UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Tesis de Licenciatura

Análisis del crecimiento poblacional de cultivos celulares a tiempos prolongados, monitoreados sobre un microscopio, de forma autónoma.

Martín Pustilnik, L.U. 534/99, mpustil@gmail.com

Directores:
Dra. Flavia Bonomo, fbonomo@dc.uba.ar
Dr. Marcelo J. Salierno, salierno@qi.fcen.uba.ar

Diciembre 2011

Resumen

En esta tesis se trabajó en la generación de un algoritmo de reconocimiento en el tiempo de un estadio celular discreto, el momento previo a la citocinesis (división celular citoplasmática).

La información es obtenida a partir de una secuencia de imágenes digitales tomadas a intervalos regulares desde un microscopio invertido.

El propósito del algoritmo es seguir el ciclo celular de cada una de las células de forma individual a lo largo de toda la secuencia de imágenes para lograr un análisis general del comportamiento del cultivo.

De esta manera se realizó un análisis poblacional del crecimiento del cultivo, tiempo medio del ciclo celular, tiempo medio del intervalo de la citocinesis. Con los datos obtenidos para el análisis se obtuvo también la trayectoria de cada célula (seguimiento celular o cell tracking).

División celular, Citocinesis, Microscopio invertido, Cell Traking

Dedicado

A Silvia, mi mama, que aunque el destino quiso que no estuviera presente, la tengo siempre presente en mi mente y en mi corazón.

Agradecimientos

A mi familia por darme el apoyo
A mis amigos por darme la fuerza

para seguir adelante.

A mis directores por haberme enseñado como escribir una tesis.
A Maria Elena por sus aportes que me ayudaron a concluirla.

Índice general

Capítulo 1

Introducción

1.1. Cell Traking

El seguimiento de un cultivo celular en experimentos clásicos de biología se realizaba en general de forma muy discreta a intervalos de varias horas, ya que esto demanda tomar el cultivo fuera del incubador para llevarlo al microscopio a tomarle una fotografía. En la actualidad, existen incubadores para mantener cultivos sobre un setup de microscopía que permite tomar fotos a intervalos regulares [2], y de esta manera obtener una información más dinámica de los cambios en los cultivos.

El siguiente desafío en estos últimos años, fue encontrar formas de analizar toda la información que se desprende de una ristra de imágenes tomadas a intervalos regulares de unos pocos minutos durante varias horas o incluso días. En base a esto, comenzaron a surgir actualmente software especializados en el seguimiento de célula única y en el seguimiento automatizado del comportamiento de todas las células que aparecen en un campo focal.

El seguimiento celular generalmente se plantea como un proceso de dos pasos [1]:

1. La segmentación(el aspecto espacial del problema): Se denomina segmentación a la parte del proceso que divide la foto en segmentos que tienen significado biológico, por ejemplo:

   a) Células

   b) Paredes celulares

2. La asociación de segmentos(el aspecto temporal): Se relacionan los segmentos la $foto_i$, con los de la $foto_{i+1}$.

En la realización del algoritmo principal, se tuvieron en cuenta las siguientes hipótesis (y axiomas) biológicas y computacionales[1]:

•El tiempo promedio para el proceso de Citocinesis es de 30 minutos.

•Este proceso se repite cada 24 horas aproximadamente, dependiendo del tipo y las condiciones celulares.

•(axioma): Cuando una célula c se divide, diremos que c no existe más y en su lugar existen dos nuevas células: c_1 y c_2.

•Las células tienen poco desplazamiento entre dos fotos consecutivas.

•Las células levantadas tienen una forma casi circular.

•No hay oclusión: No hay dos células levantadas que estén solapadas.

•Todos los objetos detectados en la foto i que cumplan con los patrones de búsqueda, son considerados células levantas[2].

• Las células levantadas forman una cantidad finita de patrones reconocibles.

1.2. Segmentación

Un método para la realizar la segmentación es comparar cada píxel respecto de un umbral [threshold] de manera que los píxels queden etiquetados para formar grupos de elementos reconocibles. Debido a su simplicidad, *Thresholding* [10] es el método más comúnmente usado. También es uno de los métodos más propensos a errores.
Sin embargo, resulta muy satisfactorio siempre que:

- Las células estén suficientemente separadas entre si. [Condición1]
- El umbral pueda distinguir lo suficiente las células respecto del fondo. [Condición2]

Existen métodos más sofisticados, que buscan patrones predeterminados [Template Matching] para identificar los segmentos. Estos métodos funcionan bien para fotos donde las células tienen una forma consistente, pero fallan cuando ocurren cambios morfológicos significativos. Otro método más popular, considera a la foto como un relieve topológico [Watershed Transform] [10], para luego inundar las áreas interiores de los relieves que se encuentren por debajo de un relieve mínimo predeterminado. Los mayores problemas de este enfoque son:

- La alta sensibilidad al ruido(de la imagen).
- La sobre segmentación

[1] Ver sección 1.5.1 Hipótesis
[2] Ver 2 Términos de biología

1.3. Asociación

Para realizar el seguimiento de cada célula, es necesario:

- Identificar qué segmentos son células
- Asociar satisfactoriamente la aparición de la misma célula en dos fotos consecutivas.

Todos los enfoques investigados utilizan el método de maximizar/minimizar una función objetivo $g(x)$, que cuantifica la efectividad de la asociación a ser aplicada.

En algunos casos $g(x)$ se vincula con la función distancia. En ese caso se calcula $g(x)$ considerando la distancia de cada segmento en la $foto_i$, respecto del mismo segmento en la $foto_{i+1}$.

Es decir, que se busca la asociación de segmentos(c_i) que minimice $g(x)$ (Ver Figura 1.1).

Por ejemplo si se tienen las asociaciones:
Asociación a: {(c_1,c_3); (c_2,c_4)} // s_1 = (c_1,c_3) ; s_2 = (c_2,c_4)
Asociación b: {(c_1,c_3); (c_2,c_5)}

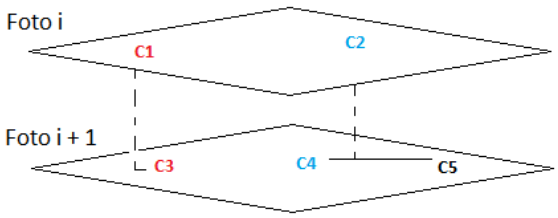


Figura 1.1: Dos fotos consecutivas con dos posibles asociaciones

El método elegirá la asociación a, por ser la que minimiza la función $\sum$i(distancia(s_i)), donde distancia(s_i) se calcula como:

$$\text{distancia}(s_i) = distanciaR^2(s_i.x,\ s_i.y)$$

En este contexto es muy difícil y engorroso el análisis durante días y en forma sistemática de cualquier cultivo. En especial un método de análisis automatizado sobre un caso particular es de gran interés para las áreas de investigación que requieren analizar procesos a largo plazo como cambios celulares morfológicos o respuestas de tratamientos que intervienen en los procesos de división, muerte y diferenciación celular, por ejemplo.

El foco de esta tesis está puesto en desarrollar una herramienta que permita obtener resultados y elaborar conclusiones a nivel biológico, de procesos que estén involucrados con cambios en la dinámica de los cultivos celulares en el tiempo. En particular se identificará el proceso de citocinesis que presenta un comportamiento singular en la célula y se analizará de esta manera el ciclo celular y la evolución poblacional en el tiempo.

1.4. Otros desarrollos

En esta dirección existen ya varios trabajos relacionados [3], mencionaremos algunos proyectos open-source:

1.4.1. CellProfiller

CellProfiller(http://www.cellprofiler.org): Es un software de análisis de imágenes celulares, diseñado para biólogos sin conocimientos técnicos en computación, ideado para analizar centenares de imágenes de manera automática.

La versión 1.0 fue desarrollada en MatLab, por lo que se necesita tener instalado MatLab para utilizar el software.

La versión 2.0 está siendo desarrollada en Python 2.5.

Fue desarrollado en el instituto Whitehead para la investigación biomédica en conjunto con el CSAIL[Computer Sciences/ Artificial Inteligence Laboratory] en el MIT[Massachusetts Institute of Technology].

1.4.1.1. CellProfiller - Segmentación

Según se explica en [4], cellProfiller (Ver Figura 1.2) utiliza su propio algoritmo de propagación, que consiste en una mejora sobre el método *Watershed* [10], para identificar las células. Cuenta además con todas las técnicas estándar ya mencionadas, como por ejemplo *Thresholding* [10].

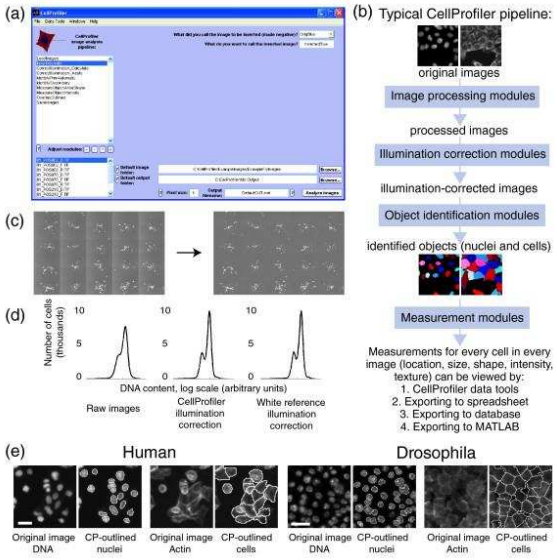


Figura 1.2: Vista general de CellProfile y sus funcionalidades más usuales

1.4.2. CellTrack

CellTrack(http://db.cse.ohio-state.edu/CellTrack/): Es un software de propósito general, para la segmentación, la asociación y el seguimiento de la migración celular.

La versión actual fue desarrollada en C++ utilizando librerías gráficas OpenCV. Fue desarrollado conjuntamente en:

- Computer Engineering Department, Middle East Technical University, Ankara, TURKEY
- Dept. of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
- Mathematical Biosciences Institute, The Ohio State University, Columbus, OH, USA

1.4.2.1. CellTrack - Segmentación y seguimiento celular

Utiliza un método de de *Thresholding* para la detección de las fronteras celulares (Ver Figura 1.3).

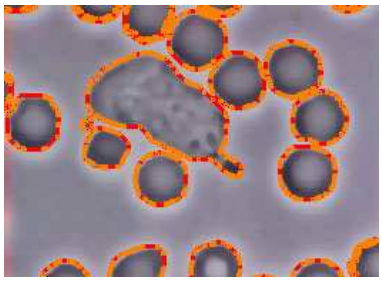


Figura 1.3: CellTrack thresholding

Y un método de *Template Matching* parametrizable para el seguimiento celular (Ver Figura 1.4).

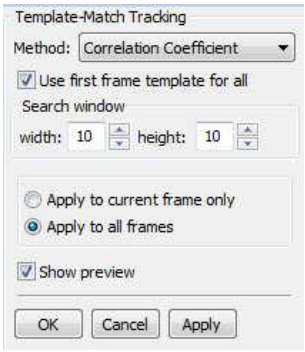


Figura 1.4: CellTrack Menu - Template Matching

CellTrack busca la mínima área rectangular r que cubra un segmento en la $foto_i$. Luego se busca en la $foto_{i+1}$ el segmento que minimice la función de seguimiento respecto de r.

En la sección 4.3.3 se comparan los resultados obtenidos en esta tesis, respecto de varios de los proyectos open-source.

1.5. Respecto del algoritmo desarrollado

Para nuestro enfoque del problema utilizamos el método *Thresholding* [10]combinado con el método *Template Matching* [10]para la etapa de segmentación.

Para la etapa de asociación, se utilizó una variación del método clásico:

- Consideramos la información de más de dos fotos para asociar dos segmentos.
- La función objetivo a ser minimizada es la distancia en R^2.

Para el procesamiento de las imágenes se adaptaron librerías conocidas de código abierto:
FreeImage: (http://freeimage.sourceforge.net/index.html)

En esta tesis se utilizaron datos biológicos obtenidos del trabajo realizado en la tesis doctoral de Marcelo Salierno: "Micromanipulación fisicoquímica sobre arreglos celulares y célula única bajo monitoreo constante", cuyo resumen se incluye en el Apéndice 6.1.

1.5.1. Hipótesis(y axiomas) biológicas y computacionales

Dividiremos a las hipótesis en biológicas (B_i) y en computacionales (H_i):

Hipótesis biológicas:

- B_1: La duración promedio del proceso de citocinesis para las células de tipo *BHK-21* es de 30 minutos (intervalo que consideramos a la célula levantada).
- B_2: El proceso de división se genera cada entre 12 y 24 horas, dependiendo del tipo y las condiciones celulares.
- B_3(axioma): Cuando una célula c se divide, diremos que c no existe más y en su lugar existen dos nuevas células: c_1 y c_2.

De esta manera, si antes de la división teníamos n células, luego de una división exitosa, tendremos $n+1$ células. (ver Apéndice 6.6 Validez de la formula $P_i \approx P_0 + L_i$ por inducción en i)

- B_4: Las células tienen poco desplazamiento entre dos fotos consecutivas. Nombraremos a esta variable como "desplazamiento máximo" (Ver: 3.1.4Parámetros del algoritmo y valores empíricos).

Hipótesis computacionales:

- H_0: Las células levantadas tienen una forma casi circular.
- H_1: No hay oclusión: No hay dos células levantadas que estén solapadas.
- H_2: Todos los objetos detectados en la $foto_i$ que cumplan con los patrones de búsqueda, son considerados células levantas.

- H_3: Las células levantadas forman una cantidad finita de patrones reconocibles.

Como la misma célula permanecerá levantada durante varias imágenes(B_1), agregaremos una hipótesis adicional para identificar cada célula individualmente:

- H_4: Si una célula no se detecta como levantada durante al menos 5 fotos consecutivas, no se contabilizara como división celular.

1.5.1.1. Hipótesis adicionales

Si bien en teoría pueden existir oclusión (recordar que el modelo es tridimensional), en la práctica esto es muy poco frecuente.

(1) Asumiremos que si se detecta una célula con centro en C_1 y radio R, la célula más cercana con centro en C_2 esta al menos a distancia $3 * R$ de C_1(Ver Figura 1.5).

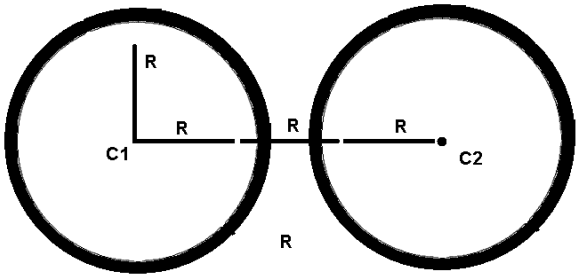


Figura 1.5: Distancia mínima entre células

Notar que:
• Esto solamente se puede asumir porque los radios de las células en estado B, son muy similares.
• Se asume un radio mínimo, que considere el radio más chico de la célula que se quiere detectar.

Las dos células de la imagen tienen un radio mayor que el radio mínimo asumido.
Esto nos evitó testear áreas que ya fueron examinadas.
El ahorro total de áreas de búsqueda fue de $\Pi * (3 * R)^2$ por cada célula.

(2) Si C_1 está en L_1 y C_1' está en L_2, se considerara que es la misma célula, siempre que se haya desplazado menos que un desplazamiento máximo admitido (Ver Figura 1.6).
Además, no tiene sentido buscar C_1 en M_2, a un distancia más allá de $C_1 + DesplazamientoMáximo$.

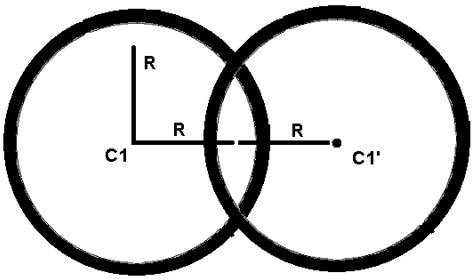


Figura 1.6: Desplazamiento máximo para la misma célula

El desplazamiento máximo de una célula levantada, entre dos fotos consecutivas: Desplazamiento Máximo $= 2 * R$

(3) Asumiremos que para cada célula levantada C_i, dentro de la misma foto ($foto_k$), y en las siguientes 5 fotos consecutivas, no puede existir otra célula levantada C_j, dentro del área de influencia de C_i.

Por lo tanto, las células levantadas detectadas para algún k en $foto_{k+1}, \ldots, foto_{k+5}$, dentro del área de influencia de C_i, serán consideradas como C_i, (ver hipótesis B_1).

Capítulo 2

Términos de biología

2.1. Conceptos Básicos

A continuación, se describen los conceptos de biología utilizados para la elaboración del trabajo:

Ciclo celular

Es un conjunto ordenado de sucesos que conducen al crecimiento de la célula y la división en dos células hijas. Se inicia en el instante en que aparece una nueva célula, descendiente de otra que se divide, y termina en el momento en que dicha célula, por división subsiguiente, origina dos nuevas células hijas [9].

División celular

Una célula progenitora se divide en dos células hijas idénticas. Esta fase incluye la mitosis, a su vez dividida en: profase, metafase, anafase, telofase; y la *citocinesis*. Si el ciclo completo durara 24 hs, la fase de mitosis duraría alrededor de 30 minutos [9].

Mitosis

Reparto de material genético nuclear.

Citocinesis

División del citoplasma.

Células levantadas

Utilizaremos los términos "*Citocinesis*", "*Mitosis*", "*División celular*" y "*Célula levantada*" para referirnos a células en la fase de *Mitosis*.

Ejemplo del ciclo celular:

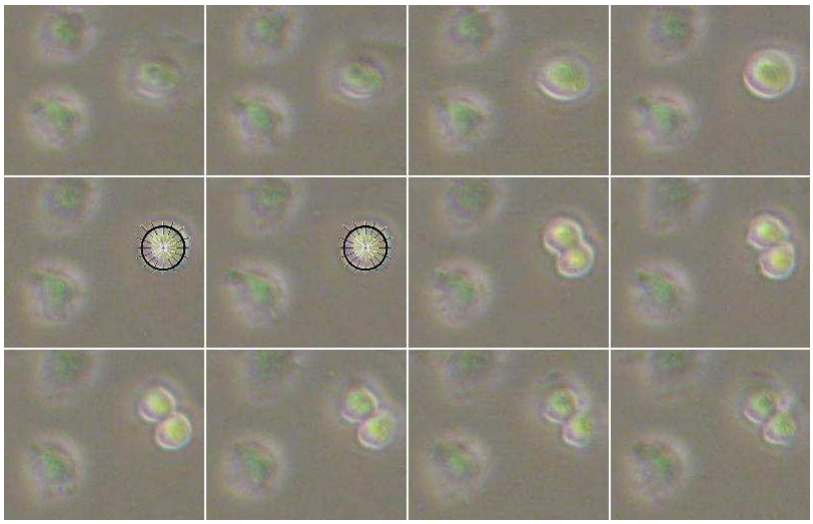


Figura 2.1: La célula c, se divide en dos nuevas

En la Figura 2.1 se muestran una de cada tres fotos. Por lo tanto la célula c fue detectada durante seis fotos (alrededor de 30').

Durante el ciclo celular, una célula pasa a diferentes estados:

- *Levantada*: Cuando comienza el proceso de división, la célula "se levanta" y toma una forma casi circular. En este caso diremos que la célula esta levantada. Al final de un proceso de división exitoso, y durante algún tiempo, se generan dos nuevas células, generadas a partir de la célula original, que comienzan su ciclo de vida "levantadas".
- *Pegada*: Para cualquier célula viva que no se encuentre en proceso de división, diremos que está pegada.

Para empezar a modelar el problema computacionalmente, representaremos el ciclo de vida celular con un diagrama de transición de estados (Ver Figura 2.2):

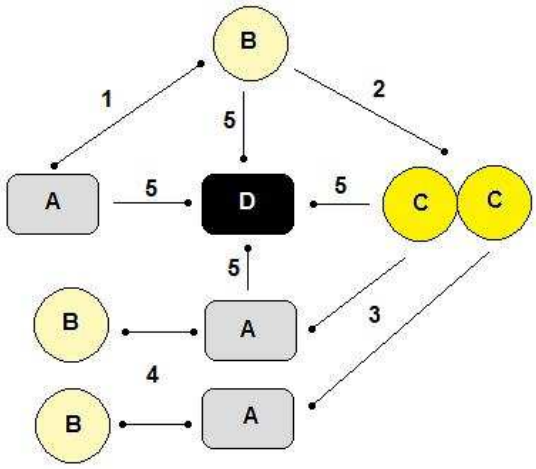


Figura 2.2: Modelo del ciclo de vida de la célula. Comienza en el estado A, pegada.

Explicaremos un ciclo de vida típico: (Los estados, y las transiciones están remarcados entre paréntesis):

- (1): la célula pasa de estar Pegada(A) a estar Levantada(B).
- (2): Comienza la División, alcanzando el estado (C).
- (3): las dos nuevas células se pegan, y vuelven al estado (A).

Luego de cierto tiempo, cada nueva célula puede comenzar nuevamente con el paso(1). En el diagrama se denota como paso(4).

Mostraremos un algoritmo que detecta eficientemente los estados B y C.

Existen también algunos pasos alternativos:

- (1.b): En algunos casos puede volver a pegarse, sin dividirse realmente.
- (1.c): También puede ocurrir que la célula en lugar de dividir su cuerpo(citoplasma) y su núcleo divida solamente su núcleo, creando dos células(dos núcleos) que comparten el mismo citoplasma. Este estado de células multinucleadas no será tomado en cuenta como división.
- (5): Las células pueden Morir(D) en cualquier momento del ciclo de vida.

Capítulo 3

Mitosis Detector

Desde que se toma la foto, hay varios procesos que se realizan antes de que el sistema genere una salida (Ver Figura 3.1).

Hay un preprocesamiento de las fotos($Paso_1$: Ver 3.3 archivos DAT) y dos formas de generar estadísticas:

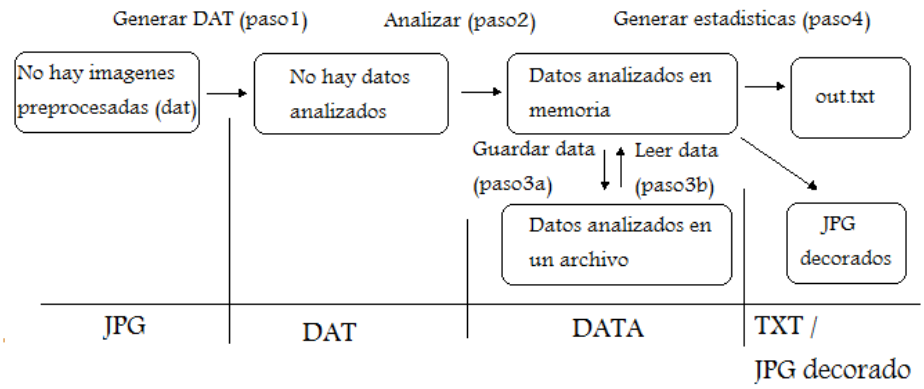


Figura 3.1: Pasos de Mitosis Detector

El Orden asintótico de ejecución [8] de todos los algoritmos fue *polinomial*. Ver apéndice 6.5 para las demostraciones.

Dependiendo del algoritmo, fue *polinomial* en cantidad de fotos o en en cantidad de células.

3.1. Algoritmo principal

Para el algoritmo principal($Paso_2$) seguimos el enfoque clásico para este tipo de algoritmos, que consiste en un procesamiento de las fotos en etapas o pasos.

Algoritmo principal del analizador

Algoritmo 3.1 Algoritmo Principal

```
Para cada dat i en DAT
        Cargar dat i en la foto i
        Procesar la foto i
        Destruir la foto i-1 para todo i > 0
Fin
Limpiar las celulas inactivas
```

Algoritmo 3.2 Procesar cada foto i

```
Para cada píxel(x,y) sin examinar, que no tenga objetos detectados
    Si se encuentra el patrón del asterisco en (x,y)
        Llamaremos celu j a esa detección
        Si no es una decteccion redundante en la foto i
            Actualizar <- (Agregar celu j en (foto i).celus)
            Si Actualizar
                Agregar celu j en las Celus generales
            Fin
            Marcar el objeto detectado(x,y) en la foto i
        Fin
    Fin
Fin
```

El cálculo del Orden de llamar i veces al algoritmo "Procesar cada foto i" se encuentra en el Apéndice 6.5.

3.1.1. Agregar una célula en el diccionario de células

Esta función recibe la coordenada donde se encontró un patrón asterisco. Puede ocurrir alguna de estas tres situaciones:

- Es una célula que ya existía, se indica diciendo:
  - ID_Cel = -1
- Era una célula ya detectada en la foto anterior(en una coordenada cercana):
  - Se adquiere el ID_Cel de la foto anterior

- Es una célula nueva: o Se genera un nueva
    - ID_Cel

Se retorna un ID_Cel, y un booleano, que indica si que se debe actualizar además, el diccionario de células principal.

Algoritmo 3.3 AgregarFotoCel(c: Coordenada, f: Foto) -> <ID_Cel, actualizar>

```
Si CelCercana(c)                          //en la foto actual
    ID_Cel = -1
    Actualizar = falso
Si CelCercanaAnt(c, ID_Cel)       // obtiene el ID_Cel, si existía, en la foto anterior
    Si ID_Cel no estaba en f
        Crear una nueva célula c1
        c1.ID_Cel = ID_Cel
        c1.c = c
        c1.Activa = verdadero
        Agregar c1 en f
    Fin
    Actualizar = Verdadero
Sino
    Crear una nueva célula c1
    ID_Cel = (Obtener nuevo ID_Cel)
    C1.ID_Cel = ID_Cel
    c1.c = c
    c1.Activa = verdadero
    c1.Nueva = verdadero
    Agregar c1 en f
    Actualizar = Verdadero
Fin
```

Algoritmo 3.4 Actualizar el diccionario principal

```
Si existe celu en Celus //Celus es el diccionario principal de células
    Celus(celu).ID_FotoHasta = ID_Foto
    Celus(celu).c = celu.c
Sino
    Crear una nueva célula c1
    c1.ID_Cel = celu.ID_Cel
    c1.ID_FotoDesde = ID_Foto
    c1.ID_FotoHasta = ID_Foto
    c1.Activa = Verdadero
Fin
```

3.1.2. Limpiar células inactivas

En cada foto existen un conjunto de células que no están en proceso de división, o simplemente son objetos mal detectados.

Llamaremos a estos objetos: "*Células inactivas*":
Existe por lo tanto un proceso para eliminar estos casos:
Ejemplo1

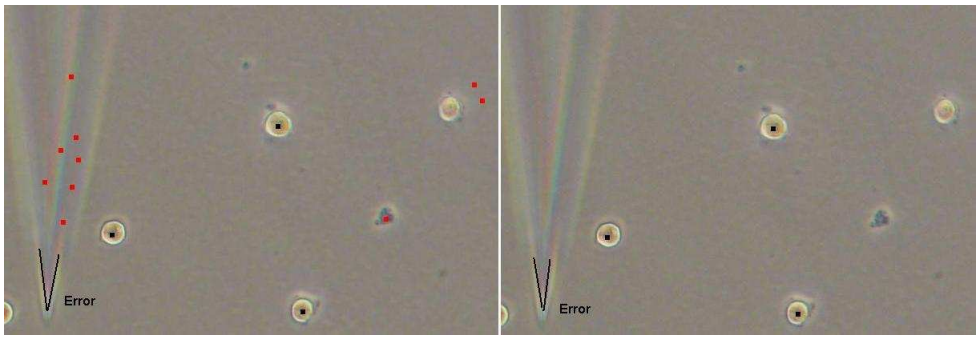


Figura 3.2: Izquierda: Con todos los objetos detectados. Derecha: Solo con células activas

Ejemplo2

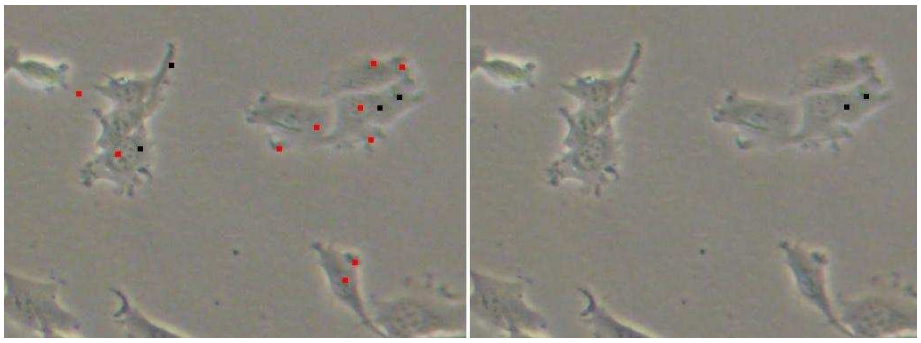


Figura 3.3: Izquierda: Con todos los objetos detectados. Derecha: Solo con células activas

Las "células inactivas" se presentaron en general, por alguno de los siguientes motivos:
En la Figura 3.2 izquierda:

- Errores en la imagen del microscopio(en la foto esta marcado con una V).
    - Este tipo de error en general produce falsos positivos.

- Manchas que aparecen en una sola foto.
- Células que comienzan el proceso de citocinesis, pero que no lo concluyen.

En la Figura 3.3 izquierda:

- Células que no están levantas, que son confundidas con células levantadas.
- Células que "desaparecen" y "reaparecen" en fotos posteriores:
    - Llamaremos a estas, células perdidas.

3.1.2.1. Reglas para limpar células inactivas

Para combatir estos problemas, se utilizaron la hipótesis B_1(El tiempo promedio para el proceso de Citocinesis es de 30 minutos), y la hipótesis B_4(Las células tienen poco desplazamiento entre dos fotos consecutivas), para construir un conjunto de reglas que el algoritmo "Limpiar células inactivas" aplica:

Regla1:

Condición:

La célula c_1 aparece en tres fotos consecutivas, pero solo es detectado en dos (Ver Figura 3.4):

Figura 3.4: $c_1$marcada en las fotos 1 y 3 como activa.

Los rectángulos negros denotan células activas.

Resultado:

Si se c_1 se detecta en la$foto_1$, se pierde en la $foto_2$, pero se encuentra otra célula c_2 en la$foto_3$, cerca de las coordenadas de c_1(Ver Figura3.5):

Se considerara a c_2 [illegible] a.

Figura 3.5: El rectángulo sin relleno representa una célula detectada por inferencia que no fue vista por el algoritmo principal

Se asume que $c_1 = c_2$ y que c_1 esta en la foto intermedia.

Regla2:

Condición:

Las células que no se detecten levantadas durante por lo menos 2 fotos (Ver Figura 3.6)

Resultado:

Figura 3.6: Son descartadas como “células activas”

En la foto y en el diccionario principal.

Regla3:
Cualquier célula que tenga una célula cercana (más cerca que $SeparacionMIN$) dentro de las cinco(5) fotos previas (Ver Figura 3.7)

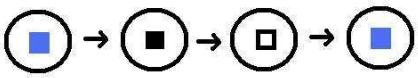


Figura 3.7: El rectángulo celeste representa una célula activa, la 1ra vez que es detectada.

Resultado:

- En la foto será descartada como “nueva”.
    - Pero permanecerá como “activa”.
- En el diccionario principal, será descartada como “activa”(la propiedad de “nueva”, no se utiliza)

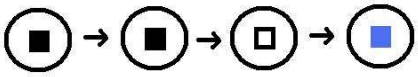


Figura 3.8: No se contabiliza dos veces la misma célula.

La población se estima contando rectángulos celestes (Ver Figura 3.8).

3.1.2.2. Algoritmo

Etapa 1

Algoritmo 3.5 Reencontrar células perdidas

```
Para cada foto i tal que  0 < i < cantidad de fotos - 5
Para cada celu j en la foto i
    Para cada foto i2 tal que      i < i2 < i + 5
    Para cada celu j2 en la foto i2
        Si DistanciaR2(celu j, celu j2) < SeparacionMIN
            Celu j2.Nueva = Falso      //unifico las dos células
            Celu j.ID_FotoHasta = j2
        Fin
```

Etapa 2

Algoritmo 3.6 Quitar falsos positivos

```
Para cada foto i
    Para cada celu j en la foto i
        Si Distancia(celu j.ID_FotoHasta, celu j.ID_FotoDesde) < 2
            celu j.Nueva = falso
            celu j. Activa = falso //Diccionario ppal
            Celus(celu j.ID_Cel). Activa = falso
        Sino
            //Observamos en las 5 fotos anteriores
            Si i > 5 And (celu j.Nueva)
                Continuacion
            Fin
        Fin
```

Algoritmo 3.7 Continuación

```
Para cada foto i2 tal que   Restar(i,6) < i2 < i
    Para cada celu j2 en la foto i2
        Si Not(celu j.ID_Cel == celu j2.ID_Cel)
            Si DistanciaR2(celu j, celu j2) < SeparacionMIN
                //Notar que queda activa
                celu j.Nueva = falso
                //Diccionario ppal
                Celus(celu j.ID_Cel). Activa = falso
            Fin
        Fin
```

$SeparacionMIN = 3 * R$

3.1.3. Estructura de datos principales

Células: Representa a todas las células en forma de un diccionario. Contiene la información global de todas las células levantadas que se pudieron detectar.

La información más importante es de cada célula es:

- El intervalo de fotos en que está levantada.

Fotos: Representa la secuencia de todas las fotos que se analizaron.

La información más importante de cada foto es:

- Una matriz m, booleana con la abstracción de la foto en el sistema.
- Un diccionario que contiene las células levantadas, detectadas solo en esa foto.

Decisiones tomadas en el diseño y en la implementación:

- La matriz m será descartada luego de ser usada, de manera de ser eficientes en el uso de memoria.
- Como la misma célula suele estar en mas de dos fotos consecutivas, también estarán "repetidas" en la estructura de datos, por razones de optimización.
    - Por ejemplo, si c_1 esta en la $Foto_1$ y en la $Foto_2$, c_1 también esta referenciada en la estructura de datos de la $Foto_1$ y de la $Foto_2$.

Mas formalmente

(0) Analizador: El sistema es un analizador, que tiene una tupla:

- <*Células*, Fotos>

(1) Células: Es el diccionario principal:

- Diccionario(ID_Cel) $\rightarrow$ Célula

(2) Célula: Es una 8-Upla con la información de una célula Célula:

- <ID_cel, ID_FotoDesde, ID_FotoHasta, X, Y, Nueva, Activa, ID_CelAsociada>

(3) Fotos: Es la secuencia de todas las fotos:

- Secuencia(Foto)

(4) Foto: Es una 3-Upla con la información de una foto:

- <ID_Foto, *Células*, Imagen>

(5) Imagen: Es una matriz de booleanos:

- Matriz(n,m): Bool

Notar que tanto el tipo Analizador como el tipo Foto, utilizan el tipo Células. Sin embargo en cada caso, el tipo Células tiene *roles diferentes*:

Atributo	Que representa en una Foto	En un Analizador
ID_Cel	Identificador único de cada célula.	(Lo mismo)
ID_FotoDesde	(No se utiliza)	Primer foto donde fue vista.
ID_FotoHasta	(No se utiliza)	Ultima foto donde fue vista.
X, Y	Posición dentro de la foto	Ultima posición donde fue vista.
Nueva	Si fue detectada por primera vez en esa foto	(No se utiliza)
Activa	Es Activa, si la detección se considera una célula. Luego de pasar por un filtro.	Es activa si se considera una célula O tiene una célula asociada.
ID_CelAsoc.	(No se utiliza)	Si cel1.ID_Cel = cel2.ID_CelAsociada: Se consideraran como la misma célula.

3.1.4. Parámetros del algoritmo y valores empíricos:

Radio Mínimo: Es el radio de referencia para calcular el diámetro interior mínimo que tendrán las células levantadas, que podrán ser detectadas por el algoritmo (Ver Figura 3.9).

C_1tiene centro en C_1 y un radio R_1
C_2tiene centro en C_2 y un radio R_2

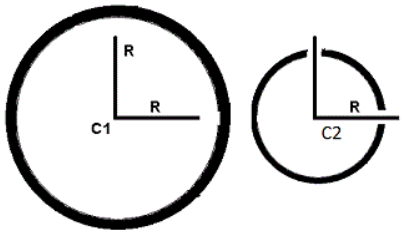


Figura 3.9: Radio interior

Denominaremos a R como el mínimo radio que puede tener una célula.
En la Figura 3.9,C_1 fue detectada por tener $R_1 > R$, mientras que C_2 no, por tener $R_2 < R$

Desplazamiento Máximo: Es la máxima distancia que se puede desplazarse una célula en dos fotos consecutivas, para ser considerada como la misma.

Separación mínima: Es la distancia mínima de separación entre los centros de cualquier par de células (c_1, c_2), dentro de la misma foto.

Umbral: Es un parámetro necesario para transformar las fotos a dos colores.

Los valores teóricos corresponden a los valores por defecto, antes de calibrar el programa para un lote en particular.

Los empíricos corresponen al lote utilizado en (4) la etapa de experimentación.

Variable	Valor teórico	Valor empírico
R = Radio mínimo	7 píxels (equivalente a 50 micrones)	
Desplazamiento máximo	$2 * R$	
Separación mínima	$3 * R$	$4 * R$
Umbral	Ver: (4.2.1) Elección del Umbral	<Rojo:120; Verde: 120; Azul: 130 >*
Patrón Asterisco - Trazo fino	$\frac{R*\sqrt{2}}{2}$	$\frac{R}{2}$
Patrón Asterisco - Trazo grueso	$R * \sqrt{2}$	$R * 2$

(*) Los valores posibles van de 0 a 255.

La elección de los valores dependen de la naturaleza de la imagen, del software de procesamiento de imágenes, y finalmente, del algoritmo de detección.

Considerando la naturaleza del lote de imágenes que se analizó, se ajustó la longitud del trazo y el umbral, para minimizar los errores de detección de células.

3.2. Patrón Asterisco "*"

En cada foto se realiza una búsqueda exhaustiva de un patrón(*Template Matching*) que identifica que objetos son células.

El patrón con el que se obtuvieron mejores resultados fue el *patrón asterisco*.

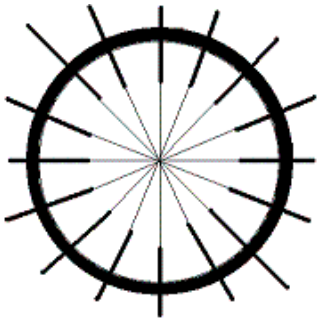


Figura 3.10: Esquema de búqueda Patrón Asterisco.

El contorno de una célula levantada, es atravesado por los rayos del Patrón Asterisco.

Los trazos buscan el centro de una célula levantada:

- La parte gruesa del trazo indica que se alcanzó el contorno celular (en color negro).
- La parte fina debe quedar situada en un área blanca, rodeada por un área negra.

Se eligió este patrón de búsqueda (Ver Figura 3.10) porque además de cumplir con todas las hipótesis, es eficiente en varios sentidos:

- Es esparzo: No es necesario examinar toda el área dentro del patrón
- Resuelve el problema de los contornos que no están cerrados, y los problemas clásicos[5] de detección de células (Ver Figuras 3.11, 3.12y3.13):

  - En la Figura 3.11, sobre-segmentación: una célula es dividida en más de un segmento
  - En la Figura 3.12, sub-segmentación: un segmento cubre dos células completamente
  - En la Figura 3.13, segmentación mezclada: una porción es identificada como célula, pero en realidad se superpone con más de una célula (no cubriendo completamente ninguna).

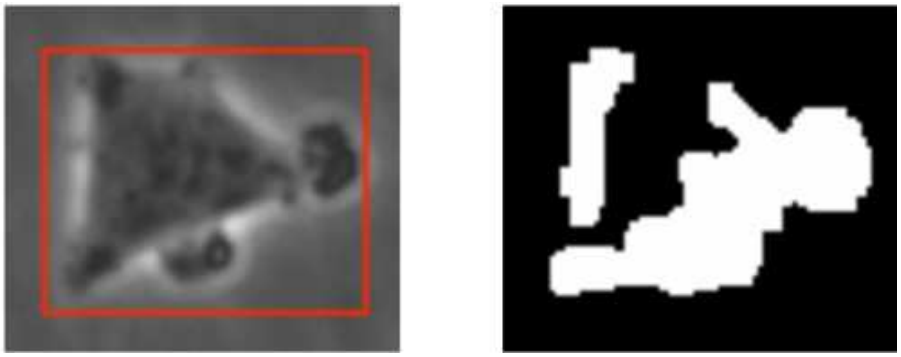


Figura 3.11: Sobre-segmentación

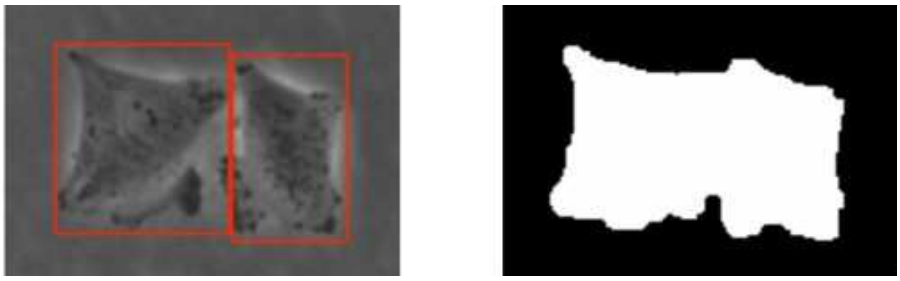


Figura 3.12: Sub-segmentación

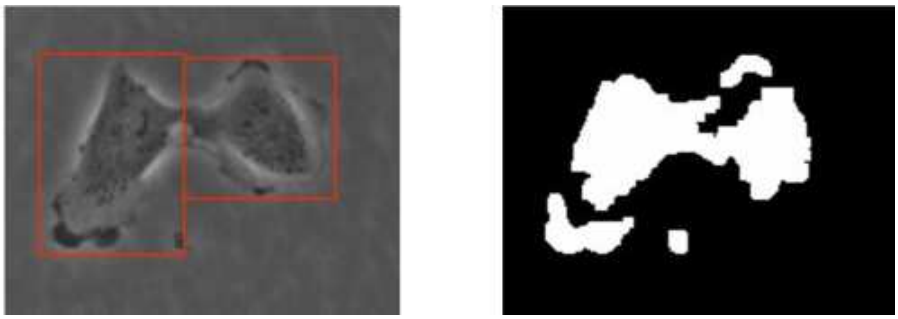


Figura 3.13: Segmentación mezclada

3.2.1. Estructura del Patrón Asterisco

Cada asterisco consta de n rayos equidistantes. Representaremos cada rayo "y" como un vector en R^2, definido de la siguiente forma:

• Origen de y: El origen (el cero) de y, está situado en centro de una célula c. Por ejemplo, c_1.

• Longitud o norma de y : Si r es el radio de c, fijaremos la longitud de y en: $\sqrt{2} * r$, es decir, $|y| = r * \sqrt{2}$.

Dividimos el rayo de longitud d, en dos. En un trazo fino y en un trazo grueso (Ver Figura 3.14):

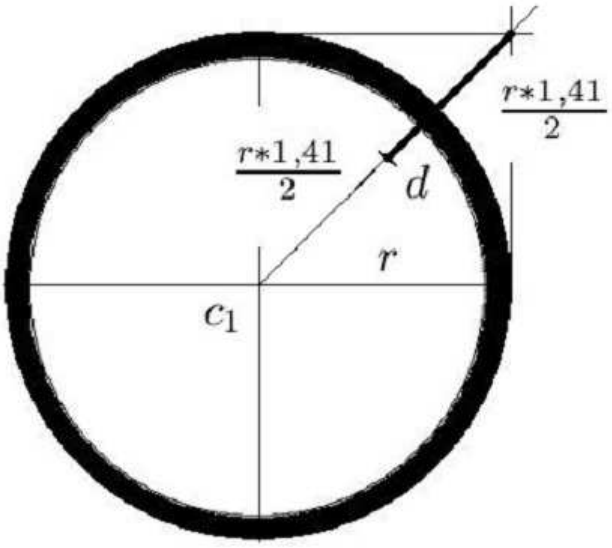


Figura 3.14: Trazo fino y Trazo grueso

Se particiono "y" de esta manera por los siguientes motivos:

1. $|y|$ tiene que ser mayor que r, para poder encontrar en algún momento, dentro de la trayectoria de "y", la frontera de la célula.
2. $|y|$ tiene que ser acotada, para poder implementar un algoritmo eficiente
3. Tenemos que garantizar: Que el trazo fino quede dentro de la frontera. Y el trazo grueso la atraviese.

Por trigonometría, sabemos que para todo triángulo rectángulo con lados de longitudes r, d, r (Ver Figura 3.15); vale que:

- $|d| > |r|$
- $|d/2| < |r|$

De hecho $|d/2|$ es exactamente $r * 0,7071...$
Entonces, se alcanza la propiedad 3.

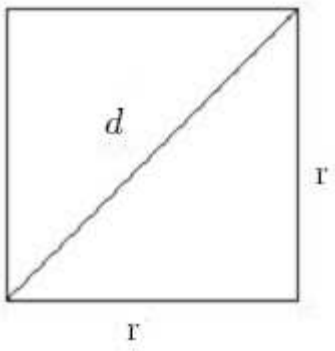


Figura 3.15: Triangulo r, d, r: $d = |y| = r * \sqrt{2}$

Para que el patrón reconozca objetos con las características de una célula, dentro de una foto, tiene que cumplirse en todos los rayos del asterisco, la siguiente propiedad:

- El trazo fino debe permanecer en un área blanca
- El trazo grueso
  - Comienza en un área blanca.
  - En algún momento tiene que encontrar un área negra.
  - Y tiene que encontrar otra área blanca dentro del límite de $d/2$.

Denominaremos a esta propiedad *PatronRayo*; al igual que a la función booleana que lo computa.

3.2.2. Algoritmo

A continuación se detalla la función, que computa la propiedad PatronRayo:
PatronRayo(c: Coordenada, d: Dirección) -> Bool

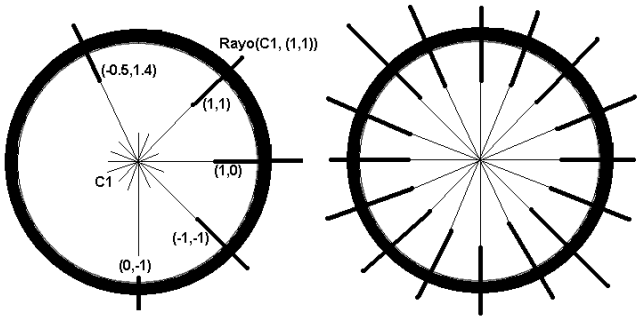


Figura 3.16: Izquierda: PatronRayo en 5 direcciones. Derecha: PatronRayo en 16 direcciones.

En nuestra implementación se utilizará en 16 direcciones, de la siguiente manera:

Algoritmo 3.8 PatronRayo(c: Coordenada, d: Dirección, conRecursion: Bool)

```
retValue = Rayo(c, d)
Si conRecursion y retValue
  retValue = retValue and  PatronRayo(c, d * -1, falso)
// Llamada sin recursión
Fin
Devolver retValue
```

Notar que la primera vez se llama con recursión, para calcular los rayos opuestos.

Para una Coordenada c, en particular llamamos a PatronRayo con las siguientes 8 direcciones:

- Vector tipo1: $(1,-1);(1,0);(1,1);(0,1)$;
- Vector tipo2: $(0{,}5,1{,}34);(-1{,}34,0{,}5);(1{,}34,0{,}5);(-0{,}5,1{,}34)$;

Las llamadas recursivas, generaran 8 direcciones adicionales de búsqueda. En total suman 16 direcciones (o ángulos), que se ven en la figura:

- Vector tipo1 * -1 = $(-1,-1);(-1,0);(-1,-1);(0,-1)$;
- Vector tipo2 * -1 = $(-0{,}5,-1{,}34);(1{,}34,-0{,}5);(-1{,}34,-0{,}5);(0{,}5,-1{,}34)$;

A fin de tener áreas de búsqueda uniformes y acotadas, se buscaron vectores con las siguientes propiedades:

- Todo par de vectores consecutivos están separados por el mismo ángulo. En nuestro caso: 22,5 grados (= 360/16).
- Todo par de vectores tiene norma2 similar.

- En nuestro caso: $|v| \approx 2$:
- |Vector tipo1| $= 1^2 + 1^2 = 2$
- |Vector tipo2| $= (0{,}5)^2 + (1{,}34)^2 = 2{,}04$

Notar que todos los vectores tienen norma en R^2 aproximadamente igual a 2: Esto es para limitar el radio de búsqueda.

Optimización: El algoritmo 3.8(Rayo), no se ejecuta casi nunca 16 veces.
Se ejecutará Rayo(para la dirección i), solo si Rayo(para la dirección i-1) es verdadero.

3.3. Estructuras intermedias - Archivos DAT

El sistema transforma cada JPG en un bitmap al aplicar la función de Umbral.

El resultado es un archivo DAT que se almacena en disco para su posterior procesamiento

Existen varias razones para trabajar con archivos intermedios:

El tamaño:

- Mantener un JPG en memoria cuesta $1600 * 1200 * 24$ bits= 5 MBytes aprox.
- Mientras que mantener un DAT: $1600 * 1200 * 1$ bit $= 0,24$ Mbytes

Abstracción: Nos da la posibilidad de poder trabajar con varios formatos de imágenes a la vez:

- BMP $>$ DAT
- JPG $>$ DAT
- ...

Deployment más económico:

- Podemos enviar la aplicación y todos los DAT por internet.
- Enviar todos los JPG es mucho más costoso.

Lectura y carga en memoria:

- El DAT es mucho más rápido de cargar que el JPG, lo cual optimizará el Algoritmo principal(paso2).

3.3.1. DAT vs JPG

- Espacio y accesos a Disco:
  - El DAT no tiene la compresión del JPG, sin embargo ocupa casi el mismo espacio.
  - Para el caso de un deployment, el DAT comprimido ocupa mucho menos que el JPG.

- Accesos "reales" a memoria:
  - Si se observa el DAT, podemos notar que la mayor parte de la estructura es blanca(o ceros en la matriz). Por lo tanto, *solo escribiremos los píxels negros en memoria.*
  - Dejando cargada una matriz esparza en memoria. (Ver Apéndice V: Carga del DAT en memoria)

- Tamaño en memoria:
  - DAT: $1600 * 1200 = 1{,}920{,}000$ bits Como hay 8 bits por Byte, son 240 Kbytes.
  - JPG:$1600*1200 = 1{,}920{,}000$ píxels Como cada píxel necesita 3 Bytes, son 5,760,000 KBytes

Capítulo 4

Experimentos

4.1. Estimación de la población celular

A continuación explicaremos una forma de estimar la población, basada en el conteo de células levantadas de cada foto.

Denominaremos P_i a la población celular estimada, en la foto i.

Definimos L_i como la cantidad de células diferentes levantadas detectadas hasta la foto i.

Por ejemplo si:

Nro. foto	# células levantadas en la foto i	# células diferentes levantadas hasta la foto i
0	10	10
10	20	13
20	25	17

Sabremos que la población es de al menos 17 células

Como L_0 y P_0 son conocidos, pues se calculan manualmente, planteamos la siguiente fórmula para estimar P_i:

$$P_i \leq P_0 + L_i$$

Por ejemplo:

- Comenzamos con 10 células
- A la hora, es decir en L_{11}, se contabilizaron 100 células distintas levantadas entre L_0 y L_{11}.

Entonces estimaremos que la población será menor o igual que 10 + 100 células.

A efectos prácticos estimaremos P_i de la siguiente manera:

$$P_i \approx P_0 + L_i$$

La población estimada en la foto i es: la suma de todas las células levantadas detectadas más la población inicial.

La demostración de esta formula se encuentra en el Apéndice 6.6.

4.2. Caso de uso

Algunos cultivos celulares, crecidos sobre sustratos planos como las cápsulas de Petri, se pueden observar bajo un microscopio invertido, permitiendo el seguimiento del comportamiento de los cultivos.

Las células crecen adheridas al fondo de las placas y su crecimiento, diferenciación, o muerte es promovido por varios factores y drogas de relevancia biológica.

En este trabajo son utilizadas células del tipo *BHK-21* a una concentración de 5x104 células/mL, monitoreadas con un microscopio invertido Leica Diavert.

Las células son iluminadas de forma perpendicular al plano de la base de la placa con una iluminación de contraste de fase, formando de esta manera una imagen bidimensional de todo el cultivo.

Las fotos (Por ejemplo Figura 4.1) fueron tomadas por una Canon *PowerShot A520* ubicada en el trilocular del microscopio. Como se explica en el apéndice 6.4 "Acerca del microscopio y la cámara", el aumento final obtenido fue de 400x.

Figura 4.1: Células con iluminación de contraste de fase

Características generales de las fotos:

- Resolución:1600x1200 píxels.
- Profundidad: 24 bits(16 millones de colores aprox.)

- Tamaño promedio en disco por foto: 140KBytes
- Formato: JPG
- Tamaño total del lote: 193MB

A efectos de mantener las estadísticas uniformes a lo largo de todo el trabajo, se trabajará con un lote de 1008 fotos tomadas a intervalos regulares durante 84 horas.

4.2.1. Elección del Umbral

Dado que la selección del Umbral se realiza respecto de su capacidad de detectar células, y que para tener el conteo exacto, el conteo debe hacerse manualmente:

No se puede automatizar la calibración del umbral mediante los métodos tradicionales[1] propuestos en [10]

4.2.1.1. Umbral - definición

Antes de analizar una foto f, transformaremos cada píxel r ($\forall r \epsilon f$) mediante la técnica *Contrast Stretching* (Ver Figura 4.2) descrita en [10]:

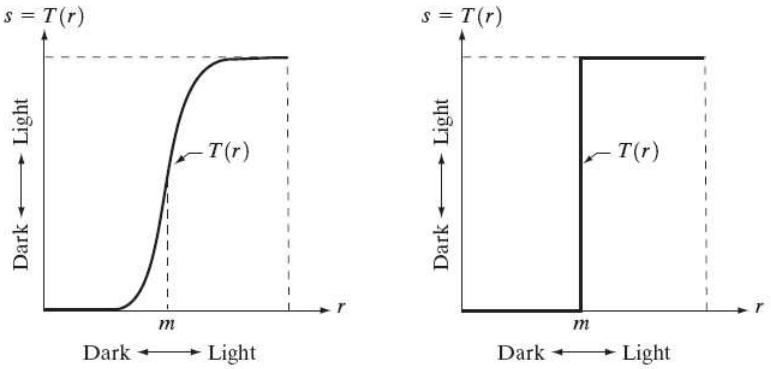


Figura 4.2: Función de transformación para el método *Contrast Stretching*

"Los valores por debajo de m son transformados en blanco.
Mientras que los valores por encima del umbral m, en negro.
$T(r)$ produce una imagen binaria.
Las funciones de mapeo $T()$ son llamadas funciones de *Thresholding*."

Cada píxel r se puede describir como una tripla: $< rojo, verde, azul >$.

[1] Por ejemplo: *Basic Adaptative Thresholding*

En el caso de un JPG, las tres variables tienen un rango de 0 a 255. De esa manera pueden representar hasta $256^3 = 16{,}777{,}216$ colores distintos dentro la misma foto.

Según el umbral seleccionado se tendrá una mejor abstracción de la foto y por lo tanto una mejor detección de objetos, a continuación se muestran algunos ejemplos de Umbrales(Ver Figura 4.3).

Parte izquierda de la foto BHK0011.JPG. Los objetos detectados son puntos rojos:

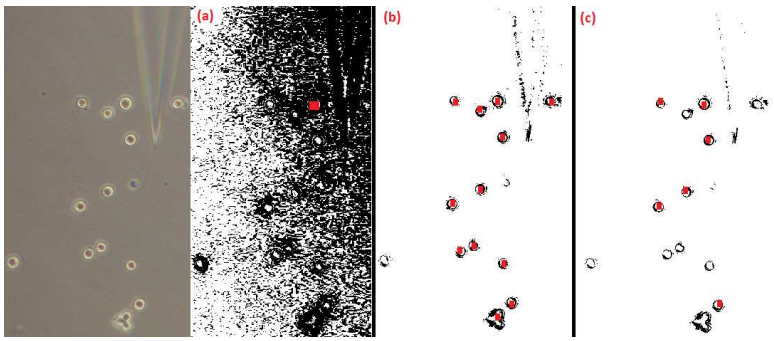


Figura 4.3: (a) Umbral(110, 110, 110) (b) Umbral(120, 120, 130) (c) Umbral(140, 140, 130)

Umbral	Cantidad de objetos detectados en toda la foto BHK0011.JPG
(a)	1
(b)	20
(c)	10

Dada la cantidad de objetos, pero más que nada la calidad de objetos (objetos detectados como células), nos quedamos con el umbral (b)

La función $T()$ particiona los píxels r en dos grupos (Ver Figura 4.4):

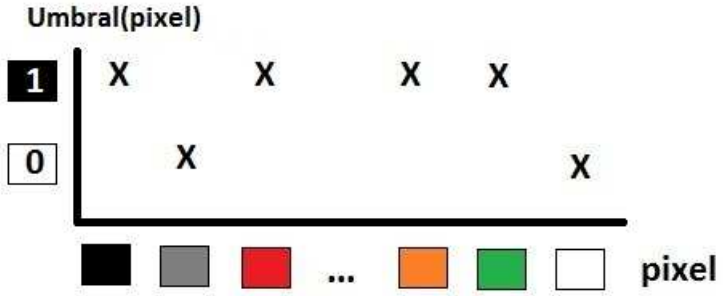


Figura 4.4: $T()$compara cada píxel con el umbral m para definir si "es un píxel blanco"(0) o si es un píxel negro"(1)

Más formalmente:
$T(r) \longrightarrow \{0, 1\}$

$$T(r) = \begin{cases} 1 & si \;\; r.rojo > 120 \;\; \wedge \;\; r.verde > 120 \;\; \wedge \;\; r.azul > 130 \\ 0 & sino \end{cases}$$

4.3. Estadísticas y resultados

Con los datos obtenidos se realizaron las siguientes estadísticas:

- Para cada foto:
  - La cantidad de objetos detectados
  - La cantidad de células nuevas detectadas
  - La población estimada hasta ese momento (L_i)
- Para cada célula:
  - El intervalo de fotos en que estuvo levantada
  - Su migración, mientras estuvo levantada
- En general:
  - El tiempo de levantamiento promedio
  - La población estimada (P_n).

Que se presentara de alguna de estas dos maneras:

- En el formato tradicional
  - En archivos de texto
- En lo que denominamos "JPG decorados"
  - Se generan nuevos JPG, agregando información adicional a los originales

Los JPG decorados se generan tomando como entrada:
• Los mismos JPG que se utilizan para generar los DAT (línea punteada en el gráfico).
• Los datos generados luego del análisis o de leer Data.txt

Las células que se detectan por primera vez se marcan con un punto rojo. Las células activas, que ya fueron detectadas en una foto previa, se marcaron con un punto negro. Se marcaron en celeste, algunas células inactivas.
Ejemplos de un JPG decorado (Figura 4.5 y 4.6):

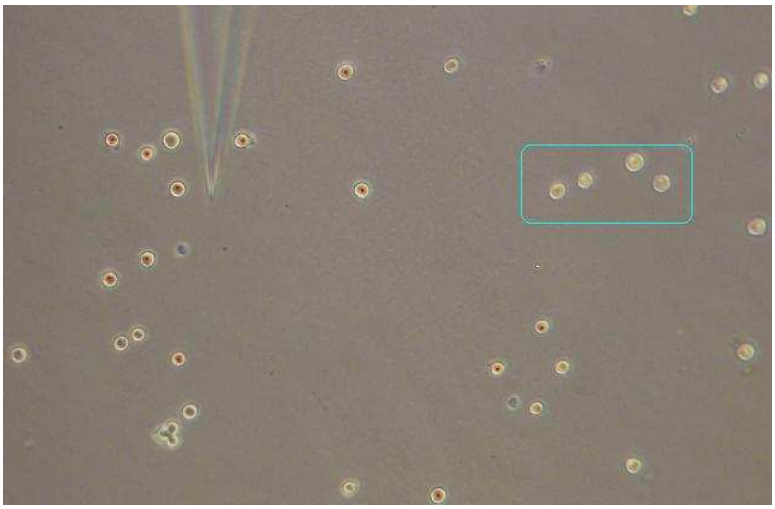


Figura 4.5: $Foto_i$

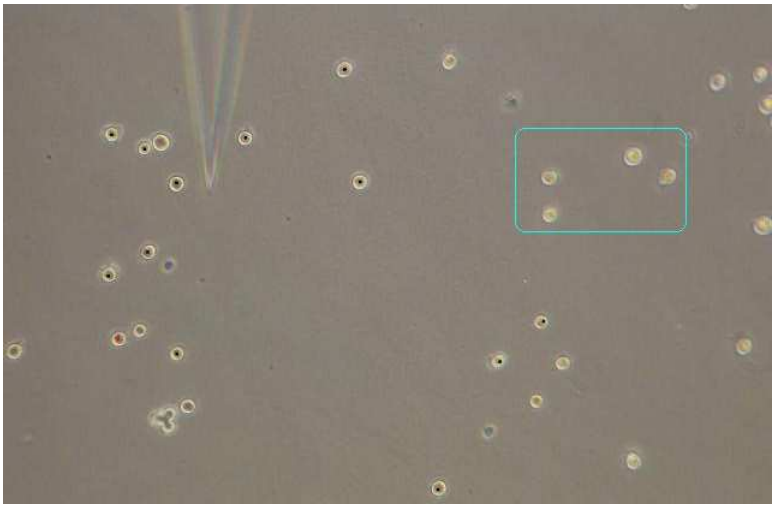


Figura 4.6: $Foto_{i+1}$

- Todas las células activas de la Figura 4.5 son rojas, porque *son todas nuevas*.
- Las células que parecen levantas, pero que no están marcadas en ninguna foto, son en realidad células que se bajaron sin dividirse, o que se desplazaron demasiado entre dos fotos consecutivas.
    - Se consideran células inactivas.

Estadística en formato tradicional (Ver Figura 4.7 y 4.8):

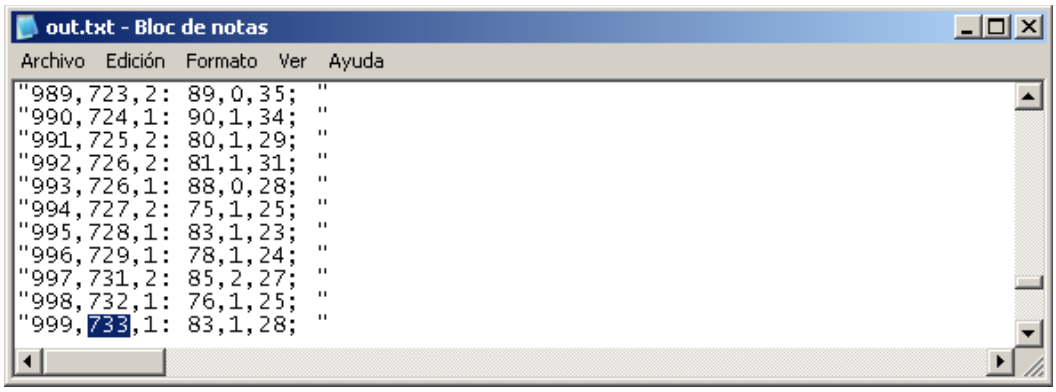


Figura 4.7: La $foto_{999}$ tiene 1 célula nueva y las células activas hasta el momento son 733

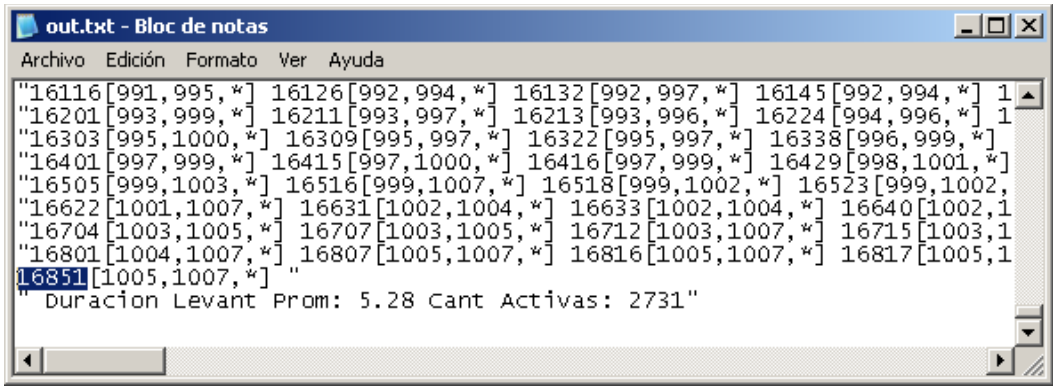


Figura 4.8: Duración del "levantamiento promedio" celular: 5.28 fotos consecutivas

Interpretación de *out.txt*:

- Se detectaron 16851 objetos, de los cuales 2731 son activos, y 741 se consideran células levantadas.
- Por ejemplo: La célula 16801 estuvo levantada desde la $foto_{1004}$ hasta la $foto_{1007}$.
- La duración de cada levantamiento, fue en promedio de 5,28 fotos. o

- Adicionamos media foto al promedio (0,5 fotos = 2,5 minutos), para compensar la precisión que se pierde al discretizar en intervalos regulares de 5 minutos.
- Entonces, *el levantamiento promedio aproximado medido fue de:* $(5,28+0,50)*5\,' = 28,90\,'$

Tiempo utilizado como hipótesis biológica	Tiempo promedio aproximado obtenido	Diferencia (%)
30'	28,90'	3,66

4.3.1. Resultados

Para calcular el error, se compararon los resultados obtenidos del algoritmo, respecto del conteo manual en las fotos.

Se consideró el método de "falsos negativos" y "falsos positivos"[6] para calcular el porcentaje de error obtenido:

- H_0: El objeto detectado es una célula activa
- H_1: El objeto detectado no es una célula activa

	H_0es cierta	H_1es cierta
Se escogió H_0	Verdadero positivo	*Error de tipo II (falso negativo)*
Se escogió H_1	*Error de Tipo I (falso positivo)*	Verdadero negativo

Por lo tanto el Error Absoluto(Err Abs) por foto es:

- Err Abs = Error de Tipo I + Error de Tipo II

Y el Error Relativo(Err Rel):

- Err Rel = Err Abs / Cantidad total de objetos en la foto

Como el error varia poco entre dos fotos consecutivas, los resultados serán expresados como promedios para cada intervalo de fotos(i..j) significativo:

(Intervalo)(i..j)	[Err Tipo I]	[Err Tipo II]	Err Abs	# total de obj.	Err Rel (%)
Int_1:(1...90)	0	1	1	25	4.00
Int_2:(91...113)	0	1	1	17	5.88
Int_3:(114...177)	1	0	1	8	12.50
Int_4:(178...195)	3	0	3	13	23.07
Int_5:(196...252)	3	0	3	20	15.00
Int_6:(253...504)	1	0	1	16	6.25
Int_7:(505...655)	2	0	2	20	10.00
Int_8:(656...736)	5	0	5	30	16.66
	Limite	de	error	aceptado	
Int_9:(737...820)	10	0	10	40	25.00
Int_{10}:(821...920)	14	0	14	60	23.33
Int_{11}:(921...1008)	25	0	25	90	27.77

Explicación de los intervalos más significativos:
Int_3: El porcentaje se debe más a la poca cantidad de objetos(solo hay 8 en promedio) que a errores inherentes del algoritmo.

Int_4e Int_5: Los errores se deben principalmente a problemas con la imagen (antes de la digitalización).

Parte de la foto BHK1911 ($foto_{191}$): En la parte izquierda hay un error (con forma de V). Y la parte derecha está fuera de foco (Ver Figura 4.9).

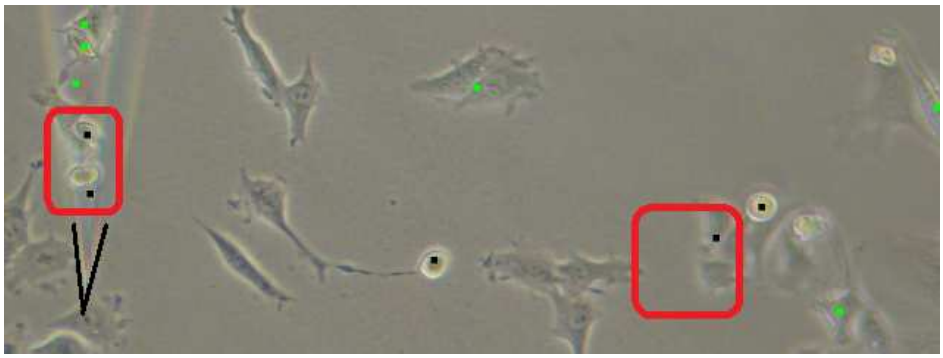


Figura 4.9: Errores en la imágenes

Si se considera el 17 % como el máximo de tolerancia permitido y sin considerar los errores de las imagenes, se puede utilizar la estimación de conteo por software hasta la $foto_{736}$ con un margen de error $< 17\,\%$.

A partir de la $foto_{737}$, la densidad de población (420 células activas aprox.) hace más difícil el conteo (tanto el manual como para nuestro algoritmo).

4.3.2. Gráficos

Población celular estimada, basada en el conteo celular (Ver Figura 4.10):

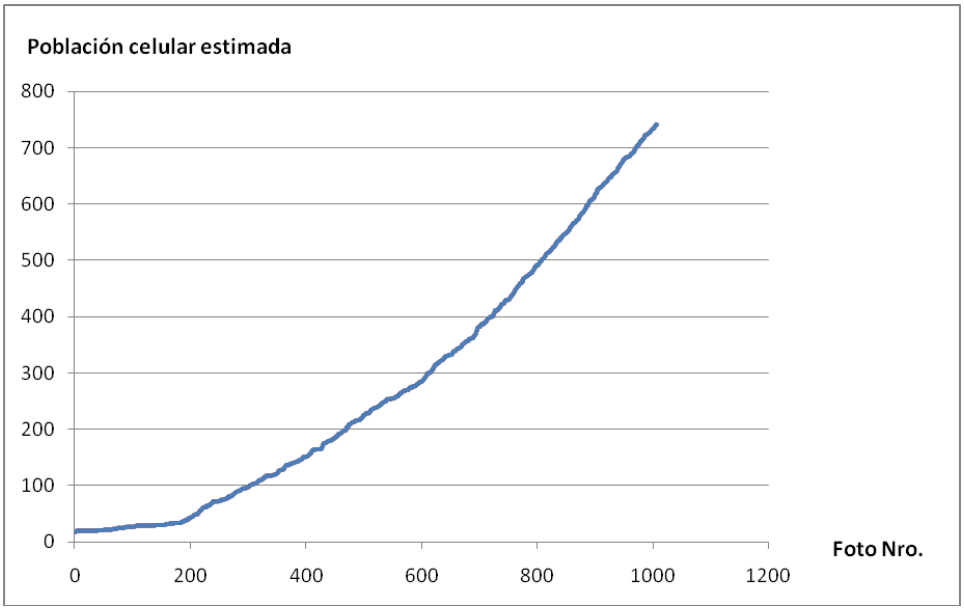


Figura 4.10: $Foto_0$: Población inicial estimada: 16. $Foto_{1008}$: Población final estimada: 741

Tiempo medio de procesamiento (Ver Figura 4.11):

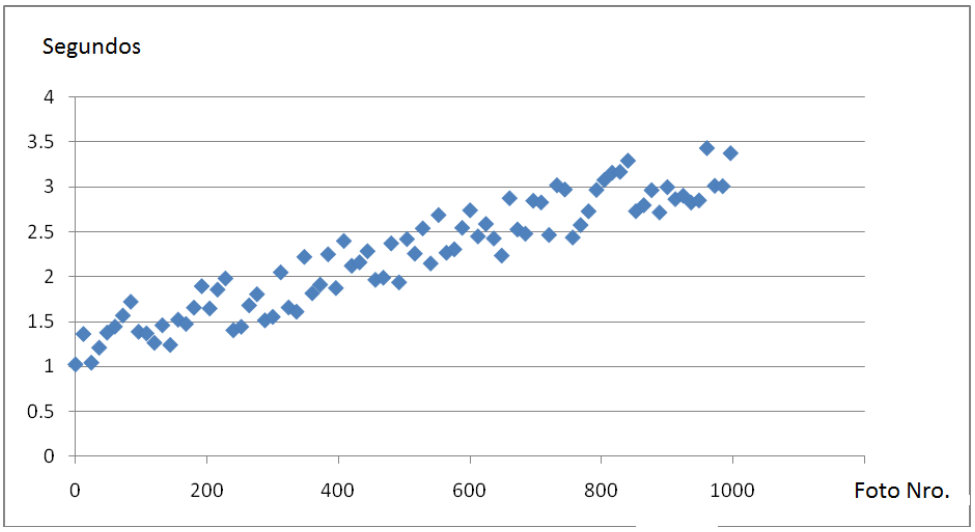


Figura 4.11: Las primeras fotos se procesaron en un 1'', mientras que las últimas en 3''

El tiempo de procesamiento promedio general por foto fue de 1,5''.

	Tiempo promedio utilizado por foto	Carga	Procesamiento	Limpiar células inactivas
Primeras fotos	1"	1 %	98 %	1 %
Últimas fotos	3"	1 %	87 %	12 %

En las últimas fotos se necesitó más tiempo para "Limpiar las células inactivas".

4.3.3. Comparación respecto de otros proyectos Open Source

Dadas las dificultades inherentes de una comparación "punto por punto", se decidió separarla en dos tipos:

- Por los resultados obtenidos.
- Por el algoritmo utilizado.

La comparación respecto de los resultados obtenidos se realizará respecto del proyecto CellTrack (citado en la introducción).

La principal diferencia con nuestro trabajo, es que CellTrack no identifica el momento en que tiene comienzo la citocinesis en si.

Sin embargo, se puede establecer una comparación, si se asume que la aparición de un nuevo segmento (una nueva célula) es el resultado final del proceso de citocinesis.

Para el análisis, las imágenes procesadas por CellTrack estarán en la parte derecha, y las nuestras del lado izquierdo.

Como resultado general, podríamos decir que cualquier célula detectada por nuestro algoritmo, también fue detectada por CellTrack (Ver Figura 4.12).

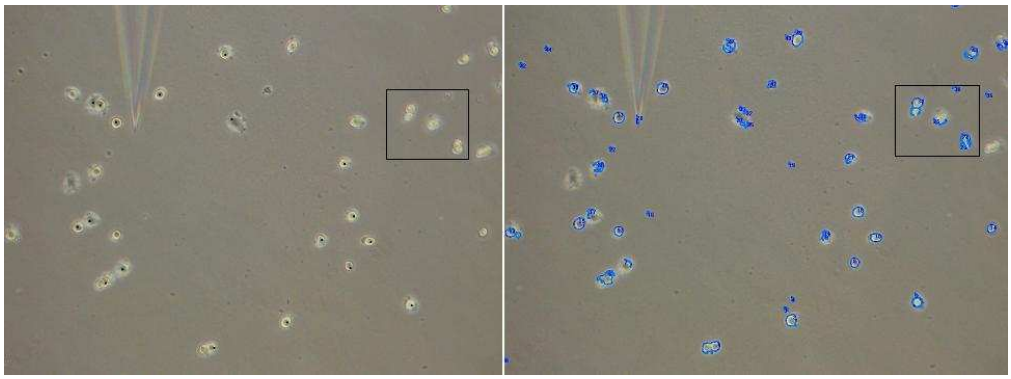


Figura 4.12: Detección de las células en la fotoBHK0131.JPG por CellTrack

Respecto a los segmentos detectados por CellTrack: Nuestro software también detectó los segmentos 23, 24, 26 y 34. Sin embargo estos no están marcados, por no ser parte de un proceso de mitosis (Ver Figura4.13):

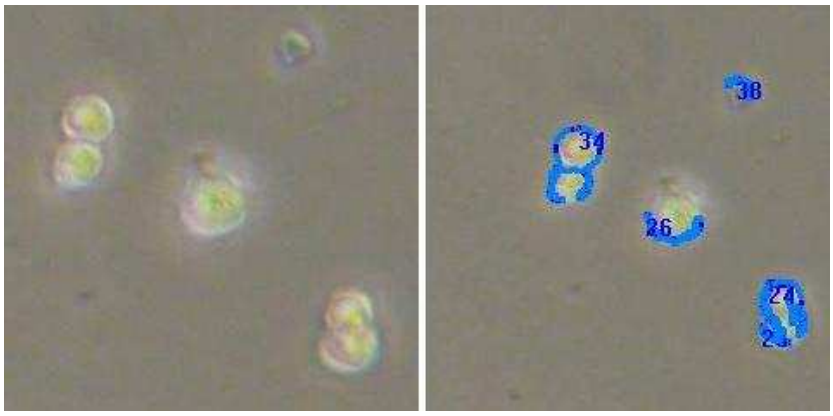


Figura 4.13: Detección de CellTrack en la misma foto que en la Figura 4.12

Respecto del proceso de citocinesis:

En general, CellTrack detectó también las mitosis, pero no en todos los casos (Ver Figura 4.14):

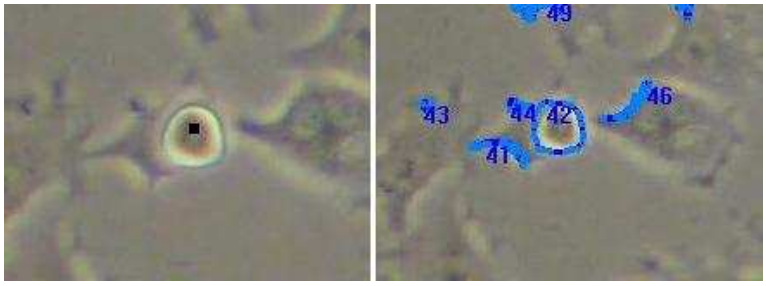


Figura 4.14: Mitosis detectada por CellTrack en la foto BHK1971.JPG

El segmento 42 es detectado en ambos lados.

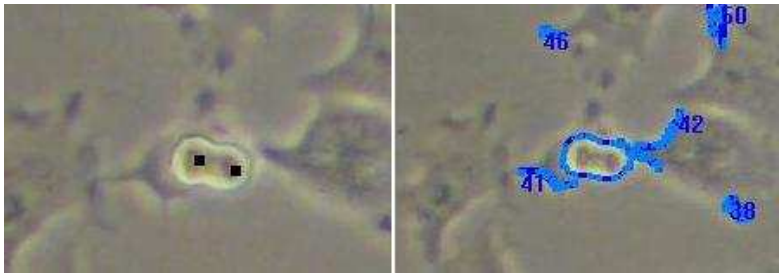


Figura 4.15: Ausencia de detección de CellTrack en la foto BHK1981.JPG

Sin embargo, CellTrack no puede identificar la nueva célula en BHK1981.JPG (Ver Figura 4.15)

Como un tercer aspecto a tener en cuenta, podemos agregar que CellTrack tampoco utiliza la misma optimización de memoria que utilizamos nosotros, al descargar la imagen de memoria luego de utilizarla (ver Algoritmo 3.1)

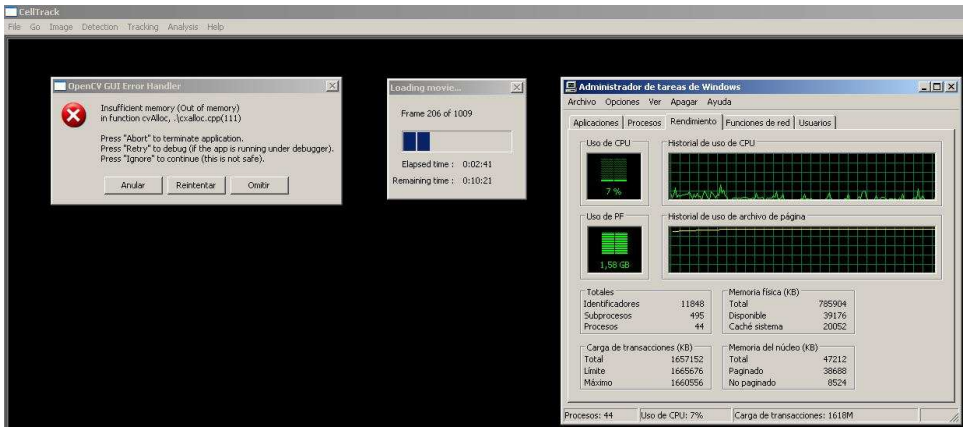


Figura 4.16: Pantalla de CellTrack en la que se observa la imposibilidad de procesar mas de 206 fotos

CellTrack no pudo procesar más de 206 fotos al mismo tiempo (Ver Figura 4.16), en la misma plataforma donde se testeó nuestro software.

Comparación respecto del algoritmo utilizado para la etapa de asociación: Proyectos open-source seleccionados:

1. "Cell tracking in video Microscopy using Bipartite Graph Matching" [7].
2. "Tracking cells in Life Cell Imaging videos using topological alignments" [5].
3. "Topaln" (http://www.picb.ac.cn/patterns/Software/topaln/), es una implementación de(2) .

La mayor similitud de nuestro proyecto, respecto del proyecto 1. y el proyecto 2., es que:

- Todos consideran la información de dos frames consecutivos a la hora de actualizar el diccionario de células.

Algunos proyectos consideran cada par de frames consecutivos(por ejemplof_1 y f_2) como un grafo bipartito(f), donde:

- Las células de cada frame son los vértices(v_i y v_j) de cada subgrafo(f_1 y f_2), tal que $v_i \in f_1$ y $v_j \in f_2$.
- El peso de las aristas que unen v_i con v_j es la distancia en R^2 tal como se describe en la Figura 1.1

El objetivo es encontrar un matching máximo(en general perfecto), respecto de una función $g(f)$.

Lo que diferencia un algoritmo de otro, es la función $g(x)$.
A continuación se describen algunos ejemplos de $g(x)$:

- Para nuestro trabajo, $g(x)$ obtiene la asociación mínima. Para nuestro ejemplo de la sección 1.3 es la asociación a.
- Para CellTrack, $g(x)$ es un algoritmo de programación lineal donde el matching máximo es la solución del sistema.
- Y en otros trabajos investigados, $g(x)$ representa el algoritmo de cuadrados mínimos, siempre considerando la distancia en R^2 entre cada vértice de v_i respecto de todos los vértices de v_j.

La mayor diferencia que encontramos fue la siguiente:

- *Nuestros algoritmos utilizan la información de más de dos frames consecutivos.*

Capítulo 5

Conclusiones

La solución presentada en este trabajo logró censar con éxito las células BHK-21 hasta cierta densidad de población, manteniendo un error por debajo del 17 %.

Con la información recopilada durante el análisis, también es posible censar otro tipo de información biologicamente relevante:

- La trayectoria de cada célula(Cell Traking).
- La velocidad de desplazamiento.
- Estas variables son importantes, por ejemplo, para la caracterización de los cultivos celulares en general.

5.1. Aportes realizados

Los aportes realizados por esta tesis son:

- una metodología para estimar la población celular

$$\underline{P_i = P_0 + L_i}$$

- Un nuevo algoritmo de detección de células levantadas

PATRÓN ASTERISCO "*"

- Se realizaron estadísticas de Población celular estimada Promedio del tiempo en que una célula permanece levantada

5.2. Trabajos a futuro

Dadas las características del algoritmo de reconocimiento creemos que podría censar con éxito muchas otras de las familias celulares utilizadas habitualmente en los laboratorios.

Como todos los algoritmos utilizados son de un orden de ejecución polinomial, podemos recomendar este método para censar lotes mucho mas grandes que el lote de 1008 fotos utilizado para los experimentos.

5.2.1. Automatización del ThresHolding

Si bien no se puede automatizar en forma general, es posible centrarse en alguna foto en particular y obtener el mejor umbral que maximize la detección.

5.2.2. Optimización de "Limpiar las células inactivas"

El objetivo es optimizar la búsqueda de células en nuestras estructuras de datos, de manera que el chequeo de células vecinas sea mas eficiente.

Utilizando la cercanía de dichas células se podría realizar las búsquedas en un hash.

Definimos la función de hash:

Sea f la función de hash que depende de las coordenadas "x" y de "y" de la célula en la foto. Entonces:

$f(x, y) = [x/40] * [y/40]$

En este caso estamos utilizando la división entera.

Notar que En nuestro caso, como $1{,}600 * 1{,}200 = 1{,}920{,}000$ tal que $0 < f(x, y) < 1{,}200$

El hash tendrá 1,200 casilleros($n = 1{,}200$), a los que se acceden en O(1).

Veamos algunos ejemplos:

Células(Coordenada)	Posición en el hash
$c_1(300, 400)$	75
$c_2(510, 700)$	223
$c_3(900, 100)$	56
$c_4(700, 510)$	223

Notar que c_2 y c_4 tienen el mismo lugar en el hash($f(c_2) = f(c_4)$).

C_2y c_4 son "vecinas".

Luego, reemplazamos la línea número 2 de "LimpiarInactivas - continuación":

- Para cada celu j_2 en la foto i_2

por:

- Para cada cada celu j_2 en la foto i_2, considerando solo aquellas que: Estén a lo sumo a un casillero del hash del $celu_j$

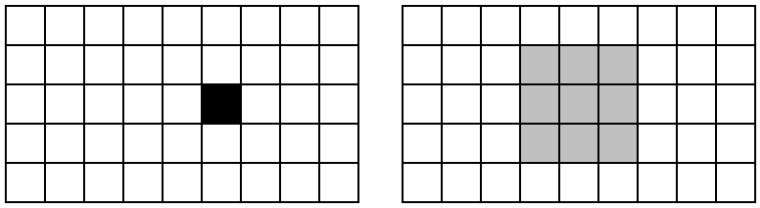


Figura 5.1: Hash de celu j

Se buscaran entre las células celu j_2, que estén en un Hash cercano a $celu_j$ (Parte derecha de Fig 5.1).

Notar que si dos células tienen hash parecido, también están a una distancia cercana en R^2!.

Como las células están *uniformemente distribuidas*, no quedaran casilleros muy poblados, por lo tanto O(FC2) pasara a ser considerado como O(1).

El costo total con esta mejora es:
$O(F * Log(C) * FCa * FC/n) \approx O(F * Log(C) * FCa)$

Capítulo 6

Apéndices

6.1. Micromanipulación fisicoquímica sobre arreglos celulares

Micromanipulación fisicoquímica sobre arreglos celulares y célula única bajo monitoreo constante

Resumen:

En este trabajo, se desarrolló un sistema de monitoreo de cultivos celulares para tiempos prolongados, totalmente autónomo. Sobre este, se adoptaron distintas estrategias para la modificación del microambiente celular mediante actuadores químicos y físicos.

Se desarrolló una técnica para generar un microgradiente molecular controlado en espacio y tiempo de manera de trabajar sobre un sector del cultivo o sobre una única célula. Por otro lado, se sintetizaron y caracterizaron compuestos de coordinación de rutenio, para la foto liberación de ligandos bioactivos aptos para la utilización como compuestos enjaulados en cultivos celulares.

Se sintonizaron las propiedades químicas y fotoquímicas de los complejos mediante el uso de diversos ligandos de manera que los complejos sean no tóxicos para los cultivos celulares y que mantengan la propiedad de fotoliberar la molécula bioactiva mediante le excitación con fotones dentro del rango del visible.

Dentro de este objetivo, se logró el primer enjaulado de glutamato que libera en el rango del visible y el primer liberador de nicotina.

En síntesis, se pudieron generar microgradientes controlados para actuar sobre arreglos celulares y sobre una única célula con monitoreo constante y

autónomo. Se pudieron caracterizar los cambios dinámicos y morfológicos de los cultivos pudiendo abrir una puerta al análisis de los cambios en la dinámica y la conformación del citoesqueleto celular en tiempo real.

6.2. Modo de uso de la aplicación

Instalación

Para ejecutar el programa principal (Analizador.exe), solo es necesario tener cuatro archivos (Ver Figura 6.1):

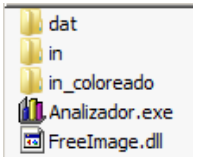


Figura 6.1: Deployment

Archivos y carpetas:

- Archivo "FreeImage.dll": Es el soporte para el procesamiento de imágenes(JPG).
- Carpeta "in": ahí deben estar los archivos JPG originales.
- Carpeta "dat": Analizador.exe dejara ahí los archivos dat
- Carpeta "in_coloreado": Analizador.exe dejara ahí las fotos procesadas.

Esta disponible además, un video en youtube (Video), con la secuencia original de fotos procesada.

Utilización:

Lo primero que hay que hacer, es generar por única vez los archivos dat: Seleccionar el $paso_1$

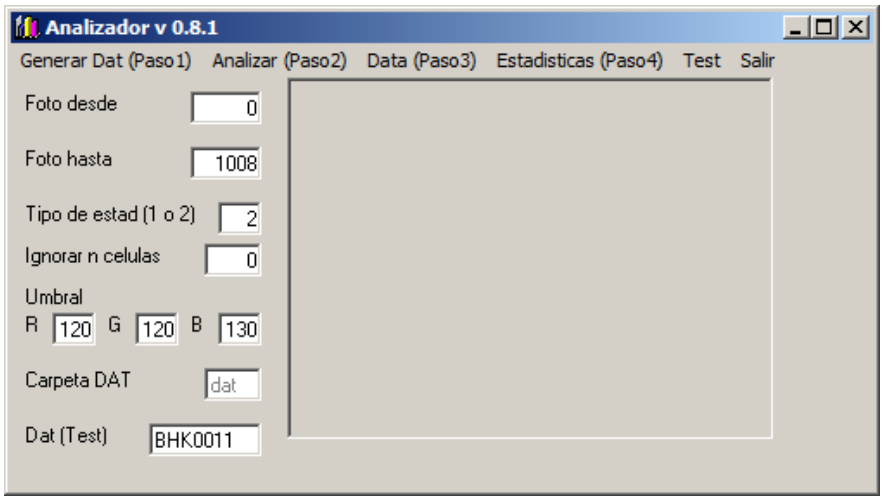


Figura 6.2: En ejecución

Los parámetros de ejecución, -en especial el Umbral-, se dejan editables, de manera que puedan ser adaptables a cualquier lote de fotos.

Si se modifica el Umbral o el lote de fotos, hay que volver a ejecutar el paso1 (Ver Figura 6.2).

De ahí en adelante, existen varias alternativas (Ver 3.1) para generar las estadísticas (paso4).

A continuación se enumeran algunas de las secuencias de uso:
La primera vez:
$paso_1 \rightarrow paso_2 \rightarrow paso_{3a}$
O bien:
$paso_1 \rightarrow paso_2 \rightarrow paso_{4txt}$

Si se ejecutó$paso_{3a}$, de ahí en adelante se puede ejecutar la secuencia:

$$paso_{3b} \rightarrow paso_{4JPG-decorado}$$

Secuencia	Tiempo de ejecución aprox. en minutos
$Paso_1$	10
$Paso_2$	23
$Paso_{3a}$	3
$Paso_{3b}$	3
$Paso_{4txt}$	1
$Paso_{4JPG-decorado}$	10
$paso_1 \rightarrow paso_2 \rightarrow paso_{3a}$	10 + 23 + 3 = 36
$paso_1 \rightarrow paso_2 \rightarrow paso_{4txt}$	10 + 23 + 1 = 34
$paso_{3b} \rightarrow paso_{4txt}$	3 + 1 = 4
$paso_{3b} \rightarrow paso_{4JPG-decorado}$	3 + 10 = 13

Se pueden obtener resultados inmediatos en 34 minutos.
Pero si se pasa por$paso_{3a}$ se pueden obtener resultados en tan solo 4 minutos.

6.3. Carga de DAT en memoria – optimización

Formato:
Cada foto tiene una matriz de booleanos de 1,600∗1,200, que suman 1,920,000 bits en total. Los archivos binarios, son, en esencia, una secuencia de Bytes. Cada matriz necesitara, entonces, un archivo de 240,000 Bytes para ser almacenada (Ver Figura 6.3y 6.4).

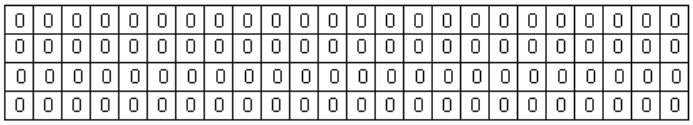


Figura 6.3: Foto vacía, para un ejemplo de $24 * 4$

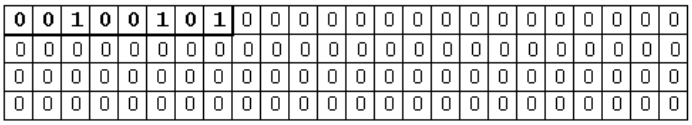


Figura 6.4: Foto luego de cargar el 1er Byte

Recpecto de LoadDat():

- Se utilizara "/" para denotar la división entera. Ej: $11/8 = 1$
- Se utilizara una clase BynaryByte para poder transformar un Byte en formato decimal, al formato binario.

- bytes() es un vector que contendrá los 240,000 Bytes(en formato decimal), de cada archivo.

Algoritmo 6.1 Cargar archivoDat en bytes(): Byte

```
indice = 0       // nro de bit
Sea j: 0 hasta 1599
    Sea k: 0 hasta 1199
        Si ( indice modulo 8 ) = 0
            Si bytes(indice / 8) = 0     //division entera
                algunUno = falso
            Sino
                algunUno = verdadero
                binaryByte.SetDecimalValue( bytes(indice / 8) )
            Fin
        Fin
        Si algunUno
            Si binaryByte.GetBinaryValue( índice modulo 8 )
                foto(j,k) = verdadero
            Fin
        Fin
        Índice ++
    Fin
Fin
```

Optimización:

Como casi toda la matriz tiene ceros, pocas veces ocurre que "algunUno == verdadero".

Como consecuencia, se accede pocas veces a las estructuras:

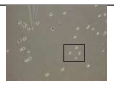
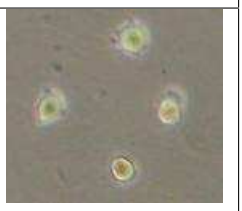
- binaryByte
- foto

6.4. Acerca del microscopio y la cámara

Se utilizó un microscopio invertido Leica Diavert en modo de contraste de fase (http://es.wikipedia.org/wiki/Microscopio_de_contraste_de_fases), con un objetivo de *20x*.

El trilocular del microscopio tiene un aumento de *10x*. Las fotos fueron tomadas por una Canon PowerShot A520, con aumento de *2x*, ubicada en el trilocular. Por lo tanto, el aumento final acumulado es de *400x*[1].

[1] El aumento final es aproximadamente 400x, considerando la resolución de las fotos y el tamaño del frame de la cámara.

Objeto	Zoom	Zoom acumulado	Imagen relativa
Caja de Petri	1x	1x	
Objetivo	20x	20x	
Trilocular	10x	200x	
Cámara	2x	400x	

6.5. Orden asintótico de los algoritmos

6.5.1. Variables

- La cantidad total de células es C.
- La cantidad total de fotos es F.
- La cantidad total de células en una foto es FC.
  - Las células activas son FCa
- El tamaño de la foto es T.

6.5.2. Algoritmo principal

Algoritmo 6.2 Procesar cada foto i

```
1 Para cada píxel(x,y) sin examinar, que no tenga objetos detectados
2   Si se encuentra el patrón del asterisco en (x,y)
        Llamaremos celu j a esa detección
3       Si no es una decteccion redundante en la foto i
            // AgregarFotoCel
4           Actualizar = (Agregar celu j en (foto i).celus)
            Si Actualizar
5               Agregar celu j en las Celus generales
            Fin
6           Marcar el objeto detectado(x,y) en la foto i
        Fin
    Fin
Fin
```

1. $O(T)$

2. Como todos los vectores tienen $Norma_2 = 2$, buscar el patrón esta en $O(1)$.

3. $O(FC)$

4. $O(AgregarFotoCel) = O(FC)$

5. $O(Log(C))$

El costo total de procesar todas las fotos es:
$O(F * T * (FC + Log(C)))$

AgregarFotoCel retorna un ID_Cel y un booleano, indicando que hay que actualizar el diccionario de células principal.

Algoritmo 6.3 AgregarFotoCel(c: Coordenada, f: Foto)

```
R1  Si CelCercana(c)                //en la foto actual
1.1     ID_Cel = -1
1.2     Actualizar = falso
R2  Si CelCercanaAnt(c, ID_Cel)   //Obtiene el ID_Cel, si existía, en la foto anterior
2.1     Si ID_Cel no estaba en f
2.2         Crear una nueva célula c1
2.3         c1.ID_Cel = ID_Cel
2.4         c1.c = c
2.5         c1.Activa = verdadero
2.6         Agregar c1 en f
        Fin
        Actualizar = Verdadero
R3  Sino
3.1     Crear una nueva célula c1
3.3     ID_Cel = (Obtener nuevo ID_Cel)
3.3     C1.ID_Cel = ID_Cel
3.4     c1.c = c
3.5     c1.Activa = verdadero
3.6     c1.Nueva = verdadero
3.7     Agregar c1 en f
3.8     Actualizar = Verdadero
    Fin
```

El orden será el máximo orden entre las ramas R1,R2 y R3.

Rama1
1.1 $O(FC)$
El resto es $O(1)$

Rama2
2.2 $O(FC)$ 2.1 y 2.6 $O(Log(FC))$
El resto es $O(1)$
$O(FC + Log(FC)) \Rightarrow O(FC)$

Rama3
3.7 $O(Log(FC))$

El resto es $O(1)$

AgregarFotoCel esta en $O(FC)$

6.5.3. Limpiar las células inactivas

Como la 2da parte de "Limpiar las células inactivas" consume mas tiempo que la primer parte, consideraremos solo la segunda parte para el calculo de Orden.

Algoritmo 6.4 Limpiar las células inactivas

```
1 Para cada foto i
2       Para cada celu j en la foto i
3               Si Distancia(celu j.ID_FotoHasta, celu j.ID_FotoDesde) < 2
3.1                 celu j.Nueva = falso
3.2                 celu j. Activa = falso //Diccionario ppal
                    Celus(celu j.ID_Cel).
3.3                 Activa = falso
4               Sino
                    //Observamos en las 4 fotos anteriores
4.1                 Si i > 5 y (celu j.Nueva)
4.2                     LimpiarInactivasParte2
                    Fin
                Fin
        Fin
Fin
```

1 $O(F)$.
2 $O(FC)$.
3 3, 3.1 y 3.2 estan todos en $O(1)$.
3.3 está en $O(Log(C))$
4 4 y 4.1 están en $O(1)$.
4.2 está en $O(FC * Log(C))$.

Por lo tanto me quedo con la parte del ELSE(rama 4), por ser la parte más costosa. 3 y 4 sumados están en $O(FC * Log(C))$.

El costo total es:
$O(F * FC_1) * O(LimpiarInactivasParte2)=$
$O(F * FC_1) * O(FC_2 * Log(C))=$
$O(F * Log(C) * FC_1 * FC_2)$

Considerando que solo se consideran las FC_1 activas:
$O(F * Log(C) * FCa * FC)$

Algoritmo 6.5 Limpiar células inactivas - Continuación

```
1 Para cada foto i2 tal que   Restar(i, 6) < i2 < i
2   Para cada celu j2 en la foto i2
3       Si Not(celu j.ID_Cel == celu j2.ID_Cel)
4           Si DistanciaR2(celu j, celu j2) < SeparacionMIN
                //Notar que queda activa
5               celu j.Nueva = falso
                //Diccionario ppal
6               Celus(celu j.ID_Cel). Activa = falso
            Fin
        Fin
    Fin
Fin
```

1 Esto se realiza 5 veces. Entonces es $O(1)$.
2 El paso 2 esta en $O(FC)$.
3, 4 y 5 se realizan todos en $O(1)$.
6 Acceder a un diccionario cuesta $O(Log(C))$.
El costo total es de $O(FC * Log(C))$

6.6. Validez de la formula $P_i = P_0 + L_i$ por inducción en i

Queremos probar que la poblacion en la foto i(P_i) es la poblacion en la foto 0 mas la cantidad de levantamientos independientes detectados hasta la foto i(L_i).

Como no estamos considerando:

- Las células que mueren. [Obs1]
- Los levantamientos que no se dividen, que son los más raros. [Obs2]

En la práctica $P_i \leq P_0 + L_i$en lugar de $P_i = P_0 + L_i$.[Obs3]

$$[\text{HI}] = P_i = P_0 + L_i$$

Caso base $i = 0$:

QPQ: $P_0 = P_0 + L_0$
Asumiremos que

- Los levantamientos iniciales son parte de la población inicial.
- Y por lo tanto $L_0 = 0$

Entonces, vale que $P_0 = P_0 + 0$

En el momento L_{i+1} se detectan k nuevas células que en el momento L_i no estaban levantadas

$$L_{i+1} = L_i + k\text{:[Prop1]}$$

Cada levantamiento produce dos nuevas células. Pero también tengo que dejar de contar las que dejan de existir(B3).

En k levantamientos, tendremos que:

- Sumar 2k a la población
- Restar k a la población

$$P_{i+1} = P_i + 2k – k \Leftrightarrow P_{i+1} = P(i) + k\text{:[Prop2]}$$

Caso inductivo, $i \Rightarrow i+1$**:**

QPQ: $P_i = P_0 + L_i \Rightarrow P_{i+1} = P_0 + L_{i+1}$

$$P_{i+1} = P_0 + L_{i+1}$$

x[Prop1]

$$\Leftrightarrow P_{i+1} = P_0 + L_i + k$$

x[HI]

$$\Leftrightarrow P_{i+1} = P_0 + P_i - P_0 + k$$

x[Prop2]

$$\Leftrightarrow P_{i+1} = P_i + k$$

Es decir, por Prop1, Prop2 y P_i, vale P_{i+1}

Bibliografía

[1] E. Meijering, O. Dzyubachyk, I. Smal, W.A. and Cappellen. Tracking in cell and developmental biology. Seminars in Cell & Developmental Biology 20(8), 2009, 894-902. doi:10.1016/j.semcdb.2009.07.004.

[2] M.J. Salierno. Micromanipulación físico-química sobre arreglos celulares y célula única bajo monitoreo constante. Tesis Doctoral, Universidad de Buenos Aires, 2009

[3] J.R. Swedlow and K.W. Eliceiri. Open source bioimage informatics for cell biology. Trends in Cell Biology 19(11), 2009, 656-660. doi:10.1016/j.tcb.2009.08.007

[4] A.E. Carpenter, T.R. Jones, M.R. Lamprecht, C. Clarke, I.H. Kang, O. Friman, D.A. Guertin, J.H. Chang, R.A. Lindquist, J. Moffat, P. Golland and D.M. Sabatini. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biology 7(10), 2006, R100. doi:10.1186/gb-2006-7-10-r100

[5] A. Mosig, S. Jäger, C. Wang, S. Nath, I. Ersoy, K. Palaniappan and S-S. Chen. Tracking cells in Life Cell Imaging videos using topological alignments. Algorithms for Molecular Biology 4, 2009, 10. doi:10.1186/1748-7188-4-10

[6] Jay Devore. Probabilidad y estadística para Ingeniería y ciencias. Cengage Learning, 2008. ISBN-13: 978-970-686-831-2 ISBN-10: 970-686-831-3.

[7] A.S. Chowdhury, R. Chatterjee, M. Ghosh and N. Ray. Cell Tracking in Video Microscopy using Bipartite Graph Matching. Proceedings of the 2010 20th International Conference on Pattern Recognition, Istambul, 2010, 2456-2459. doi:10.1109/ICPR.2010.601

[8] Thomas H. Cormen, Leiserson, E. Charles, Rivest, L. Ronald. Introduction to Algorithms 1Ed. MIT Press and McGraw-Hill, 1990. ISBN 0-262-03141-8.

[9] H. Lodish et al. Biología celular y molecular, Editorial Médica Panamericana, Buenos Aires, 2005. ISBN 950-06-1974-3.

[10] R.C. González, R.E. Woods. Digital Image Processing 2Ed. Prentice Hall, 2002, ISBN 0-201-18075-8.