Proyecto
Coloreando imágenes del imperio ruso

Antonio José Moya Díaz

9 de junio de 2013

Laboratorio Multimedia

Ingeniería en Telecomunicación

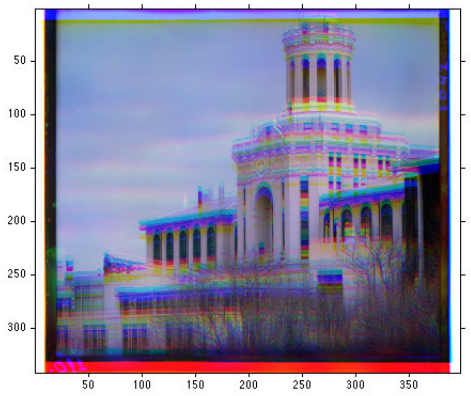
Índice general

1 | Descripción de la tarea

Nos han sido suministradas una serie de imágenes en escala de grises que contienen los 3 canales de color alineados en una columna como se muestra en la figura 1.1a.

Los canales de color se muestran ordenados, de arriba hacia abajo, Azul, Verde y Rojo. La tarea consiste en segmentar la imagen y superponer los distintos canales con el fin de obtener una imagen en color. Sin embargo, dichos canales no se encuentran alineados, como se puede ver en la figura 1.1b, siendo éste nuestro objetivo principal; obtener una alineación correcta de los canales de color obteniendo una imagen sin artificios visuales..

(a) Imagen original



(b) Canales superpuestos

Figura 1.1: Imágenes de la tarea

Hay que tener en cuenta que algunas de las imágenes suministradas están en alta resolución (más de 9 Megapíxeles) por lo que un proceso de alineamiento (registrado) convencional podría derivar en unos altísimos tiempos de ejecución así como consumir una cantidad de memoria muy superior a las disponibles en computadores personales. Por tanto habrá que mostrar especial atención e interés en que el algoritmo desarrollado sea

eficiente, consiguiendo llevar a cabo la tarea con el menor consumo de recursos posible y en el menor tiempo posible.

Adicionalmente se propone, una vez conseguido el registro de las imágenes, aplicar cualquier clase de mejora visual a la imagen.

2 | Registro de imágenes

Aunque se trate quizá de una definición algo sesgada, vamos entender como registro de 2 imágenes el mapeo o proyección de una imagen sobre la otra. Ésto es, hallar de alguna manera si la primera imagen está contenida en la segunda y, de ser así, en qué región de ésta.

De forma mucho más concreta y pensando en nuestra tarea, entendemos como registro el alineamiento de 2 imágenes que, representando el mismo contenido (paisaje, persona u objeto) y desde la misma perspectiva, no tienen dicho contenido localizado en el mismo área del interior de la imagen.

2.1 Distancia entre dos imágenes

Para lograr el alineamiento vamos a considerar 2 efectos. En primer lugar vamos a necesitar algún tipo de métrica que nos diga cuánto se parece, o se diferencia, una imagen de otra. En segundo lugar necesitaremos mover una imagen sobre la otra para, ayudándonos de esa métrica, determinar cual es el desplazamiento que existe entre ellas.

Podemos definir distintos tipos de métricas de la distancia, como se nos indicó en el guión, como por ejemplo la distancia por mínimos cuadrados, o la correlación normalizada. En el desarrollo presentado en este documento escogimos implementación de la correlación normalizada que trae integrada MATLAB en la función llamada *normxcorr2*. Esta función tiene la ventaja de que ya incluye la iteración que mueve una imagen sobre la otra, si bien, tiene la desventaja de la necesidad de saber interpretar el resultado que nos devuelve.

En la figura 2.1 tenemos representado el resultado de la función cuando las dos imágenes que necesita de entrada son la misma. Es decir, está calculando una autocorrelación normalizada en 2 dimensiones.

La correlación es un cálculo que se hace de forma simétrica y que, por tanto, considera desplazamientos positivos y negativos. De forma simplificada la correlación es un cálculo de la energía de la composición de 2 señales, imágenes en nuestro caso, de forma

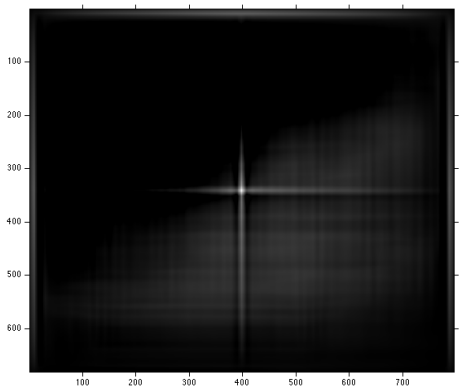


Figura 2.1: Autocorrelación con normxcorr2

que dará un valor máximo cuando en el desplazamiento entre éstas sea mínimo. En el caso ideal, cuando las 2 imágenes son la misma, dicho máximo estará localizado en la posición correspondiente al desplazamiento 0. Teniendo en cuenta la simetría anteriormente comentada, en dicho caso ideal, ese desplazamiento 0 corresponderá justamente con el punto central de la matriz de correlación (ver figura 2.1).

¿Y qué pasa si las dos imágenes no son iguales? Más allá, ¿y si las imágenes son muy parecidas pero no exactamente la misma? Entonces la correlación cruzada de ambas imágenes dará un máximo que estará desplazado del centro. De tal manera que la distancia euclídea sobre la matriz de correlación de las coordenadas entre nuestro máximo y el punto central de la matriz será, justamente, el desplazamiento que nos da el alineamiento entre las 2 imágenes.

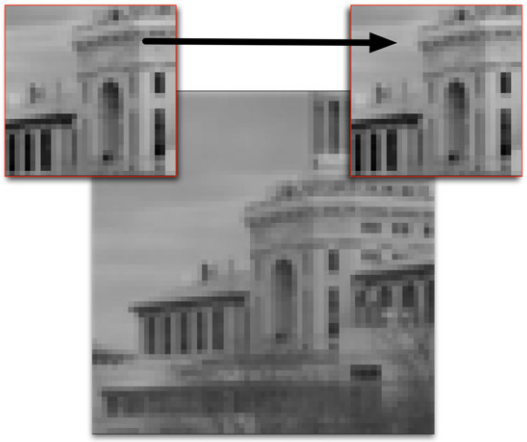


Figura 2.2: Desplazamiento de normxcorr2

2.2 Peculiaridades sobre la correlación

La forma habitual de trabajar con la función *normxcorr2* es pasarle dos imágenes de forma que una de ellas sea mayor, en sus dos dimensiones, que la otra. Entonces el algoritmo intentará buscar la imagen pequeña dentro de la grande.

Para hacer el movimiento supone el centro de la imagen pequeña como un punto ancla, que es el que desplaza, y lo hace sobre toda la imagen grande. En la figura 2.2 observamos este proceso en sus posiciones inicial y final para la primera fila.

De ésta forma, si tenemos una imagen de tamaño M_1xN_1, y otra imagen de tamaño M_2xN_2, con $M_1, N_1 \leq M_2, N_2$, el tamaño de la matriz de correlación será $(M_1 + M_2 - 1)$x$(N_1 + N_2 - 1)$. A efectos de programación, para que el cálculo de la distancia al centro sea fácil de programar y robusto, conviene que la matriz de correlación tenga sus dos dimensiones impares. Se puede demostrar de la fórmula anterior que eso se consigue cuando las dimensiones de las imágenes de entrada sean pares o impares dos a dos. Es decir, que M_1 y M_2 tengan la misma paridad. Igual con las N.

Más adelante se comentarán las implicaciones de estos detalles sobre nuestro algoritmo.

3 | Persiguiendo la eficiencia

Ahora que sabemos alinear dos imágenes debemos considerar el otro objetivo que nos habíamos marcado, la eficiencia. Para ello, como se propone, se ha implementado el método de la pirámide. Lo detallamos a continuación.

3.1 Método de las pirámides

Para reducir los tiempos de ejecución lo que necesitamos hacer es intentar reducir al mínimo el número de operaciones que tiene que hacer la máquina para llevar a cabo el algoritmo. Basándose en esa idea el método de las pirámides lo que hace es realizar los pasos que, en principio, conllevarían más operaciones en imágenes más pequeñas.

Así la idea es crear un conjunto de pasos donde, de la imagen original, crearíamos varias copias reducidas en un factor dado, usualmente potencia de 2. Así, en la copia más pequeña crearíamos una ventana de recorte de un tamaño dado, superior al desplazamiento, y calcularíamos éste. Una vez calculado el desplazamiento, d, sabemos que su tamaño, en el nivel superior, estará en el rango $(d, 2d)$, por lo que en el nivel superior podremos crear una ventana de recorte más pequeña.

Así iríamos subiendo niveles hasta llegar a la imagen original, donde tendríamos una ventana presumiblemente pequeña, por lo que podremos calcular el desplazamiento sin un retardo significativo de cálculo.

3.2 Consideraciones sobre el método

La primera y más importante observación que debemos hacer es que la ventana usada al comenzar la iteración, en el nivel más pequeño, debe ser lo suficientemente grande como para ser más grande que el desplazamiento escalado a ese nivel. En caso contrario la imagen a alinear, a efectos de la correlación, 'no estaría dentro' de la otra, por lo que la métrica para calcular el desplazamiento podría dar cualquier resultado inesperado.

Hemos también de tener en cuenta las limitaciones de la métrica, la correlación. Especialmente cuando los desplazamientos de una imagen sobre otra son demasiado pe-

queños. La correlación se sirve de puntos o áreas característicos o singulares de la imagen para realizar su cálculo y que éste represente la alineación de las imágenes. Así pues, cuanto menor sea la ventana en cada nivel, mayor es la probabilidad del cálculo incorrecto del desplazamiento a través de la correlación. Por otro lado una ventana demasiado grande implicaría demasiadas operaciones a realizar por tanto perderíamos eficiencia. Es por ello que debemos establecer algún tipo de solución de compromiso. Como se apuntará más adelante, se ha optado por imponer un tamaño mínimo de ventana en cada paso, de forma que aunque el desplazamiento sea menor, se seguirá tomando ese tamaño mínimo.

Finalmente debemos plantearnos cuántos niveles de pirámide crear. En el algoritmo que presentaré más adelante se han establecido 3 niveles: 1, 1/2 y 1/4. Se probó a incluir un cuarto nivel, pero los tiempos de ejecución subían considerablemente. Téngase en cuenta que un nivel más implica repetir todos y cada uno de los pasos que sean necesarios para calcular el desplazamiento. Por tanto, aunque al disminuir el tamaño, y las ventanas de recorte, disminuyamos el número de operaciones a realizar, existirá un límite a partir del cuál, introducir un nivel de división más no quite operaciones, sino que añada más.

4 | Algoritmo implementado

El algoritmo implementado se compone en total de 3 ficheros. La función *registralmagen*, la función *recortarBordes*, y un script de demostración y automatización de la obtención de resultados.

4.1 registralmagen

Esta función es la encargada de calcular el vector de desplazamiento de una imagen sobre otra. Notar que no realiza el registro como tal, sino que solo calcula cuál sería. Recibe como entrada las 2 imágenes a registrar, una de las cuales la tomará como 'base' y otra como 'registable' (que será la que desplazará sobre la imagen base), y un tercer parámetro que estará relacionado con el tamaño de la ventana en la primera iteración (en el nivel más bajo, con la imagen con mayor reducción).

Este tercer parámetro (llamémosle p) por sencillez en las siguientes iteraciones, será multiplicado por 2 en la primera iteración. Los recortes a tomar de cada imagen se harán desde el centro, ya que se presupone que será en el centro de la misma donde tendrá la mayor cantidad de puntos significativos que maximizarán la métrica. Hemos de tener en cuenta que cogeremos 2 recortes de distinto tamaño para la imagen base y la imagen registrable (IB e IR, respectivamente, en el código). Así por tanto, tenemos que para la imagen registrable la ventana en la primera iteración tendrá un tamaño de 4p+1, es decir el área desde el centro de la imagen $\pm 2p$.Y para la imagen base, la filosofía es parecida, el centro $\pm 4p$ en este caso, es decir, el doble tamaño que la ventana anterior.

Así, si para las imágenes grandes hemos cogido un valor inicial de 20, significará que nuestras ventanas en el primer nivel tendrán un tamaño de $2 \cdot 2 \cdot 20 + 1 = 81$, es decir, $81x81$ mientras que para la imagen registrable, y $2 \cdot 4 \cdot 20 + 1 = 161$, por tanto, $161x161$.

Otro detalle importante es que se sesgan los resultados posibles devueltos por la correlación. Es decir, si seleccionamos en una primera iteración, como en el ejemplo que acabamos de comentar, una ventana de 81 pixeles, significa que el desplazamiento máximo que vamos a permitir será de ± 40 pixeles. Como vimos antes, la correlación para este

caso tendrá un tamaño de 80+160+1=241 pixeles. En algunos casos esto podría darnos falsos vectores de desplazamiento, por lo que limitamos, haciendo cero, los valores de correlación que estén más allá de una ventana de ± 40 pixeles del centro de la correlación.

Algunos parámetros a destacar son, por ejemplo, que se ha determinado una ventana mínima que da unos buenos resultados para la mayoría de imágenes de ± 10 píxeles, tanto para imágenes grandes como para imágenes pequeñas.

Por contra, serán necesarios dos valores de ventana inicial según la imagen de entrada sea grande o pequeña. El resultado es, aparentemente, muy sensible a este parámetro, por ejemplo, un valor demasiado grande podría desbordarnos en imágenes pequeñas. Por tanto en la función se ha dejado como valor a introducir por el usuario. En cualquier caso, en base a las pruebas realizadas, 8 parece ser un valor adecuado para las imágenes pequeñas, mientras que en las grandes se ha determinado un valor adecuado de 20.

Los demás detalles que pueda tener esta función no son más que meras tretas matemáticas para ajustar índices y obtener así el vector de desplazamiento correcto.

4.2 recortarBordes

Una vez ajustada y alineada la imagen, podemos pasársela a esta función para que realice el recorte. Esta función, al igual que la anterior, no realiza el recorte propiamente dicho, sino que calcula los límites que habría que tomar de la imagen para que los bordes quedaran fuera. En otras palabras, define un marco de recorte.

El procedimiento de recorte propuesto se basa en que todas las imágenes poseen, aunque sea muy fino, un borde negro. Sabiendo ésto y teniendo en mente siempre el objetivo de la eficiencia, se pensó en primera instancia proceder como sigue.

En primer lugar seleccionar solo un pequeño área de la imagen para cada borde, de forma que cada uno de esos pequeños áreas (4 en total) contengan cada uno un borde distinto.

Una vez recortados éstos trozos de imagen calculamos su proyección[1] horizontal para los bordes verticales, y su proyección vertical para los bordes horizontales. Si representamos para uno de los bordes, en este caso el superior, podemos encontrarnos con algo como lo que apreciamos en la figura 4.2.

Por supuesto, estamos considerando la imagen en negativo, para que el negro se convierta en blanco y de un valor máximo en la proyección.

Tendremos un esquema similar para los 4 bordes. Nos bastará, por tanto, establecer unos umbrales en las proyecciones y detectar los puntos que los superen. Notar que

[1] En los comentarios del código llamo a estas proyecciones 'histogramas'

Figura 4.1: Areas de selección (aproximadas) para detección del borde

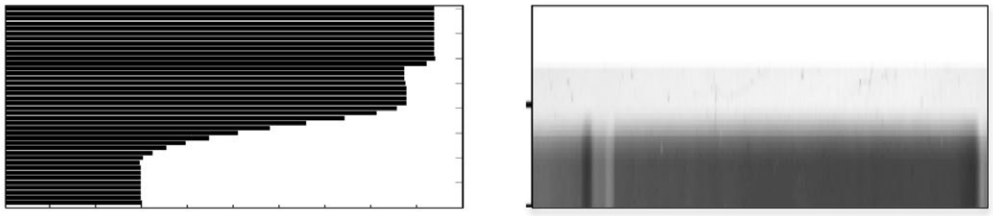


Figura 4.2: Area de recorte y su proyección vertical

habrá muchos valores que superan el umbral, mientras que nosotros solo buscamos un valor, un índice. Nos bastará con coger el último valor que supere el umbral, para los bordes izquierdo y superior, y el primer valor lo supera, para los bordes derecho e inferior.

Hasta ahora ya tenemos una forma más o menos eficiente de realizar el recorte, pero se pensó en hilar todavía un poco más fino. Para ello se procedió de manera similar a las pirámides. Esto es, realizamos el mismo procedimiento descrito con anterioridad, pero en una imagen un cuarto de la original. Así el cálculo de las proyecciones es mucho más rápido. No obstante, al igual que pasaba con los vectores de desplazamiento, los índices de los bordes calculados serán un cuarto más pequeños, por lo que es necesario

multiplicarlos por 4 para tener los índices reales en la imagen a tamaño original. No obstante, el factor que se usa es 5.5 en lugar de 4, en un intento de tomar el borde por exceso, para paliar las posibles imperfecciones de esta forma de calcular los bordes.

Haciendo este paso adicional conseguimos bajar los tiempos algunas décimas de segundo.

Deficiencias

Esta forma de calcular los bordes presenta algunas deficiencias más o menos importantes.

La primera está en la elección del umbral. Yo propongo que su valor sea 3/4 del máximo de la imagen en esa dimensión, suponiendo que el máximo será un borde. Esto es para intentar evitar cortar demasiado en aquellas imágenes que, tras la corrección del desplazamiento para el registrado, se hayan quedado sin borde. Sin embargo este umbral es bastante restrictivo a la hora de considerar el máximo que indica el corte, pudiendo ocurrir que un borde que no sea completamente negro, no supere el umbral y no sea detectado, como ocurre en la imagen 5.2c.

Distintos tipos de umbrales fueron probados, siendo éste el que daba mayor cantidad de buenos recortes y, a su vez, no recortando la imagen en evidente exceso.

La otra gran deficiencia es que estamos detectando los bordes como áreas negras. Es decir, si el contenido de nuestra foto, junto al borde que pretende ser detectado, tiene un nivel de negro comparable con el borde, será detectado como tal y, por tanto, recortado. De igual manera podría ocurrir, y de hecho ocurre en varias de las imágenes propuestas, si en lugar de bordes negros estuviésemos considerando bordes blancos.

4.3 demo.m

Finalmente se incluye un script en el que se automatizan algunas tareas de estadísticas y medición de tiempos. Notar que los algoritmos anteriores no realizan el alineamiento ni el recorte, sino que solo calculan los datos para ello. Por tanto ambas tareas hay que hacerlas 'a mano' a posteriori. Esas son alguna de las tareas que se automatizan en este fichero.

Notar que, como se propone, se alinean los canales azul y rojo sobre el canal verde, siendo éste el que no sufre desplazamiento ninguno.

Otra mejora que se incluye en este fichero es una ecualización del histograma de la imagen. Para ésta no se ha conseguido optimizar de ninguna manera. Sus resultados en tiempo no son demasiado malos, en general, pero son por sí solos comparables a todo el resto del procedimiento por lo que, si se desea ecualizar, los tiempos se incrementarán aproximadamente al doble.

5 | Resultados

A continuación se presentan los resultados obtenidos. Tanto las propias imágenes como los tiempos.

5.1 El equipo

Siempre que se considere un estudio de los tiempos de ejecución es conveniente aclarar las condiciones de contorno sobre las que dichas mediciones fueron realizadas.

Se traba de MacBook Pro Core 2 Duo de 64 bits a 2,26 GHz, 4 GB de RAM. El sistema operativo es Mac OS X 10.8.3 y la versión de MATLAB R2013a de 64 bits.

5.2 Tiempos de ejecución

Como se puede apreciar en el script, se han tomado diversas medidas de los distintos procedimientos que componen el proyecto.

Los resultados a continuación surgen tras lanzar el proceso completo a todas las imágenes, 20 veces cada una.

Tarea	Imágenes grandes	Imágenes pequeñas
Registro canal rojo	0.8440	0.0335
Registro canal azul	0.8730	0.0339
Corte de bordes	0.2739	0.0057
Proceso total	4.3573	0.1061

Tabla 5.1: Tiempo de ejecución para una batería de 20 iteraciones

En la tabla 5.1 tenemos los tiempos medios para 20 iteraciones de cada grupo de fotos. Comentar que no se ha usado la ecualización en este caso. Al no estar optimizada es el proceso más lento, consumiendo él solo en torno a 2 segundos para las imágenes

grandes.

Como se puede observar, si sumamos los tiempos parciales no obtenemos el tiempo del proceso total. Como ya se apuntó, las funciones solo calculan los índices. El proceso de superposición de las imágenes ya alineadas y el recorte hay que hacerlos 'a mano'. Son esas funciones adicionales, crear una matriz nueva, asignar las bandas de color, etc, las responsables de esos tiempo de la tabla.

5.3 Vectores de desplazamiento

Como ya se apuntó anteriormente, el canal verde se mantiene fijo y son los canales rojo y azul los que alineamos sobre éste. Por tanto, en la tabla 5.2 podemos observar los vectores de desplazamiento obtenidos para cada canal, rojo y azul, de cada imagen. Se muestran en la misma forma que los genera MATLAB, primero la coordenada de la fila (coordenada Y) y luego la de la columna (coordenada X).

Imagen	Canal Rojo	Canal Azul
00125v.jpg	[4 -1]	[-5 -2]
00149v.jpg	[5 0]	[-4 -2]
00153v.jpg	[7 2]	[-7 -3]
00154v.jpg	[6 -1]	[-5 -1]
00163v.jpg	[-2 0]	[3 -1]
00270v.jpg	[8 1]	[-3 -1]
00398v.jpg	[6 1]	[-5 -3]
00564v.jpg	[6 -1]	[-5 -1]
01167v.jpg	[7 -2]	[-5 0]
31421v.jpg	[5 0]	[-8 -0]
00458u.tif	[42 26]	[-40 -7]
00911u.tif	[120 -6]	[-12 7]
01043u.tif	[24 8]	[15 -11]
01047u.tif	[47 13]	[-23 -20]
01657u.tif	[58 4]	[-48 -7]
01861a.tif	[76 24]	[-68 -38]

Tabla 5.2: Vectores de desplazamiento obtenidos para cada imagen

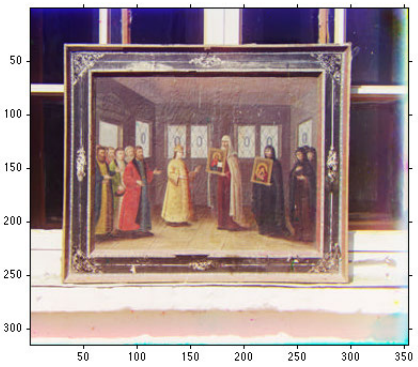
En base a los resultados visuales que observaremos en el próximo apartado, éstos valores parecen ser correctos. No obstante, y especialmente en las imágenes grandes, cabe la posibilidad de un ligero error de 1 o 2 pixeles, inapreciable a simple vista.

5.4 Imágenes resultado

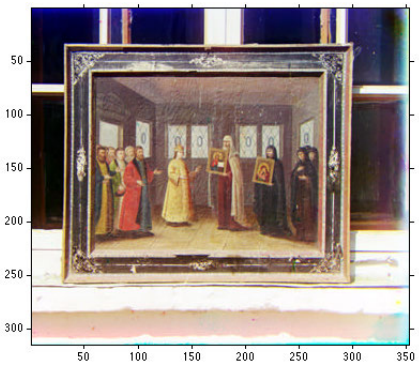
A continuación se muestran las imágenes finales obtenidas, con y sin ecualización del histograma. Todas con el algoritmo de recorte de bordes aplicado.

(a) Imagen 00125v.jpg alineada

(b) Imagen 00125v.jpg ecualizada

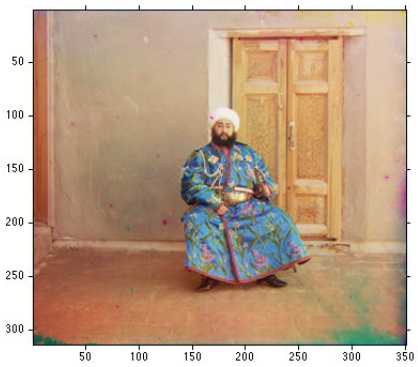


(c) Imagen 00149v.jpg alineada

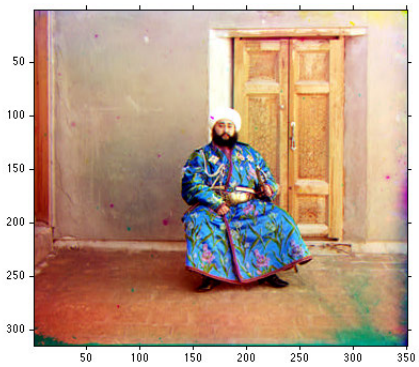


(d) Imagen 00149v.jpg ecualizada

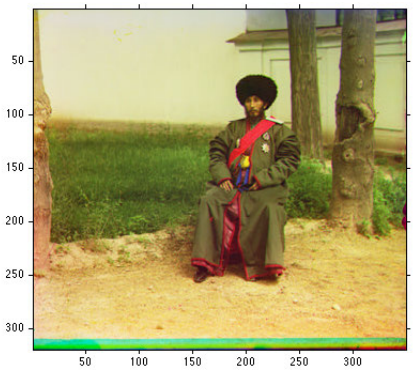
Figura 5.1: Imágenes 00125v.jpg y 00149v.jpg



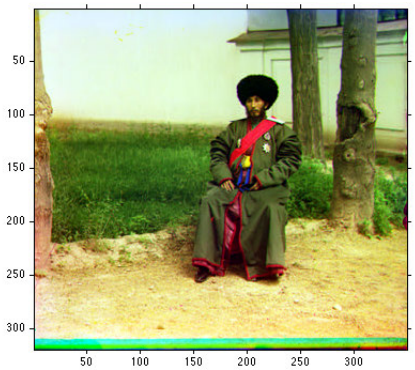
(a) Imagen 00153v.jpg alineada



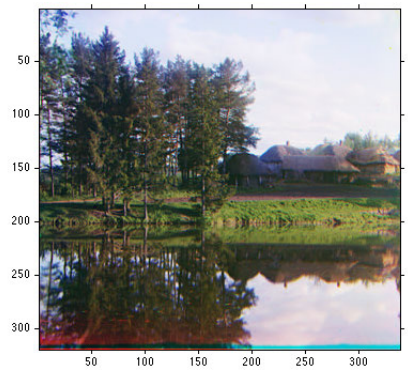
(b) Imagen 00153v.jpg ecualizada



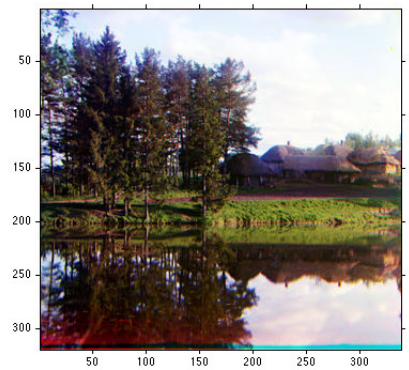
(c) Imagen 00154v.jpg alineada



(d) Imagen 00154v.jpg ecualizada

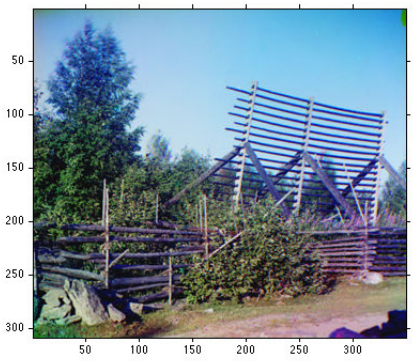


(e) Imagen 00163v.jpg alineada

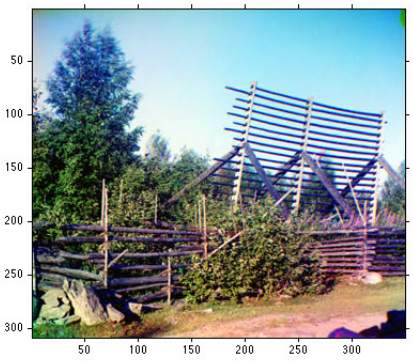


(f) Imagen 00163v.jpg ecualizada

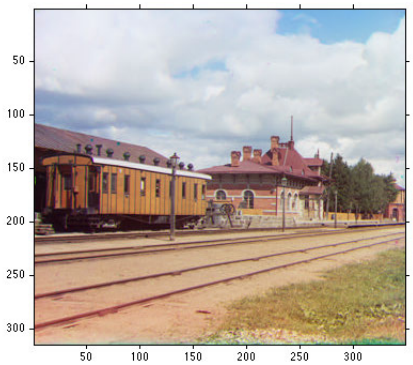
Figura 5.2: Imágenes 00153v.jpg, 00154v.jpg y 00163v.jpg



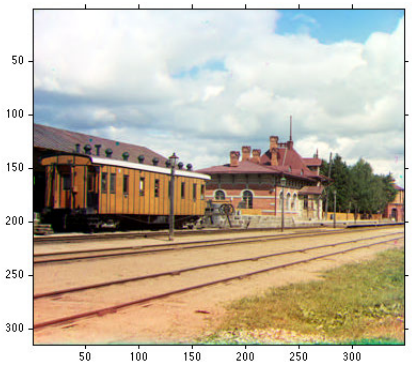
(a) Imagen 00270v.jpg alineada



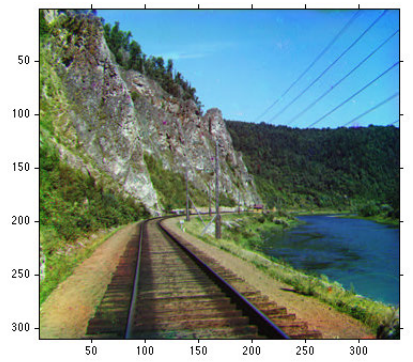
(b) Imagen 00270v.jpg ecualizada



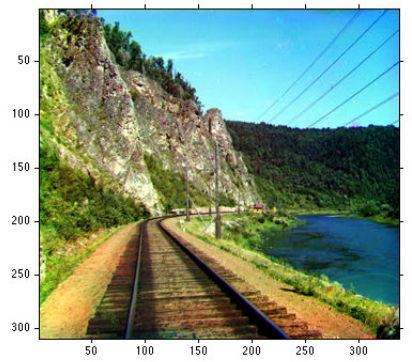
(c) Imagen 00398v.jpg alineada



(d) Imagen 00398v.jpg ecualizada

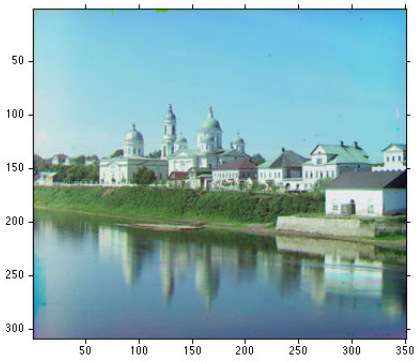


(e) Imagen 00564v.jpg alineada

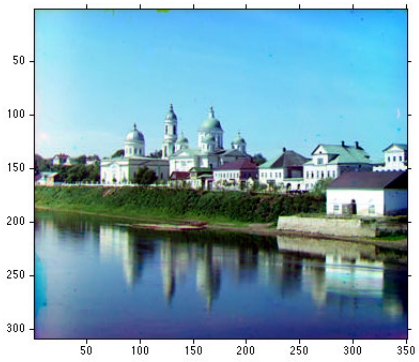


(f) Imagen 00564v.jpg ecuaizada

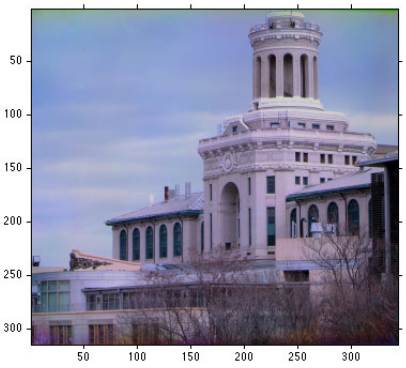
Figura 5.3: Imágenes 00270v.jpg, 00398v.jpg y 00564v.jpg



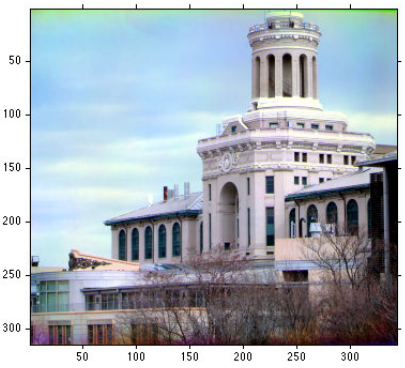
(a) Imagen 01167v.jpg alineada



(b) Imagen 01167v.jpg ecualizada



(c) Imagen 31421v.jpg normalizada



(d) Imagen 31421v.jpg ecualizada

(e) Imagen 00458u.tif alineada

(f) Imagen 00458u.tif ecualizada

Figura 5.4: Imágenes 01167v.jpg, 31421v.jpg y 00458u.tif

(a) Imagen 00911u.tif alineada

(b) Imagen 00911u.tif ecualizada

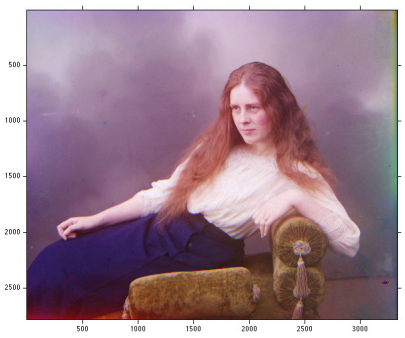
(c) Imagen 01043u.tif alineada

(d) Imagen 01043u.tif ecualizada

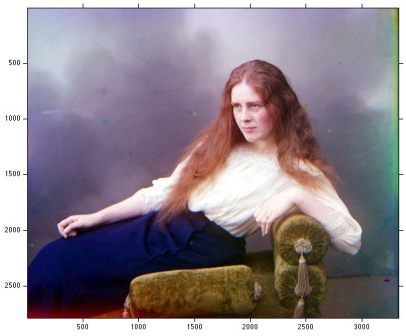
(e) Imagen 01047u.tif alineada

(f) Imagen 01047u.tif ecualizada

Figura 5.5: Imágenes 00911u.tif, 01043u.tif y 01047u.tif



(a) Imagen 01657u.tif alineada



(b) Imagen 01657u.tif ecualizada

(c) Imagen 01861a.tif alineada

(d) Imagen 01861a.tif ecualizada

Figura 5.6: Imágenes 01657u.tif y 01861a.tif