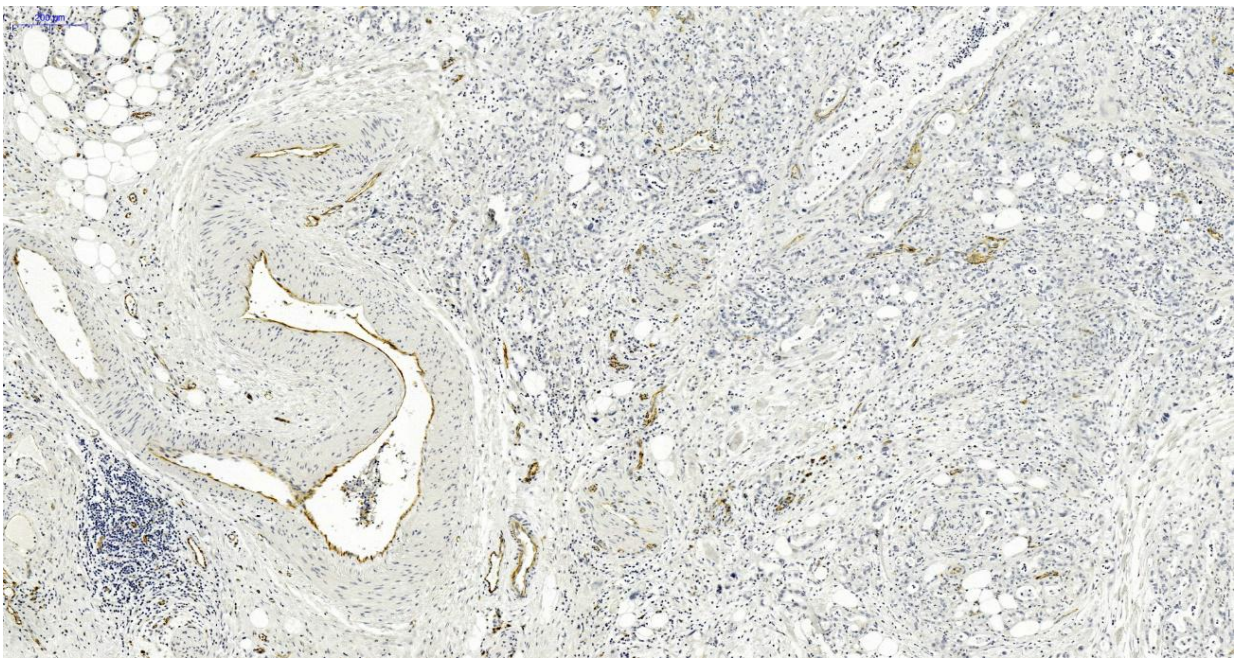
RETO1: Discriminar los puntos azules de la imagen



Original

El objetivo del reto es separar todos los puntos azules del resto de la imagen de forma que en la nueva imagen sólo estén representados los puntos azules.

Sabemos que la imagen está representada por tres matrices, una para el color Verde, otra para el Azul y otra para el Rojo, así cada píxel es la composición de los valores de cada una de estas matrices para un determinado par de coordenadas i e j, quedando cada píxel representado por tres valores entre 0 y 255, a este modelo se le denomina modelo de colores RGB.

En este modelo, podemos interpretar los colores como puntos en el espacio cúbico comprendido entre las tres dimensiones, Red, Green y Blue (RGB) y por lo tanto, una solución sencilla al problema propuesto consiste en filtrar el espacio RGB de forma que los colores comprendidos en el subespacio de interés conserven su valor y el resto se conviertan a blancos.

Sea $S \in \{r, g, b\}$ tal que $\mathrm{r, g, b} \in [0,255]$ el espacio de colores RGB.
e $I(i, j) \in S$ una imagen representada como una matriz de puntos de S

Y sean $r_{\min}, r_{\max}, g_{\min}, g_{\max}$ y $b_{\min}, b_{\max}$ los valores mínimos y máximos del subespacio en el que se comprenden los colores deseados.

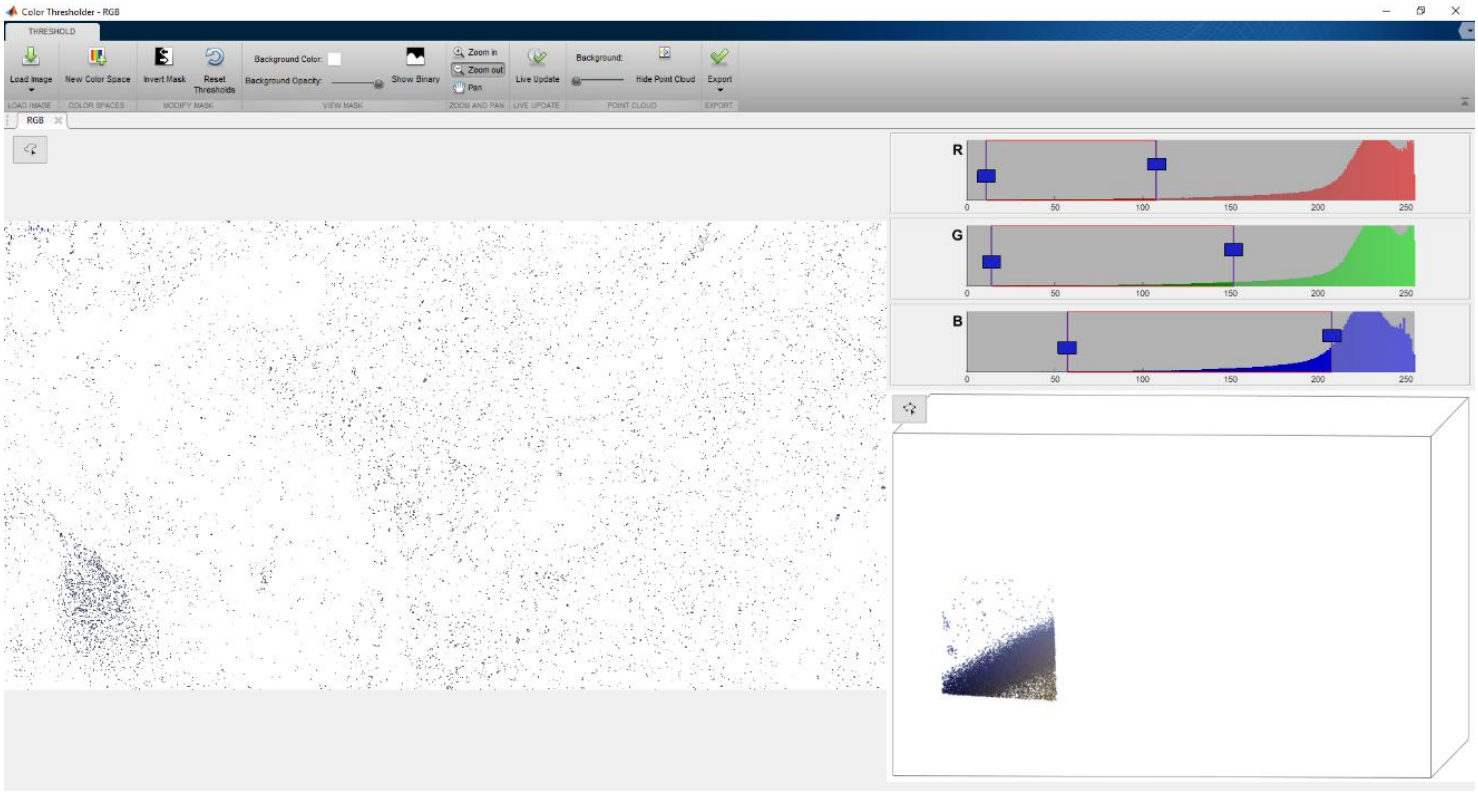
La imagen resultado es el producto de

$$I^*(i, j) = \begin{cases} I(i, j) & , r_i \in [r_{\min}, r_{\max}] \wedge g_i \in [g_{\min}, g_{\max}] \wedge b_i \in [b_{\min}, b_{\max}] \\ (255,255,255) \end{cases}$$

Utilizando la herramienta de **Color T resholder** de MatLab he seleccionado y ajustado

h

manualmente el rango de colores que me interesaba, se distingue fácilmente el la parte derecha que secciones del espacio se van a filtrar.



Color Thresholder tool

Luego he exportado la función de filtrado a otro fichero donde se puede modificar libremente y están disponibles los valores del filtro para cada capa de color, entonces he establecido el fondo de color blanco ya que el experto así lo requería.

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace
and
%  range for each channel of the colorspace were set within the app.
The
%  segmentation mask is returned in BW, and a composite of the mask
and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 18-Sep-2018
%-------------------------------------------------------
```

```
% Convert RGB image to chosen color space
I = RGB;

% Define thresholds for channel 1 based on histogram settings
channel1Min = 11.000;
channel1Max = 108.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 14.000;
channel2Max = 152.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 57.000;
channel3Max = 208.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to 255 white Backgroung.
maskedRGBImage(repmat(~BW,[1 1 3])) = 255;

end
```

Básicamente, lo que esta función hace es crear una máscara binaria a partir de los límites marcados anteriormente y los valores de cada pixel, luego aplica la máscara a la nueva imagen conservando o eliminando los valores originales.

Finalmente, cabe destacar que este filtrado es sólo una solución posible y que solo filtra espacios continuos y con geometrías de prismas rectangulares. Por otro lado, su sencillez también lo hace bastante eficiente, ya que es de orden lineal en el número de evaluaciones a cada pixel y en la cantidad de memoria requerida.

Imagen resultante