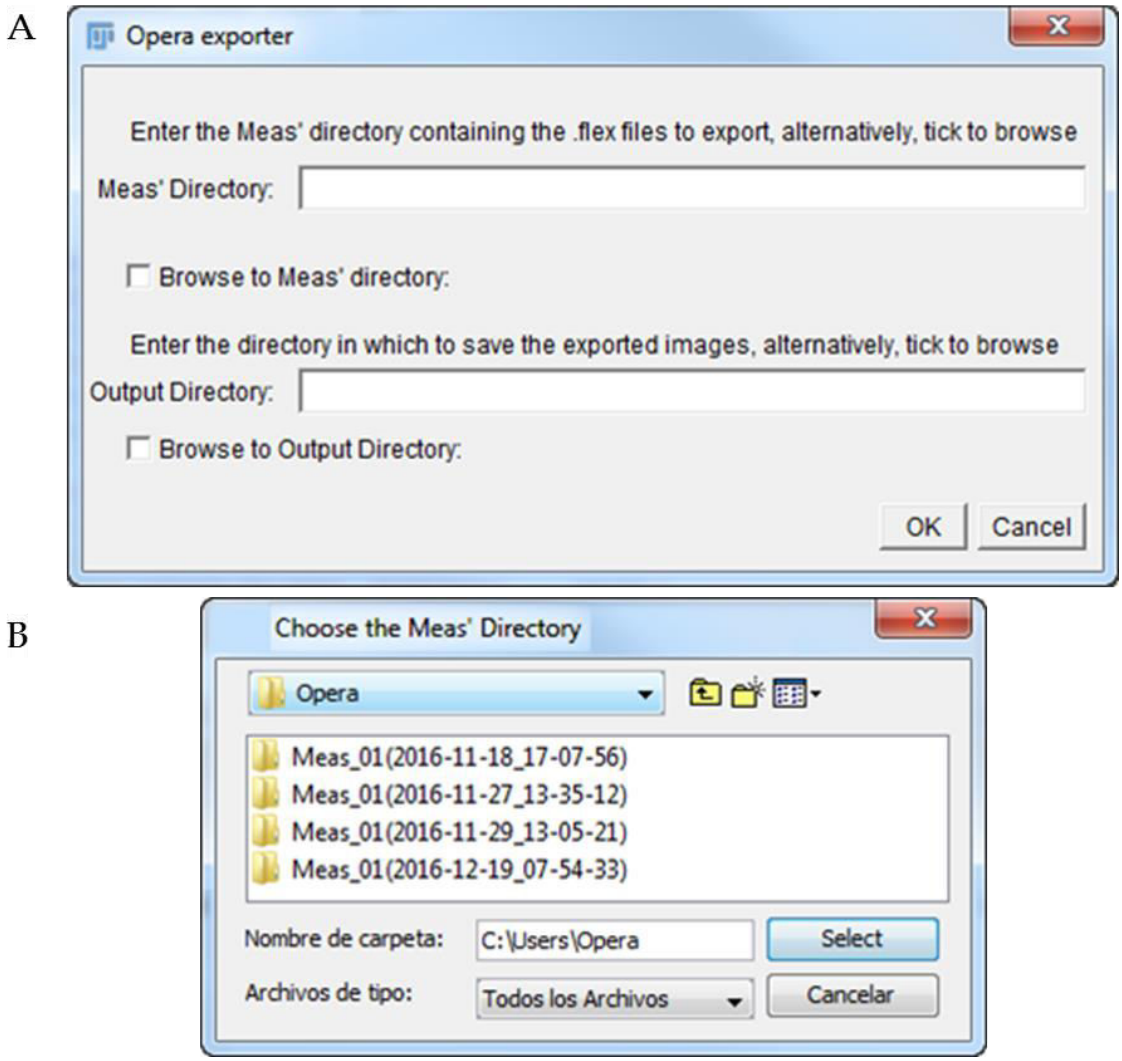
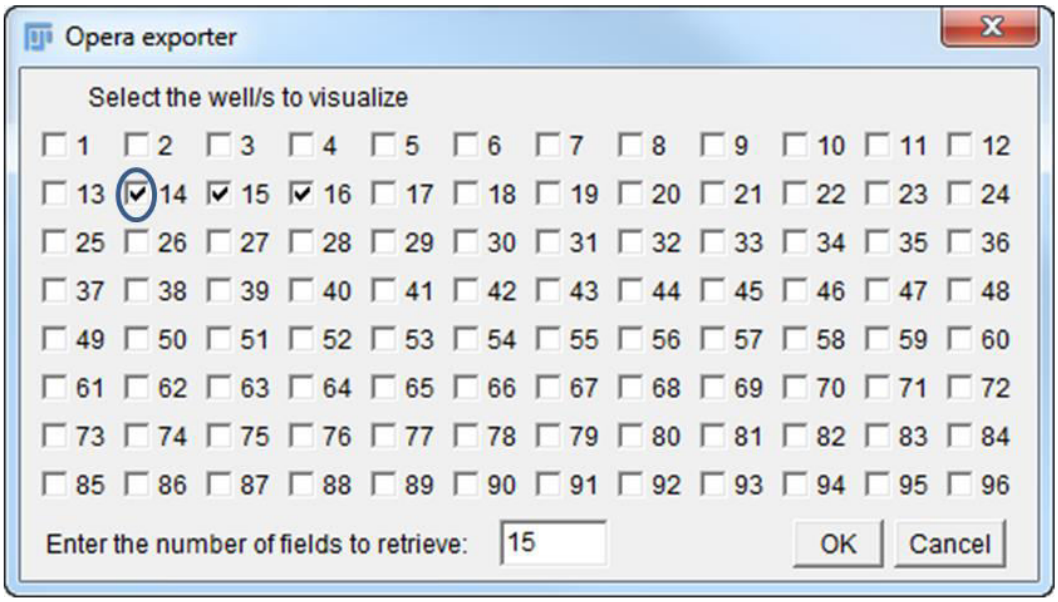


Figura 18. Interfaces gráficas de usuario 1 (A) y 2 (B) de Opera Exporter. Permiten obtener el directorio a las imágenes originales y el directorio en que se guardarán los resultados. GUI-2 (B) aparece si se selecciona cualquiera de las casillas de verificación en GUI-1 (A), facilita introducir directorios mediante un navegador. Interfaces muy similares se mostrarán en las aplicaciones iMSRC Stitching Module y TMA Manager.

A continuación, se muestra una interfaz con 96 casillas de verificación y un cuadro de texto (GUI-3 en Figura 17: 2 y Figura 19: A). Las 96 casillas de verificación siguen la misma distribución que los 96 pocillos de las placas utilizadas para la captura de las imágenes (Figura 7: A), y permiten seleccionar los pocillos cuyas imágenes se quiere visualizar. En el cuadro de texto, el usuario debe introducir el número de campos por pocillo que desea inspeccionar. Su valor se asignará a la variable *nFields*, que permitirá, entre otras cosas, limitar el número de imágenes que se mostrará al usuario, lo cual es útil para disminuir el tiempo de computación en el caso de que el número de campos por pocillo sea muy elevado.

A



B

```
selectedWellsArray=newArray(96);
nSelectedWells=0;
for(i=1;i<97;i++)
{
(1) if(Dialog.getCheckbox()==true)                          → i=14 (true)
    {
(2) selectedWellRow=1+floor((i-1)/12);                      → 2 (1+≈(14-1)/12)
    selectedWellColumn=i-(12*(selectedWellRow-1));          → 2 (14-(12*(2-1))
    if(selectedWellRow<10)
    {selectedWellRow="00"+selectedWellRow;}                 → 002
    else
    {selectedWellRow="0"+selectedWellRow;}                  → 002
    if(selectedWellColumn<10)
    {selectedWellColumn="00"+selectedWellColumn;}
    else
    {selectedWellColumn="0"+selectedWellColumn;}
(3) selectedWellsArray[nSelectedWells]=selectedWellRow+selectedWellColumn;
    nSelectedWells++;                                       → selectedWellsArray(002002,na, …)
    }
    if(i==96)
    {selectedWellsArray=Array.trim(selectedWellsArray,nSelectedWells);}
}                                                           → selectedWellsArray(002002,002003,002004)
```

Figura 19. GUI-3 (**A**) y algoritmo para la extracción del nombre de los pocillos seleccionados por el usuario en el formato de los archivos de Opera (**B**). **A**: GUI-3 permite seleccionar el número de imágenes y pocillos a visualizar. **B**: de cada una de las casillas de verificación seleccionadas en GUI-3 (1), se extrae el nombre del pocillo en el formato de los archivos .flex (2) y se introduce en selectedWellsArray (3). Se muestra el ejemplo para la casilla 14. nSelectedWells contiene el número de pocillos seleccionados. selectedWellsArray se declara e inicializa como un vector de 96 elementos al principio del código, los elementos vacíos se eliminan al final del mismo. na: no asignado.

Al pulsar el botón *OK*, el valor de cada una de las casillas de verificación en GUI-3 se comprueba ordenadamente (de izquierda a derecha y de arriba abajo). Cuando este coincide con su estado seleccionado (*true*, Figura 19: B1), se obtiene el nombre del pocillo siguiendo el formato de los archivos .flex (ver nota al pie en la página 37, Figura

19: B2) y se asigna al vector *selectedWells Array*. Cada vez que esto ocurre, se incrementa en uno el valor de la variable *nSelectedWells*, que contendrá el número de pocillos seleccionados por el usuario (Figura 19: B3).

Los valores de las variables que codifican el directorio a las imágenes (*flexDir*), el nombre de los pocillos seleccionados por el usuario (*selectedWellsArray*) y el número de campos a inspeccionar por pocillo (*nFields*), alimentando un bucle que depende del número de pocillos seleccionados (*nSelectedWells*), permiten abrir, secuencialmente, las imágenes que el usuario desea visualizar, utilizando el comando run("Bio-Formats Importer", …) (Figura 20 y Figura 17: 3). Conviene recordar que las imágenes no serán visibles al usuario hasta el momento en que se cree la siguiente interfaz, dado que la aplicación se está ejecutando en modo batch.

```
for(j=1; j<=nSelectedWells;j++)
{

  for(i=1; i<=nFields;i++)
  {
    if(i<10)
    {run("Bio-Formats Importer", "open=["+flexDir+selectedWellsArray[j-1]+"00"+i+".flex] ...");}
    else if(i>=10 && i<100)
    {run("Bio-Formats Importer", "open=["+flexDir+selectedWellsArray[j-1]+"0"+i+".flex] ...");}
    else if(i>=100 && i<1000)
    {run("Bio-Formats Importer", "open=["+flexDir+selectedWellsArray[j-1]+i+".flex] ...");}
  }
  MAX_WELLS_TO_VISUALIZE=3;
  if((j)%MAX_WELLS_TO_VISUALIZE==0)
  {
    wellsToVisualize=MAX_WELLS_TO_VISUALIZE;
    imagePicker();
  }
  else if(j==nSelectedWells)
  {
    wellsToVisualize= nSelectedWells-floor(nSelectedWells/MAX_WELLS_TO_VISUALIZE)
                                                                    *MAX_WELLS_TO_VISUALIZE;
    imagePicker();
  }
}
```

Figura 20. Parte del código de *Opera Exporter* que permite abrir las imágenes .flex seleccionadas por el usuario en GUI-3. Las imágenes de hasta un máximo de tres pocillos, se abren secuencialmente, utilizando las variables: *nSelectedWells*, *flexDir*, *selectedWellsNamesArray*, *nFields* y *MAX_WELLS_TO_VISUALIZE*. El número máximo de pocillos a visualizar simultáneamente se puede cambiar modificando el valor de la variable *MAX_WELLS_TO VISUALIZE*. Los puntos suspensivos indican omisión de código para facilitar la lectura.

El procesado continuará cada vez que se abran todas las imágenes de un número de pocillos múltiplo de *MAX_WELLS_TO_VISUALIZE* ("X" en Figura 17: 3) o al llegar al número total de pocillos seleccionados por el usuario (Figura 20). Limitar la apertura de imágenes en grupos de pocillos garantiza la eficiencia de Opera Exporter, que podría verse comprometida si tuviera que procesar muchas imágenes a la vez. Se decidió limitar el número de pocillos a visualizar simultáneamente a tres (valor por defecto de *MAX_WELLS_TO_VISUALIZE*) porque: 1- permite que la aplicación funcione de forma satisfactoria aun cuando el número de imágenes por pocillo sea elevado, 2- permite visualizar las imágenes con un tamaño adecuado en un monitor

convencional y 3- se adapta a la mayoría de diseños experimentales (como fruto de nuestra experiencia, el factor más determinante a la hora de decidir el número de pocillos a visualizar simultáneamente suele ser el número de réplicas por experimento, que no suele exceder de tres). Sin embargo, este número puede cambiarse fácilmente, para adaptar la aplicación a cualquier diseño experimental, modificando el valor de la variable *MAX_WELLS_TO_VISUALIZE* en el código.

A continuación (Figura 20) se ejecuta la función *imagePicker**, responsable de la creación de la interfaz de visualización y selección de imágenes (GUI5 en Figura 17: 4). Este es un proceso complejo que implica manipular las imágenes anteriormente abiertas, nombrarlas en un formato intuitivo para el usuario (ver nota al pie de la página 37) y generar una interfaz para la fusión de canales (si hay más de uno, GUI-4 en Figura 17: 4). En primer lugar, las imágenes abiertas se transforman en una única pila de imágenes (Figura 21: A), de la que se extrae un conjunto de pilas con todas las imágenes de cada pocillo (incluye todas las imágenes de todos los campos y canales capturados en un pocillo, Figura 21 :B). Si se capturó más de un canal, cada una de las pilas anteriores se procesará secuencialmente, dividiéndose en un conjunto de pilas por canal y pocillo, que se fusionarán en una pila multicanal (RGB) por pocillo (Figura 21: C). Si solo se capturó un canal, las pilas permanecerán inalteradas (a 16-bit, en escala de grises). Cuando todas las pilas se hayan procesado de esta forma, cada una de las pilas resultantes se transformará en un montaje de imágenes, cada uno de los cuales mostrará todos los campos de un pocillo (Figura 17: 4 y Figura 21: D).

* La función *imagePicker*, no pertenece a la API del lenguaje macro de ImageJ, se declara y define en el código de la aplicación Opera Exporter.

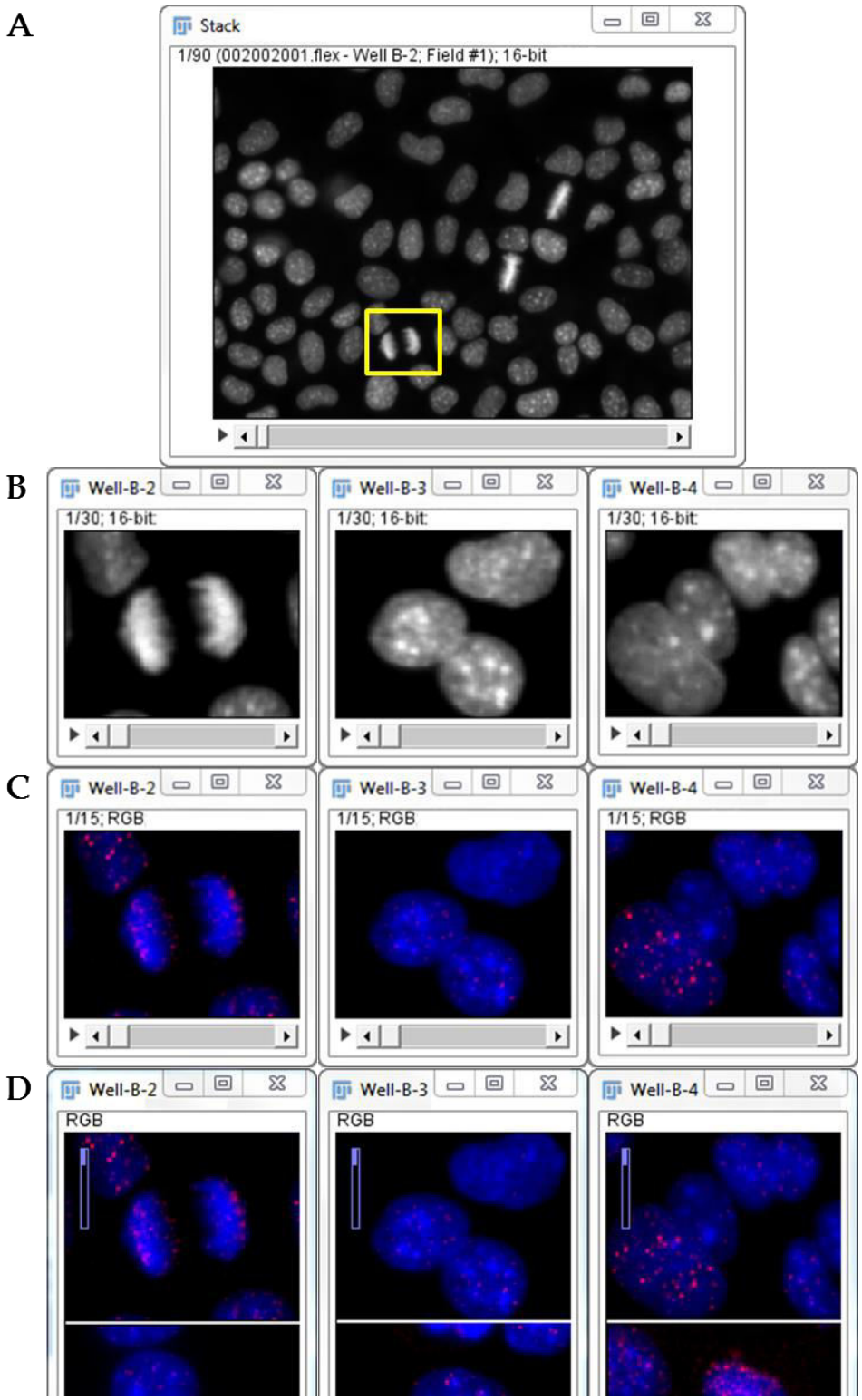


Figura 21. La función *imagePicker* de Opera Exporter transforma un conjunto de imágenes en escala de grises de diferentes canales, campos y pocillos, en un conjunto de montajes de imágenes multicanal (RGB) por pocillo, que serán la base para la formación de GUI-5. Las imágenes abiertas (no mostrado) se transforman en una pila (A) que se divide en pilas por pocillo (B). En cada una de ellas se fusionan los canales (C) antes de convertirlas en montajes (D). A partir de B, las pilas se nombran siguiendo el formato de las placas multipocillo convencionales. La fusión de canales (C) está mediada por GUI-4, solo se realiza si se capturó más de un canal, en caso negativo, los montajes en D permanecen en escala de grises. B, C y D muestran detalles del inserto en A. A, B y C no se muestran al usuario, la función *imagePicker* continuará hasta dar lugar a GUI-5.

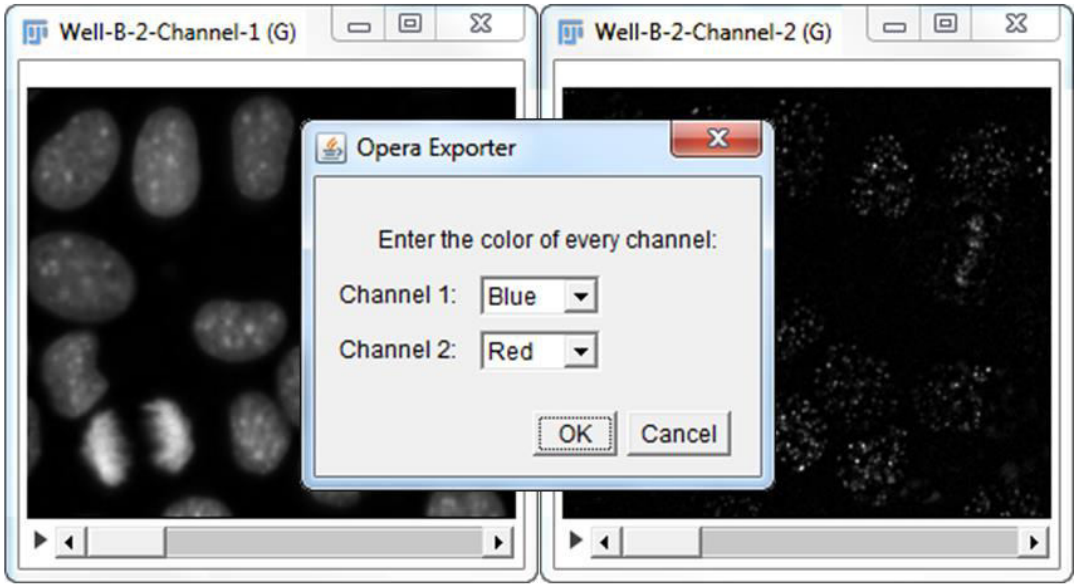
La conversión de las imágenes en una pila (Figura 21: A) es inmediata, gracias al comando Images to Stack de ImageJ. La conversión de la pila de imágenes de diferentes pocillos y canales, en pilas con imágenes de un solo pocillo (Figura 21: B y Figura 17: 4), se realiza alimentando el comando Make Substack de ImageJ con las variables *sliceNumber*, *wellsToVisualize*, *i*, *nFields* y *nChannels* (Figura 22: C), de forma similar a la discutida en la Figura 15. Cada una de estas pilas se nombrará con el nombre del pocillo del que provienen las imágenes, el cual se extrae de los metadatos de la misma.

Figura 22. Obtención de pilas con las imágenes de un pocillo en Opera Exporter. Detalle de las cabeceras de la pila original (A, Figura 21:A) y de la pila que contiene las imágenes del primer pocillo (B, Figura 21: B), que muestran parte de los metadatos, incluyendo el nombre del pocillo del que proceden las imágenes (Well B-2 en B). El código en C permite extraer, de la pila original, pilas con las imágenes de un pocillo (comando Make Substack) y asignarles el nombre del mismo. Se muestra como ejemplo la extracción de la pila con las imágenes del primer pocillo. *wellsToVisualize* contiene el número de pocillos del que está compuesta la pila original, *sliceNumber* contiene el número de imágenes en la pila original, *nFields* el número de campos por pocillo en la pila original y *nChannels* el número de canales. getImageInfo(), permite acceder al nombre del pocillo en los metadatos y asignarlo a la variable *alphanumericName*.

Transformar la pila de imágenes que contiene todas las imágenes de un pocillo con los canales separados, en una pila de imágenes multicanal (RGB) (Figura 21: C), requiere de un proceso más complejo. Este solo se llevará a cabo si el número de canales es mayor a 1 (es decir, 2 ó 3, Figura 17: 4). Cada pila de imágenes procedentes de un pocillo se procesará secuencialmente. En primer lugar, la primera de estas pilas

se descompone en una pila de imágenes por cada canal. A continuación, la primera vez que se ejecute la función *imagePicker*, se generará una interfaz gráfica en la que el usuario deberá seleccionar (entre varias opciones de un menú desplegable) la paleta de falso color a asignar a cada canal (GUI4 en Figura 17: 4 y Figura 23: A). La selección de canales se asignará a variables que alimentarán el comando Merge Channels de ImageJ, en un proceso similar al de la aplicación DeltaVision Stack Manager (Figura 16), con cuatro pequeñas diferencias: 1-no se asigna ninguna paleta de falso color si solo hay un canal, 2-no existe la opción de asignar la paleta de grises (dado que el equipo Opera no captura imágenes de campo claro), 3-la interfaz se adapta al número de canales (se ejecuta una interfaz distinta dependiendo del número de canales) y 4-las paletas de falso color se introducen una vez, pero se pueden procesar un número indefinido de pilas de imágenes de forma automática, en vez de una sola pila de imágenes. Para que la asignación de colores en la interfaz gráfica solo se realice una vez por ejecución, se declaran cuatro variables globales al inicio del código: *channel(1-3)Lut* y *flag*. Las tres primeras garantizan que la selección de canales perdure tras finalizar el bucle donde se asigna su valor y la cuarta garantiza que la interfaz de usuario para la selección de canales solo se ejecute una vez (Figura 23: B).

A



B

```
var flag = true;                                                    (1)
var channel1Lut, channel2Lut, channel3Lut, nChannels, montageImageBorder;

if (flag==true)                                                     (2)
{
 //GUI4 Dialog's box creation*****************************************************
 Dialog.create("Opera Exporter");
 Dialog.addMessage("Enter the color of every channel:");
 colorArray = newArray("Blue", "Green", "Red");
 defaultValuesColorArray = newArray("Blue", "None", "Red","Green", "None", "None", "Red");
 for(j=0; j<nChannels; j++)
 {Dialog.addChoice("Channel "+j+1+":",colorArray,defaultValuesColorArray[j*nChannels]);}
 setBatchMode("exit and display");
 Dialog.show;
 //GUI4 Dialog's box values retrieving*****************************************
 setBatchMode(false);
 channel1Lut = Dialog.getChoice();
 channel2Lut = Dialog.getChoice();                                  (3)
 if(nChannels==3)
 {channel3Lut = Dialog.getChoice();}
 //End of dialog's box values retrieving***************************************
 flag = false;                                                      (4)
}
```

Figura 23. GUI-4 y parte del código responsable de su creación. **A**: GUI-4 permite visualizar cada canal y asignarle un LUT para su posterior fusión. **B**: *Channel(1-3)Lut* se declaran al principio del *script* como variables globales (1) para asegurar la resiliencia del valor asignado en 3. La variable *flag* se declara e inicializa como variable global al principio del código (1), se verifica al principio del bloque que genera la interfaz de usuario (2), y se le asigna el valor *false* al finalizar el mismo (4), para garantizar que esta interfaz de usuario solo se genera una vez. Se ha omitido el código responsable de ordenar las imágenes de la interfaz, para facilitar la comprensión.

La conversión de las pilas de imágenes en montajes (Figura 21: D) se realiza alimentando el comando Make Montage de ImageJ con la variable *nFields* (que codifica el número de campos por pila), en un bucle para todas las pilas abiertas. Para evitar que se produzca un error que detenga la macro en el caso de que se intente realizar un montaje de una única imagen (cuando el usuario haya indicado en GUI-3 que solo

desea visualizar una imagen por pocillo), se comprueba que *nFields*>1 (en caso negativo, la macro continuará normalmente, Figura 17: 4).

Una vez se han transformado todas las pilas de imágenes en montajes, la creación de la interfaz gráfica de visualización y selección de imágenes es bastante sencilla: el código sale del modo batch, aplica un *zoom* sobre los montajes, los ordena, activa la ventana principal de ImageJ, activa la herramienta multipoint y ejecuta el comando waitForUser, mostrando un texto con instrucciones (GUI5 en Figura 17: 4 y Figura 24). Esto permite al usuario navegar por cada uno de los montajes, visualizar las imágenes que los componen y seleccionar aquellas que sean de su interés haciendo clic con el ratón sobre las mismas. Los montajes se han nombrado anteriormente como los pocillos de donde se generaron y las imágenes tienen sus canales fusionados (si se capturó más de uno) lo que facilita el proceso de selección.

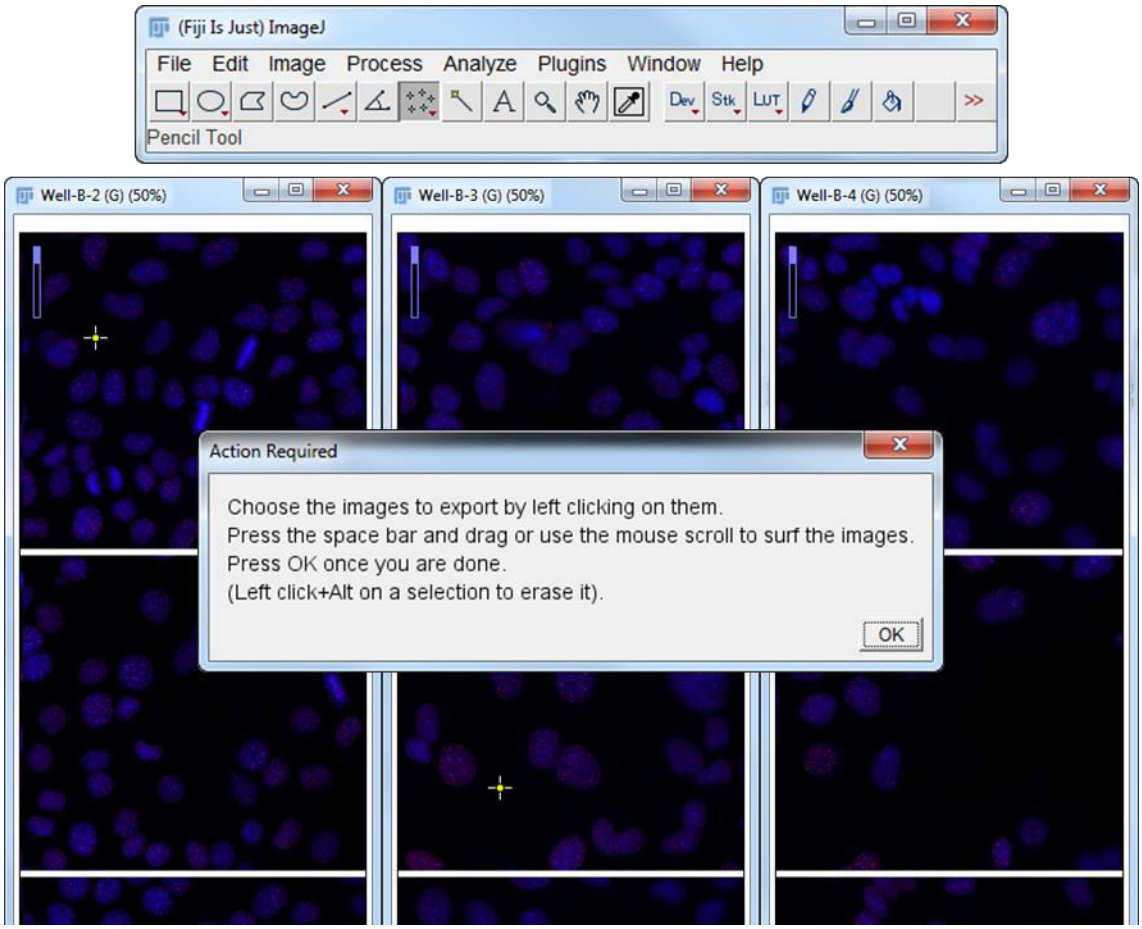


Figura 24. Detalle de GUI-5. Muestra tres montajes de imágenes. Cada uno de ellos contiene tantas imágenes de un pocillo como se especificó en GUI-3. El usuario puede navegar por los montajes, comparar imágenes de diferentes pocillos a la vez y seleccionar las de interés para su exportado. Se muestran dos imágenes seleccionadas (crucetas).

Una vez elegidas las imágenes de interés, el usuario debe presionar el botón *OK* para continuar la ejecución de la aplicación. Al hacerlo, se ejecuta el comando Measure de ImageJ para cada montaje de imágenes que contiene una selección, generándose una ventana de resultados (Figura 25: A y Results en Figura 17: 4). El comando Set

Measurements, ejecutado con anterioridad, con la opción display activada, garantiza que el nombre del montaje aparezca en esta ventana, junto con las coordenadas de la selección realizada por el usuario (Figura 25: A), esta información se utilizará posteriormente (Figura 17: 5) para identificar los archivos .flex del que provienen las imágenes seleccionadas. Finalmente, se cierran todos los montajes y termina la función *imagePicker*. A continuación, se abrirán las imágenes de más pocillos, si es que queda alguno por procesar, y se repetirá todo el proceso desde 3 en Figura 17, omitiéndose la creación de la interfaz gráfica para la asignación de canales (GUI-4 en Figura 17: 4, y Figura 23: A). En el momento en que todos los pocillos seleccionados por el usuario se hayan procesado, se dispondrá de una ventana de resultados (Results en Figura 17) con las coordenadas de las selecciones realizadas por el usuario y el nombre del pocillo del que proceden siguiendo el formato: "Well-(*fila en letra)*-(*columna en número)*" (Figura 25: A).

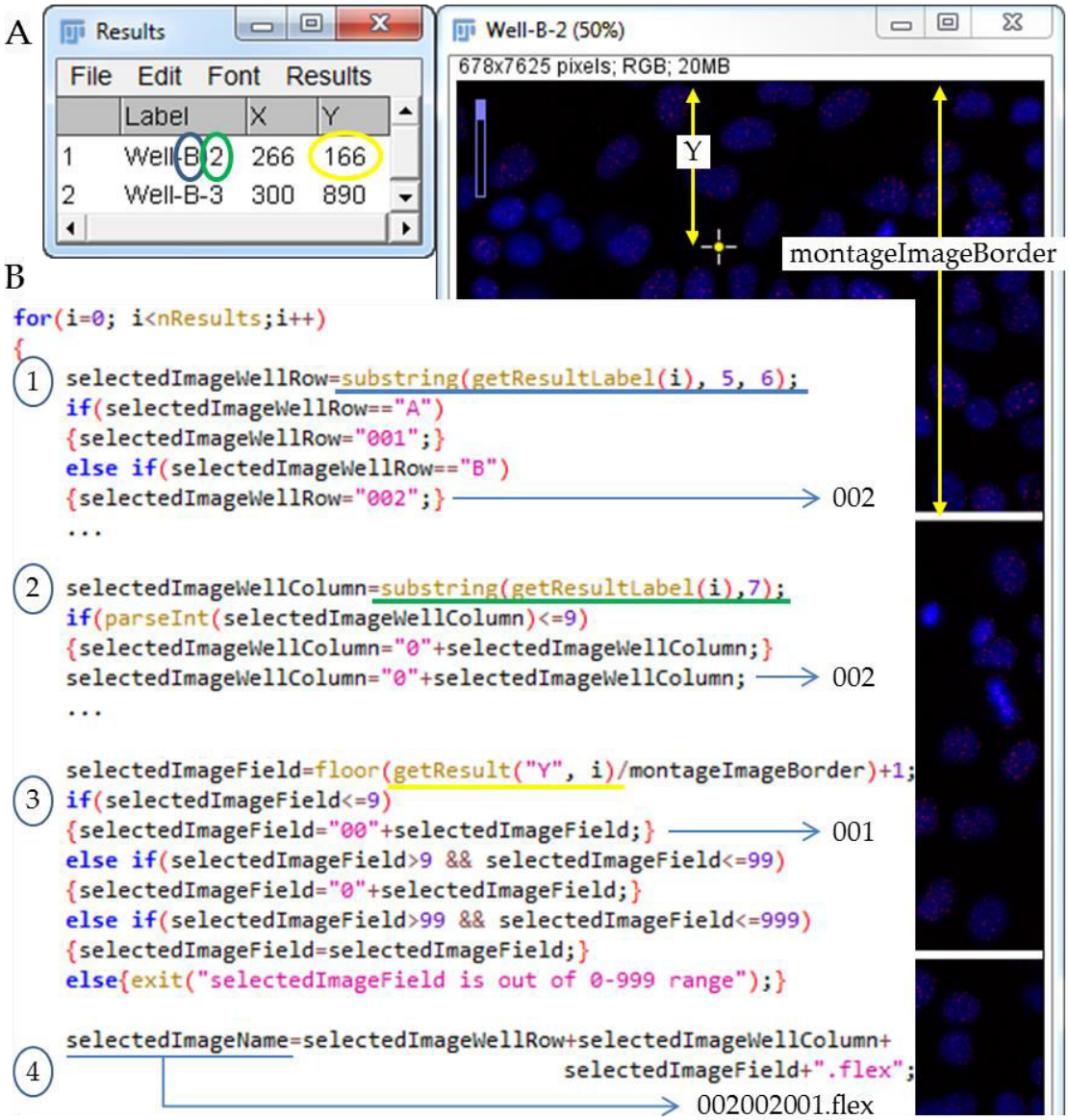


Figura 25. Algoritmo para la obtención del nombre del archivo que contiene la imagen seleccionada en GUI-5 a partir de la ventana de resultados. **A**: imagen seleccionada por el usuario mediante la herramienta Multipoint de ImageJ y ventana de resultados con las coordenadas de la cruceta. **B**: código encargado de obtener el nombre del archivo .flex de las imágenes seleccionadas en el montaje. El nombre del fichero del archivo .flex que codifica cada imagen seleccionada (B4) se extrae de la ventana de resultados a partir del nombre del montaje (B1 y B2) y la coordenada en "y" de la cruceta (B3). B muestra el ejemplo de la selección en A. El código subrayado en B es el responsable de obtener el valor resaltado en la ventana de resultados de A. La relación entre la coordenada "y" de la cruceta y montageImageBorder, permite identificar el campo del que procede la imagen seleccionada. Los puntos suspensivos indican omisión de líneas de código. *montageImageBorder*=508 píxeles.

Si tras visualizar las imágenes de todos los pocillos seleccionados en GUI-3, el usuario no ha seleccionado ninguna de ellas, se muestra un mensaje "No images have been selected", y la macro finaliza. En caso contrario, la macro transferirá las imágenes seleccionadas al directorio definido en GUI-1 (Figura 18). Para hacerlo, es necesario saber el nombre del archivo .flex que contiene la imagen que seleccionó el usuario, el cual se obtiene de la ventana de resultados (Figura 25 y nota al pie de la página 37). Los tres primeros dígitos del archivo .flex se extraen del nombre del montaje, comprobando

cada una de las posibles opciones (A, B, C, D, E, F, G y H) y asignándole el valor correspondiente (001 para la fila A, 002 para la fila B, etc.) a una variable *selectedImageWellRow* (Figura 25: B1). Los tres dígitos siguientes se extraen de forma similar mediante una correspondencia: 1-001, 2-002, 3-003, etc. (Figura 25: B2) y asignando el valor obtenido a una variable *selectedImageWellColumn*. Los tres últimos dígitos se asignan a una variable *selectedImageField*. Su cálculo es un poco más complejo: las imágenes del montaje están ordenadas por campos de forma creciente, de manera que la primera imagen del montaje se corresponde con el primer campo del pocillo y así consecutivamente. El campo de la imagen seleccionada por el usuario (*selectedImageField*), se puede obtener estudiando la relación entre la coordenada "y" de la cruceta y la altura de una imagen en el montaje (*montageImageBorder*, Figura 25: A):

selectedImageField≈Y/*montageImageBorder*,

siendo *selectedImageField* un número natural y "≈" el resultado del redondeo hacia arriba. La coordenada y de la cruceta (Y) se puede obtener mediante el comando getResult de ImageJ y *MontageImageBorder* se puede calcular fácilmente dividiendo la altura total del montaje (se obtiene al ejecutar el comando getHeight de ImageJ) entre el número de campos (*nFields*). El lenguaje macro de ImageJ solo permite redondear hacia abajo (comando floor), por lo que la ecuación anterior queda como sigue:

selectedImageField=floor(getResult("Y", i)/montageImageBorder)+1 (Figura 25: B3). Finalmente, se concatenan las variables *selectedImageWellRow*, *selectedImageWellColumn* y *selectedImageField*, se añade la cadena de caracteres ".flex" y el resultado se asigna a la variable *selectedImageName* (Figura 25: B4).

selectedImageName (Figura 25: B4) permite copiar el archivo .flex a la carpeta de resultados, a continuación se ejecuta el código que abrirá, fusionará los canales y guardará las imágenes seleccionadas en formato .tif (Figura 17: 5), siguiendo procedimientos similares a los mostrados en la Figura 20 y Figura 16, respectivamente.

Al terminar la macro, se muestra un mensaje "The macro is done", junto con el directorio donde el usuario puede encontrar las imágenes seleccionadas.

La aplicación se validó seleccionando imágenes al azar de diferentes campos y pocillos y contrastando las imágenes obtenidas con el *software* de visualización propietario de Opera (Acapella).

OBJETIVO 3. DESARROLLAR UNA APLICACIÓN QUE PERMITA FUSIONAR MOSAICOS REGULARES E IRREGULARES EN SERIE.

Se desarrolló una única aplicación (iMSRC Stitching Module) para cumplir los objetivos propuestos. iMSRC Stitching Module permite la fusión en cadena de colecciones de imágenes procedentes del módulo Matrix Screener de Leica. Si la colección de imágenes contiene varios canales y planos, o si estas se capturaron en un microscopio confocal, iMSRC Stitching Module es capaz de realizar la fusión de canales, proyección de planos y corregir la rotación introducida por el escáner de forma automatizada. Si se han capturado mosaicos irregulares utilizando iMSRC (Figura 8: E), iMSRC Stitching Module crea las imágenes no capturadas hasta completar una colección de imágenes como la de un mosaico regular, lo cual permite fusionarlas eficientemente usando Grid/Collection Stitching. La aplicación es autoexplicativa, interfaces gráficas con instrucciones guían al usuario a lo largo del proceso.

La Figura 26 muestra la captura y fusión de seis mosaicos irregulares en serie utilizando iMSRC e iMSRC Stitching Module.

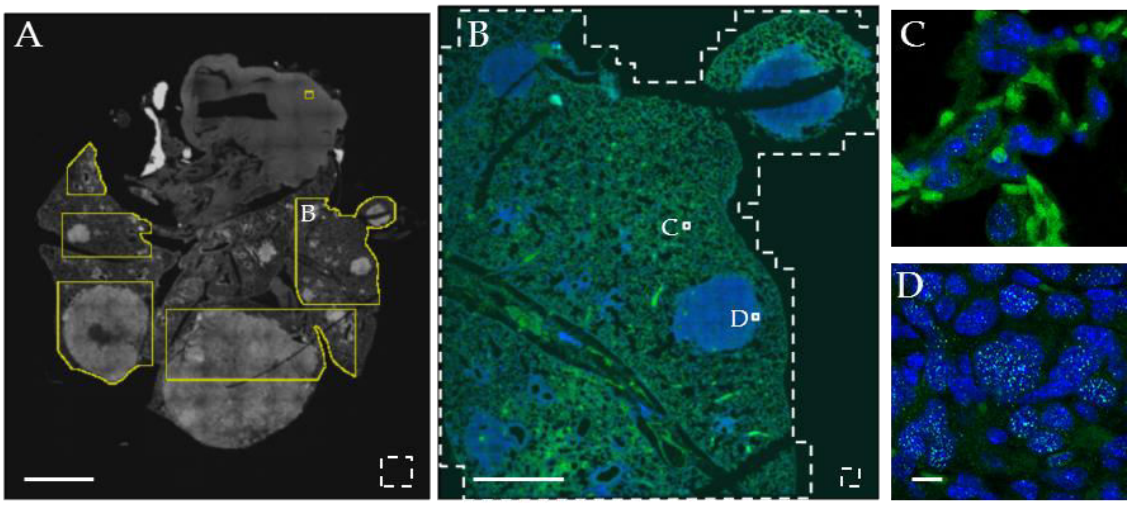


Figura 26. Captura y fusión automatizada de mosaicos irregulares en serie utilizando iMSRC e iMSRC Stitching Module. **A**: mosaico de búsqueda. **B**: mosaico irregular de la zona marcada como B en A. **C** y **D**: imagen aumentada en los cuadros amarillos marcados en B. Las seis regiones marcadas en A en amarillo se capturaron y fusionaron automáticamente utilizando iMSRC e iMSRC Stitching Module. A se capturó con un objetivo 10X, B con un objetivo 40X y *zoom* 2. Los cuadros blancos en A y B muestran el tamaño aproximado de un campo. La zona fuera de las líneas discontinuas en B no se capturó, evitándose la captura de 96 campos. Las barras de escala en A, B y D miden 3 mm, 1 mm y 7 micras, respectivamente.

El código completo de la aplicación puede encontrarse en el apéndice. El experimento mostrado como ejemplo en la Figura 26 se capturó con un microscopio de barrido láser confocal Leica SP5X. En el apartado de material y métodos pueden

consultarse las características de este equipo, las condiciones de la primera y segunda capturas, y la rutina de análisis para la extracción de coordenadas.

Cuando se capturan mosaicos usando el módulo de captura Matrix Screener, la estructura de archivos originada, si se activa la opción de autoexportado en .tif, sigue siempre el mismo patrón (Figura 9). El código de la aplicación mostrado en la Figura 27, sirve de base para generar rutinas de procesado de cualquier colección de imágenes como esta.

```
1  rawChamberNamesArray = getFileList(slidesDir);
2  nChambers = lengthOf(rawChamberNamesArray);
3  for(z=0;z<nChambers;z++)
4  {//Loop for every chamber in slidesDir
5    chamberRelativePath = replace(rawChamberNamesArray[z], "/", "\\");
6    chamberDir = slidesDir+chamberRelativePath;
7    rawFieldNamesArray = getFileList(chamberDir);
8    nFields = lengthOf(rawFieldNamesArray);
9    for(i=0; i<nFields;i++)
10   {//Loop for every Field
11     fieldRelativePath = replace(rawFieldNamesArray[i], "/", "\\");
12     fieldDir = chamberDir+fieldRelativePath;
13     rawImageNamesArray = getFileList(fieldDir);
14     nRawImageNames = lengthOf(rawImageNamesArray);
15     for (j=0; j<nRawImageNames;j++)
16     {
17       imagesDir = fieldDir+rawImageNamesArray[j];
18       if( endsWith(imagesDir, "C00.ome.tif") == 1)
19       {
20         imagesDir=fieldDir+rawImageNamesArray[j];
21         j=nRawImageNames;
22       }
23     }
24     for(c=0; c<nChannels;c++)
25     {//Loop for every channel
26       run("Image Sequence...", "open=["+imagesDir+"] number=999 file=--C0"+c+" sort");
27     }//end of Loop for every channel
28   }//end of Loop for every Field
29 }//end of Loop for every chamber in slidesDir
```

Figura 27. Código de iMSRC Stitching Module que permite abrir, secuencialmente, las imágenes .tif en capturas de Matrix Screener. Las variables *slidesDir* y *nChannels* se toman de las GUI (ver texto). Este código puede ajustarse para el procesado automático de cualquier experimento capturado con Matrix Screener.

Grosso modo, cualquier instrucción que se quiera ejecutar una vez por experimento (Figura 28: 1), debe añadirse fuera del bucle para cada mosaico (Figura 27, antes de la línea 1 o después de la 29); cualquier instrucción que se quiera ejecutar una vez por canal, campo y mosaico (Figura 28: 2), debe añadirse en el bucle para cada canal (Figura 27, líneas 25 a 27); cualquier instrucción que se quiera ejecutar una vez por campo (Figura 28: 3), debe añadirse en el bucle para cada campo (Figura 27, líneas 10 a 28); y cualquier instrucción que se desee ejecutar una vez por mosaico (Figura 28: 4) debe introducirse en el bucle para cada mosaico (Figura 27, líneas 4 a 29)*.

* El procesado de las imágenes de la segunda captura de los objetivos 4, 5 y 6 de esta tesis, se ha llevado a cabo siguiendo este procedimiento.

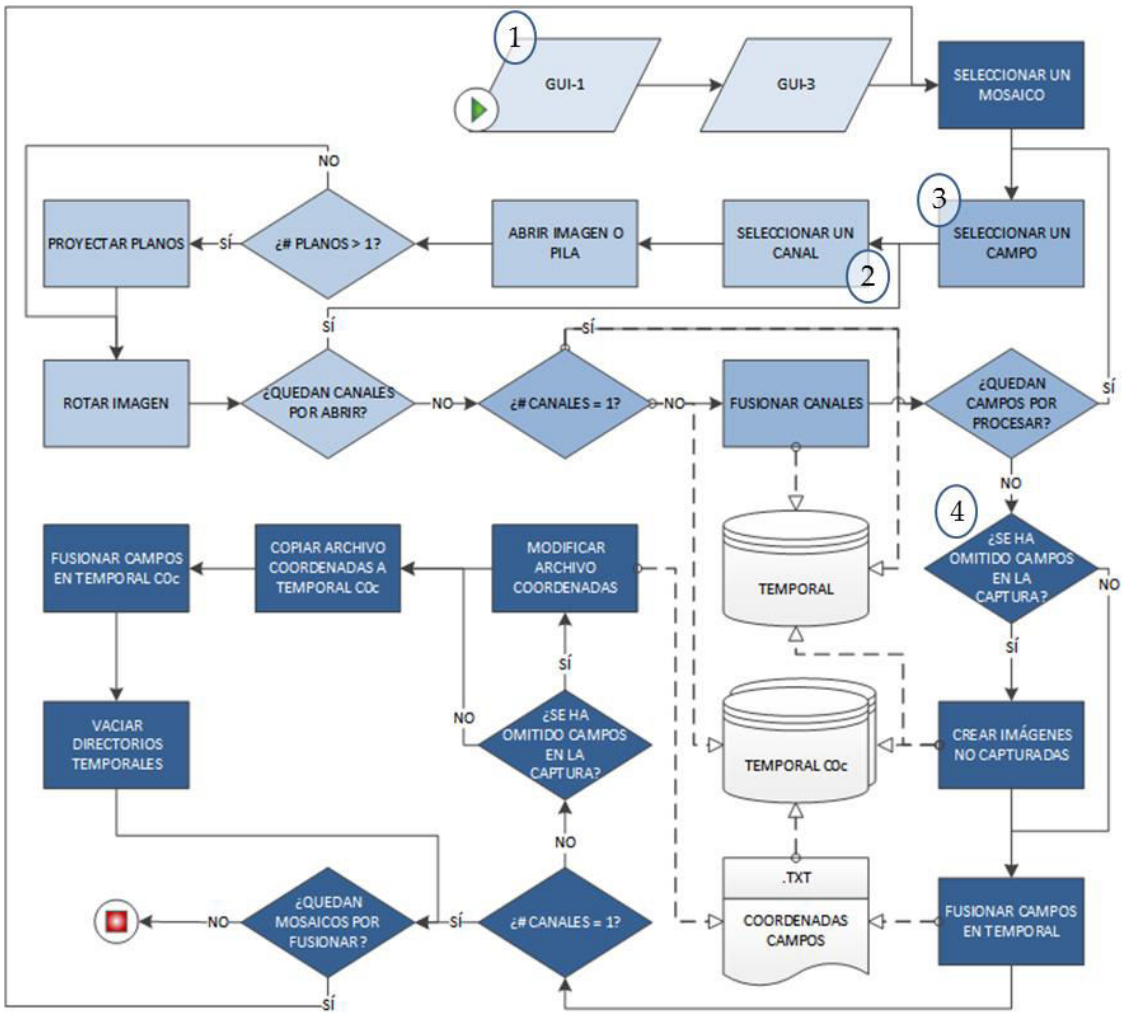


Figura 28. Diagrama de flujo de la aplicación iMSRC Stitching Module. Permite obtener los mosaicos mono y multicanal de una colección de imágenes autoexportadas de Matrix Screener de Leica, aun cuando se haya omitido campos en la captura usando iMSRC. **1**: obtención de directorios y variables mediante GUIs. **2**: bucle de apertura, proyección y rotación de canales. **3**: Bucle de fusión y guardado de los canales de cada campo. **4**: bucle de fusión de mosaicos. No se muestran las interfaces alternativas de obtención de directorios por navegación, GUI-2 y GUI-4, ni el directorio de resultados donde se guardan los mosaicos. El directorio Temporal se crea en todos los casos, contendrá los campos 8-bit de los mosaicos monocanal, o RGB de los mosaicos multicanal. Si se capturó más de un canal (c), se crearán tantos directorios Temporal C0c como canales, cada uno de ellos contendrá imágenes 8-bit de un canal. Los directorios temporales se vacían cada vez que se selecciona un nuevo mosaico y se eliminan al terminar la macro.

Para que una colección de imágenes capturada con Matrix Screener pueda combinarse en un mosaico, las imágenes de cada campo deben abrirse secuencialmente por canales, proyectarse en un único plano, rotarse (si se capturaron en un microscopio confocal, Figura 28: 2) y fusionarse en un único canal (Figura 28: 3). A continuación, se deben guardar los canales independientes y la fusión de canales en carpetas distintas (Temporal C0c y Temporal, respectivamente, en Figura 28), con el nombre de las coordenadas en el mosaico (Figura 28: 3). Por último, se deben fusionar los campos en mosaicos, primero los de la carpeta Temporal y después los de las carpetas Temporal C0c (Figura 28: 4). Todas estas operaciones deben repetirse para cada mosaico. iMSRC Stitching Module automatiza todo este proceso, obteniendo la información necesaria

de varias interfaces gráficas de usuario (Figura 28: 1). GUI-1 y GUI-3 se generan siempre, GUI-2 y GUI-4 (no mostrado), solo si se desea introducir directorios usando un navegador. Si el mosaico es irregular, iMSRC Stitching Module genera imágenes temporales en las posiciones que no se capturaron, estas no aparecerán en el mosaico final, pero permiten la fusión eficiente de estos mosaicos (Figura 28: 4). La aplicación se ejecuta en batch, por lo que el usuario solo visualiza interfaces gráficas.

BLOQUE DE CÓDIGO PARA CADA EXPERIMENTO

Cuando se ejecuta iMSRC Stitching Module (Figura 28: 1), aparece una primera interfaz (GUI-1 en Figura 28: 1 y Figura 29: A) en la que el usuario debe introducir la ruta a la carpeta Slide--S00 (Figura 9), bien rellenando un cuadro de texto o, si se selecciona la casilla de verificación, usando un navegador (Figura 18: B). El valor introducido se almacenará en la variable *slidesDir*. Al pulsar *OK*, se muestra una segunda interfaz con texto, campos rellenables, una casilla de verificación y menús multiopción (GUI-3 en Figura 28: 1 y Figura 29: B), parecida a la de la aplicación DeltaVision Stack Manager (Figura 14). En el primer cuadro de texto (Figura 29: B1) se debe introducir el número de canales capturados; en el conjunto de menús desplegables a continuación (Figura 29: B2), hay que asignar un color a cada uno de ellos siguiendo la nomenclatura de la aplicación Matrix Screener (Figura 9). Los valores introducidos se asignarán a las variables *nChannels* y *red*, *green*, *blue*, *gray*, *cyan*, *magenta* y *yellowChannel* respectivamente, que serán las principales responsables de la fusión posterior de canales. En el siguiente cuadro de texto (Figura 29: B3), se debe introducir la ruta de la carpeta donde se desea guardar los resultados. Aparece una opción por defecto (creada a partir de la ruta introducida en la primera interfaz) que referencia a una carpeta Results-"nombre del experimento" dentro de la carpeta Slide--S00. El usuario puede ignorarla y escribir una nueva o pulsar la casilla de verificación siguiente para introducir una ruta usando un navegador (Figura 18: B). La ruta elegida por cualquiera de los métodos se guardará en *resultsDir*. Finalmente, el usuario debe introducir el valor del ángulo de rotación de la imagen (*rotationAngle*) y el umbral de regresión (*regressionThreshold*) que desea utilizar en el algoritmo de fusión de imágenes Grid/Collection Stitching de ImageJ (este último suele determinarse empíricamente, Figura 29: B4).

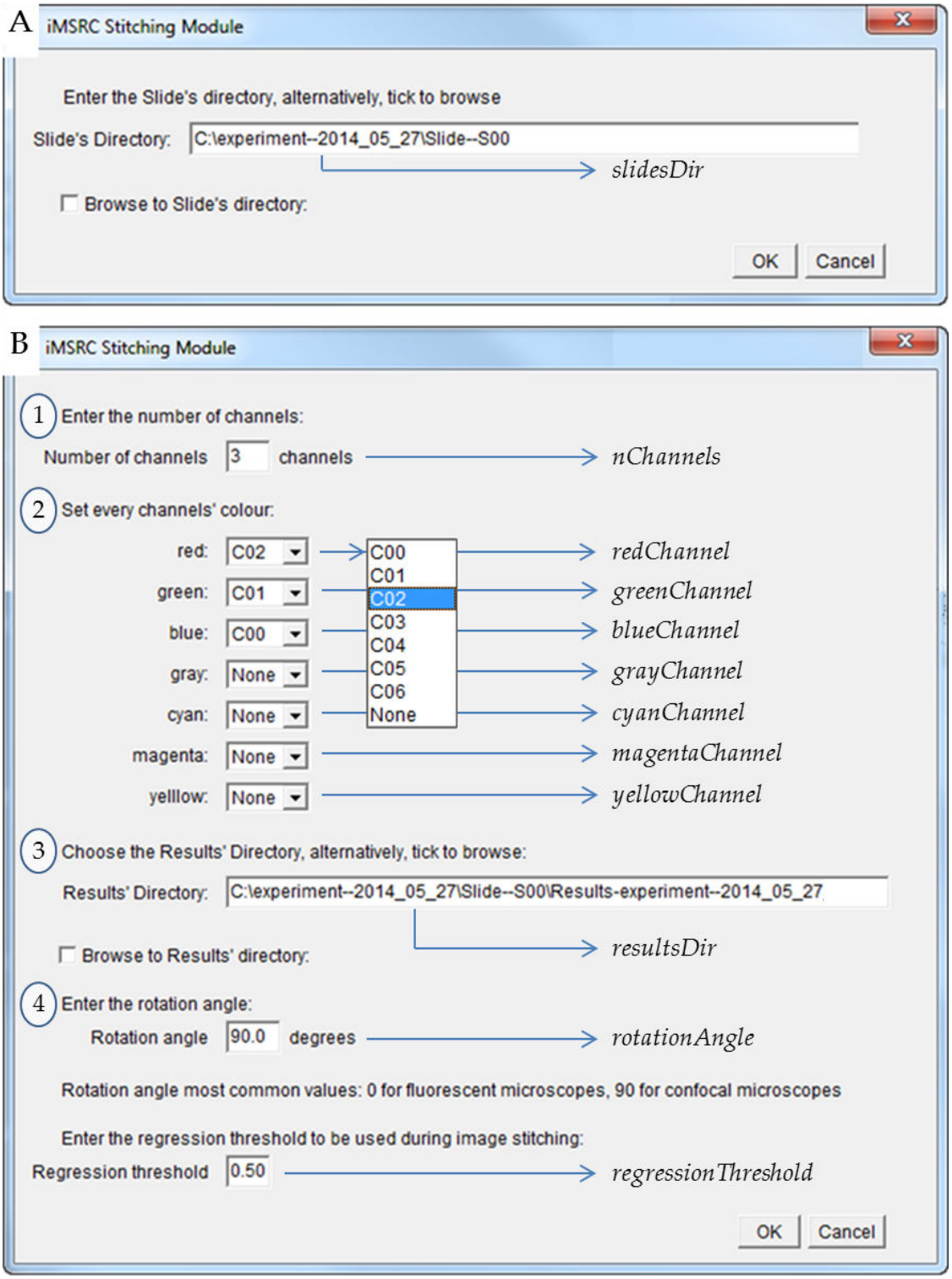


Figura 29. Interfaces gráficas de usuario GUI-1 (A) Y GUI-3 (B) de iMSRC Stitching Module. Muestra sus valores por defecto y el menú multiopción extendido. Las flechas apuntan a las variables a las que se asignarán los valores. Cuando se introduce una ruta en A, se genera automáticamente una ruta de resultados en B3 que se puede usar o modificar. Si se selecciona cualquiera de las casillas de verificación en A y B, se accede a una interfaz gráfica de usuario (GUI-2 o GUI-4) para la selección de directorios mediante un navegador.

Los valores introducidos por defecto en la interfaz se corresponden con los del experimento más repetido en nuestra unidad: captura de tres canales siguiendo el orden azul, verde y rojo, en un microscopio confocal (ángulo de rotación de 90º, porque el escáner de nuestros microscopios confocales rota la imagen - 90º con respecto a su posición original) y un umbral de regresión medio.

Al pulsar *OK*, se asignan los valores introducidos a sus respectivas variables. A continuación, se crea el directorio de resultados, si no existe ya, usando la variable *resultsDir* (Figura 29: B3), en él se guardarán los mosaicos. También se crean los directorios temporales, donde se guardarán las imágenes de cada campo para su fusión en mosaicos. En el caso en que el número de canales sea uno, se genera un único directorio temporal (Temporal en Figura 28). Si el número de canales es mayor a 1, se genera un directorio temporal (Temporal en Figura 28), donde se guardará la fusión de canales para la generación de mosaicos multicanal, y un directorio temporal por canal (c, Temporal-C0c en Figura 28), donde se guardarán los canales independientes para la generación de mosaicos monocanal.

Al final de la macro, cuando la última de las carpetas Chamber se haya procesado, se eliminarán las carpetas temporales y se mostrará un mensaje: "The macro is done".

BUCLE PARA CADA CANAL

La primera vez que se ejecuta este bucle (Figura 28: 2), se abren las imágenes del primer mosaico, plano y canal. La apertura de imágenes se realiza usando el comando Image Sequence de ImageJ (Figura 27: línea 26), este comando permite abrir imágenes individuales, si solo se capturó un plano por campo, o una pila de imágenes, si se capturaron varios planos. Después se realiza la proyección máxima (si hay más de un plano) y se rota la imagen. La proyección de la imagen se realiza utilizando el comando z-Project de ImageJ. La rotación de la imagen permite corregir la rotación introducida por el escáner (Figura 30). Esta se realiza utilizando el comando Rotate de ImageJ y el ángulo de rotación que el usuario introdujo en GUI-3 (que se asignó a la variable *rotationAngle)*.

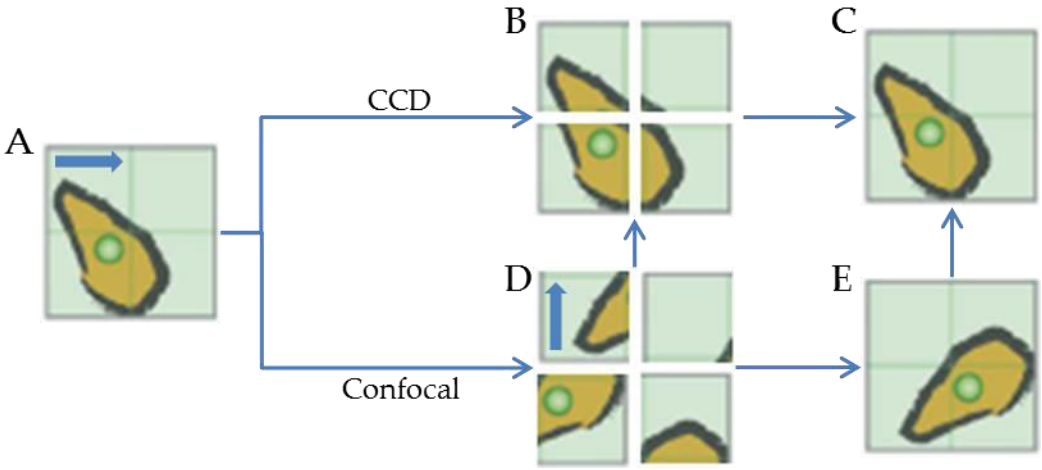


Figura 30. Diferencias en la captura y fusión de mosaicos entre un CCD y un escáner con un ángulo de rotación de -90 grados. Los campos capturados con el CCD (B) tienen la misma orientación que la realidad (A), al igual que el mosaico resultante de su fusión (C). Si un escáner introduce una rotación en la imagen, los campos capturados (D) están rotados respecto a la realidad (A), al igual que el mosaico resultante de su fusión (E). La flecha en A muestran la dirección y sentido de escaneo, la flecha en D muestra la dirección y sentido de la asignación de píxeles del escáner, respectivamente. La rotación del escáner se puede corregir rotando los campos (flecha de D a B) o los mosaicos (flecha de E a C). Grid/Collection Stitching no puede fusionar eficientemente una colección de campos como la capturada en D, por lo que cada uno de ellos debe rotarse individualmente antes de su fusión.

Una vez procesado el primer canal de un campo, se abre y procesa el siguiente (Figura 28: 2).

BUCLE PARA CADA CAMPO

Cada vez que se ejecuta este bucle (Figura 28: 3), al inicio del mismo se extraen los valores de X e Y del nombre del campo y se guardan en un par de vectores (*fieldsXValuesArray* y *fieldsYvalues Array*, respectivamente). Estos se utilizarán en el bucle para cada mosaico para determinar si se han omitido campos en la captura.

Una vez procesadas todas las imágenes para cada canal de un campo, si solo se capturó un canal, la imagen que lo codifica se guarda en la carpeta temporal (Temporal en Figura 28), en formato 8-bit. Si se capturaron varios canales, estos se fusionan en una única imagen y, a continuación, las imágenes multi y monocanal se guardan en directorios distintos: la imagen fusión lo hace en un directorio temporal, en formato RGB, y las imágenes monocanal lo hacen, en formato 8-bit, en un directorio temporal–C0c (Temporal-C0c en Figura 28, c: ordinal por canal). Todas las imágenes se guardan con el nombre de la carpeta Field. La fusión de canales se realiza alimentando el comando Merge Channels de ImageJ con las variables *nChannels* y *red*, *green*, *blue*, *gray*, *cyan*, *magenta* y *yellowChannel*, extraidas de GUI-3 (Figura 29: B1 y B2), de un modo similar al mostrado en la Figura 16.

La fusión y guardado de canales se repetirá para cada uno de los campos de cada mosaico.

BUCLE PARA CADA MOSAICO

Al inicio de este bucle (Figura 28: 4), se obtiene el número de campos de cada mosaico y se almacena en la variable *nFields (*Figura 27: línea 8).

Una vez procesados todos los campos de un mosaico, se extraen los valores de X e Y máximos (*xMa*x e *yMax*) de los vectores *fieldsXValuesArray* y *fieldsYvalues Array,* respectivamente usando la función Array.getStatistics. Si se omitieron campos durante la captura (es decir, si ([*xMa*x+1]x[*yMax*+1*])-nFields>0),* se ejecuta la función *createUncapturedFields()*, ("Crear imágenes no capturadas" en Figura 28: 4) que crea y añade los campos que faltan a todas las carpetas temporales (ver apartado "Función *createUncapturedFields()*", al final de esta sección). Después se ejecuta el módulo Grid/Collection Stitching de ImageJ sobre la carpeta Temporal, aplicando el umbral de regresión introducido en GUI-3 (Figura 29: B). Esto genera: 1- la imagen mosaico y 2- un archivo en Temporal, que contiene la información de la posición de cada campo del mosaico (Coordenadas Campos, en Figura 28). La imagen mosaico se guarda en la carpeta de resultados con el nombre: "nombre del experimento"-"nombre del Chamber"–RGB (si se capturó más de un canal).

El archivo de coordenadas de los campos del mosaico permite fusionar juegos de imágenes procedentes de canales distintos ahorrando tiempo de computación. Si este es el caso (es decir, si el número de canales es mayor a uno, Figura 28: 5), el archivo de posiciones se copia a cada una de las carpetas Temporal-C0c y se ejecuta el módulo Grid/Collection Stitching sobre cada una de esas carpetas, utilizando el archivo anterior. El resultado se guarda en la carpeta de resultados con el mismo nombre que antes menos las tres últimas letras (RGB), que serán: C00, para el primer canal, C01, para el segundo, etc. De esta forma, cuando el número de canales es mayor a uno, la fusión de mosaicos sobre la imagen multicanal (más informativa) optimiza la posición de cada campo, su transmisión a los canales individuales a través del archivo de coordenadas, garantiza la coherencia del resultado y permite ahorrar tiempo de procesado.

El archivo de coordenadas de la imagen mosaico debe procesarse si se omitieron campos durante la captura, para eliminar la información relativa a los campos que no se capturaron (de lo contrario, los campos no capturados afectarán el resultado de la fusión alterando la imagen en la coordenada 0, 0 o añadiendo filas y campos vacíos).

Finalmente, se vaciarán todas las carpetas temporales y se repetirá el procedimiento anterior sobre la siguiente carpeta Chamber.

FUNCIÓN CREATEUNCAPTUREDFIELDS()

La función *createUncapturedFields()* no pertenece a la API de ImageJ, se declara y define en el código.

El módulo Grid/Collection stitching de ImageJ, como se explicó en la introducción, funciona mucho más eficientemente si conoce la posición relativa de cada campo (Figura 10). Esta información la puede extraer del nombre de los campos si se dan unas condiciones: 1- los campos se encuentran en el mismo directorio, 2- el nombre de los campos sigue un patrón regular conocido y 3- no faltan campos.

Cuando se captura un mosaico irregular en Matrix Screener, usando iMSRC, se genera una estructura de carpetas como la descrita en la Figura 9, pero sin las carpetas Field de los campos no capturados. Al ejecutarse la aplicación iMSRC Stitching Module, se crea una carpeta (Temporal) que contiene las imágenes de cada uno de los campos, menos los deshabilitados. En este momento, se cumplen todas las condiciones para que Grid/Collection stitching pueda funcionar óptimamente, excepto la tercera (es decir, faltan campos, porque no se capturaron expresamente). La función *createUncaptureFields()* (Figura 28: 4) soluciona este problema generando las imágenes que faltan y guardándolas junto a las demás con el nombre adecuado. Para hacerlo, genera secuencialmente el nombre de todos los posibles campos entre Field—X00—Y00 y Field—X*xMax*—Y*yMax*, y comprueba si existe una imagen con ese nombre en la carpeta Temporal (ejecutando la función File.exists). En caso negativo, la crea, nombra y guarda en cada una de las carpetas temporales. Para la creación de imágenes, las dimensiones de las mismas se toma de las existentes y su contenido varía. En las carpetas temporales que contienen imágenes de un solo canal, se añaden imágenes de 8 bits de contenido aleatorio. En la carpeta temporal que contiene las imágenes con la fusión de canales (cuando en número de canales es mayor a 1), se crean imágenes RGB de contenido aleatorio. Cuando se ejecuta el módulo Grid/Collection Stitching sobre la carpeta Temporal, en la que previamente se ha ejecutado la función *createUncapturedFields()*, las imágenes introducidas dan lugar a coeficientes de regresión significativamente más bajos que el resto de imágenes (al contener información poco coincidente en la región de solapamiento entre campos). Esto permite establecer un umbral de regresión tal que se descarten, por lo que no aparecerán en el mosaico final.

El bloque de código responsable del procesado de las imágenes (fusión de canales, planos y rotación), se verificó eligiendo campos al azar, procesándolos manualmente y contrastando la imagen resultante con la obtenida de forma automática. Esto se repitió en colecciones de imágenes con varios canales, planos focales y capturados con microscopios fluorescentes y confocales.

El incremento en la eficiencia de la captura de mosaicos irregulares resulta bastante obvio, al igual que el hecho de que esta dependerá de la forma y tamaño del

objeto a capturar. No obstante, esta se evaluó en un tejido a modo de ejemplo. Para hacerlo, se realizó un mosaico de búsqueda del mismo (Figura 31: A) y se capturó y fusionó el mosaico dos veces, la primera vez se realizó un mosaico regular (Figura 31: B) y la segunda uno irregular (Figura 31: C). La captura de mosaicos irregulares usando iMSRC permitió ahorrar 25 horas de captura y 4GB de espacio de memoria en disco.

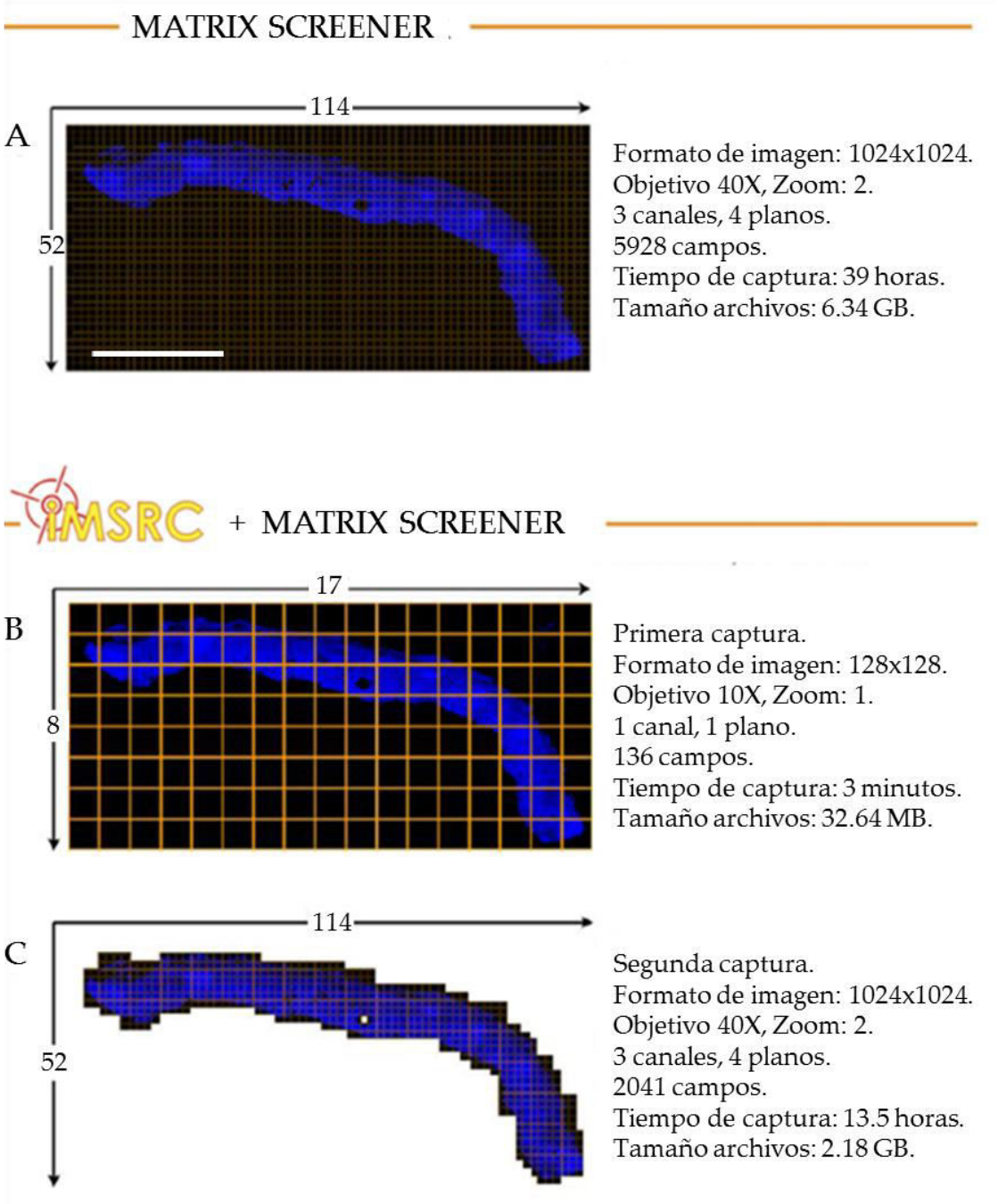


Figura 31. Comparativa de la captura de mosaicos regulares e irregulares usando iMSRC. **A**: mosaico de búsqueda. **B**: mosaico regular. **C**: mosaico irregular Se indican las condiciones y tiempo de captura, el número de campos capturados y el tamaño de los archivos generados. iMSRC permitió ahorrar 25 horas de captura y 4 GB de espacio de memoria en disco en este ejemplo concreto. Barra de escala: 5 mm. Tomado de Carro et al. (2015).

OBJETIVO 4. CAPTURA INTELIGENTE DE CÉLULAS CANCEROSAS CIRCULANTES.

Este procedimiento se diseñó para ser llevado a cabo por parte del personal técnico de la Unidad de Microscopía Confocal, utilizando un microscopio de barrido láser confocal Leica SP5X. Las características del equipo pueden consultarse en el apartado de material y métodos.

Poner a punto un sistema de captura automático de células cancerosas circulantes, basado en la aplicación iMSRC, requiere: 1- configurar perfiles de primera y segunda captura, 2- desarrollar una rutina de análisis para detectar candidatos a célula cancerosa circulante (Figura 4: D y Figura 5), y 3- desarrollar una rutina de procesado de las imágenes resultantes (Figura 27 y nota al pie de la página 95). Conviene tener presente que este diseño experimental presenta la particularidad de producir tres mosaicos de búsqueda, uno para las células positivas para CK (canal verde), otro para las células positivas para CD45 (canal rojo) y otro para la tinción nuclear (canal azul).

CONFIGURACIÓN DE LOS PERFILES DE PRIMERA Y SEGUNDA CAPTURA

Los perfiles de la primera y segunda captura utilizados en el experimento mostrado como ejemplo en la Figura 33, pueden encontrarse en el apartado de material y métodos.

Los perfiles de primera captura deben permitir detectar candidatos a células cancerosas y, a la vez, facilitar la formación rápida de los mosaicos de búsqueda. Teniendo esto en cuenta, se maximizó la velocidad del escáner y se minimizó el número de planos, el *zoom*, la resolución de las fotografías, el aumento del objetivo de captura y el promediado de imágenes. Además, fue necesario diseñar una rutina de autofoco predictivo enfocada sobre las células pegadas a la base de la placa. La fusión de campos en mosaicos se realizó en Matrix Screener, con la función de autoexportado en .tif activada.

El resultado de la primera captura son 9 imágenes mosaico .tif: 3 planos en el canal azul (Hoechst 33342), 3 planos en el canal verde (CK-FITC) y 3 planos en el canal rojo (CD45-Cy3). Cada mosaico de búsqueda comprende toda el área de un pocillo que contiene una muestra de células cancerosas circulantes. Se deben capturar tres planos porque la muestra contiene un gran número de esferas magnéticas (ver material y métodos) que le aportan volumen (Figura 32). La captura y formación de los mosaicos de búsqueda se realiza en 25-30 minutos.

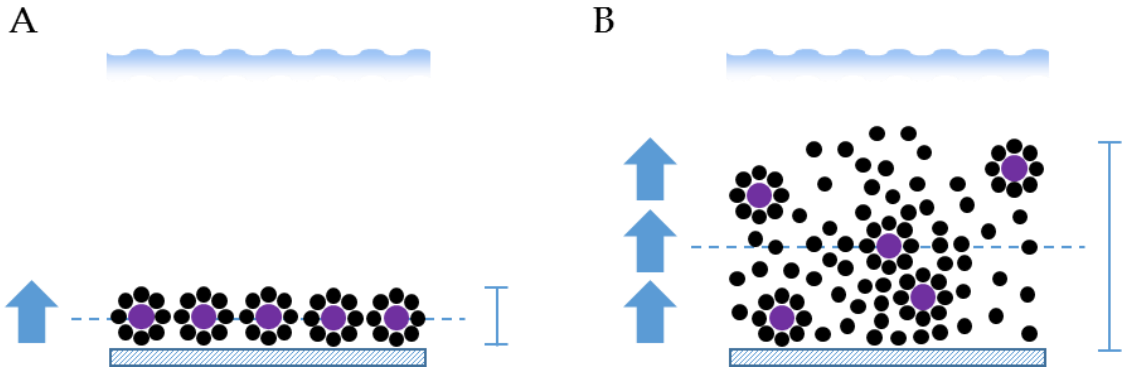


Figura 32. Relación entre el volumen de la muestra, el número de planos focales a capturar en la primera captura y el rango de búsqueda de un hipotético patrón de autofoco asistido por software en la segunda captura. Sección transversal de una muestra, desde la base de un pocillo (abajo) a la superficie del medio de inmersión (arriba). **A**: muestra sin volumen. Las células (círculos violeta) se localizan en un plano. Un solo plano focal (flecha azul) permite capturar todas las células en la primera captura. Una hipotética rutina de autofoco asistida por software, en la segunda captura, tendría un rango de búsqueda estrecho (línea continua azul), centrado sobre las células (línea discontinua azul). **B**: muestra con volumen. Las células se localizan en varios planos. Es necesario adquirir varios planos focales durante la primera captura, para fotografiar todas las células. Una hipotética rutina de autofoco asistida por software, en la segunda captura, tendría un rango de búsqueda ancho, centrado sobre la muestra (línea discontinua azul). La rutina de autofoco fallará en B cuando haya células superpuestas. Los círculos negros representan esferas magnéticas.

Los mosaicos de búsqueda en este experimento cumplen una doble función. Los mosaicos verde y azul son la base para la detección de células positivas para el canal verde. Los mosaicos verde, rojo y azul, también se utilizan para aproximar el número de células verdes, rojas y total (no mostrado), cuando el número de células cancerosas circulantes en la segunda captura es mayor a 15 (ver apartado "Segunda captura").

Los perfiles de segunda captura se diseñaron para permitir la detección inequívoca de células cancerosas circulantes. Para ello, se utilizaron objetivos de grandes aumentos, *zoom*, un formato que cumpliera el teorema de Nyquist, una apertura de *pinhole* óptima y un número de planos adecuado. La velocidad del escáner y el número de imágenes a promediar se optimizó en base a la relación señal/ruido. Los canales se capturaron secuencialmente para evitar el cruce de señal. Se diseñó una rutina de autofoco que se acabó descartando por problemas de fiabilidad (en ocasiones, obtenía focos erróneos de forma impredecible) y porque fallaba cuando dos células positivas se localizaban una sobre la otra. La rutina de autofoco utilizaba el reflejo del láser en la base del pocillo para determinar el centro de la muestra (utilizando el autofoco predictivo y sumándole una distancia fija). A continuación, se realizaba una búsqueda de foco en todo el rango de la muestra utilizando la señal verde.

Las ventanas de detección de señal de los detectores, en los perfiles de primera y segunda captura, se ajustaron para permitir la detección de señal específica de cada fluoróforo, utilizando sus espectros de emisión y excitación y los controles citados en el apartado de material y métodos.

ANÁLISIS DE IMAGEN

La rutina de análisis Circulating Cancer Cells permite detectar candidatos a célula cancerosa en los mosaicos de búsqueda. El código de la misma puede consultarse en el apéndice. La macro no se ejecuta en modo batch, por lo que todas las imágenes son visibles al usuario.

La identificación de candidatos a células CK positivas en los mosaicos de búsqueda se realiza en base a su tamaño, brillo en el canal verde y brillo en el canal azul. Consideramos objetos susceptibles de ser una célula positiva para CK a aquellos objetos que brillan más en verde que el fondo (son inmunopositivos para FITC), brillan más en azul que el fondo (contienen núcleo, condición indispensable de toda célula eucariota) y su tamaño es compatible con el de una célula cancerosa o un agregado de dos células (5-50 píxeles en estas condiciones de captura, como fruto de nuestra experiencia).

La rutina de análisis que incluye iMSRC no permite identificar este tipo de células, porque solo puede procesar un mosaico de búsqueda de 8-bit (ver nota al pie en la página 28). Como necesitamos detectar objetos en base a dos imágenes (canal azul y verde), se tuvo que diseñar una rutina de análisis *ex profeso*. Dicha rutina obtiene objetos marcados en el canal verde, comprueba que también estén marcados en el canal azul y extrae una imagen binaria con los objetos que cumplen ambas condiciones. Esta imagen sí puede ser utilizada por la rutina de análisis de iMSRC, permitiendo capturar los candidatos a célula cancerosa circulante.

El tamaño celular se determinó empíricamente, midiendo un gran número de células CK positivas en el canal verde. Aunque este parámetro es bastante robusto, puede ajustarse en base a la calidad de la tinción y a la cantidad de agregados de cada muestra. En cuanto al umbral de segmentación de los canales verde y azul, se buscó un sistema automático para determinarlos. Con esta finalidad, se testaron todos los algoritmos automáticos de segmentación que incluye la distribución FIJI. Ninguno de ellos dio resultados satisfactorios y robustos, por lo que el umbral de segmentación se determina empíricamente, en cada una de las imágenes de cada experimento. En ocasiones, cuando las muestras contienen artefactos, puede ser útil clasificar los objetos en base a su circularidad, aunque este parámetro no se utiliza normalmente.

La imagen binaria con los candidatos a célula cancerosa se obtiene de la siguiente manera: se abren las imágenes de todos los planos del mosaico de búsqueda para el canal verde y se obtiene la proyección máxima (Figura 33: B). Se asigna un umbral de detección (Figura 33: E) y se extrae una máscara binaria con el número de objetos por encima del mismo, de tamaño comprendido entre los 5 y 50 píxeles (Figura 33: G). Esta imagen contiene los objetos verdes más brillantes que el fondo que son susceptibles de tener el tamaño adecuado para ser células. A continuación, se abren las imágenes de todos los planos del mosaico para el canal azul, se obtiene la proyección máxima

(Figura 33: I) y se realiza una operación lógica tipo AND con la máscara binaria anterior. El resultado es una imagen en escala de grises, que es la proyección máxima del canal azul en la que se ha omitido toda señal no susceptible de ser un objeto verde del tamaño de una célula (Figura 33: H). Finalmente, se binariza la imagen anterior, asignando un umbral de detección para el canal azul (Figura 33: F) y extrayendo los objetos entre 5 y 50 píxeles. La imagen resultante comprende los objetos nucleados positivos para verde con un tamaño compatible con el de una célula (Figura 33: C). Dicha imagen se analiza mediante la rutina básica de iMSRC, con un umbral de detección entre 1 y 255, sin límite de tamaño. El resultado es un archivo de segunda captura, modificado con las coordenadas de los candidatos a células cancerosas circulantes, que permite capturarlas a grandes aumentos (Figura 33: D), de forma automática, utilizando Matrix Screener.

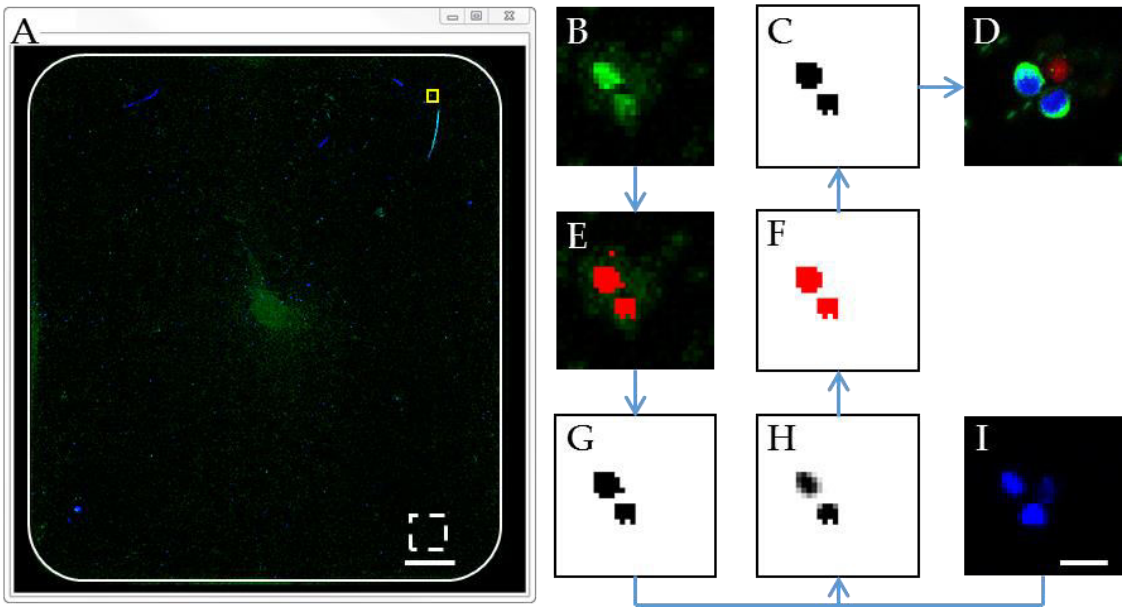


Figura 33. Algoritmo para la obtención de las coordenadas de las células positivas para CK (verde). **A** muestra la proyección de planos y la fusión de los canales azul y verde del mosaico de búsqueda. **B** e **I** son detalles del inserto amarillo en A para el canal verde (CK) y azul (Hoechst 33342), respectivamente. Se asigna un umbral de detección al canal verde (**E**), se extraen los objetos entre 5-50 píxeles (**G**) y se realiza una operación lógica AND con la imagen que contiene el canal azul (**I**). A la imagen resultante (**H**), se le aplica un umbral de detección (**F**) para obtener una imagen binaria con los objetos entre 5-50 píxeles (**C**), que se analiza con iMSRC. El resultado es un archivo de captura de Matrix Screener que permite capturar automáticamente los candidatos a célula CK positiva a grandes aumentos (**D**). El color rojo en E y F muestra los píxeles seleccionados al aplicar el umbral de segmentación. Los perfiles de los objetos en C y G se almacenan en el ROI Manager, esto permite superponerlos en las imágenes originales para contrastar la calidad de la detección. El cuadrado de líneas discontinuas en A representa el tamaño de un campo, el cuadrado de línea continua blanca delimita los bordes del pocillo. Las flechas indican el orden de procesado de las imágenes. Escalas: A, 1mm; B-I, 1 µm.

El resultado de cada análisis (Figura 33: G y C) se almacena en el ROI* Manager. Es habitual repetir el proceso de análisis varias veces hasta dar con unos umbrales de

* ROI: región de interés, de sus siglas en inglés: region of interest.

segmentación satisfactorios, lo cual suele comprobarse superponiendo las selecciones obtenidas en el ROI Manager sobre las imágenes de cada uno de los canales. Asignar un umbral en el canal azul (Figura 33: F) es sencillo, dado el elevado número de eventos positivos (Figura 33: I). Para asignar un umbral en el canal verde (Figura 33: E), es útil encontrar un evento positivo (suele aparecer durante el proceso, Figura 33: B) o bien asignarlo, como fruto de la experiencia, en función del fondo. Suelen asignarse umbrales poco conservadores, que resulten en la captura a grandes aumentos de un número bajo de falsos positivos. Si en la segunda captura solo se encuentran eventos positivos o no se encuentra ningún evento, se repite el proceso disminuyendo el umbral de segmentación del canal verde y/o azul a modo de comprobación. La imagen binaria final siempre queda como registro. El procedimiento se validó alternando muestras control y experimentales, desconociendo el analizador su naturaleza.

La rutina Circulating Cancer Cells, guía al usuario a lo largo de todo el proceso, la Figura 34 muestra el diagrama de flujo de la misma.

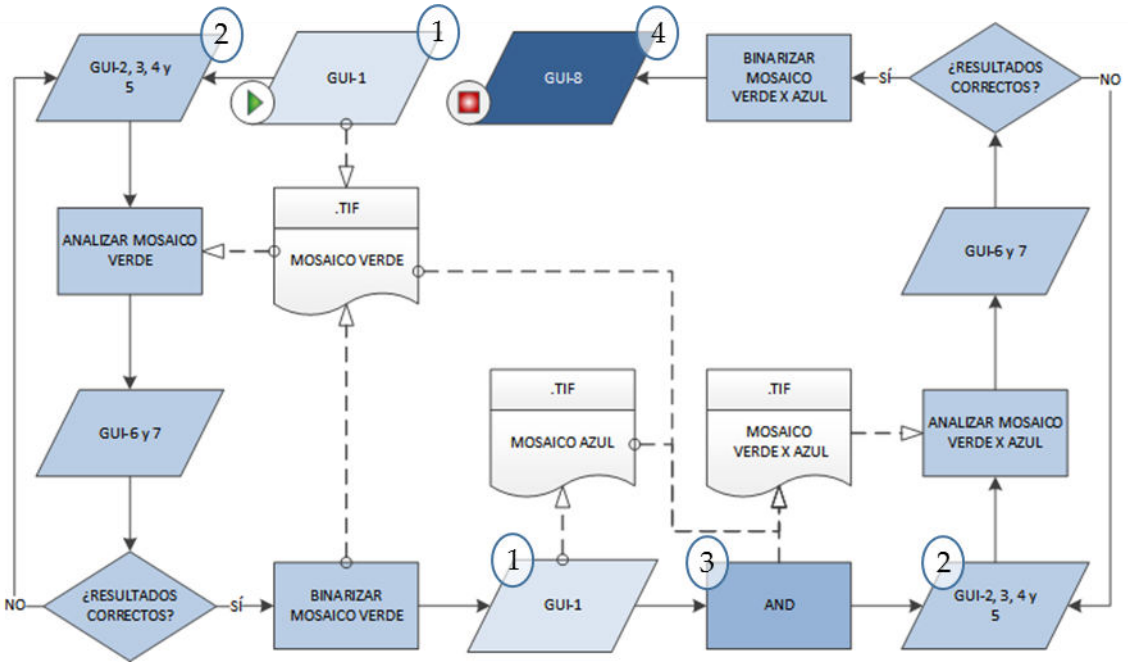


Figura 34. Diagrama de flujo de Circulating Cancer Cells. Una serie de interfaces gráficas guían al usuario en el procesado de las imágenes mosaico de un canal azul y verde, para extraer una imagen binaria con los candidatos a célula cancerosa circulante. **1**: el usuario abre el mosaico verde la primera vez que se muestra GUI-1 y el azul la segunda vez que lo hace. **2**: cada mosaico se analiza en base a los valores de brillo, tamaño y circularidad introducidos en GUI-2, GUI-3, GUI-4 y GUI-5. GUI-6 muestra los objetos detectados sobre la imagen original, en función de los cuales el usuario puede decidir (en GUI-7) repetir el paso de detección o binarizar la imagen con los objetos detectados. **3**: una operación AND permite obtener un mosaico con los objetos azules positivos para CK. **4**: tras GUI-8, el usuario dispone de una imagen binaria con candidatos a células cancerosas que iMSRC puede analizar. Los perfiles de estos se almacenan en el ROI Manager, lo que permite superponerlos en los mosaicos originales para comprobar el resultado del proceso.

El usuario debe realizar la proyección máxima de las imágenes de los mosaicos de búsqueda manualmente, abrirlas cuando se lo pida la macro (Figura 35: A y Figura 34: 1), introducir los valores de los parámetros de detección de objetos (umbral de segmentación, tamaño mínimo y máximo, y umbral de circularidad Figura 35: B, C, y Figura 34: 2) y guardar el resultado (Figura 35: E y Figura 34: 4). Cada vez que segmente una imagen, la macro le permitirá visualizar el resultado sobre la imagen original y repetir el proceso, hasta que esté satisfecho con el mismo (Figura 35: D y Figura 34: 2). Finalmente, los resultados se almacenan en el ROI Manager, junto con la imagen binaria. El usuario puede abrir las imágenes originales y mostrar los perfiles a candidato a célula cancerosa sobre ellas, para contrastar los resultados y, si está satisfecho con los mismos, guardar la imagen binaria (Figura 35: E, Figura 34: 4). Los valores de los umbrales de segmentación, circularidad y tamaño de objetos, se registran en el *Log*, por si el usuario quiere guardarlos o consultarlos.

Figura 35. Interfaces gráficas de usuario de Circulating Cancer Cells. **A**: GUI-1. **B**: GUI-2. **C**: GUI-3, GUI-4 Y GUI-5, respectivamente. **D**: GUI-6 y GUI-7, respectivamente. **E**: GUI-8. Las interfaces de la A a la D se repiten para cada uno de los canales (primero para el verde y luego para el azul), sustituyendo "green" por "blue" en A y B. Al finalizar C, los objetos que cumplen los requisitos introducidos en B y C, se muestran sobre la imagen original. Si el usuario presiona "No" en D, la aplicación vuelve a B, si presiona "Yes", se obtiene la imagen binaria con los objetos detectados y la macro continúa. Los botones "Cancel", detienen la macro. Al abrirse la imagen azul, se realiza una operación AND entre esta y la imagen binaria con los objetos en el canal verde.

La detección de coordenadas de los candidatos a célula cancerosa circulante suele

llevarse a cabo en 15 minutos, como promedio.

SEGUNDA CAPTURA

Se cargó el archivo de segunda captura modificado (con las coordenadas de los candidatos a célula cancerosa) en la aplicación Matrix Screener, se enfocaron y se capturaron las imágenes manualmente con la función de exportado en .tif activada. El resultado fue una colección de imágenes por candidato a célula cancerosa, cada una de ellas con tres canales y un número variable de planos.

Se acordó con el director del proyecto que, para cumplir los objetivos del mismo, se capturarían un máximo de 15 células cancerosas circulantes por experimento. Cuando el número de células cancerosas circulantes es mayor que 15, se fotografían las 15 primeras células cancerosas circulantes y se estima el número total de las mismas a partir de la imagen mosaico.

La captura de una célula cancerosa circulante en estas condiciones se lleva a cabo en 3 minutos.

PROCESADO DE IMÁGENES

La colección de imágenes a grandes aumentos, generada en la segunda captura, se procesa automáticamente para la fusión de colores y planos. El resultado se envía al investigador para su caracterización. Este determina qué células, de las capturadas (Figura 33: D), son células cancerosas circulantes, descartando posibles falsos positivos. El código para el procesado automático de la colección de imágenes procedente de la captura de grandes aumentos, se ejecuta una vez por célula capturada y es una adaptación del código visto en iMSRC Stitching Module, para el procesado de campos capturados en un mosaico (ver nota al pie en la página 95 y Figura 27). El procesado de las imágenes de la segunda captura tarda 10 segundos por célula capturada.

La duración de todo el proceso no excede de las 2 horas por muestra.

OBJETIVO 5. CAPTURA INTELIGENTE DE CÉLULAS MADRE CANCEROSAS.

Este procedimiento se diseñó para ser llevado a cabo por parte del personal técnico de la Unidad de Microscopía Confocal, utilizando un microscopio de barrido láser confocal Leica SP5X. Las características del equipo pueden consultarse en el apartado de material y métodos.

Poner a punto un sistema de captura automático de células positivas para un biomarcador, basado en la aplicación iMSRC, requiere: 1- configurar perfiles de primera y segunda captura, 2- desarrollar una rutina de análisis para detectar candidatos a célula positiva aislada en un micropocillo (Figura 4: D y Figura 5), y 3- desarrollar una rutina de procesado de las imágenes resultantes (Figura 27 y nota al pie de la página 95). Conviene tener presente que este diseño experimental presenta la particularidad de producir dos mosaicos de búsqueda, uno para el canal verde (biomarcador) y otro para el canal rojo (micropocillos, Figura 11: D y E).

CONFIGURACIÓN DE LOS PERFILES DE PRIMERA Y SEGUNDA CAPTURA

Los perfiles de la primera y segunda captura, utilizados en el experimento mostrado como ejemplo en la Figura 37 y Figura 38, respectivamente, pueden encontrarse en el apartado de material y métodos.

Los perfiles de la primera captura se diseñaron para permitir la detección de la señal autofluorescente en el canal verde y los contornos de los micropocillos en el canal rojo. Se priorizó la rapidez, tanto en la captura, como en la formación de los mosaicos de búsqueda (de forma similar a la explicada en el objetivo 4). Para incrementar todavía más la velocidad de captura, los canales rojo y verde se tomaron simultáneamente; para agilizar la formación de mosaicos, solo se capturó un área del chip de micropocillos que contuviera varios miles de ellos. La ventana de detección del canal rojo se fijó para captar la reflexión del láser 561 en las paredes de los micropocillos. La fusión de campos en mosaicos se realizó en Matrix Screener, con la función de autoexportado en .tif activada. En estas condiciones, la captura del mosaico de búsqueda no excede de los 5 minutos.

El resultado de la primera captura son dos mosaicos de búsqueda (Figura 11: D y E): una imagen de un canal verde con la señal autofluorescente y una imagen de un canal rojo con la señal del contorno de los pocillos (ambas en formato .tif).

Los perfiles de la segunda captura se diseñaron para obtener imágenes a una resolución espacial tal que permitiera visualizar inequívocamente células positivas

para la marca autofluorescente, y a una resolución temporal tal que permitiera visualizar mitosis. Estos se diseñaron de forma similar a la explicada en el objetivo 4.

Teniendo en cuenta que la muestra contenía células vivas, en ambos perfiles se minimizó la potencia del láser, se abrió el *pinhole*, se utilizó el detector híbrido cuando era posible, se redujo el formato, y se evitó el uso del *zoom*, aun cuando esto supusiera no cumplir el teorema de Nyquist.

Las ventanas de detección de señal de los detectores en el canal verde, en los perfiles de primera y segunda captura, se ajustaron en base al espectro de emisión de la señal autofluorescente.

ANÁLISIS DE IMAGEN

La rutina de análisis que incluye por defecto iMSRC, no permite identificar los pocillos que contienen una única célula madre tumoral porque, tal y como se explicó en el objetivo 4, esta rutina no permite procesar dos mosaicos de búsqueda. Para solucionarlo, diseñamos una macro *ex profeso*, denominada Single Cell Well Detector.

El código de Single Cell Well Detector puede encontrarse en el apéndice, la Figura 36 muestra su diagrama de flujo.

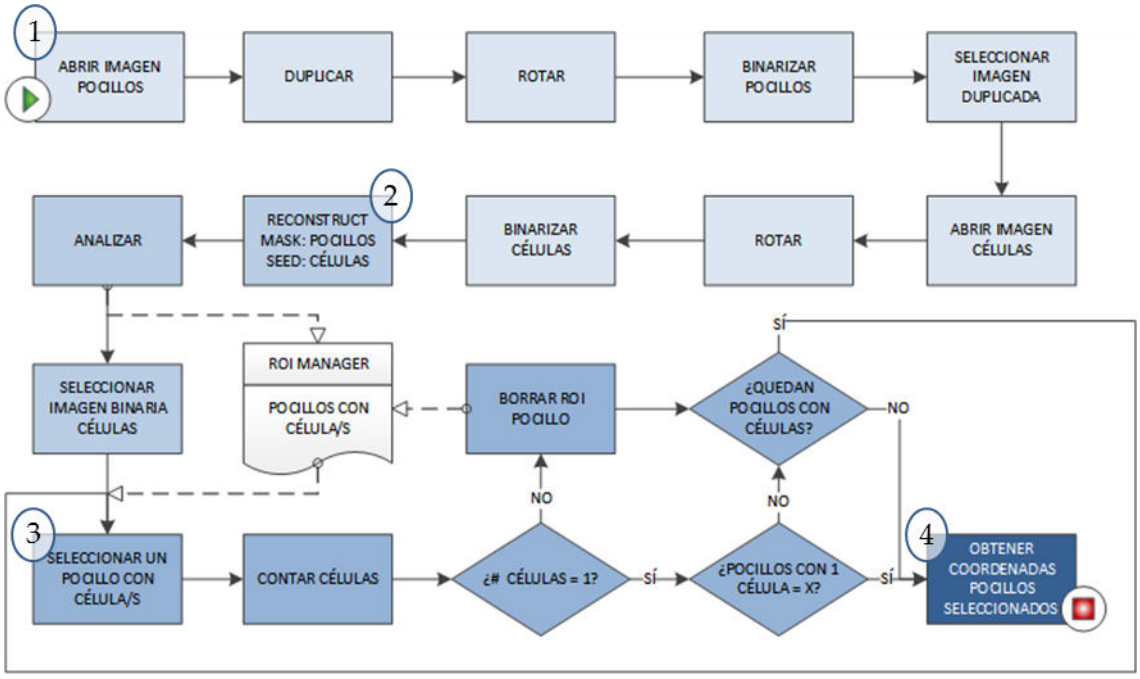


Figura 36. Diagrama de flujo de Single Cell Well Detector. Esta macro extrae las coordenadas de los micropocillos que contienen una célula autofluorescente, a partir de dos imágenes: una de los pocillos y otra de las células. **1**: bloque responsable de binarizar las imágenes mosaico que contienen los pocillos y las células. La imagen de las células se abre a partir de la de los pocillos, para lo cual es necesario duplicarla antes. **2**: parte encargada de obtener los perfiles de los pocillos que contienen células. La imagen reconstruida contiene los pocillos con al menos una célula, sus selecciones se almacenan en el ROI Manager. **3**: bucle de selección de las ROIs de los pocillos con una única célula. "X" es un valor arbitrario, codificado en la variable *limit*, que permite limitar el número de resultados. **4**: sección dedicada a obtener las coordenadas de los pocillos con una única célula.

Single Cell Well Detector se ejecuta directamente desde iMSRC*. En su interfaz gráfica de usuario, se debe introducir el umbral de segmentación del canal verde y las rutas a la imagen mosaico roja y a los perfiles de primera y segunda captura. iMSRC generará directamente los perfiles de segunda captura modificados, que el usuario puede cargar en Matrix Screener para capturar los objetos detectados. La macro funciona en batch y carece de GUIs, por lo que no muestra ninguna información al usuario.

Brevemente, se binarizan las imágenes mosaico que contienen los micropocillos y las células (Figura 36: 1 y Figura 37: A, B, C y D). A partir de estas, se genera la imagen binaria de los pocillos que contienen células (Figura 37: E), mediante una operación de reconstrucción que utiliza la imagen binaria que contiene los pocillos como máscara y la imagen binaria que contiene las células como semilla (Figura 36: 2). Dicha imagen se analiza y las selecciones de los pocillos que contienen alguna célula se almacenan en el ROI Manager (Figura 36: 2 y ROI Manager en Figura 37). A continuación, estas selecciones se superponen una a una en la imagen binaria que contiene a las células (Figura 37: F), se cuenta el número de células por ROI y se eliminan las ROI con un número de células distinto de 1 (Figura 36: 3). Finalmente, se obtiene la imagen binaria de los pocillos que contienen una única célula (Figura 37: G) y se extraen sus coordenadas (Figura 36: 4 y Results en Figura 37).

* iMSRC permite implementar fácilmente rutinas de análisis de ImageJ. Si se desea hacer uso de su interfaz gráfica de usuario para introducir parámetros, se debe modificar la macro de la siguiente manera: - introducir la sentencia *open("#file#")* en el punto donde se quiera abrir la imagen mosaico, - introducir la sentencia setThreshold(#thresholdMin#, #thresholdMax#), en el punto donde se quiera segmentar la imagen, e introducir la sentencia run("Analyze Particles...", "size=#size#-#maxsize# pixel circularity=#circularity#-1.00 show=Nothing display exclude clear"), en el punto donde se quiera extraer las coordenadas de los objetos de interés.
iMSRC sustituirá las variables entre las almohadillas por la ruta a la imagen mosaico, los niveles de segmentación mínimo y máximo, los tamaños mínimo y máximo y circularidad de los objetos a detectar, respectivamente, introducidos por el usuario en su GUI.

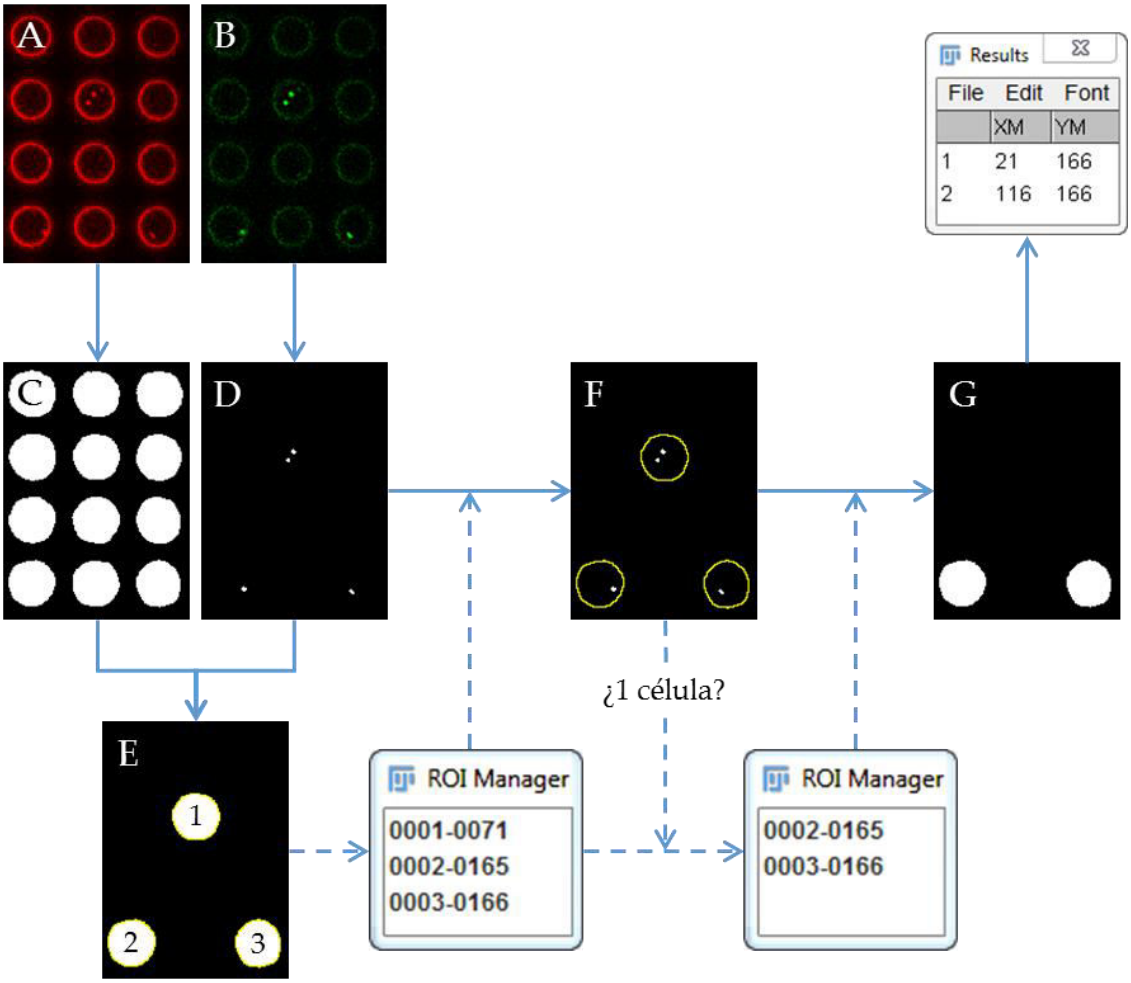


Figura 37. Algoritmo para la extracción de las coordenadas de los micropocillos que contienen una única célula. **A** y **B**: detalle de los mosaicos de búsqueda que contienen los micropocillos y las células positivas para el biomarcador, respectivamente. **C** y **D**: imagen binaria de los pocillos y células, resultado de segmentar A y B. **E**: resultado de una operación de reconstrucción que usa D como semilla y C como máscara, permite incluir en el ROI Manager los pocillos que contienen células (el número indica la selección en el ROI Manager). **F**: resultado de mostrar los elementos del ROI manager en D, permite discriminar las selecciones de pocillos que contienen una única célula. **G**: imagen binaria de los pocillos que contienen una célula, se obtiene de F, dibujando sobre esta las selecciones del ROI Manager. **Results**: coordenadas de los pocillos en G. Se muestran en línea discontinua las actividades que hacen uso del ROI Manager.

Conviene destacar algunos detalles del proceso anterior: 1- la imagen verde (Figura 36: 1, "abrir imagen células") se abre a partir de la roja usando el comando open next de ImageJ, para lo cual es necesario haberla duplicado antes*, 2- las imágenes mosaico deben rotarse 90 grados a la derecha para corregir la rotación introducida por el escáner y asegurar que las coordenadas obtenidas de la imagen se corresponden con las de la platina (Figura 30 y Figura 36: 1), 3- el umbral de binarización de los pocillos se ajusta automáticamente mediante el comando Auto

* Open next es un comando de ImageJ que abre la imagen siguiente a otra abierta con anterioridad. Al hacerlo, cierra la primera, por lo que si se quiere conservar esta, conviene duplicarla antes.

Local Threshold de ImageJ usando el método Niblack. El umbral de segmentación de las células debe asignarlo el experimentador empíricamente antes de ejecutar la macro, e introducirlo en la GUI de iMSRC, 4- el conteo de pocillos que contienen una única célula se detiene cuando se analizan todas las ROIs o cuando se alcanza un límite. El límite lo determina el usuario modificando el valor de la variable *limit* directamente sobre el código de la aplicación. Esto evita obtener más posiciones de las que realmente se pueden capturar. En este caso, el límite se fijó en 100 pocillos, que es el número máximo de capturas que permiten completar un ciclo cada 10 minutos, 5- la imagen binara con los pocillos que contienen una única célula se obtiene de la imagen binaria que contiene las células. Para hacerlo, se selecciona la totalidad del área de esta imagen y se rellena de negro, a continuación, se seleccionan las ROIs de los pocillos con una única célula del ROI Manager y se rellenan de blanco. Las coordenadas del centro de masas de los pocillos se obtienen analizando la imagen generada (Figura 36: 4).

SEGUNDA CAPTURA

La captura de los mosaicos de búsqueda y su análisis en iMSRC no excede de los 10 minutos por chip. El resultado es un archivo con las condiciones de captura de alta resolución y las coordenadas de los pocillos que contienen una única célula autofluorescente. Este puede cargarse en Matrix Screener para realizar un experimento de videomicroscopía en el que se capture una fotografía en foco de dos canales (verde y campo claro), cada 10 minutos, durante 72 horas, para cada una de las coordenadas detectadas.

La segunda captura produce una colección de imágenes autoexportadas de Matrix Screener (como la descrita en la Figura 9), que contiene 100 carpetas Field, 2 canales por carpeta y 432 imágenes por canal (una imagen cada 10 minutos durante 72 horas).

PROCESADO DE IMÁGENES

El resultado informativo para este tipo de experimento es un video multicanal por posición, que muestre las células positivas y negativas para el biomarcador. Para obtenerlo, se repite el siguiente procedimiento para cada uno de los 100 pocillos capturados: 1- abrir ordenadamente todas las imágenes de campo claro en una pila, 2- abrir ordenadamente todas las imágenes del canal verde en una pila, 3- contrastar el canal verde, 4- fusionar los dos canales, 5- guardar el resultado en formato .avi con el nombre del campo.

El procesado de los resultados se llevó a cabo mediante una macro modificada de la vista con anterioridad para la captura de mosaicos (ver Figura 27 y nota al pie en la página 95).

El resultado obtenido se resume en la Figura 38.

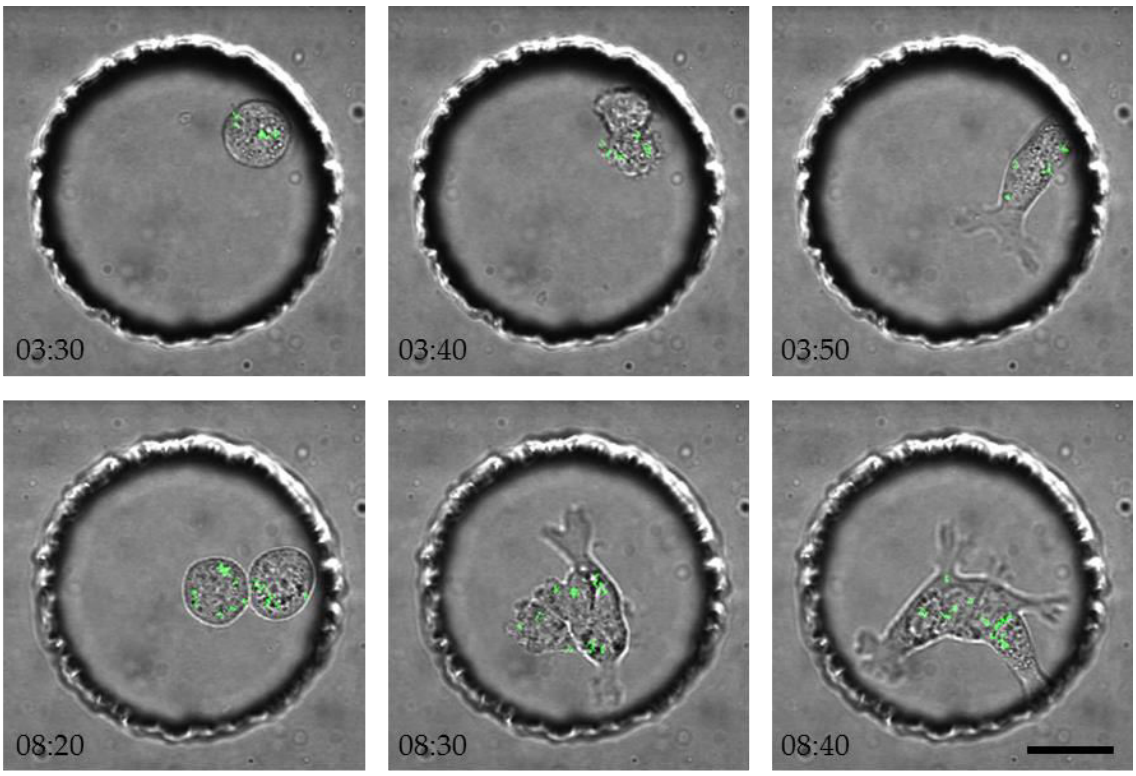


Figura 38. Secuencia temporal de imágenes de un pocillo que inicialmente contenía una única célula. Se muestran dos divisiones mitóticas consecutivas. Los números indican el intervalo de tiempo en formato horas: minutos (hh:mm). La barra de escala mide 20 micras.

OBJETIVO 6. CAPTURA INTELIGENTE DE *TISSUE MICROARRAYS*.

La captura automatizada de TMAs se ha realizado en nuestra unidad en microscopios confocales, en microscopios de fluorescencia convencionales, capturando piezas de tejido completas y capturando un número limitado de campos por pieza de tejido. Todos estos procedimientos son similares, se muestra aquí la captura automatizada de un máximo de 5 campos por tejido de TMA en un microscopio fluorescente convencional. Los lectores interesados en conocer los detalles de la captura automatizada de los tejidos completos de un TMA utilizando un microscopio confocal, podrán encontrarlos en las siguientes publicaciones: Carro et al. (2015) (Figura 5) y Alonso-Curbelo et al. (2014).

El procedimiento mostrado a continuación se diseñó para ser llevado a cabo por parte del personal técnico de la Unidad de Microscopía Confocal, utilizando un microscopio fluorescente Leica DMI6000 B. Las características del equipo pueden consultarse en el apartado de material y métodos.

Poner a punto un sistema de captura de TMAs, basado en la aplicación iMSRC, requiere: 1- configurar perfiles de primera y segunda captura, 2- desarrollar una rutina de análisis para la detección y extracción de coordenadas (TMA Field Sampler, Figura 4: D y Figura 5), y 3- desarrollar una rutina para el procesado y visualización de las imágenes obtenidas (TMA Manager).

CONFIGURACIÓN DE LOS PERFILES DE PRIMERA Y SEGUNDA CAPTURA

Los perfiles de la primera y segunda captura, utilizados en el experimento mostrado como ejemplo en la Figura 40 y en la Figura 45, respectivamente, pueden encontrarse en el apartado de material y métodos.

Los perfiles de la primera captura se diseñaron para permitir diferenciar tejidos y para la que la captura y fusión de mosaicos fuera rápida. Para ello, se utilizó un objetivo de bajo aumento, se minimizó el número de canales, y se incrementó el *binning* y la ganancia de la cámara en la medida de lo posible. La captura y fusión de campos se realizó en Matrix Screener, con la función de autoexportado en .tif activada.

El resultado de la primera captura es un mosaico de búsqueda (ver Figura 40: A y Figura 45: 1, más adelante), que muestra la autofluorescencia de los tejidos de un TMA.

Los perfiles de la segunda captura se diseñaron para obtener imágenes a una resolución espacial tal que permitiera visualizar inequívocamente la marca fluorescente. Para hacerlo, se utilizó un objetivo de gran aumento y se minimizó el

binning. Se capturaron 5 planos focales, para garantizar la captura de señal en varios planos de los tejidos.

ANÁLISIS DE IMAGEN (TMA FIELD SAMPLER)

TMA Field Sampler es una macro de análisis que permite extraer las coordenadas de un número máximo de campos por tejido. Se diseñó para ser utilizada por personal de la Unidad de Microscopía Confocal, con conocimientos especializados en procesado de imágenes.

La Figura 39 muestra el diagrama de flujo de TMA Field Sampler, su código completo puede encontrarse en el apéndice.

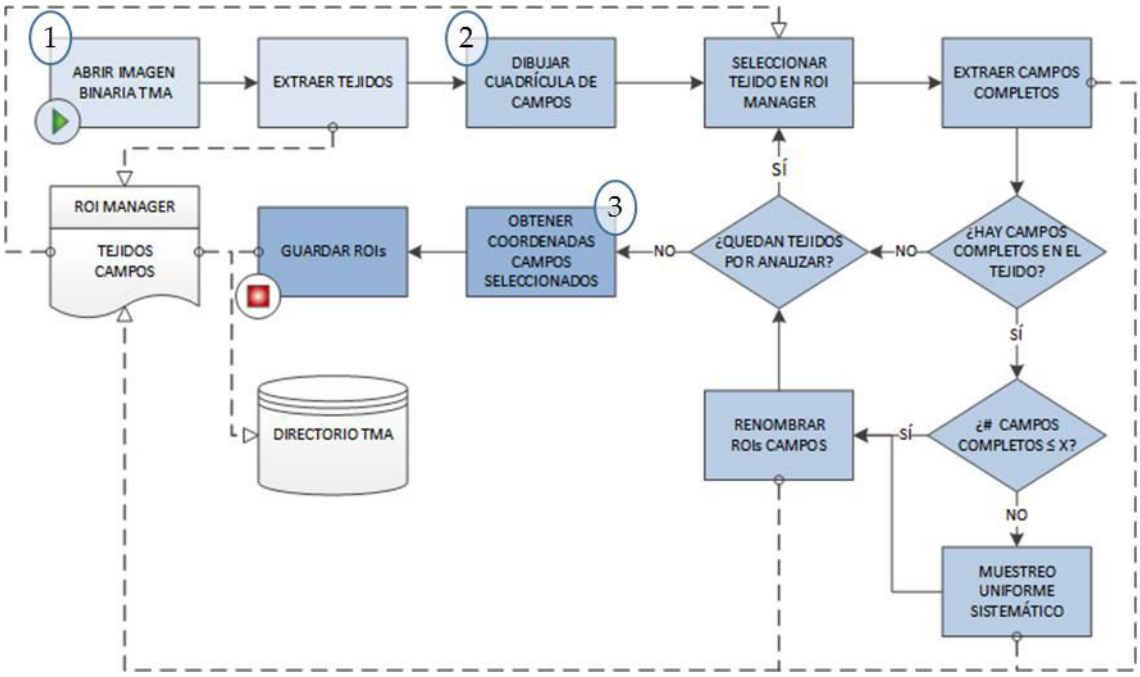


Figura 39. Diagrama de flujo de TMA Field Sampler. Esta macro selecciona un número de campos por tejido menor o igual a un número "X", y extrae sus coordenadas para su posterior captura automática. **1**: parte de la aplicación responsable de almacenar los perfiles de los tejidos en el ROI Manager. **2**: bloque responsable de obtener un número de campos por tejido igual o menor a un número "X" y almacenarlos en el ROI Manager. "X" es un número personalizable, almacenado en la variable FIELDS_TO_SAMPLE_PER_SLIDE. **3**: parte responsable de obtener las coordenadas de los campos almacenados en el ROI Manager y de guardar las selecciones de campos y tejidos. El ROI Manager contendrá las selecciones de tejidos y campos resultantes de varias operaciones de análisis, nombradas de forma intuitiva, estas se guardarán por separado para ser utilizadas posteriormente por TMA Manager.

TMA Field Sampler se ejecuta directamente desde iMSRC (ver nota al pie en la página 114). En su interfaz gráfica de usuario, se deben introducir las rutas a la imagen mosaico y a los perfiles de primera y segunda captura. iMSRC generará directamente los perfiles de segunda captura modificados, que se pueden cargar en Matrix Screener

para fotografiar automáticamente los objetos detectados. La macro funciona en batch y carece de GUIs, por lo que no muestra ninguna información al usuario.

Para que esta macro funcione correctamente, después de la captura del mosaico de búsqueda y antes de ejecutar TMA Field Sampler, se deben realizar manualmente una serie de acciones: 1- duplicar la imagen mosaico en el mismo directorio y nombrarla añadiendo el sufijo "–Copy" al nombre original, 2- binarizar la imagen mosaico original y sobrescribir el resultado (tras la binarización, Figura 40: B, el valor de cada uno de los píxeles de los tejidos del TMA debe valer 1 ,255 en ImageJ, y el del resto de la imagen 0), y 3- adaptar el código de la aplicación a cada caso concreto, asignando a las constantes *FIELDS_TO_SAMPLE_PER_TISSUE* y *GRID_SIZE*, el número máximo de campos que se desean capturar por tejido y el valor en píxeles (en el mosaico de búsqueda) del lado de un campo de la segunda captura, respectivamente.

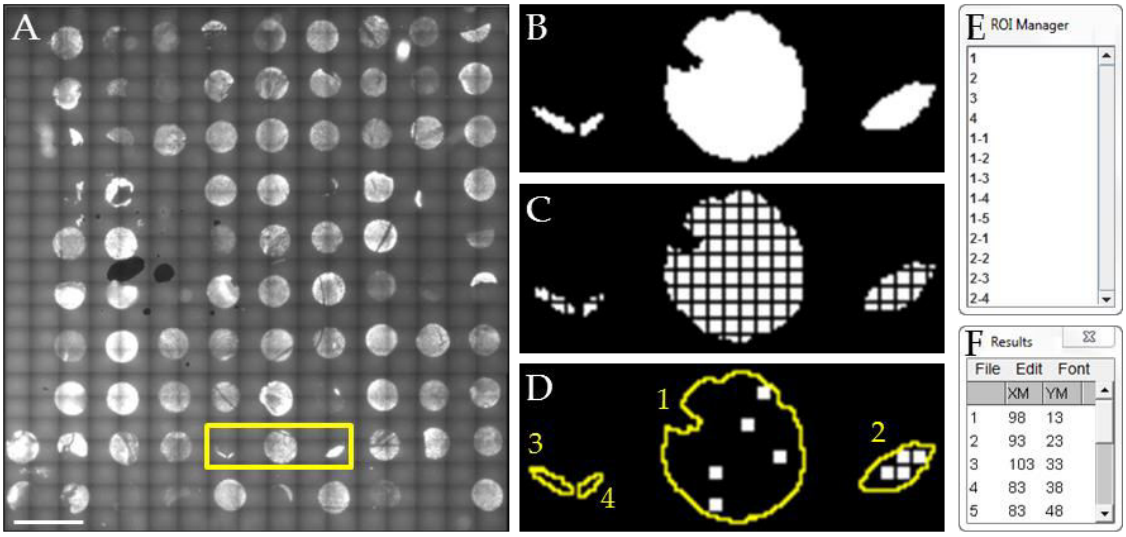


Figura 40. Ejemplo de funcionamiento de TMA Field Sampler. Mosaico de búsqueda de un TMA en escala de grises (**A**), binarizado (**B**), tras superponer una cuadrícula (**C**) y tras muestrear un máximo de 5 campos completos por tejido (**D**). **E** muestra la codificación de las selecciones de los tejidos (del 1 al 4) y campos (1-1, 1-2, etc.) en el ROI Manager. **F** muestra la ventana de resultados con las coordenadas de los campos seleccionados. B, C y D son un detalle tomado del inserto en A. D muestra la selección y nombre de los tejidos en el ROI Manager (E) en amarillo. Barra de escala: 2 mm.

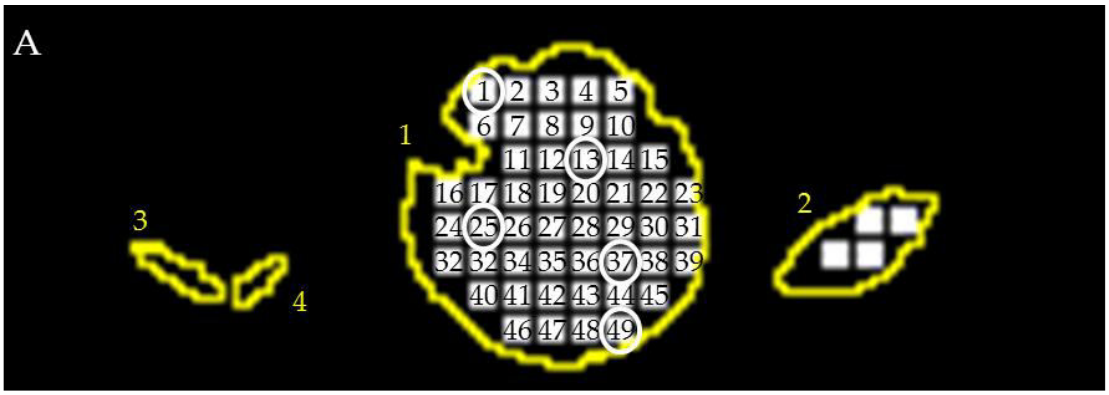
Para extraer de la imagen mosaico binaria las coordenadas de los campos por tejido que el usuario desea fotografiar, TMA Field Sampler realiza tres pasos: 1- obtener las selecciones de los tejidos y almacenarlas en el ROI Manager (Figura 39: 1), 2- a partir de las selecciones anteriores, conseguir las selecciones de los campos a capturar y almacenarlas en el ROI Manager (Figura 39: 2), y 3- obtener las coordenadas de los campos seleccionados en 2 (Figura 39: 3).

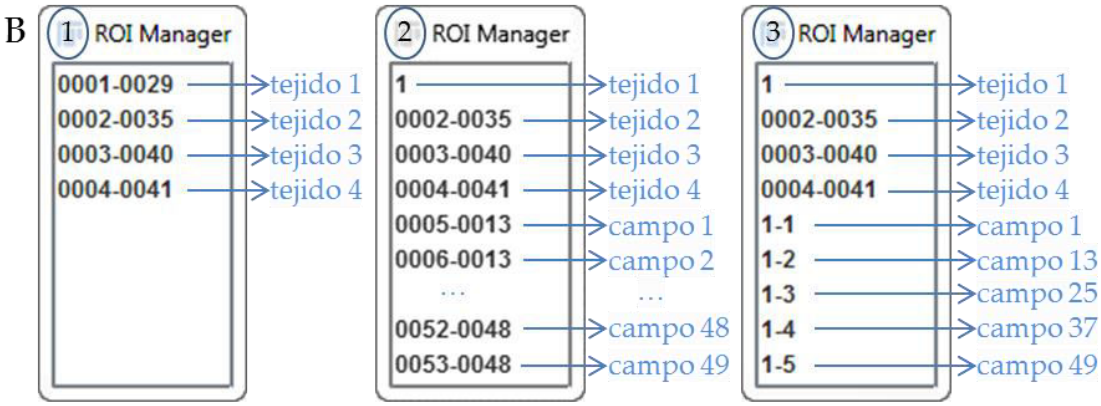
1- Obtener las selecciones de los tejidos (Figura 39: 1) en el ROI Manager es sencillo, solo hay que abrir la imagen mosaico binaria (Figura 40: B) y ejecutar el comando Analyze Particles de ImageJ con la opción Add to Manager seleccionada.

2- Obtener las selecciones de los campos a fotografiar en la segunda captura (Figura 39: 2) es más complejo, dado que, habitualmente, el número de campos completos por tejido no coincidirá con el número de campos que el usuario quiere capturar. Por ejemplo: habrá tejidos en los que no se pueda capturar ningún campo completo, tejidos en los que se pueda capturar menos campos que los que el usuario desea capturar y tejidos en los que se pueda capturar más campos que los que el usuario quiere capturar (Figura 40: C). Para solucionarlo, se obtiene el número de campos completos por tejido, se compara con el número de campos que el usuario desea capturar y, en función del resultado, se realizan una serie de acciones distintas, como se verá a continuación.

Para obtener el número de campos por tejido, se dibuja, sobre la imagen mosaico binaria del TMA, una cuadrícula del tamaño de *GRID_SIZE* (Figura 40: C). A continuación, se procesan todos los tejidos de forma secuencial, del primer al último elemento del ROI Manager, de la siguiente manera: se selecciona un elemento del ROI Manager, se renombra con un ordinal y se ejecuta el comando Analyze Particles, con la opción Add to Manager activada y limitando los resultados a aquellos objetos cuyo tamaño sea igual al área de un campo completo de la cuadrícula (*GRID_AREA*, calculado a partir de *GRID_SIZE*, Figura 41: C, línea 5). Al hacerlo, aparecen en el ROI Manager los perfiles de los campos completos de un tejido (Figura 41: B). Estos se pueden contar comparando el número de elementos en el ROI Manager antes y después de cada operación de análisis (*nFieldsPerTissue*, Figura 41: C, línea 7).

Para obtener el número de campos a capturar por tejido, si no hay ningún campo completo dentro del tejido seleccionado (Figura 40: C y D, tejidos 3 y 4, Figura 39: 2, "¿Hay campos completos en el tejido?") no se hace nada. Si el número de campos completos es menor o igual al máximo de campos a muestrear por tejido (*FIELDS_TO_SAMPLE_PER_TISSUE*, Figura 40: C y D, tejido 2 y Figura 39: 2, "¿# campos completos ≤ X?"), estos se seleccionan en el ROI Manager y se renombran con números siguiendo el patrón: "tejido-campo" (2-3, por ejemplo, se corresponde con el tercer campo completo del segundo tejido, Figura 40: E). Si hay más campos completos que el número máximo de campos que se quiere capturar por tejido (Figura 40: C y D, tejido 1 y Figura 39: 2, "¿# campos completos ≤ X?"), se realiza una selección de campos, se renombran como en el caso anterior y se eliminan los campos no seleccionados (Figura 41).





C

```
for(i=1;i<=TISSUE_SLICES;i++)
{//bucle para cada tejido
 initialObjectsNumber=roiManager("count");                    → 4
 roiManager("select", i-1);
 roiManager("Rename", i);
 run("Analyze Particles...", "size="+GRID_AREA+"-"+GRID_AREA+" add");
 finalObjectsNumber=roiManager("count");                      → 53
 nFieldsPerTissue=finalObjectsNumber-initialObjectsNumber;
                        ...                                   → 49
 fieldsToEraseArray=newArray(nFieldsPerTissue-FIELDS_TO_SAMPLE_PER_TISSUE);
 tissueFieldOrdinal=1;                                        → 5
 numberOfSampledFields=0;
 numberOfFieldsToErase=0;
 samplingPeriodicity=
   floor((nFieldsPerTissue-FIELDS_TO_SAMPLE_PER_TISSUE)/(FIELDS_TO_SAMPLE_PER_TISSUE-1))+1;
 for(j=initialObjectsNumber; j<finalObjectsNumber;j++)        → 12
 {
  if(tissueFieldOrdinal==1+(samplingPeriodicity*numberOfSampledFields)
                                  && numberOfSampledFields<FIELDS_TO_SAMPLE_PER_TISSUE)
  {
   roiManager("select", j);
   numberOfSampledFields++;
   roiManager("Rename", i+"-"+numberOfSampledFields);
  }
  else
  {
   fieldsToEraseArray[numberOfFieldsToErase]=j;
   numberOfFieldsToErase++;
  }
  tissueFieldOrdinal++;
 }
    roiManager("select", fieldsToEraseArray);
    roiManager("Delete");
}//fin del bucle para cada tejido
```

Figura 41. Ejemplo de selección de campos cuando el número de campos por tejido (*nFieldsPerTissue*) es mayor al número de campos por tejido que se quiere seleccionar (*FIELDS_TO_SAMPLE_PER_TISSUE*). **A**: muestra los perfiles de 4 piezas de tejido (en amarillo) y los campos completos en cada uno de ellos (cuadrados blancos). Los números en amarillo indican la pieza de tejido, los números en negro, los campos completos del primer tejido. Los círculos blancos muestran los campos seleccionados al ejecutarse el código en C. **B**: ROI Manager, antes (**1)** y después (**2**) de extraer los campos completos de un tejido (líneas 1 y 6 en C) y al finalizar el bucle para el primer tejido en C (**3**, línea 32 en C). Se muestran en azul las correspondencias entre las selecciones en A y los elementos del ROI Manager, así como la posición de cada elemento dentro del mismo. Los puntos suspensivos indican omisión de elementos. **C**: parte del código de TMA Field Sampler responsable de la selección de campos en el ejemplo descrito. Las flechas azules apuntan al valor de algunas variables. *GRID_AREA* es el área en píxeles de un campo completo. Los puntos suspensivos indican omisión de código.

Lo mismo se repite para cada pieza de tejido, de manera que, al finalizar el bucle, se obtiene una selección de tejidos en el ROI Manager numerados por orden, y un conjunto de campos completos nombrados ordenadamente por pieza de tejido (Figura 39 "ROI Manager" y Figura 40: E).

3- Obtener las coordenadas de los campos seleccionados (Figura 39: 3) es sencillo: se seleccionan una a una las selecciones de los campos en el ROI Manager (Figura 40: E) y se obtienen sus coordenadas (Figura 40: F) utilizando el comando Measure de ImageJ. La selección de tejidos y la selección de campos se guardarán, de forma separada (con los nombres "tissueRois-Roiset" y "picturesRois-Roiset", respectivamente), en el directorio de la imagen mosaico (Figura 39: "Directorio TMA"). Ambas serán utilizadas por la aplicación TMA-Manager (como se verá más adelante).

SEGUNDA CAPTURA

El resultado del paso anterior son unos perfiles de segunda captura modificados, que el usuario puede cargar en Matrix Screener para capturar los campos seleccionados.

Se capturaron 5 planos focales y tres canales en un total de 396 posiciones distribuidas en 80 piezas de tejido. El resultado fueron 5940 archivos .tif, en una colección de imágenes autoexportadas de Matrix Screener (como la descrita en la Figura 9), con 396 carpetas Chamber, cada una con una sola carpeta Field y dentro de ella, 15 imágenes de 3 canales y 5 planos focales.

PROCESADO DE IMÁGENES (TMA MANAGER)

Para poder interpretar los resultados obtenidos, el usuario necesita visualizar las imágenes capturadas y compararlas entre ellas. La colección de imágenes resultado de la segunda captura tiene dos inconvenientes: 1- las imágenes no están procesadas para la fusión de canales, proyección de planos y rotación del escáner; y 2- el usuario no conoce a qué pieza de tejido pertenece cada una de las imágenes capturadas.

La aplicación TMA Manager soluciona estos problemas utilizando las selecciones de los campos y de los tejidos generadas por TMA Field Sampler. Para hacerlo, realiza dos funciones: 1- fusiona los canales, proyecta los planos, rota las imágenes y las guarda en una estructura de carpetas nueva, con una nomenclatura sencilla que identifica las imágenes capturadas con la pieza de tejido de la que procede; y 2- genera un visor de datos que permite al usuario navegar por el mosaico del TMA y visualizar las imágenes de los tejidos haciendo clic sobre los mismos. La parte de TMA Manager responsable del procesado de datos se ideó para ser utilizado por parte de personal especializado, el visor de datos se diseñó para ser utilizado por parte de personal sin formación en procesado y análisis de imagen.

La Figura 42 muestra el diagrama de flujo de la aplicación.

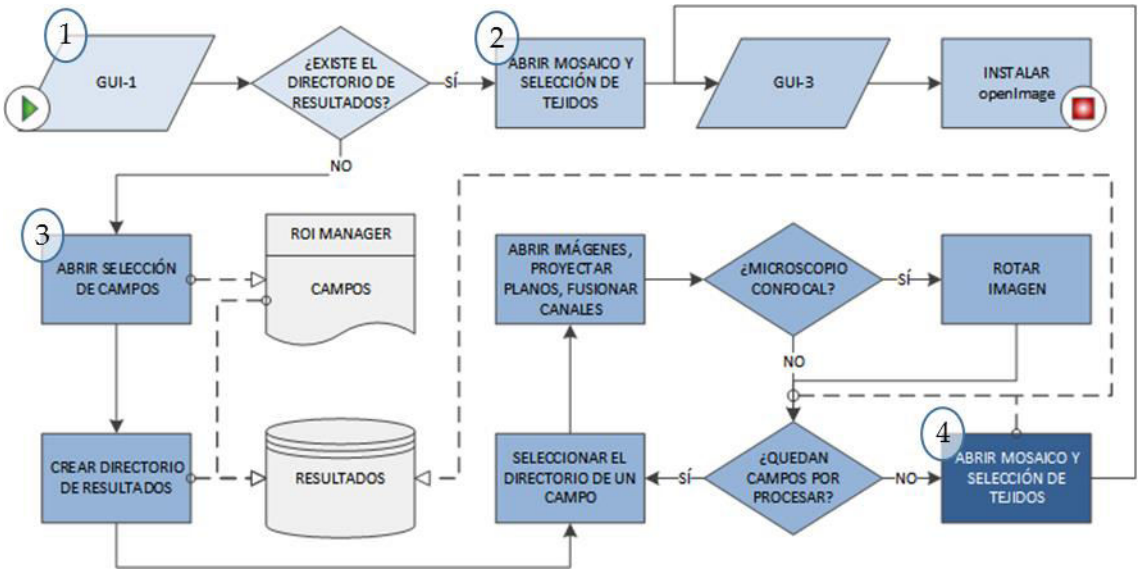


Figura 42. Diagrama de flujo de TMA Manager. Esta aplicación: 1- crea una colección de tifs multicanal, proyectados e interpretable por el usuario, a partir de una colección de archivos .tif de varios canales y planos, no interpretable por el usuario, y 2- genera una interfaz que permite abrir y visualizar las imágenes de forma intuitiva. **1**: GUI-1 permite introducir los directorios de las imágenes y de las selecciones del ROI Manager y definir si las imágenes se capturaron en un microscopio confocal. **2**: parte del código responsable de la visualización de imágenes. GUI-3 muestra la imagen mosaico del TMA con la selección de tejidos (tissueRois-Roiset) superpuesta. La función *openImage* permite abrir y visualizar las imágenes capturadas en la pieza de tejido seleccionada en GUI-3. **3**: parte del código responsable del procesado de las imágenes, solo se ejecuta si estas no están ya procesadas. Las imágenes se guardan en un directorio ("RESULTADOS"), distribuidas y nombradas de forma intuitiva. **4**: Creación y guardado de una imagen de referencia. No se muestra la interfaz alternativa de obtención de directorios por navegación GUI-2.

TMA Manager debe implementarse como parte de la rutina de captura de TMAs que usa iMSRC, una vez se hayan realizado la primera y segunda capturas, y se haya ejecutado la macro TMA Field Sampler. Así, en el momento de su ejecución, existirán dos colecciones de imágenes autoexportadas de Matrix Screener (similares a la de la Figura 9):

1- El resultado de la primera captura de iMSRC, que contendrá, dentro de la carpeta Chamber--U00--V00, como se ha explicado anteriormente, además de los campos del mosaico:
    - La imagen mosaico binaria del TMA (tal y como la nombró la aplicación de autoexportado de MatrixScreener).
    - La imagen mosaico en escala de grises del TMA (que será una copia de la anterior antes de binarizar y se llamará igual con el sufijo - Copy).
    - El conjunto de selecciones de los tejidos y de los campos capturados (tissueRois-Roiset y picturesRois-Roiset, respectivamente).
2- El resultado de la segunda captura de iMSRC.

Para que TMA Manager funcione correctamente, los archivos openImage.ijm y openImageMacroResilence.txt deben estar instalados en la carpeta Macros de FIJI. openImage.ijm contiene la función *openImage**, que será utilizada por TMA Manager para abrir las imágenes seleccionadas por el usuario. openImageMacroResilence.txt es un archivo de texto en blanco en el que se escribirá la ruta a la carpeta de resultados generada por TMA Manager, necesaria para la correcta ejecución de *openImage*.

TMA Manager tiene tres partes: 1- una GUI para facilitar la introducción de directorios (Figura 42: 1), 2- un bloque responsable de procesar las imágenes de la segunda captura, generar un directorio de resultados intuitivo (Figura 42: 3) y una imagen de referencia (Figura 42: 4), y 3- una parte encargada de generar un visor de imágenes (Figura 42: 2).

1- Cuando se ejecuta la aplicación, se genera una interfaz gráfica de usuario (Figura 42: 1 y Figura 43) con el mismo diseño seguido en la aplicación Opera Exporter (Figura 18: A, ver nota al pie de la página 81), con dos cuadros de texto, 3 casillas de verificación y texto con instrucciones. Los dos primeros cuadros de texto y casillas de verificación (Figura 43: 1 y 2) servirán para que el usuario introduzca las rutas a las carpetas Slide--S00 de la primera y segunda captura de iMSRC, respectivamente (Figura 9), su valor se asignará a las variables *first* y *secondScanSlidesDir*. La última casilla de verificación (Figura 43: 3) debe marcarse si la captura se realizó en un microscopio confocal o dejarse sin marcar si se hizo en un microscopio fluorescente convencional, su valor se asignará a la variable *confocalMicroscope*. *first* y *secondScanSlidesDir* permitirán obtener las rutas a la imagen mosaico en escala de grises del TMA, a tissueRois-Roiset, a picturesRois-Roiset, al directorio de resultados, a las imágenes a visualizar y a las imágenes a

* La función *openImage*, no pertenece a la API del lenguaje macro de ImageJ, se declara y se define en el código del archivo openImage.ijm.

procesar, las cuales se utilizarán más adelante en la aplicación. *confocalMicroscope*, permitirá corregir la rotación del escáner, rotando las imágenes capturadas con un confocal 90 grados a la derecha (Figura 30).

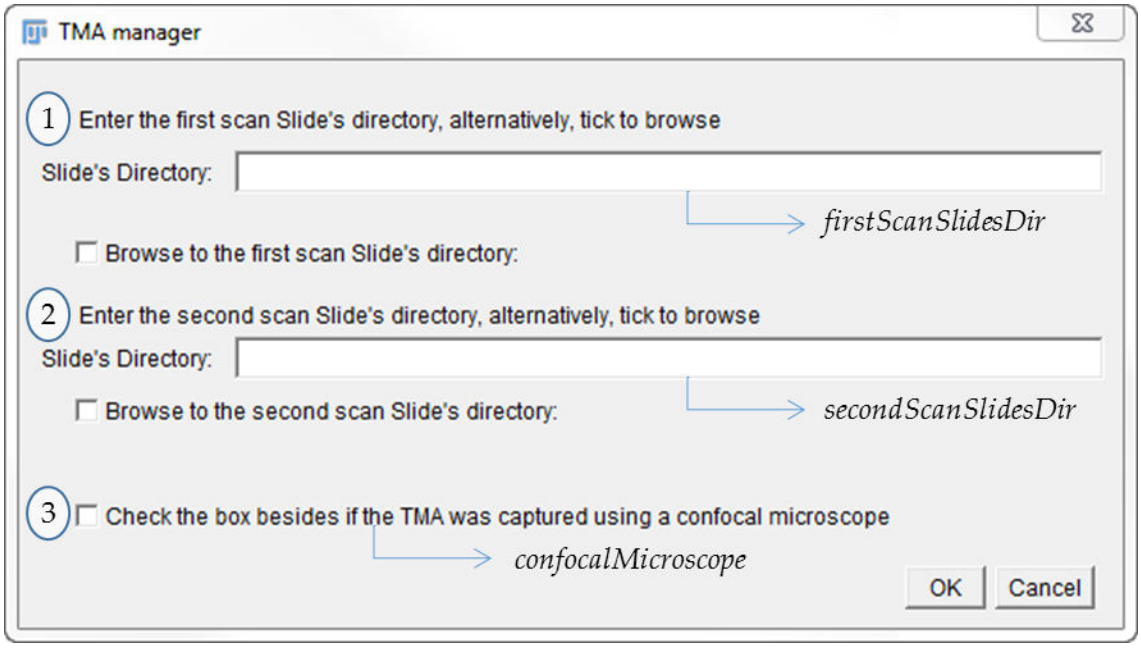


Figura 43. Primera interfaz gráfica de usuario (GUI-1) de TMA Manager. Permite introducir las rutas a las imágenes de la primera (**1**) y segunda capturas (**2**). Si se marca la casilla de verificación en **3**, las imágenes se rotarán 90 grados a la derecha para corregir la rotación del escáner. No se muestra GUI-2, responsable de introducir directorios usando un navegador. Las flechas apuntan a las variables a las que se asignarán los valores introducidos por el usuario.

2- Al pulsar *OK* en GUI-1, el programa genera la ruta a una carpeta de resultados (*resultsDir*) y la guarda en el archivo openImageMacroResilence.txt. A continuación, verifica si este directorio existe (Figura 42: 1, "¿Existe el directorio de resultados?"). Si el directorio de resultados no existe, se crea y se guarda en él el resultado de la fusión de canales, la proyección de planos (si hay más de uno) y la rotación (si procede de un microscopio confocal) de cada una de las imágenes tomadas en la segunda captura (Figura 42: 3 y Figura 44). El código para la fusión de canales y proyección de planos se escribe para cada caso concreto modificando el bloque de código en la línea 99 de la aplicación*. A continuación, se abrirá la imagen mosaico en escala de grises y la selección de tejidos (tissueRois-Roiset) en el ROI Manager, se superpondrá la selección de tejidos a la imagen (Figura 42: 4) y se guardará en el directorio de resultados, como imagen de referencia, para que el usuario pueda consultarla cuando lo desee (esta imagen es muy similar a GUI-3, ver Figura 45: 1, más adelante). La carpeta de resultados tiene una estructura comprensible, con todas las imágenes

* Este código se tomó de iMSRC Stitching Module. Ver nota al pie de la página 95 y Figura 27.

capturadas ordenadas en una carpeta por pieza de tejido, nombrada como las piezas de tejido en la imagen de referencia (Figura 44). Por ejemplo, la carpeta 1 de la Figura 44: 3, contendría todas las imágenes capturadas a grandes aumentos del tejido nombrado como 1 en la imagen de referencia.

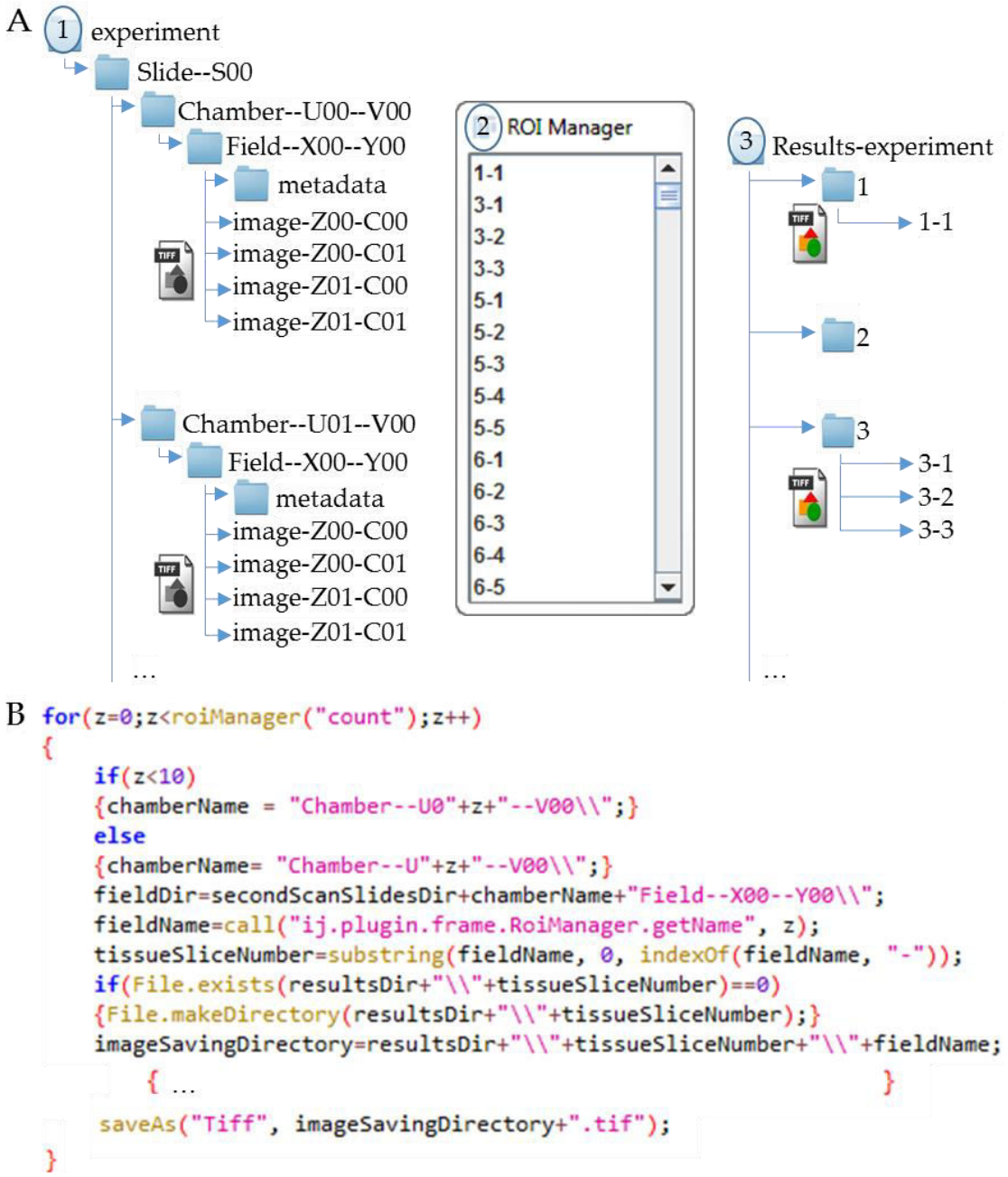


Figura 44. Algoritmo para la generación de una estructura de carpetas ordenada por piezas de tejido. **A**: La aplicación TMA Manager convierte una colección de imágenes no intuitiva (**A: 1**), en una colección de imágenes procesadas e interpretable por el usuario (**A: 3**), a partir del nombre de las selecciones de campos capturados en el ROI Manager (**A: 2**). Los puntos suspensivos indican estructuras que se repiten. Se han acortado algunos nombres en A: 1 por motivos didácticos. **B** muestra el código encargado de la generación de directorios, los puntos suspensivos indican la posición del código responsable de abrir y procesar las imágenes, que debe adaptarse para cada caso (no mostrado). *SecondScanSlidesDir* y *resultsDir* contienen los directorios a las carpetas experiment y Results-experiment, respectivamente.

Si, al pulsar el botón *OK* tras la primera interfaz de usuario, el programa detecta que existe un directorio de resultados (Figura 42: 1, "¿Existe el directorio de resultados?"), se ejecuta directamente el código para la visualización de imágenes (Figura 42: 2), omitiéndose todo el bloque de procesado. Esto permite ahorrar una gran cantidad de tiempo de computación en aquellos casos en que se deseen visualizar imágenes ya procesadas previamente.

3- La generación de un visor de datos para visualizar las imágenes capturadas, se basa en la creación de una interfaz gráfica de usuario (GUI-3, Figura 42: 2). Esta interfaz tienen 3 componentes: 1- la imagen mosaico del TMA en escala de grises, 2- el ROI Manager con las selecciones de los tejidos (tissueRois-Roiset), y 3- un cuadro con instrucciones (Figura 45). En el caso de que sea la primera vez que se ejecute la macro, los componentes 1 y 2 ya están abiertos (se abrieron en el bloque de procesado, para formar la imagen de referencia), en el caso de que no sea la primera vez que se ejecute, estos se abren. A continuación, se muestra el mensaje <<Click OK, then click on ROI´s number and press "o" to open images.>> (Figura 45). Al pulsar OK, la aplicación instala la macro openImage, salvada en el directorio de macros de FIJI y finaliza (Figura 42: 2). Cuando el usuario pulsa sobre el nombre de una selección de tejido (ya sea en el ROI Manager o en la imagen) y pulsa la tecla "o", se ejecutará la macro openImage. Esta abrirá todas las imágenes que se capturaron en ese tejido en forma de una pila de imágenes, junto con la herramienta de modificación de brillo y contraste. En este punto, el usuario dispone de toda la funcionalidad de FIJI para procesar, analizar, guardar, etc. las imágenes a su gusto.

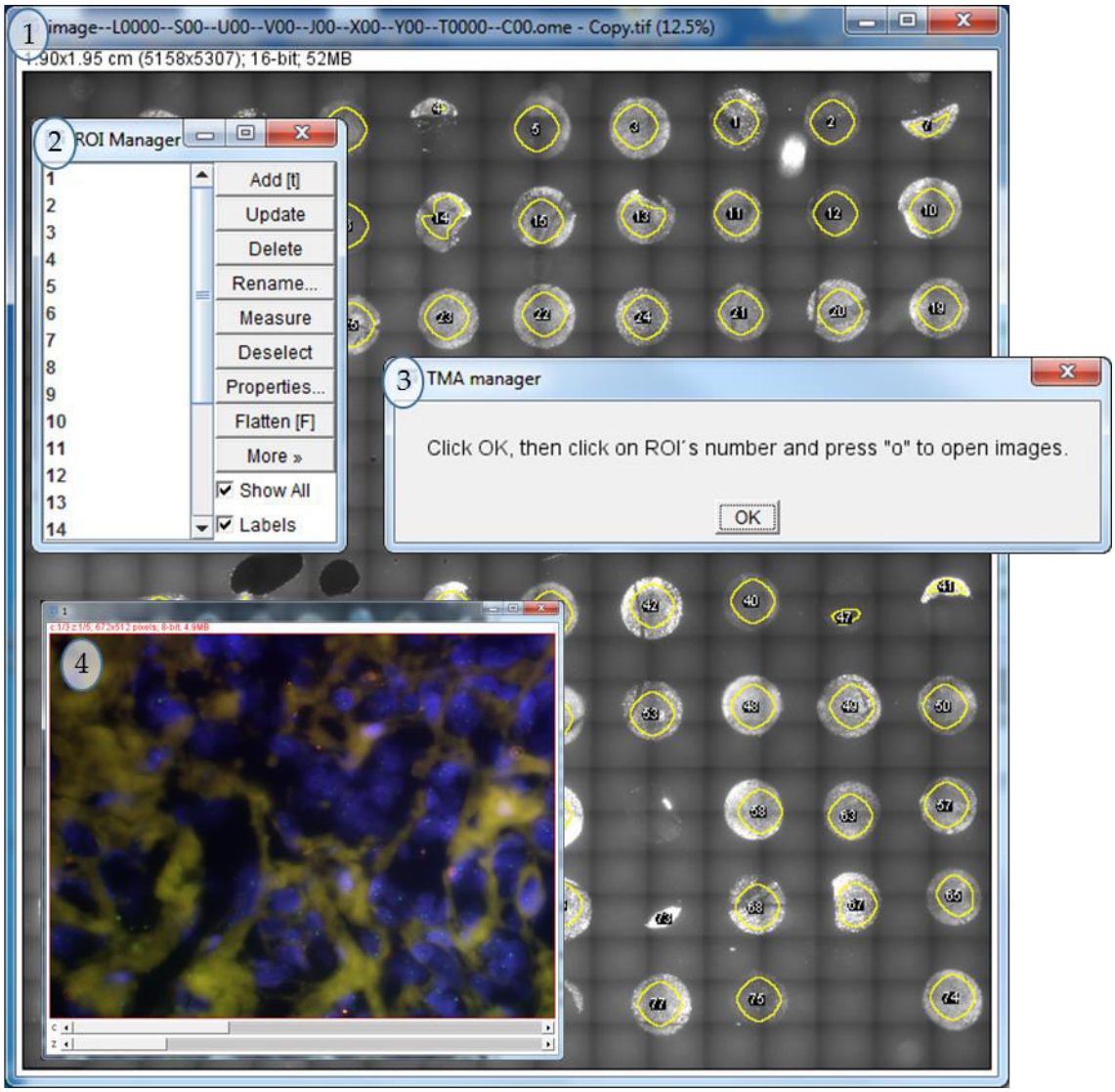


Figura 45: tercera interfaz gráfica de usuario de TMA Manager (GUI-3). Esta interfaz consta de tres elementos: **1**: la imagen mosaico del TMA, **2**: el ROI Manager y **3**: un texto con instrucciones. El ROI Manager (2) contiene las selecciones de los tejidos (tissueROI_ROIset), las cuales se muestran superpuestas en la imagen mosaico del TMA (1). Al hacer clic sobre la selección de un tejido en el ROI Manager o en la imagen, y pulsar la tecla "o", se abren las imágenes de los campos de ese tejido en una pila (**4**), cada una de ellas tiene los canales fusionados, los planos proyectados y está rotada (si se capturó en un microscopio confocal). Una copia de la imagen mosaico del TMA con las selecciones de los tejidos superpuestas se guarda en el directorio de resultados como imagen de referencia, para que el usuario pueda consultarla cuando lo desee.

El código completo de la función *openImage* puede encontrarse en el apéndice.

Su funcionamiento es muy sencillo: cuando el usuario activa una selección de un tejido y pulsa la tecla "o", la macro extrae el nombre de la selección en el ROI Manager y lo concatena a la ruta al directorio de resultados (*resultsDir*, salvada en openImageResilence.txt), a continuación abre todas las imágenes en esa ruta como una pila.

Como término medio, la duración de todo el procedimiento (primera y segunda capturas, ejecución de las macros, etc.) en el microscopio fluorescente, suele ser de 20 minutos, 30 en el caso del microscopio confocal.

Para validar el funcionamiento de la macro TMA Field Sampler, se obtuvo una imagen mosaico del TMA con las selecciones de los campos superpuestas. A continuación, se revisó en Matrix Screener que cada una de las posiciones de la segunda captura, se correspondía con las posiciones de las selecciones en la imagen del TMA.

Para validar el funcionamiento de la macro TMA Manager, se contrastó el procesado automático de las imágenes con el manual, para ver que el resultado era el mismo. Para comprobar el correcto funcionamiento del visor de datos, se abrieron imágenes al azar usando el visor y se comprobó en Matrix Screener, con los perfiles de segunda captura, que las imágenes abiertas de cada pieza de tejido pertenecían a la misma.

DISCUSIÓN

<<– Vayamos a los jardines. Porque ¿acaso no está escrito: "Un poco de aire fresco te vendrá bien"?
– ¿Lo está?
– Lo pone aquí. Si lo sabré yo.>>

Terry Pratchet, El ladrón del tiempo.

El *software* libre ha cambiado la forma en que se construyen los programas y cómo se hace negocio con los mismos[34,36]. En su día, la aparición de aplicaciones de complejidad y robustez tales como el sistema operativo GNU/Linux (de miles de millones de líneas de código)[42], demostró que este puede llegar a alcanzar los mismos (o mejores) estándares que el *software* propietario. Pese a todo, alrededor del *software* libre se siguen dando una serie de falsas creencias muy extendidas, como que lo desarrollan amateurs de forma altruista o que es incapaz de generar interés económico. Conviene recordar que, en promedio, el 50% del código del *software* libre lo desarrollan programadores contratados por compañías o instituciones (pudiendo llegar al 95% en algunos casos), siendo estos porcentajes mucho mayores en proyectos de *software* libre liderados por empresas (92% como media, pudiendo llegar al 100% de forma puntual)[47]. Por otra parte, solo en el año 2006, las compañías comerciales invirtieron 1.2 billones de euros en desarrollar *software* libre[46]. Aun así, al *software* libre de uso científico se le sigue atribuyendo, como se ha expuesto en la introducción: 1- problemas de documentación y falta de soporte, 2- baja eficiencia y 3- problemas de responsabilidad legal.

Respecto al primer punto, hay que tener presente que se pueden encontrar proyectos de *software* libre en diferentes grados de madurez[29], que forman parte de proyectos más o menos ambiciosos y que llaman la atención de un mayor o menor número de desarrolladores. Esto ha llevado (tal y como se ha expuesto en la introducción) a que varios autores[20,35,50] hayan hecho una serie de recomendaciones para la selección de *software* libre, con los cuales no podemos estar más de acuerdo. Curiosamente, todos destacan la cantidad y calidad de la documentación y soporte como medida directa de la calidad del proyecto, así como la existencia de una comunidad de usuarios activa a través de varios canales. Nuestra experiencia personal con el proyecto ImageJ/FIJI es muy positiva en este aspecto, y nuestra recomendación no puede ser más clara: no ser usuario de un proyecto de *software* libre si no proporciona suficiente documentación y soporte, bien de forma explícita (a través de cursos, congresos, *wiki*s, etc.) o a través de una comunidad de usuarios muy activa (foros, listas de distribución, etc.).

En cuanto al segundo punto, Krylov et al. (2015) argumentan que el *software* libre favorece la interoperabilidad frente a la eficiencia. Estos autores proceden del campo de la investigación química, en el que experimentos exhaustivos de detección de moléculas con actividad biológica, que generan gran cantidad de volúmenes de datos, imponen condiciones de computación muy exigentes. Otros autores del mismo[58] y otros campos con exigencias similares (como la neurociencia)[131], describen protocolos, basados en *software* libre de reconocido prestigio, que destacan la interoperabilidad de sus plataformas como un valor añadido.

Por último (aunque la responsabilidad del uso de *software* está supeditada a la legislación regional o nacional donde reside el usuario del mismo), la única diferencia

sustancial legal entre *software* libre y propietario de la que tenemos noticia (en el campo del análisis de imagen), es que el *software* libre no puede ser utilizado para el diagnóstico médico. Esto es así porque, en este caso concreto, la ley exige al fabricante que se comprometa, mediante un contrato comercial, a informar al usuario de la existencia de un error (si lo hubiera) y a distribuirle una solución. Esta limitación puede salvarse comercializando *software* libre bajo una licencia dual (*i. e.* Osirix, programa de imagen médica[37,56]), como se vio en la introducción. Hay que tener presente que la legislación no suele regular la eficacia o eficiencia del producto[37], pero sí ampara las cláusulas de limitación de garantía y responsabilidad legal, las cuales, por otra parte, son una práctica muy extendida en las licencias de *software* comercial y libre, por lo que no son una característica exclusiva de ninguno de ellos. En otras palabras: ningún desarrollador garantiza el correcto funcionamiento de sus programas. Desde nuestro punto de vista, toda esta coyuntura implica que la responsabilidad (al menos moral) del empleo de los mismos, queda en manos del usuario, lo cual obliga a que detrás de cada aplicación haya un profesional que sepa lo que está haciendo, lo que debería ser imperativo en cualquier caso.

Finalmente, cabe destacar, y creemos que esta tesis es un claro ejemplo de ello, una de las principales ventajas del *software* libre respecto al propietario: la capacidad de generar nuevas aplicaciones de forma rápida aun en ausencia de interés comercial. Su paralelismo con la actividad científica, en cuanto a su capacidad para facilitar la obtención, divulgación y reproducibilidad de resultados, ya ha sido expuesta anteriormente. En todo caso, en nuestra opinión, los científicos deberían evitar, en la medida de lo posible, por los motivos expuestos en la introducción, el uso de modelos de *caja negra* como el *software* propietario. Quizá no sea necesario hacer público el código fuente para acabar con este tipo de modelos; si las compañías informáticas se muestran recelosas a hacerlo, deberían, como mínimo, publicitar el nombre de los algoritmos que utilizan, manteniendo en secreto su implementación. De esta forma, los usuarios podrían revisar el funcionamiento de los programas y las firmas no verían amenazado su negocio. Como resultado, estas competirían entre ellas en la implementación eficiente de algoritmos. Pese a todo lo expuesto, no pretendemos demonizar al *software* propietario. Las ventajas asociadas al mismo y a los equipos integrados de captura comerciales (facilidad de uso, garantía de un nivel mínimo de desarrollo y documentación, optimización *hardware-software*, etc.) no deben ser subestimadas. Además, actualmente, muchas de las compañías que los producen, están haciendo esfuerzos por dotar a sus productos de mayor flexibilidad* e

* Implementando herramientas de grabado de comandos, editores de macros y lenguajes de guion propios y externos que permiten desarrollar flujos de trabajo y aplicaciones, por ejemplo.

interoperabilidad* y aumentando la diversidad del *hardware* que sus programas pueden controlar.

Esperamos haber convencido a los lectores de la capacidad de ImageJ/FIJI. En todo caso, queda claro que el *software* libre es muy numeroso y fácil de conseguir, pero de calidad variable[20]. Toda esta perspectiva, junto con el panorama de la microscopía actual (elevado grado de automatización de los equipos y producción de gran cantidad de datos que deben ser analizados, procesados, visualizados y mantenidos por herramientas informáticas especializadas) hace prácticamente inevitable que el microscopista tenga que añadir nociones de análisis y procesado de imágenes, informática y programación a su currículo, o al menos, debe conocer, dominar, saber evaluar y elegir la solución que le permita resolver sus necesidades de la forma más eficiente posible[21].

Hemos visto cómo enfoques relativamente sencillos (modelos de desarrollo de *software* ágil, grabado de comandos y programación con lenguajes de guion específicos de programa) nos han permitido elaborar flujos de procesado, análisis y visualización de imágenes. Y cómo estos permiten implementar nuevas herramientas e incrementar la productividad de procesos, el uso de equipos y el tiempo del personal dedicado, en el contexto de la investigación biomédica. Otros abordajes más complejos (programación en lenguajes de propósito general, por ejemplo), podrían haber dado lugar a soluciones más eficientes, pero a costa de dedicar más tiempo a su aprendizaje e implementación. Las aplicaciones desarrolladas en esta tesis muestran un rendimiento más que aceptable, por lo que el modelo utilizado, visto *a posteriori*, permite un buen balance entre los recursos dedicados y el resultado obtenido.

A continuación, realizaremos un análisis pormenorizado de cada una de las herramientas generadas:

DELTAVISION STACK MANAGER

Esta aplicación ha tenido muy buena aceptación por parte de los usuarios, que la han usado regularmente durante un periodo de más de 5 años.

DeltaVision Stack Manager satisfizo con creces los objetivos de su desarrollo en el momento de su lanzamiento: 1-liberar a los técnicos de tareas de adiestramiento en el uso de aplicaciones de análisis de imagen y 2-agilizar el procesado de las imágenes procedentes de este equipo. En cuanto al primer punto, usuarios sin conocimientos especializados son capaces de utilizar esta aplicación tras unos pocos minutos de

* Permitiendo que ejecuten rutinas de ImageJ en su entorno, por ejemplo, o abriendo sus formatos de imágenes: bien permitiendo guardarlas junto con todos sus metadatos como OME-TIFF o importando los mismos a sus formatos propietarios.

ayuda inicial por parte de los técnicos de la unidad, lo cual es, con diferencia, mucho menos tiempo del necesario para enseñar el procesado manual de pilas de imágenes. En cuanto al segundo punto, el tiempo de procesado de las imágenes mediante el uso de la aplicación es mucho menor comparado con el procesado manual, incluso cuando este último se lleva a cabo por parte de personal especializado.

Hoy en día, los avances y corrección de errores en las opciones de importación de archivos .dv de la herramienta OME Bio-Formats (concretamente, la posibilidad de visualizar los datos en forma de *Hyperstacks* y la posibilidad de separar canales y planos focales) ha facilitado significativamente el procesado de estos archivos, especialmente cuando el objetivo es extraer pilas monocanal. En esta situación, esta aproximación funciona con un rendimiento similar al nuestro. Para el resto de casos, DeltaVision Stack Manager sigue siendo más eficiente, por lo que su uso no ha disminuido considerablemente en la unidad.

Posibles actualizaciones de mejora que podrían implementarse son:

1- Actualizar la herramienta a las nuevas capacidades del plugin OME Bio-Formats (especificadas en el párrafo anterior). Permitiría incrementar el rendimiento de la aplicación. En vez de abrir un archivo de un experimento con imágenes fuera y dentro de foco, por ejemplo, se podría explorar la posibilidad de abrir directamente solo los planos focales en foco, esto ahorraría el uso de muchos recursos al equipo, por lo que la macro funcionaría más rápidamente o permitiría procesar videos más largos, con mayor resolución y/o profundidad de color. También se podría explorar la posibilidad de abrir los archivos en pilas de imágenes separadas por canales y planos focales; esto evitaría a la macro tener que obtener estas pilas del archivo original, con el mismo efecto explicado en el caso anterior.

2- Integrar la importación de los archivos .dv dentro de la aplicación. Se podría generar una GUI para introducir el directorio de los archivos a procesar y automatizar su apertura, evitando al usuario tener que hacerlo manualmente.

3- Simplificar la interfaz gráfica de la aplicación. La información referente al número de canales, número de planos focales y LUTs de cada canal, se podría obtener directamente de los metadatos del archivo. Esto permitiría eliminar los campos 2, 3 y 4 de la Figura 14.

4- Actualizar la herramienta para el procesado en serie de múltiples archivos. Actualmente, DeltaVision Stack Manager debe ejecutarse una vez por cada archivo .dv (o lo que es lo mismo, por cada posición espacial en un experimento de videomicroscopía multiposición). Se podría modificar la aplicación para que el usuario la ejecutara una vez y se procesaran todos los archivos .dv de una carpeta de forma automática,

uno detrás de otro, guardándose los resultados en un directorio determinado. Esto incrementaría la productividad de forma muy considerable. Además, su implementación sería relativamente sencilla.

5- Actualizar el *output* para que las pilas de imágenes multicanal resultantes (de haber alguna) tuvieran formato *composite* (permite modificar el brillo y contraste de cada canal independientemente, y visualizar el resultado de forma instantánea) en vez de RGB. Podría ser especialmente útil cuando uno de los canales fuera de campo claro, ya que este tiende a enmascarar los canales de fluorescencia y no puede separarse del resto de colores en una imagen RGB. Esta actualización se podría implementar de forma inmediata, editando unas pocas líneas de código. Sin embargo, no se ha implementado porque no se ha detectado esta necesidad por parte de los usuarios y hacerlo restringiría el uso del resultado a los programas capaces de manejar imágenes en formato *composite*, o bien obligaría a exportar el resultado en RGB.

6- Modificar el algoritmo de la macro. El algoritmo actual extrae todos los canales en foco de una pila, los fusiona a gusto del usuario y luego cierra los canales que este no quiere conservar. Se puede dar la situación de que se extraiga un canal, no se utilice en ningún paso de fusión y luego se cierre, lo cual es un despilfarro de recursos. No es habitual que se capturen canales que luego no se van a utilizar, pero este caso hipotético se podría prevenir extrayendo solo los canales que el usuario va a fusionar o desea conservar. Hacerlo sería muy sencillo, basta con cambiar la macro para que, por ejemplo, solo extraiga el canal 1 si el usuario comunica (a través de GUI-1) su intención de visualizarlo (solo o junto a otros canales).

Pensamos que esta aplicación podría ser de interés, fuera de nuestra unidad, para usuarios de cualquier equipo de microscopía que no sea capaz de corregir las diferencias de planos de foco durante la captura, ni de procesar experimentos de videomicroscopía multifocal de forma eficiente.

OPERA EXPORTER

Antes del lanzamiento de esta aplicación, los usuarios interesados en visualizar imágenes del equipo Opera tenían dos opciones: 1- esperar a la disponibilidad del equipo de captura o de análisis de la unidad, y visualizar las imágenes una a una en el *software* propietario (con una eficiencia variable, dependiendo del número de licencias en uso), o 2- exportar las imágenes en formato .tif, abrirlas en FIJI siguiendo el patrón

fila-columna-campo explicado en la introducción, y procesar los canales manualmente para cada imagen (importar las imágenes mediante el uso de OME Bio-Formats en este caso es complicado porque este complemento da problemas, si no se utiliza correctamente, cuando el número de imágenes es alto).

El uso de Opera Exporter ha permitido a los usuarios visualizar varias imágenes simultáneamente desde sus ordenadores personales, sin necesidad de desplazarse a la unidad y sin problemas de disponibilidad de equipos, elegir las imágenes siguiendo un patrón intuitivo (el dibujo de la placa), asignarles una paleta de falso color de forma sencilla y seleccionar aquellas de interés para su exportación en formato. tif. El aprendizaje del uso de la aplicación es muy rápido y puede usarse sin conocimientos especializados. Como consecuencia, los equipos de captura y análisis están menos congestionados y su uso se centra en su función principal: la captura de imágenes y la ejecución de rutinas de análisis; los técnicos de la unidad, por otra parte, tienen más tiempo para dedicar a otras tareas.

El rendimiento de Opera Exporter no se ve afectado por la selección de un número elevado de pocillos a visualizar, porque estos se procesan independientemente en grupos de tres, tal y como se ha explicado en los resultados. El número de imágenes a abrir por pocillo puede limitarse por el usuario a través de la interfaz (Figura 19), en el caso de que se den problemas de eficiencia o se desee agilizar el proceso.

Posibles intervenciones de mejora sobre la aplicación serían:

1- Mejorar la velocidad de importado de las imágenes. Actualmente, este es el paso que consume más tiempo de procesado. Se deberían explorar otras funciones de apertura, otras librerías de metadatos alternativos a OME Bio-Formats o explorar las actualizaciones de esta librería.

2- Adaptarla para su uso en placas de más de 96 pocillos (las placas de menos de 96 pocillos pueden visualizarse con la aplicación actual). Esta mejora no se ha implementado porque no se ha detectado esta necesidad.

3- Adaptar la aplicación para que pueda utilizarse cuando se capturen 4 canales. En el equipo Opera no se suelen capturar más de tres canales, porque hacerlo suele implicar problemas de cruce de señal. Sin embargo, esta mejora se podría implementar si fuera necesario.

4- Mejorar el diseño de la plantilla de la placa (Figura 19) para que respete la nomenclatura habitual de los pocillos (A1-H12). Esta mejora sería fácil de implementar para nombrar cada una de las casillas de verificación individualmente (por ejemplo, nombrar a la casilla de verificación 13 como B1, la 14 como B2, etc.). Implementar esta mejora para crear una columna con la numeración de las filas de las casillas de verificación (por ejemplo, una fila de 12 casillas de verificación precedidas por una A), requeriría modificar la API del lenguaje macro de ImageJ.

5- Guardar las imágenes seleccionadas por el usuario siguiendo la nomenclatura anterior. Curiosamente, no se ha detectado tal necesidad, quizás porque, una vez seleccionadas, el número de imágenes suele ser muy bajo.

6- Mejorar la opción de selección de campos a abrir por pocillo (Figura 19). Esta opción se diseñó para restringir la apertura de imágenes cuando abrir todas las imágenes capturadas en un pocillo pudiera comprometer los recursos del equipo. Tal y como está diseñada actualmente, la aplicación abre las imágenes hasta el valor introducido por el usuario (por ejemplo, si el usuario introduce el valor "15" se abrirán las imágenes de los quince primeros campos capturados). La aplicación no permite seleccionar las imágenes a abrir (por ejemplo: abrir las imágenes de los quince últimos campos capturados o las 15 imágenes de los campos del centro del pocillo). Se podría modificar la macro para permitir seleccionar intervalos o grupos de imágenes, modificando la interfaz y el código de apertura de imágenes.

7- Introducir un nuevo campo que permita al usuario elegir el número de pocillos cuyas imágenes se van a mostrar simultáneamente (limitado a 3 por defecto). La implementación de esta mejora es inmediata, pero podría tener efectos negativos en el rendimiento de la aplicación, dependiendo de las capacidades del equipo en que se ejecute y del número de pocillos y campos por pocillo a visualizar.

En cuanto a su posible difusión fuera del CNIO, pensamos que podría despertar cierto interés, especialmente en aquellos laboratorios en que el equipo Opera se use frecuentemente y no dispongan de medios económicos para invertir en núcleos de análisis, estaciones de análisis o *software* propietario especializado.

IMSRC STITCHING MODULE

iMSRC Stitching Module permite que colecciones de imágenes .tif, procedentes de la aplicación Matrix Screener de Leica, se puedan fusionar en mosaicos. Su uso requiere estar familiarizado con el módulo Grid/Collection Stitching de ImageJ y tener conocimientos mínimos de imagen (canales, LUTs, ángulo de rotación, etc.). La aplicación funciona tanto si se ha usado o no iMSRC, para capturar mosaicos regulares o irregulares.

Su diseño ha permitido aumentar la eficiencia de la captura de mosaicos, por partida doble:

1- Contribuyendo al desarrollo de una herramienta de captura de mosaicos irregulares. iMSRC Stitching Module es una aplicación integrada dentro de iMSRC y distribuida junto con esta. iMSRC Stitching Module permite a iMSRC fusionar mosaicos irregulares grandes, rápidamente y sin errores. Como se ha visto en resultados, una herramienta de captura y fusión de mosaicos como iMSRC, supone un importante ahorro en recursos económicos y una considerable ganancia de disponibilidad de equipamiento (Figura 31).

2- Permitiendo fusionar mosaicos en un equipo distinto al de captura. La aplicación iMSRC no está limitada a un número de licencias, se puede ejecutar en cualquier ordenador (distinto al del equipo de captura) y permite fusionar mosaicos en serie. Esto hace que la fusión de mosaicos se pueda realizar en múltiples equipos, en varios mosaicos de forma secuencial y en ausencia de personal (habitualmente por la noche). La consecuencia principal es un gran incremento del rendimiento de los equipos, dado que el tiempo empleado en la fusión de imágenes queda disponible para la captura.

Posibles actuaciones de mejora de iMSRC Stitching Module serían:

1- Aumentar el rendimiento de la aplicación cuando se capturen 2 ó 3 canales y no se capturen imágenes de campo claro. En estas condiciones, los canales individuales de la imagen mosaico pueden extraerse de la imagen mosaico RGB, lo cual es mucho más eficiente que realizar la fusión de campos de los mismos. Esta mejora se ha implementado parcialmente en nuestra unidad. Solo requeriría integrarla en el código. Alternativamente, se podría modificar la interfaz gráfica de usuario para que esta permitiera decidir la salida de datos (RGB, canales independientes o los dos) a su gusto.

2- No proyectar las imágenes de distintos planos, sino seleccionar el plano en foco para su fusión. Requeriría obtener el plano en foco mediante el análisis de la dispersión del valor de gris de la imagen en cada plano. Simplificaría la captura de mosaicos difíciles de enfocar. Esta mejora ya se ha implementado en nuestra unidad.

3- Asignar automáticamente el umbral de regresión. Actualmente, el umbral de regresión se determina por ensayo-error. Generalmente, no suele ser complicado encontrar un umbral de regresión que produzca una buena fusión, aunque esto depende de la muestra y de la experiencia del usuario. Aun así, determinar el umbral de regresión puede suponer una pérdida de productividad en aquellos casos en que la computación sea larga, por ejemplo, cuando se quieren fusionar mosaicos en serie o mosaicos muy grandes.

Los índices de regresión entre imágenes adyacentes quedan registrados al realizarse la fusión del mosaico, es más, se puede hacer el cálculo de los índices de regresión sin necesidad de fusionar el mosaico posteriormente. Se podría implementar un algoritmo que analizara el valor de los índices de regresión de los campos generados en mosaicos irregulares. Esto permitiría determinar el valor mínimo del umbral de regresión que elimina todos los campos generados, lo cual se podría utilizar como una primera aproximación a la hora de asignar un umbral de regresión. También se podría estudiar la distribución de los índices de regresión de los campos de un mosaico sintético, fusionarlo utilizando varios umbrales de regresión, comparar el resultado obtenido con el mosaico sintético, y determinar el rango de umbrales de regresión que producen un resultado óptimo. De tener éxito, se incrementaría la eficacia de la fusión de mosaicos y se simplificaría el uso de la aplicación (eliminándose el campo para la introducción del umbral de regresión en GUI-3, Figura 29: A4), haciéndola accesible a usuarios sin formación en algoritmos de fusión de mosaicos.

4- Automatizar la lectura del número de canales, sus LUTs y el ángulo de rotación, que introduce el usuario de forma manual en GUI-3 (Figura 29: A1, A2 y A4). Se podría modificar la macro para extraer esta información del archivo de metadatos de la imagen. Esto simplificaría el uso de la aplicación (eliminando los campos A1, A2 y A4 de GUI-3de la Figura 29), haciéndola accesible a usuarios sin conocimientos en análisis de imagen.

5- Adaptar la aplicación para su uso completamente automático. Actualmente, la fusión de mosaicos en serie está limitada a los mosaicos capturados en un mismo experimento, con el mismo número y distribución de canales, ángulo de rotación y que se puedan fusionar con el mismo umbral de regresión. La implementación de los puntos 3 y 4 de este apartado, permitiría fusionar mosaicos de forma independiente de la configuración de canales, del ángulo de rotación y del umbral de regresión necesario para la fusión. Si se modificara la aplicación para procesar varios experimentos en serie, mejoraría el rendimiento del proceso, ya que no habría límite a la cantidad de mosaicos que se podrían procesar por sesión en un equipo.

iMSRC Stitching Module ya se ha distribuido fuera del laboratorio junto con la aplicación iMSRC. Esta aplicación puede obtenerse gratuitamente a través de la red (http://imsrc.bioinfo.cnio.es/) o a través de Leica, en virtud a un acuerdo de distribución adquirido con esta compañía.

Aparte de su uso junto con iMSRC, iMSRC Stitching Module también podría tener interés para los usuarios de ImageJ que necesiten fusionar mosaicos en serie.

CAPTURA INTELIGENTE DE CÉLULAS CANCEROSAS CIRCULANTES

Los objetivos perseguidos en este punto se han alcanzado parcialmente. Si bien se ha conseguido desarrollar un método de captura de células cancerosas circulantes capaz de analizar una muestra en 1 hora 30 minutos, no se ha conseguido automatizar todo el procedimiento. Actualmente, el proceso de captura consiste en: 1- capturar tres mosaicos de búsqueda, 2- obtener las coordenadas de los candidatos a células cancerosas circulantes, 3- capturar imágenes de estos a grandes aumentos y 4- procesar las imágenes para la fusión de canales y planos. El primer paso consume 30 minutos, el segundo 15 minutos, el tercero 3 minutos por célula detectada y el cuarto 10 segundos por célula. (El máximo número de células capturadas en cualquier caso es de 15, como se vio en resultados).

Conviene recordar que en el paso 3 el investigador tiene que ajustar manualmente el foco a cada una de las posiciones de captura y que las imágenes obtenidas se procesan posteriormente para la fusión de canales y proyección de planos, adaptando un código y ejecutándolo una vez por cada captura (ver nota al pie en página 95 y Figura 27).

Posibles intervenciones de mejora serían:

1- Disminuir el tamaño del soporte de las muestras. Permitiría disminuir la superficie de los mosaicos de búsqueda, como consecuencia estos se capturarían más rápido (el paso 1 se realizaría en menos tiempo). Si, por ejemplo, la muestra se introdujera en un portaobjetos de 8 micropocillos (8-well Chambered Coverglass, Thermo Fisher Scientific, Waltham, Massachusetts, USA) de 0.8 cm^2 de superficie cada uno (en vez del portaobjetos que se está utilizando actualmente, con micropocillos de 1 cm^2) se tendrían que capturar 48 campos menos, lo que ahorraría, aproximadamente, 6 minutos de captura por mosaico de búsqueda.

2- Reducir el número de esferas magnéticas libres en las muestras. Las esferas magnéticas son autofluorescentes y aportan volumen a la muestra. La autofluorescencia de las esferas magnéticas dificulta segmentar las células positivas en el paso 2 y da lugar a falsos positivos que obligan a visitar posiciones sin interés en el paso 3. Una muestra con excesivo volumen obliga a que los perfiles de captura de bajo aumento requieran de la captura de tres planos focales en el paso 1 e impide utilizar una rutina de autofoco en el 3, imposibilitando la automatización del proceso, (Figura 32). Una muestra sin esferas magnéticas libres permitiría capturar un único plano focal en la primera captura, utilizar la herramienta de autofoco en la segunda captura (automatizándola completamente), y

facilitaría el análisis de los mosaicos de búsqueda, disminuyendo la cantidad de falsos positivos.

3- Implementar un sistema de binarización más eficiente. Agilizaría el paso 2, que se podría automatizar completamente. En este sentido, se podría explorar la posibilidad de implementar un algoritmo de segmentación basado en aprendizaje automático (*machine learning*).

4- Automatizar el procesado de las imágenes para que se ejecute una sola vez por experimento. Solo requeriría adaptar parte del código de la aplicación iMSRC Stitching Module a una nueva estructura de carpetas.

La primera mejora permitiría ahorrar aproximadamente 12 minutos de captura, la segunda 20, la tercera 12 y la cuarta 1 minuto.

De las intervenciones de mejora planteadas, la segunda es la que mayor impacto tendría en el proceso global; sin embargo, su implementación es complicada debido a la rigidez de los procedimientos en pruebas de diagnóstico médico. La implementación de todos estos puntos debería permitir automatizar completamente la captura de una muestra, disminuir el tiempo empleado en hacerlo por debajo de los 45 minutos, y permitir la captura de múltiples muestras en serie.

Por otra parte, el diseño experimental propuesto presenta una ventaja respecto al conteo manual que no se había tenido en cuenta: se genera un registro exhaustivo de imágenes y del protocolo de análisis. Las imágenes de grandes aumentos de células cancerosas circulantes pueden ser evaluadas por varios observadores en caso de duda o vueltas a evaluar en caso de resultados inesperados; además, las imágenes mosaico permiten obtener información adicional, como la estima de células totales, que se pueden utilizar para la normalización de datos (número de células cancerosas del total de células) o como punto de control de calidad (un número anormal de células en la imagen mosaico puede poner de manifiesto problemas en el protocolo de extracción de muestras, por ejemplo). El protocolo de análisis y sus anotaciones (valores de los umbrales de segmentación, tamaño, circularidad, etc., utilizados) permiten repetirlo en cualquier momento, poniendo de manifiesto posibles errores en este paso.

El proyecto de colaboración con el Laboratorio de Oncología Traslacional del Hospital General Universitario Gregorio Marañón de Madrid sigue en funcionamiento, habiéndose procesado decenas de muestras. La eficiencia del procedimiento de captura de células cancerosas circulantes desarrollado es mayor al conteo manual y muestra un margen de mejora muy prometedor. Si los estudios de doble ciego en muestras control ponen de manifiesto que la metodología presentada es capaz de detectar células cancerosas circulantes de forma eficaz, y el seguimiento de los pacientes y demás estudios dan lugar a resultados favorables, se desarrollará una herramienta de diagnóstico sencilla y poco invasiva.

CAPTURA INTELIGENTE DE CÉLULAS MADRE CANCEROSAS

El desarrollo de este procedimiento permitió alcanzar plenamente los objetivos perseguidos. Los resultados derivados de su implementación han sido publicados [130].

El buen funcionamiento de la aplicación Single Cell Well Detector se debe al sistema de detección. Por una parte, las propiedades ópticas de los micropocillos permitieron implementar un método de detección de pocillos muy robusto. Por otra parte, el biomarcador permitió discriminar células en base a parámetros muy sencillos (brillo y tamaño).

El procedimiento desarrollado puede resultar de interés para aquellos laboratorios interesados en el seguimiento de marcadores a través del linaje celular.

CAPTURA INTELIGENTE DE *TISSUE MICROARRAYS*

TMA Field Sampler se diseñó para ser utilizado por miembros de la Unidad de Microsocopía Confocal, con el objetivo de permitir la captura automatizada de TMAs usando iMSRC. TMA Manager se diseñó para ser utilizado por usuarios de la unidad, como herramienta de procesado y visualización de la colección de imágenes resultantes.

Todos los objetivos perseguidos en el diseño de estas aplicaciones se han alcanzado con éxito. Esto ha permitido dotar a la Unidad de Microscopía Confocal, a coste cero, de la funcionalidad de un escáner digital convencional y de dos escáneres digitales de barrido láser confocal (este último tipo de equipamiento es exclusivo de nuestra unidad, hasta donde nosotros sabemos). Como consecuencia, el tiempo de captura de un TMA ha mejorado de forma muy significativa en ambos tipos de equipos, pasando a ser de varias horas a 20 ó 30 minutos, respectivamente. Esto supone un ahorro muy considerable de tiempo, recursos y ganancia de disponibilidad de equipos.

Por otra parte, los usuarios aprenden a utilizar TMA Manager en varios minutos, a través de un correo electrónico con instrucciones sencillas. Esto hace que tanto el personal técnico como los usuarios ganen tiempo, que no tienen que dedicar ni a enseñar ni a aprender a procesar imágenes.

Actualmente, varios investigadores del CNIO utilizan estas herramientas en sus proyectos, tanto en microscopios confocales como en fluorescentes convencionales, mostrando un grado de satisfacción muy alto. Algunos ejemplos de su uso ya han sido publicados[28,132].

Algunas posibles intervenciones de mejora serían:

1- Generar una interfaz gráfica en TMA Manager similar a la de la Figura 29: B1 y B2. Esto evitaría tener que adaptar manualmente la fusión de canales modificando el código. Su implementación debería ser bastante sencilla, dado que solo habría que adaptar un código que ya está escrito.

2- Cambiar la estructura de la carpeta de resultados generada por TMA Manager para hacerla todavía más intuitiva: 1- no generar carpetas vacías cuando haya selecciones de tejidos que no contienen ningún campo completo (Figura 44: 3), y 2- añadir "Tejido" y "campo" al nombre de las carpetas y de los archivos .tif (por ejemplo: la carpeta "1" en la Figura 44: 3 pasaría a llamarse "Tejido-1" y la imagen "1-1", se llamaría "Tejido 1-campo 1"). Ambas mejoras son bastante inmediatas, bastaría con modificar el código en la Figura 44: B y la macro *openImage*.

3- Adaptar ambas aplicaciones para que permitieran capturar y procesar varios TMAs en serie. Este diseño experimental aumentaría todavía más el rendimiento de la captura de TMAs, liberando tiempo del personal de la unidad y permitiendo la captura desatendida de varias muestras fuera del horario laboral.

4- Actualizar el algoritmo de selección de campos de TMA Field Sampler (Figura 41) para que permita realizar muestreos aleatorios, uniformes y sistemáticos y/o muestreos aleatorios sin repetición. Permitiría automatizar la captura de muestras estadísticamente representativas, lo que posibilitaría utilizar iMSRC en estudios cuantitativos completamente automatizados.

5- Automatizar el duplicado de la imagen mosaico del TMA y la creación de openImageResilence.txt. Ambas mejoras son de fácil implementación, facilitarían ligeramente el uso de TMA Field Sampler y TMA Manager, respectivamente.

Pensamos que estas aplicaciones serán de gran interés para usuarios con equipamiento Leica, interesados en la captura de TMAs y muestras similares, que quieran adquirir la funcionalidad de un escáner digital gratuitamente. También serán de interés para usuarios interesados en adquirir un escáner digital confocal.

CONCLUSIONES

<<La respuesta de la vida, el universo y todo lo demás es... 42.>>

Douglas Adams, Guía del autoestopista galáctico.

1- Hemos desarrollado herramientas de procesado, análisis y visualización de imágenes automáticas, gratuitas, accesibles, modificables, distribuibles y basadas en GUIs y diseños intuitivos. Esto permite independizar captura y análisis, curvas de aprendizaje muy cortas, uso por parte de personal no especializado y aumentar la eficiencia de procesos. Como consecuencia, los equipos de captura están más disponibles, disminuye el coste económico de los experimentos, los usuarios y técnicos ganan tiempo y los resultados se obtienen en menos tiempo.

2- DeltaVision Stack Manager procesa pilas de imágenes procedentes de experimentos de videomicroscopía multicanal y/o multifocal del equipo DeltaVision. Permite extraer las imágenes en foco, modificar brillo, contraste y profundidad, aplicar LUTs y separar y fusionar canales.

3- Opera Exporter se usa en colecciones de imágenes capturadas con el equipo Opera de HCS. Permite visualizar, simultáneamente, grupos de imágenes escogidas intuitivamente de una interfaz, y seleccionar las de interés. Estas se copian en formato nativo y .tif a una carpeta de resultados. Si hay más de un canal, estos se fusionan automáticamente.

4- iMSRC Stitching Module fusiona campos en mosaicos en colecciones de imágenes procedentes de la aplicación Matrix Screener. Si los campos contienen imágenes de varios canales y planos, estos se fusionan y proyectan, respectivamente. Si se capturó más de un mosaico, estos se procesan en serie. Si faltan campos, la aplicación funciona igualmente. Puede modificarse para procesar automáticamente, como se desee, cualquier colección de imágenes como la citada.

5- Circulating Cancer Cells detecta candidatos a célula cancerosa circulante sobre mosaicos de búsqueda de la aplicación iMSRC. Una modificación de iMSRC Stitching Module procesa las imágenes resultantes. Estas aplicaciones han permitido implementar un sistema de captura inteligente, que puede dar lugar al desarrollo de una nueva herramienta de diagnóstico clínico.

6- Single Cell Well Detector identifica células positivas aisladas dentro de micropocillos sobre mosaicos de búsqueda de la aplicación iMSRC. Una modificación de iMSRC Stitching Module procesa las imágenes resultantes. Han permitido implementar un sistema de captura inteligente para el seguimiento del linaje celular y caracterizar un nuevo biomarcador de células madre cancerosas.

7- TMA Field Sampler funciona sobre mosaicos de búsqueda de la aplicación iMSRC y muestrea un número definido de campos en piezas de tejido. TMA Manager automatiza el procesado de las imágenes resultantes, las guarda en una estructura de carpetas interpretable y posibilita visualizarlas intuitivamente. Estas aplicaciones han permitido implementar un sistema de captura inteligente para la captura de TMAs y una herramienta de visualización de los resultados obtenidos.

BIBLIOGRAFÍA – WEBGRAFÍA

<<While a calculator on the ENIAC is equipped with 10000 vacuum tubes and weighs 30 tons, computers of the future may have only 1000 vacuum tubes and weigh only 1.5 tons.>>

Popular mechanics, 1949

1. Zuidevaart, H.J. The 'true inventor' of the telescope. A survey of 400 years of debate in *The Origins of the Telescope* (Albert van Helden, Sven Dupré, Rob van Gent, Huib Zuidervaart, Amsterdam, 2012).
2. Arndt-Jovin, D. J., Robert-Nicoud, M., Kaufman, S. J. & Jovin, T. M. Fluorescence digital imaging microscopy in cell biology. *Science* **230,** 247–256 (1985).
3. Leung, B. O. & Chou, K. C. Review of super-resolution fluorescence microscopy for biology. *Appl. Spectrosc.* **65,** 967–980 (2011).
4. Hogg, J. & West, T. History of the invention and improvements of the microscope. in *The microscope, : its history, construction, and applications: being a familiar introduction to the use of the instrument, and the study of microsopical science, / with upwards of 500 engravings... by Tuffen West* (G. Routledge, London, 1887). doi:10.5962/bhl.title.27389
5. Amos, B. Lessons from the history of light microscopy. *Nat. Cell Biol.* **2,** E151-152 (2000).
6. Sailem, H. Z., Cooper, S. & Bakal, C. Visualizing quantitative microscopy data: History and challenges. *Crit. Rev. Biochem. Mol. Biol.* **51,** 96–101 (2016).
7. Overney Gregor, O. N. The History of Photomicrography. (2011). Available at: http://www.microscopy-uk.org.uk/mag/artmar10/history_photomicrography_ed3.pdf. (Accessed: 8th December 2016)
8. Oberholzer, M., Ostreicher, M., Christen, H. & Brühlmann, M. Methods in quantitative image analysis. *Histochem. Cell Biol.* **105,** 333–355 (1996).
9. Weibel, E. R., Kistler, G. S. & Scherle, W. F. Practical stereological methods for morphometric cytology. *J. Cell Biol.* **30,** 23–38 (1966).
10. Meunier, P., Aaron, J., Edouard, C. & Vignon, G. Osteoporosis and the replacement of cell populations of the marrow by adipose tissue. A quantitative study of 84 iliac bone biopsies. *Clin. Orthop.* **80,** 147–154 (1971).

11. Paddock, S. W. & Eliceiri, K. W. Laser Scanning Confocal Microscopy: History, Applications, and Related Optical Sectioning Techniques in *Confocal Microscopy* (Springer New York, New York, 2014).

12. Lipkin, L. E. & Lipkin, B. S. Computers in the Clinical Pathologic Laboratory: Chemistry and Image Processing. *Annu. Rev. Biophys. Bioeng.* **4,** 529–577 (1975).

13. Harmon, L. D. & Knowlton, K. C. Picture Processing by Computer. *Science* **164,** 19–29 (1969).

14. Johnston, A. R. & Powell, R. V. Optics at the Jet Propulsion Laboratory. *Appl. Opt.* **9,** 271 (1970).

15. Hornick, J. E. & Hinchcliffe, E. H. It's all about the pentiums: The use, manipulation, and storage of digital microscopy imaging data for the biological sciences. *Mol. Reprod. Dev.* **82,** 508–517 (2015).

16. Riley, R. S., Ben-Ezra, J. M., Massey, D., Slyter, R. L. & Romagnoli, G. Digital photography: A primer for pathologists. *J. Clin. Lab. Anal.* **18,** 91–128 (2004).

17. Wallack, S. Digital Image Storage. *Vet. Radiol. Ultrasound* **49,** S37–S41 (2008).

18. Hedvat, C. V. Digital microscopy: past, present, and future. *Arch. Pathol. Lab. Med.* **134,** 1666–1670 (2010).

19. The quest for quantitative microscopy. *Nat. Methods* **9,** 627–627 (2012).

20. Carpenter, A. E., Kamentsky, L. & Eliceiri, K. W. A call for bioimaging software usability. *Nat. Methods* **9,** 666–670 (2012).

21. Peng, H. Bioimage informatics: a new area of engineering biology. *Bioinformatics* **24,** 1827–1836 (2008).

22. Cardona, A. & Tomancak, P. Current challenges in open-source bioimage informatics. *Nat. Methods* **9,** 661–665 (2012).

23. Eliceiri, K. W. *et al.* Biological imaging software tools. *Nat. Methods* **9,** 697–710 (2012).

24. Myers, G. Why bioimage informatics matters. *Nat. Methods* **9,** 659–660 (2012).

25. Edelstein, A., Amodaj, N., Hoover, K., Vale, R. & Stuurman, N. Computer Control of Microscopes Using µManager in *Current Protocols in Molecular Biology* (John Wiley & Sons Inc., Hoboken, 2010).

26. Edelstein, A. D. *et al.* Advanced methods of microscope control using µManager software. *J. Biol. Methods* **1,** 10 (2014).

27. Conrad, C. *et al.* Micropilot: automation of fluorescence microscopy–based imaging for systems biology. *Nat. Methods* **8,** 246–249 (2011).

28. Carro, A., Perez-Martinez, M., Soriano, J., Pisano, D. G. & Megias, D. iMSRC: converting a standard automated microscope into an intelligent screening platform. *Sci. Rep.* **5,** 10502 (2015).

29. Swedlow, J. R. & Eliceiri, K. W. Open source bioimage informatics for cell biology. *Trends Cell Biol.* **19,** 656–660 (2009).

30. de Laat, P. B. Copyright or copyleft?: An analysis of property regimes for software development. *Res. Policy* **34,** 1511–1532 (2005).

31. Stahl, M. T. Open-source software: not quite endsville. *Drug Discov. Today* **10,** 219–222 (2005).

32. Lindman, J., Paajanen, A. & Rossi, M. Choosing an Open Source Software License in Commercial Context: A Managerial Perspective. 3*6th EUROMICRO Conference on Software Engineering and Advanced Applications*, 237–244 (IEEE, 2010). doi:10.1109/SEAA.2010.26

33. Hamilton, N. A. Open Source Tools for Fluorescent Imaging in *Methods in Enzymology* (Elsevier, Amsterdam, 2012).

34. Cusumano, M. A. The Changing Software Business: Moving from Products to Services. *Computer* **41,** 20–27 (2008).

35. Karopka, T., Schmuhl, H. & Demski, H. Free/Libre Open Source Software in Health Care: A Review. *Healthc. Inform. Res.* **20,** 11 (2014).

36. Riehle, D. The Commercial Open Source Business Model in *Value Creation in E-Business Management* (Springer Berlin Heidelberg, 2009). https://doi.org/10.1007/978-3-642-03132-8_2.

37. Ratib, O., Rosset, A. & Heuberger, J. Open Source software and social networks: Disruptive alternatives for medical imaging. *Eur. J. Radiol.* **78,** 259–265 (2011).

38. Harrington, D. P. Open-Source Software Opportunities and Challenges. *J. Am. Coll. Radiol.* **3,** 14–15 (2006).

39. gnu.org. *What is free software?* Available at: https://www.gnu.org/philosophy/free-sw.en.html. (Accessed: 8th December 2016)

40. Free Software Foundation. Available at: https://www.fsf.org/. (Accessed: 8th December 2016)

41. Open Source Initiative. Available at: https://opensource.org/. (Accessed: 8th December 2016)

42. Raymond, E. S. *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary*. (O'Reilly, Beijing, 2001).

43. Crowston, K. & Howison, J. Assessing the Health of Open Source Communities. *Computer* **39,** 89–91 (2006).

44. Schneider, C. A., Rasband, W. S. & Eliceiri, K. W. NIH Image to ImageJ: 25 years of image analysis. *Nat. Methods* **9,** 671–675 (2012).

45. Schindelin, J. *et al.* Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9,** 676–682 (2012).

46. Gaughan, G., Fitzgerald, B. & Shaikh, M. An Examination of the Use of Open Source Software Processes as a Global Software Development Solution for Commercial Software Engineering. *35th Euromicro Conference on Software Engineering and Advanced Applications*, 20–27 (IEEE, 2009). doi:10.1109/SEAA.2009.86.

47. Wasserman, A. I. & Capra, E. Evaluating Software Engineering Processes in Commercial and Community Open Source Projects. *First International Workshop on Emerging Trends in FLOSS Research and Development*, 1–5 (IEEE, 2007). doi:10.1109/FLOSS.2007.6.

48. Chang, S., Lee, J. & Yi, W. A Practical Management Framework for Commercial Software Development with Open Sources. *IEEE International Conference on E-Business Engineering*, 164–171 (IEEE, 2010). doi:10.1109/ICEBE.2010.82.

49. Ghosh, S. S., Klein, A., Avants, B. & Millman, K. J. Learning from open source software projects to improve scientific review. *Front. Comput. Neurosci.* **6,** (2012).

50. Nagy, P. Open Source in Imaging Informatics. *J. Digit. Imaging* **20,** 1–10 (2007).

51. SourceForge. Available at: https://sourceforge.net/. (Accessed: 8th December 2016).

52. Janamanchi, B., Katsamakas, E., Raghupathi, W. & Gao, W. The State and Profile of Open Source Software Projects in health and medical informatics. *Int. J. Med. Inf.* **78,** 457–472 (2009).

53. gnu.org-licenses. Available at: https://www.gnu.org/licenses/license-list.html. (Accessed: 8th December 2016).

54. Licenses & Standards | Open Source Initiative. Available at: https://opensource.org/licenses/. (Accessed: 8th December 2016).

55. MySQL. Available at: https://www.mysql.com/. (Accessed: 8th December 2016).

56. Rosset, A., Spadola, L. & Ratib, O. OsiriX: An Open-Source Software for Navigating in Multidimensional DICOM Images. *J. Digit. Imaging* **17,** 205–216 (2004).

57. OsiriX | The world famous medical imaging viewer. Available at: http://www.osirix-viewer.com/. (Accessed: 8th December 2016).

58. Stöter, M. *et al.* CellProfiler and KNIME: Open Source Tools for High Content Screening in *Target Identification and Validation in Drug Discovery* (Humana Press, Totowa, 2013).

59. Prevedello, L. & Khorasani, R. Should You Use Open-Source Software Applications in Your Practice? *J. Am. Coll. Radiol.* **9,** 751–752 (2012).

60. Bush, J. Open-source software: just what the doctor ordered? *Fam. Pract. Manag.* **10,** 65, 69 (2003).

61. J, M. P., Graham, W., Thomas, K., Helen, B. & Andrej, O. Open Source and Healthcare in Europe in *Time to Put Leading Edge Ideas into Practice,* 963–967 (IOS Press Ebooks, 2009). doi:10.3233/978-1-60750-044-5-963

62. Solomon, R. W. Free and Open Source Software for the Manipulation of Digital Images. *Am. J. Roentgenol.* **192,** W330–W334 (2009).

63. Robertson, M. N. *et al.* Open source drug discovery – A limited tutorial. *Parasitology* **141,** 148–157 (2014).

64. McDonald, C. J. *et al.* Open Source software in medical informatics--why, how and what. *Int. J. Med. Inf.* **69,** 175–184 (2003).

65. Riehle, D. The Economic Motivation of Open Source Software: Stakeholder Perspectives. *Computer* **40,** 25–32 (2007).

66. Krylov, A. I. *et al.* What Is the Price of Open-Source Software? *J. Phys. Chem. Lett.* **6,** 2751–2754 (2015).

67. Noticias, análisis y descargas de software, guía de las mejores apps - Softonic. Available at: https://www.softonic.com/. (Accessed: 8th December 2016).

68. KNIME | Open for Innovation. Available at: https://www.knime.org/. (Accessed: 8th December 2016).

69. Kitware Inc. - leading edge, high-quality software. Available at: http://www.kitware.com/. (Accessed: 8th December 2016).

70. Glencoe Software | Viewing, sharing, analysis and management of life science images and data. Available at: https://glencoesoftware.com/. (Accessed: 8th December 2016).

71. Oracle | Integrated Cloud Applications and Platform Services. Available at: https://www.oracle.com/index.html. (Accessed: 8th December 2016).

72. The world's open source leader. Available at: https://www.redhat.com/en. (Accessed: 8th December 2016).

73. Packt Publishing | Technology Books, eBooks & Videos. Available at: https://www.packtpub.com/. (Accessed: 8th December 2016).

74. Collins, T. ImageJ for microscopy. *BioTechniques* **43,** S25–S30 (2007).

75. Murray, S. *et al.* Open science, open access and open source software at Open Medicine. *Open Med. Peer-Rev. Indep. Open-Access J.* **2,** e1-3 (2008).

76. Linkert, M. *et al.* Metadata matters: access to image data in the real world. *J. Cell Biol.* **189,** 777–782 (2010).

77. de Chaumont, F. *et al.* Icy: an open bioimage informatics platform for extended reproducible research. *Nat. Methods* **9,** 690–696 (2012).

78. Kamentsky, L. *et al.* Improved structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. *Bioinformatics* **27,** 1179–1180 (2011).

79. Jones, T. R. *et al.* CellProfiler Analyst: data exploration and analysis software for complex image-based screens. *BMC Bioinformatics* **9,** 482 (2008).

80. Jones, T. R. *et al.* Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning. *Proc. Natl. Acad. Sci.* **106,** 1826–1831 (2009).

81. Sommer, C., Straehle, C., Kothe, U. & Hamprecht, F. A. Ilastik: Interactive learning and segmentation toolkit. *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro* ,230–233 (IEEE, 2011). doi:10.1109/ISBI.2011.5872394.

82. Berthold, M. R. *et al.* KNIME: The Konstanz Information Miner in *Data Analysis, Machine Learning and Applications* (Springer Berlin Heidelberg, 2008). doi:10.1007/978-3-540-78246-9_38.

83. R Core Team. *R: a language and environment for statistical computing*. (the R Foundation for Statistical Computing, Vienna, 2015).

84. Allan, C. *et al.* OMERO: flexible, model-driven data management for experimental biology. *Nat. Methods* **9,** 245–253 (2012).

85. Goldberg, I. G. *et al.* The Open Microscopy Environment (OME) Data Model and XML file: open tools for informatics and quantitative analysis in biological imaging. *Genome Biol.* https://doi.org/10.1186/gb-2005-6-5-r47 (2005).

86. Yoo, T. S. *et al.* Engineering and algorithm design for an image processing Api: a technical report on ITK--the Insight Toolkit. *Stud. Health Technol. Inform.* **85,** 586–592 (2002).

87. Schroeder, W. J., Martin, K. M. & Lorensen, W. E. The Design and Implementation of an Object-oriented Toolkit for 3D Graphics and Visualization. *Proceedings of the 7th Conference on Visualization* (IEEE Computer Society Press, 1996). https://dl.acm.org/citation.cfm?id=244979.245018.

88. Jonasson, H. *Determining project requirements*. (Auerbach Publications, Boca Ratón, 2008).

89. Royce, Winston W. Managing the Development of Large Software Systems: Concepts and Techniques. *Proceedings of the 9th international conference on Software Engineering,* 328-338 (1970). https://dl.acm.org/citation.cfm?id=41765.41801.

90. Kumar, D. & Mishra, K. K. The Impacts of Test Automation on Software's Cost, Quality and Time to Market. *Procedia Comput. Sci.* **79,** 8–15 (2016).

91. Meso, P. & Jain, R. Agile Software Development: Adaptive Systems Principles and Best Practices. *Inf. Syst. Manag.* **23,** 19–30 (2006).

92. Manifesto for Agile Software Development. Available at: http://agilemanifesto.org/. (Accessed: 8th December 2016).

93. Pitt-Francis, J. *et al.* Chaste: using agile programming techniques to develop computational biology software. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **366,** 3111–3136 (2008).

94. Collen, M. F. The origins of informatics. *J. Am. Med. Inform. Assoc. JAMIA* **1,** 91–107 (1994).

95. Aho, A. V. Software and the future of programming languages. *Science* **303,** 1331–1333 (2004).

96. Ousterhout, J. K. Scripting: higher level programming for the 21st Century. *Computer* **31,** 23–30 (1998).

97. Gilbert, D. Bioinformatics software resources. *Brief. Bioinform.* **5,** 300–304 (2004).

98. Sadowski, M. I., Grant, C. & Fell, T. S. Harnessing QbD, Programming Languages, and Automation for Reproducible Biology. *Trends Biotechnol.* **34,** 214–227 (2016).

99. Kindlmann, G., Chiw, C., Seltzer, N., Samuels, L. & Reppy, J. Diderot: a Domain-Specific Language for Portable Parallel Scientific Visualization and Image Analysis. *IEEE Trans. Vis. Comput. Graph.* **22,** 867–876 (2016).

100. Loui, R. P. In Praise of Scripting: Real Programming Pragmatism. *Computer* **41,** 22–26 (2008).

101. Milicchio, F., Rose, R., Bian, J., Min, J. & Prosperi, M. Visual programming for next-generation sequencing data analytics. *BioData Min.* **9,** (2016).

102. Open Bioinformatics Foundation. Available at: https://www.open-bio.org/wiki/Main_Page. (Accessed: 8th December 2016).

103. Stajich, J. E. Open source tools and toolkits for bioinformatics: significance, and where are we? *Brief. Bioinform.* **7,** 287–296 (2006).

104. Mangalam, H. The Bio* toolkits--a brief overview. *Brief. Bioinform.* **3,** 296–302 (2002).

105. de Brevern, A. G., Meyniel, J.-P., Fairhead, C., Neuvéglise, C. & Malpertuy, A. Trends in IT Innovation to Build a Next Generation Bioinformatics Solution to Manage and Analyse Biological Big Data Produced by NGS Technologies. *BioMed Res. Int.* **2015,** 1–15 (2015).

106. CNIO: Centro Nacional de Investigaciones Oncológicas. Available at: http://www.cnio.es/es/index.asp. (Accessed: 8th December 2016).

107. Megías, D. *et al.* Novel lambda FRET spectral confocal microscopy imaging method. *Microsc. Res. Tech.* **72,** 1–11 (2009).

108. Brown, L. G. A survey of image registration techniques. *ACM Comput. Surv.* **24,** 325–376 (1992).

109. Chen, C.-Y. & Klette, R. Image Stitching — Comparisons and New Techniques in *Computer Analysis of Images and Patterns***,** 615–622 (Springer Berlin Heidelberg, 1999). https://doi.org/10.1007/3-540-48375-6_73.

110. Chow, S. K. *et al.* Automated microscopy system for mosaic acquisition and processing. *J. Microsc.* **222,** 76–84 (2006).

111. Ma, B. *et al.* Use of Autostitch for automatic stitching of microscope images. *Micron* **38,** 492–499 (2007).

112. Bevilacqua, A. & Gherardi, A. Real-time whole slide mosaicing for non-automated microscopes in histopathology analysis. *J. Pathol. Inform.* **4,** 9 (2013).

113. ImageJ. Available at: https://imagej.nih.gov/ij/. (Accessed: 8th December 2016).

114. ImageJ. Available at: http://imagej.net/Welcome. (Accessed: 8th December 2016).

115. ImageJ overview — Bio-Formats 5.2.4 documentation.

116. Welcome to the ImageJ Information and Documentation Portal [ImageJ Documentation Wiki]. Available at: http://imagejdocu.tudor.lu/. (Accessed: 8th December 2016).

117. github-ImageJ. *GitHub* Available at: https://github.com/imagej. (Accessed: 8th December 2016).

118. Técnicas de Análisis de Imagen: Tratamiento y Cuantificación de la Imagen Digital. *Transmitting Science* Available at: https://www.transmittingscience.org/courses/imaging/tecnicas-analisis-imagen-tratamiento-cuantificacion-imagen-digital/. (Accessed: 14th May 2018).

119. EMBL Course - Bioimage Data Analysis. Available at: https://www.embl.de/training/events/2016/IMA16-01/index.html. (Accessed: 8th December 2016).

120. Conference - ImageJ. Available at: http://imagej.net/Conference. (Accessed: 8th December 2016).

121. Hartig, S. M. Basic Image Analysis and Manipulation in ImageJ in *Current Protocols in Molecular Biology* (John Wiley & Sons, Inc., Hoboken, 2013).

122. The Open Microscopy Environment — OME. Available at: https://www.openmicroscopy.org/site. (Accessed: 8th December 2016).

123. Fiji is just ImageJ. Available at: https://fiji.sc/. (Accessed: 8th December 2016).

124. Fiji - wiki. Available at: http://imagej.net/Fiji. (Accessed: 8th December 2016).

125. Preibisch, S., Saalfeld, S. & Tomancak, P. Globally optimal stitching of tiled 3D microscopic image acquisitions. *Bioinformatics* **25,** 1463–1465 (2009).

126. Gasch, C., Ffrench, B., O'Leary, J. J. & Gallagher, M. F. Catching moving targets: cancer stem cell hierarchies, therapy-resistance & considerations for clinical intervention. *Mol. Cancer* **16,** (2017).

127. García-Rojo, M., Bueno-García, G., Peces-Mateos, C., González García, J. & Carbajo Vicente, M. Preparaciones digitales en los servicios de Anatomía Patológica (II). Análisis de soluciones existentes. **38,** 207–220 (2005).

128. Landini, G. Advanced shape analysis with ImageJ. *Proceedings of the Second ImageJ User and Developer Conference,* 116–121 (Luxembourg, 2008). http://imagejconf.tudor.lu/archive/imagej-user-and-developer-conference-2008/copy_of_programme/presentations/advanced-shape-analysisis-with-imagejj.

129. Harb, W. *et al.* Mutational Analysis of Circulating Tumor Cells Using a Novel Microfluidic Collection Device and qPCR Assay. *Transl. Oncol.* **6,** 528–538 (2013).

130. Miranda-Lorenzo, I. *et al.* Intracellular autofluorescence: a biomarker for epithelial cancer stem cells. *Nat. Methods* **11,** 1161–1169 (2014).

131. Freeman, J. Open source tools for large-scale neuroscience. *Curr. Opin. Neurobiol.* **32,** 156–163 (2015).

132. Alonso-Curbelo, D. *et al.* RAB7 Controls Melanoma Progression by Exploiting a Lineage-Specific Wiring of the Endolysosomal Pathway. *Cancer Cell* **26,** 61–76 (2014).

GLOSARIO

<<Para ir a donde no se sabe, hay que ir por donde no se sabe.>>

Juan de Yepes Álvarez.

Abstracción

Técnica utilizada en computación para disminuir la complejidad de los lenguajes de programación. Consiste en suprimir detalles de implementación en el código para crear un lenguaje fácil de leer y escribir. El lenguaje binario se considera el nivel de abstracción más bajo, los lenguajes de programación general suponen un nivel de abstracción medio y los lenguajes de guion un nivel de abstracción alto.

Como ejemplo, véase la diferencia de realizar la suma de 1+1 en el lenguaje de guion propio de ImageJ (a la izquierda), en el lenguaje de programación java (en medio) y código máquina (inventado, a la derecha).

```
print(1+1);    public class ejemplo {                    00000010 10000000
               public static void main (String[] args){  00000000 00010001
               public int a=1;                           01100010 01101101
               public int b=1;                           11110100 00010011
               public int c= a+b;                        10010100 11100011
               System.out.println(c);                    00100010 10101010
               }                                         00101011 01011101
               }                                         …
```

En la medida en que disminuye el nivel de abstracción el código se vuelve más largo y más difícil de leer y escribir. El código máquina es más eficiente porque lo ejecuta el microprocesador directamente, el resto de códigos deben ser traducidos a código máquina antes de ser ejecutados.

Algoritmo

Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. Por ejemplo, se puede llenar un vaso de agua sosteniendo un vaso en la mano derecha con la abertura hacia arriba, tomando una jarra llena de agua con la mano izquierda con la abertura hacia arriba, alineándolos en la vertical, inclinando la jarra ligeramente en dirección al vaso, esperando a que se llene el vaso, devolviendo la jarra a la posición original, etc., o pidiendo a alguien que lo haga por ti. Los dos son ejemplos de algoritmos distintos que solucionan el mismo problema.

Análisis de imagen

Cuerpo de conocimientos que permite extraer datos de las imágenes.

AND

Operación lógica entre dos imágenes binarias (generalmente de igual tamaño y forma). Esta operación se produce entre los píxeles que ocupan la misma posición. Da como resultado una imagen en la que se muestran encendidos los píxeles que lo estaban en las dos imágenes originales a la vez.

API/Interfaz de programación de aplicaciones

De sus siglas en inglés: *Application Programming Interface*. Conjunto de métodos, escritos en un lenguaje de programación, con una finalidad común, organizados y explicados en una interfaz. Permiten a un programador escribir código de forma eficiente. Su funcionalidad la determina el número y tipo de métodos que la componen. Suele traducirse a un lenguaje de programación más complejo antes de su ejecución.

Aplicación

Software que ayuda de forma directa a un usuario, a través de una GUI, a realizar una tarea. En ocasiones se utiliza como sinónimo de programa, aunque técnicamente no lo son.

Autofoco:

Metodología que permite enfocar un objetivo sobre una muestra de forma automática.

Asistido por *software*: el principal componente del sistema de autofoco es un programa. Típicamente, el autofoco por *software* consiste en capturar una pila de imágenes a distinta altura, centrada en el campo a capturar. Un algoritmo informático determina la altura de la imagen enfocada (el algoritmo más conocido, asigna la posición de foco a la imagen con la mayor desviación típica de los niveles de gris de cada píxel). La herramienta de **autofoco predictivo** (*predictive focus*) de Leica, es una herramienta de autofoco asistido por *software* que se utiliza en el módulo Matrix Screener de formación de mosaicos. Permite determinar la posición del foco de todos los campos de un mosaico a partir del valor de foco de tres de sus esquinas. Para utilizarlo, un usuario debe introducir el foco de la muestra en tres puntos del mosaico, el algoritmo de autofoco calcula el plano que determinan estos tres puntos y asigna a cada uno de los posibles campos de un mosaico la altura de dicho plano.

Asistido por *hardware*: el principal componente del sistema de autofoco es un mecanismo físico. Típicamente, se basa en el análisis de la reflexión de una fuente de luz instalada en el objetivo. Un detector calcula la intensidad de la luz reflejada en la muestra para varias distancias focales y toma una distancia de referencia (por ejemplo, la distancia del objetivo en que la reflexión es máxima). A continuación, el usuario determina un foco y se calcula la distancia entre la posición de referencia y la posición de foco. Para recuperar la posición de foco, se varía gradualmente la distancia del objetivo a la muestra, se recupera la posición de referencia, y, a partir de esta, la de foco.

Batch mode*, modo *batch

Modo de ejecución de macros y *scripts* de ImageJ en el que no se muestran imágenes para aumentar la eficiencia de procesado. Ver herramientas de ejecución en serie (*batch processing tools*), en la página 23.

Código

Conjunto de instrucciones que un ordenador debe seguir para ejecutar una tarea.

Código fuente: está escrito en un lenguaje de programación. Puede ser leído y entendido por un programador. No es directamente ejecutable por una computadora, debe ser traducido a código máquina por un compilador o un intérprete.

Código máquina/binario/ejecutable: escrito como sucesión de ceros y unos (ver abstracción). No puede ser interpretado por un programador ni traducido a código fuente.

Compilador (lenguaje compilado)

Programa que traduce código fuente a máquina. Funciona traduciendo todo el programa de golpe. El resultado es un archivo en código máquina que puede ser ejecutado muy eficientemente.

***Composite* (referido a una pila o imagen)**

Modo de gestión de imágenes multicolor. Cada uno de los colores se mantiene en imágenes separadas de una misma pila. El color de cada imagen se codifica en escala de gris con un LUT. Una barra de desplazamiento permite seleccionar cada una de las imágenes de la pila y modificar sus características. Los cambios se visualizan automáticamente para la mezcla de todos los colores. Son pilas de imágenes a todos los efectos (ver entrada correspondiente).

Flujo de trabajo

Sucesión ordenada de acciones que permiten llevar a cabo una tarea compleja. En microscopía abarcan la captura, procesado y análisis de imagen, el tratamiento estadístico de datos y la representación de resultados.

Grid/Collection stitching

Módulo de fusión de mosaicos de ImageJ que funciona en tres pasos: 1-calcula la posición relativa de dos imágenes mediante el análisis de correlación cruzada de cada una de las posibles traslaciones que se pueden dar entre ellas. El análisis de correlación cruzada se basa en el cálculo del índice de regresión entre los niveles de gris de dos imágenes. El índice de regresión vale 1 cuando dos imágenes son idénticas en una zona de solapamiento y cae hasta 0 en la medida en que no lo son. En una primera aproximación, Grid/Collection Stitching considera que la posición relativa de dos

imágenes en un mosaico coincide la traslación que devolvió el índice de regresión máximo. 2- si hay más de dos imágenes, la posición de cada una de ellas viene definida por sus vecinas, por lo que puede haber resultados incongruentes. En este caso, se aplica un algoritmo de optimización de posición global que usa, entre otros, el índice de regresión máximo calculado en el paso anterior. Si este es menor a un umbral definido por el usuario, se considera que el par de imágenes no son solapantes, y estas no se tienen en cuenta para calcular sus posiciones relativas. Si, siguiendo este criterio, una imagen no solapa con ninguna de sus vecinas, se elimina del mosaico. 3- una vez definida la posición de cada campo, se pueden aplicar diversos algoritmos para determinar el nivel de gris de las zonas de solapamiento.

Cuando el orden de los campos y el grado de solapamiento entre ellos son desconocidos, Grid/Collection Stitching calcula el índice de regresión de todas las traslaciones de cada una de las posibles combinaciones de imágenes de un mosaico tomadas de dos en dos. Cuando se conocen estos datos, Grid/Collection Stitching solo calcula el índice de regresión para las traslaciones en el área de solapamiento de las imágenes contiguas, lo cual es mucho más eficiente. En este último caso, Grid/Collection Stitching, obtiene el orden de los campos del nombre de la imagen y asume que el mosaico es regular (si falta alguna imagen, la macro se interrumpe).

Al ejecutarse este módulo, se genera un archivo que contiene las coordenadas de los campos. Este archivo se puede utilizar para volver a fusionar la misma colección de imágenes saltándose los pasos 1 y 2, lo cual es mucho más eficiente.

Grid/Collection Stitching asume que todos los campos de un mosaico se localizan dentro de una única carpeta.

GUI/Interfaz Gráfica de Usuario/Interfaz/Interfaz de usuario

De sus siglas en inglés: *Graphical User Interface*. Sistema de comunicación entre un ordenador y un usuario que permite al segundo recibir información mediante imágenes (generalmente a través de un monitor) e introducirla rellenando campos de texto, seleccionando casillas de verificación, opciones en menús desplegables etc. (generalmente a través de un teclado o ratón).

HCS/Cribado de alto rendimiento

De sus siglas en inglés: *High Content Screening*. Técnica microscópica consistente en la captura y análisis automático de un número elevado de muestras con células. Suele requerir preparar muestras celulares o tisulares (habitualmente para marcar los componentes que van a ser estudiados), obtener fotografías, procesar y analizar las capturas por análisis de imagen, y visualizar y tratar los datos obtenidos.

Hyperstack

Pila de imágenes que codifica más de tres dimensiones. Habitualmente, x e y más z, t o el número de canales (c). Ver la entrada "Pila de imágenes". La navegación a través de las dimensiones se realiza mediante barras de desplazamiento. Al igual que las pilas de imágenes, pueden codificarse en un único archivo, pero no todos los programas de visualización de imágenes son capaces de gestionarlos.

IDE/Entorno de desarrollo integrado

De sus siglas en inglés: *Integral Development Environment*. Aplicación que facilita al programador el desarrollo de *software. S*uele incluir herramientas de edición de código, coloreado de sintaxis, autocompletado de palabras, etc.

Intérprete (lenguaje interpretado)

Programa que traduce código fuente a máquina instrucción a instrucción. Cada instrucción es leída, traducida, ejecutada y la traducción borrada en tiempo real, por lo que la ejecución de estos programas es más lenta que la de un programa ya compilado. Es útil en programación porque permite ejecutar código fuente, facilitando la detección y corrección de errores.

Licencia

Contrato legal que vincula a un usuario y al propietario de un *software*. Permiten mantener la sostenibilidad de los modelos de desarrollo de *software*.

Tipo *copyright*: la propia del *software* propietario. Su objetivo es defender los intereses económicos del propietario, evitando la divulgación del código fuente.

Tipo *copyleft*: la propia del *software* libre. Su objetivo es defender los derechos del usuario a utilizar, estudiar, modificar y distribuir el *software*, facilitando la divulgación del código fuente.

Log

Tipo de interfaz gráfica de usuario. Sirve de registro de la actividad del *software* o para mostrar mensajes al usuario durante la ejecución de un programa.

LUT/Paleta de falso color

De sus siglas en inglés: *Look Up Table*. Representación visual del valor de intensidad de los píxeles de una imagen monocanal. Lo más habitual (en una imagen de 8 bit) es representar cada uno de los posibles 256 valores de intensidad en niveles de gris de luminosidad equidistante entre 0 (negro absoluto) y 255 (blanco absoluto), pero existen LUTs que los codifican en niveles de colores (entre negro y rojo absoluto, por ejemplo), combinaciones de colores o distribuciones no lineales de niveles. Los LUTs modifican el modo de visualizar una imagen, pero no afectan al contenido de la misma.

Macro

Ver página 23 (“Grabado de comandos”).

Make Montage

Comando de ImageJ capaz de transformar una pila de imágenes en un montaje. Sigue la sintaxis: run("Make Montage...", "columns=int1 rows=int2 scale=fract1 first=int3 last=int4 increment=int5 border=int6 font=int7 label"). Los argumentos modifican el montaje final: int1 e int2 especifican el número de filas y columnas en las que se distribuirán las imágenes, fract1 su escala de reproducción, int3, 4 y 5 permitirán seleccionar las imágenes de la pila que aparecerán en el montaje (ver Make Substack) e int6 determinar el grosor en pixeles del borde de las mismas. El argumento opcional label permite marcar con un título cada una de las imágenes del montaje, int7, determina el tamaño de la letra del mismo. Los argumentos int son del tipo entero, los del tipo fract son decimales exactos entre 0 y 1 (incluidos).

Make Substack

Comando de ImageJ capaz de generar pilas de imágenes a partir de otra. Sigue la sintaxis: run(“Make Substack…”, “slices: int1-int2-int3”), siendo int1, 2 y 3, tres números enteros que codifican el rango de imágenes de la pila a procesar (int1-int2) y un incremento (int3). Por ejemplo, run(“Make Substack…”, “slices: 3-9-2”), sobre una pila de 9 o más imágenes, genera un pila con la tercera, la quinta, la séptima y la novena imágenes de la pila original.

Merge Channels

Comando de ImageJ capaz de unir imágenes o pilas de imágenes monocanal en una imagen o pila multicanal. Sigue la siguiente sintaxis: run("Merge Channels...", " "red=*None* green=*None* blue=*None* gray=*None* create keep"). El valor de los argumentos: red, green, blue y gray (cuyo valor por defecto es *None*) deben ser sustituidos por el nombre de la imagen o pila que vaya a codificar el canal rojo, verde, azul y gris, respectivamente (si es que los hay). Create y keep son parámetros opcionales, su presencia asegura que el resultado es una imagen o pila multicanal tipo *composite* y que se mantienen abiertas las pilas monocanal, respectivamente.

Módulo/*plug-in*

Bloque de código que permite extender las capacidades de un programa. Se diferencian de los *scripts* en que se suelen escribir en lenguajes de programación de propósito general (ver página 22, “Lenguajes de programación de propósito general”).

Monocanal

Imagen en la que cada píxel codifica un único valor con información relativa a la intensidad de la luz. Si se capturó con un sensor sensible a una longitud de onda (o se seleccionó la luz incidente, mediante un filtro, por ejemplo), la imagen captura la intensidad de luz para un color determinado, de lo contrario, la imagen codifica la intensidad de luz para una mezcla de colores. Estas imágenes suelen visualizarse en escala de grises o con LUTS (paletas de falso color).

Multicanal

Imagen cuyos píxeles codifican, separadamente, varios valores con información relativa a la intensidad de la luz de varias longitudes de onda. Se suelen capturar con un sensor sensible al color o mediante varias exposiciones de un sensor insensible al color en el que se ha seleccionado el color de la luz incidente (mediante la aplicación de filtros, por ejemplo).

Lo más habitual es representar estas imágenes en un espacio RGB. En este caso, cada píxel codifica un valor de intensidad (a una profundidad de 8-bit) para cada uno de los colores primarios (rojo, verde y azul). A cada posible mezcla de los tres valores de intensidad se le asigna el color que el ojo humano percibiría en una situación real.

Las imágenes RGB multicanal pueden descomponerse en una imagen monocanal por canal primario, y viceversa (varias imágenes monocanal pueden fusionarse en una única imagen RGB).

Multi-TIFF

Archivo .tif que codifica varias imágenes. Ver Pila/Pila de imágenes.

Pila/Pila de imágenes

Conjunto de imágenes que se muestran en un mismo marco. Generalmente cada imagen representa diferentes canales, planos focales o sucesiones temporales de un mismo campo. Solo es visible una imagen a la vez, pero existen medios para seleccionar la imagen que se quiere visualizar. Pueden codificarse en un único archivo (por ejemplo multi-TIFF), pero no todos los programas de visualización de imágenes son capaces de gestionarlos. En ocasiones codifican tres dimensiones de un campo, que suelen ser, habitualmente, las dimensiones espaciales del mismo (x: anchura del campo, y: altura del campo) más la dimensión temporal (t) o la profundidad (z).

Procesado de imagen

Conjunto de técnicas que permiten modificar una imagen, generalmente con la finalidad de permitir extraer información de la misma.

Programa

Conjunto de instrucciones que se pueden ejecutar en un ordenador.

RAID

De sus siglas en inglés: *Redundant Array of Independent Disks*. Conjunto redundante de discos independientes. Sistema de almacenamiento de datos que se compone de varias unidades independientes. La distribución de los datos obedece a criterios de capacidad, fiabilidad y eficiencia de ejecución. Para asegurar la integridad de los datos en caso de fallo o daño del sistema, por ejemplo, estos se distribuyen de forma redundante en los diversos componentes.

Reconstrucción/*reconstruct*

Operación lógica que requiere de dos imágenes binarias, denominadas semilla y máscara. Se ejecuta en dos pasos: 1- operación lógica tipo AND entre las imágenes anteriores y 2- selección de los objetos en la máscara que contienen al menos un píxel encendido como resultado de la operación anterior. El resultado es una imagen como la máscara en la que desaparecen los objetos que no tienen ningún elemento común con la semilla.

ROI

Región de interés (de sus siglas en inglés: *region of interest*). En ImageJ, las ROIs son selecciones que se superponen a la imagen sin modificarla. Permiten limitar los análisis (conteo de objetos, por ejemplo) a las ROIs seleccionadas. Pueden crearse manualmente o como resultado de un paso de análisis.

ROI Manager

Aplicación de ImageJ que permite guardar, cargar e interactuar con ROIs (seleccionando, renombrando o eliminando alguna de ellas, por ejemplo). Puede aparecer como resultado de un paso de análisis, en cuyo caso contendría las selecciones de los objetos que se detectaron en el mismo. Las ROIs obtenidas en una imagen pueden mostrarse sobre otra.

Roiset

Conjunto de ROIs. ROI Manager permite guardarlo como un archivo de texto comprimido que puede ser abierto posteriormente.

Script

Secuencia de instrucciones que permiten automatizar una tarea. Se diferencia de una macro en que está escrito en un lenguaje de guion (ver página 22, "Lenguajes de Guión") formalmente definido en una API, puede crear GUIs y tiene mecanismos de control del flujo (bucles) y variables. Con frecuencia se utilizan para acceder a librerías y juegos de herramientas, permitiendo implementar algoritmos complejos de forma sencilla.

Software

Sustantivo genérico que hace referencia a todos los constituyentes no físicos de un ordenador (macros, *scripts*, programas, aplicaciones, etc.).