

Ejercicio Obia

Juan Sebastian Vinasco Salinas

October 14, 2025

Contents

1	Ejercicio practico de Geo-OBIA	1
1.1	Fuente de datos	1
1.2	Detección de bordes o zonas de alta frecuencia con sobel	2
1.3	Segmentación	3
1.3.1	Segmentación	3
1.3.2	Vectorización y Extracción de características	7
1.4	Automatización con python	9
1.5	Tarea	11

1 Ejercicio practico de Geo-OBIA

El presente documento ilustras los pasos para replicar el ejercicio propuesto en la clase de OBIA.

Como nota al margen se usara el termino “segmentación” en el sentido de las ciencia de *Sensores Remotos* y no en el sentido usado en las ciencias de *Visión por computador* que no son intercambiables.

Adicionalmente, los ejercicios mezclaran el uso de QGIS (versión 3.40.9-Bratislava) y de Monteverdi (que es la interfaz gráfica de Orfeo Toolbox en su versión 8.1.2), tenga en cuenta que monteverdi solo esta disponible para esta versión o anteriores, OTB 9.x o superior descontinúan monteverdi en favor de la integración con QGIS.

1.1 Fuente de datos

La fuente de datos propuesta para el ejercico es el almacen de datos de pruebas de la biblioteca Orfeo Toolbox, tenga en cuenta que algunas de

estas imágenes no tienen sistema de referencia asignado y al momento de ser visualizadas en QGIS, esto puede generar problemas.

Datos:

- Fuente de datos ejercicio extracción de bordes
- Fuente de datos ejercicio segmentación y extracción de características

1.2 Detección de bordes o zonas de alta frecuencia con sobel

Una aproximación rápida para la extracción de características lineales es el uso de filtros como lo son el filtro de Sobel, este filtro en particular explota el cálculo de derivadas direccionales para determinar las zonas donde los cambios en los pixeles son más altos, permitiendo así la identificación de elementos lineales o bordes:

1. Primer paso: Desplegar los algoritmos de OTB.

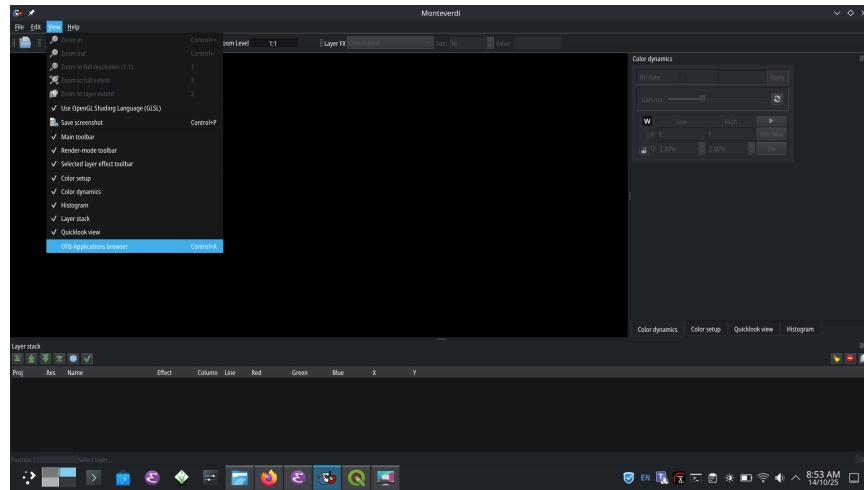


Figure 1: Desplegar panel de algoritmos

1. Seleccionar el panel de detección de bordes
1. Aplicar el algoritmo de filtro de sobel
 - Seleccione el algoritmo deseado en el panel de la izquierda (marcado en color verde)

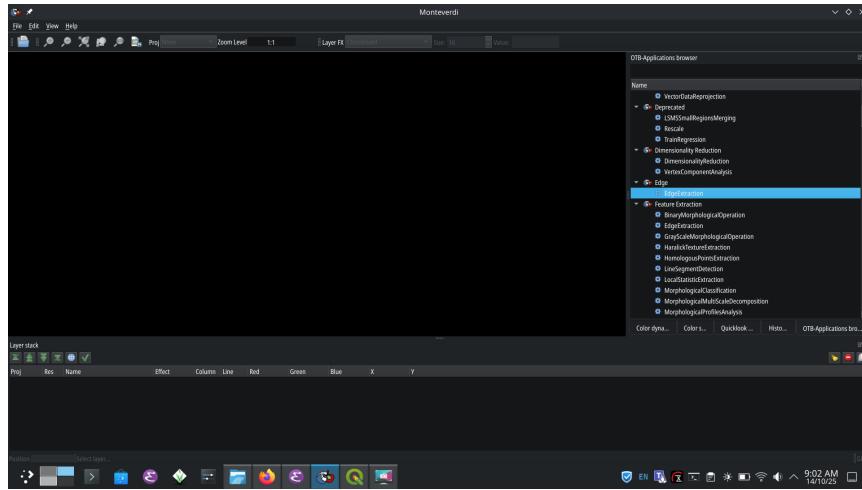


Figure 2: Selección panel de detección de bordes

- Introduzca la ruta completa de las imágenes de entrada (marcado en amarillo) y salida (marcado en azul claro)
- Seleccione el algoritmo a aplicar (marcado en blanco) (están disponibles también el algoritmo de gradiente y el de touzi)

Finalmente comparamos las imágenes de entrada y salida:

1.3 Segmentación

1.3.1 Segmentación

Similar a el ejercicio anterior, pero esta vez usando QGIS como interfaz gráfica, aplicaremos un algoritmo de segmentación siguiendo los siguientes pasos:

1. Abra la caja de herramientas de geoprocisos de QGIS, despliegue el “proveedor” OTB, en su categoría “Segmentación” y seleccione la aplicación “Segmentación” (marcado en verde en la imagen)
2. Seleccione la imagen objetivo de la lista desplegables (marcado en amarillo)
3. Seleccione el algoritmo que deseé aplicar y parametrización según su criterio (marcado en azul claro) (La lista de algoritmos son: meanshift, connected componentes, watershed, mprofiles)

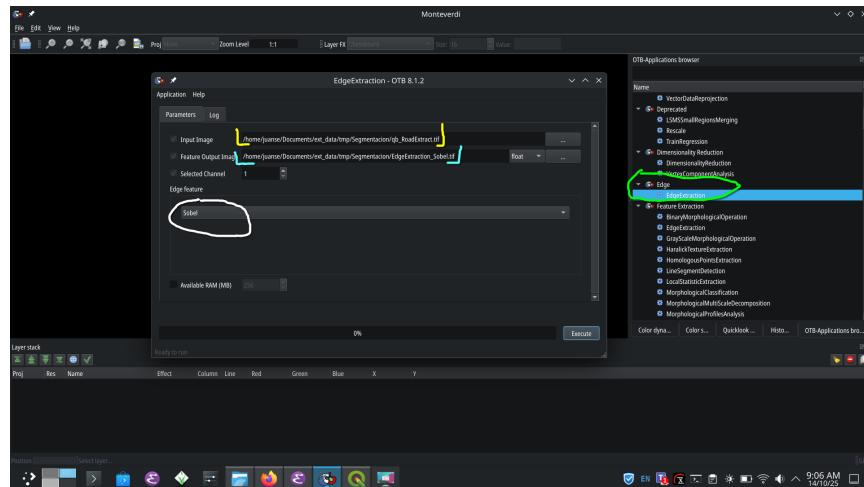
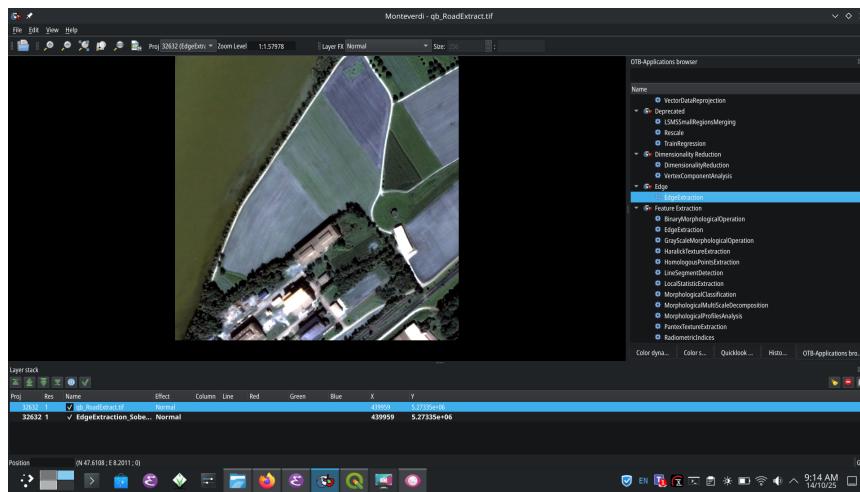
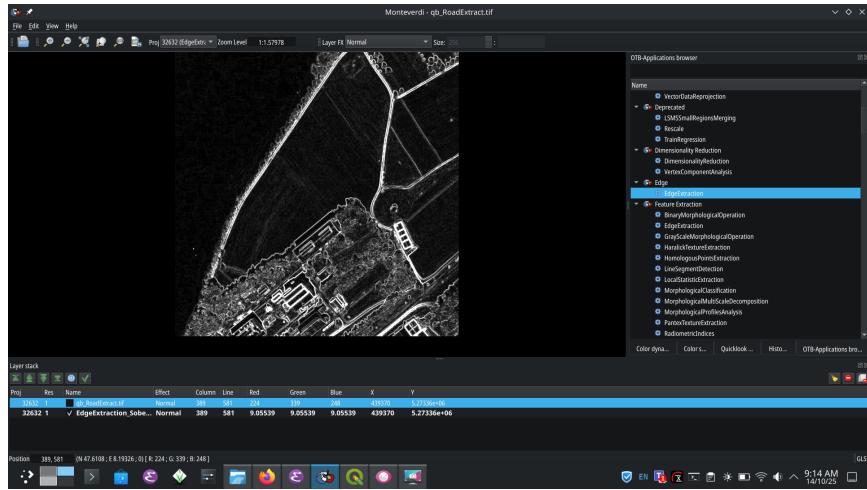


Figure 3: Aplicar el algoritmo de filtro de sobel





4. Seleccione el modo **raster** (marcado en rojo) (paso importante el modo vectorial tiende a fallar)
5. Seleccione un archivo de salida (marcado en azul oscuro)

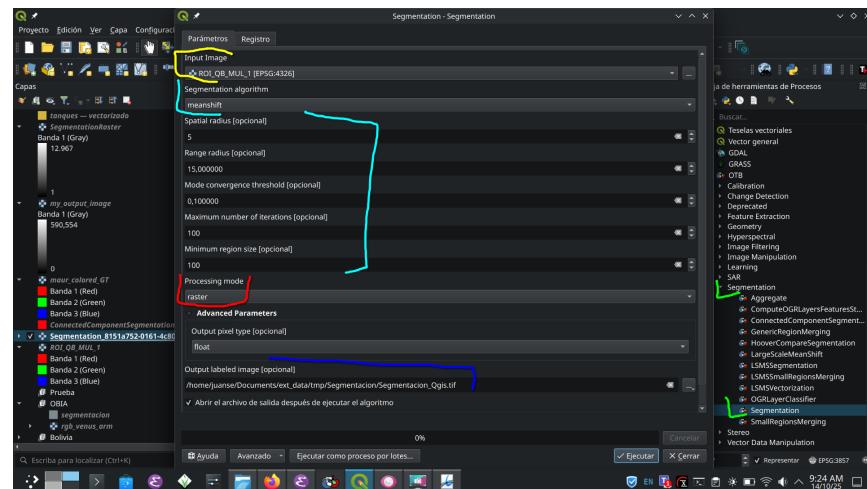


Figure 4: Segmentación meanshift

La salida del modulo, es como la siguiente

Con el objetivo de mejorar la visualización, podemos asignar un color diferente y aleatorio a cada uno de los segmentos. Para ello dirigase a propiedades, como en la siguiente imagen

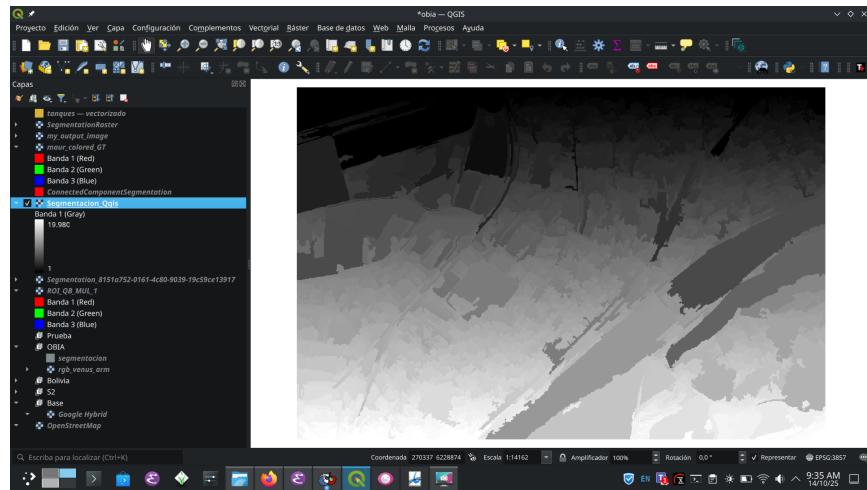


Figure 5: Salida de la Segmentación Meanshift

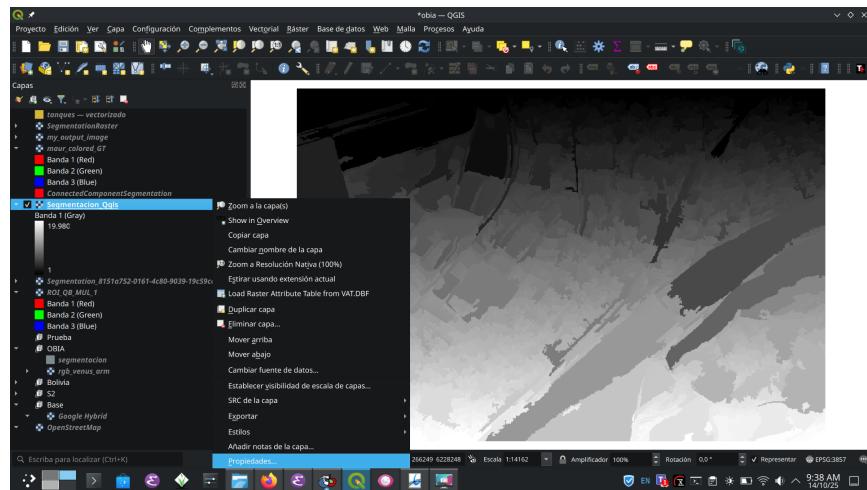


Figure 6: Propiedades de la imagen

A continuación seleccione el panel de simbología, categorize por “Valores en paleta/unicos”, y seleccione una paleta de color aleatoria, luego presione los botones de clasificar, aplicar y por ultimo aceptar. Obtendrá una imagen como la de la derecha.

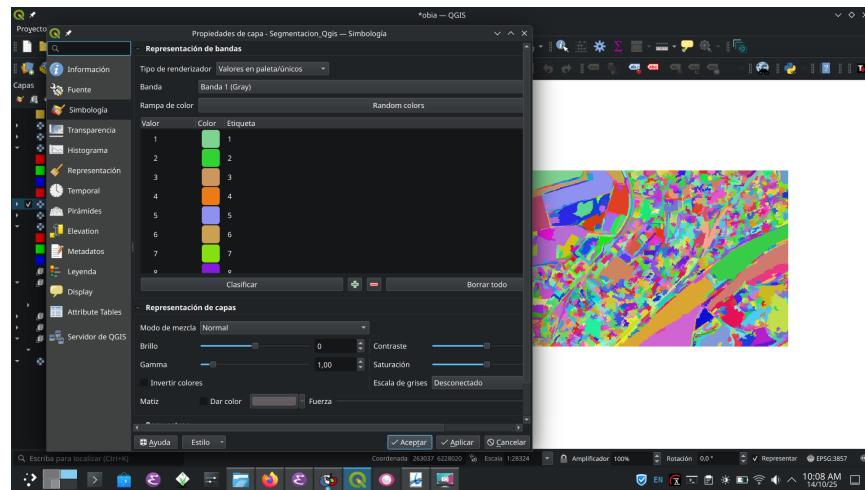


Figure 7: Asignación de colores aleatorios

1.3.2 Vectorización y Extracción de características

Una vez, se tienen los resultados de una segmentación es posible continuar el análisis utilizando herramientas vectoriales, para ello lo primero es convertir la segmentación en una capa vectorial, como se puede ver en la siguiente imagen. Siguiendo los siguientes pasos:

1. Abrir la herramienta de poligonizar (marcado en verde)
2. Seleccione la imagen a convertir (marcado en verde)
3. Introduzca una ruta de salida para la imagen (marcado en verde)

Luego, procedemos a buscar los tanques pretoleros que se pretenden extraer, como se visualizan en la imagen a continuación.

Seguidamente se le da estilo a la capa vectorial siguiendo el mismo procedimiento que antes, se selecciona los segmentos que se superponen con los tanques y se le da click derecho sobre la capa, exportar y se selecciona la opción “**Guardar solamente objetos espaciales seleccionados**”

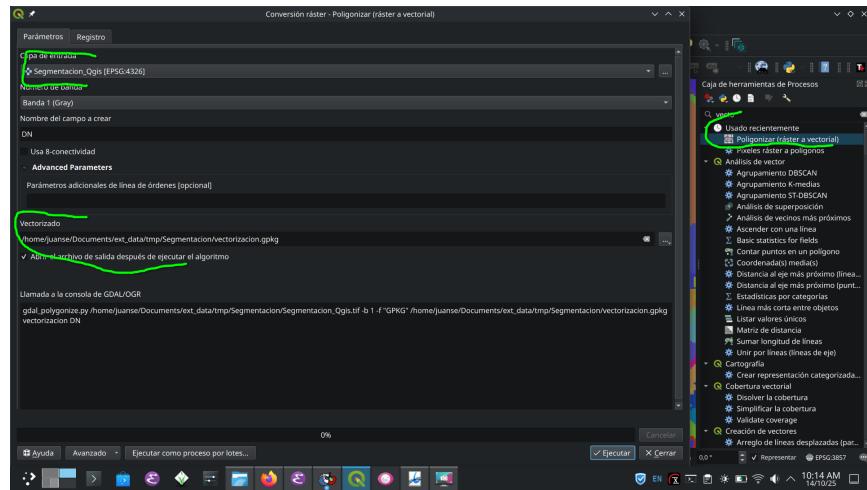


Figure 8: Vectorizar

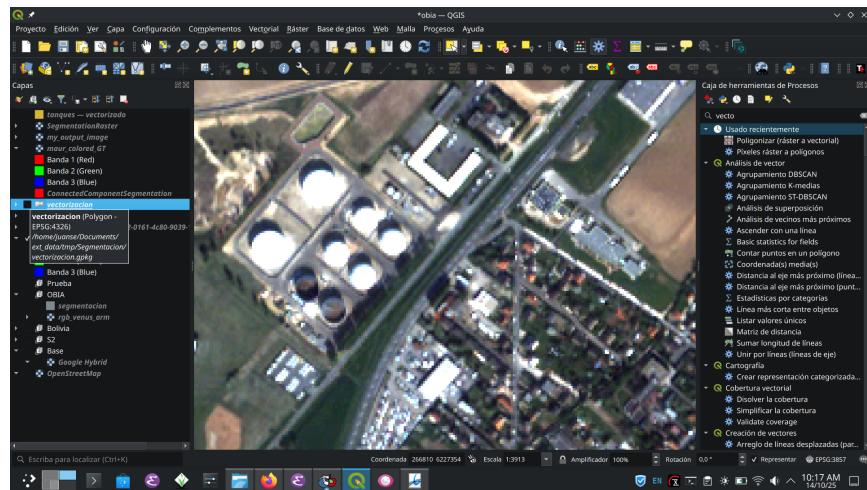


Figure 9: Objetivos de la vectorización

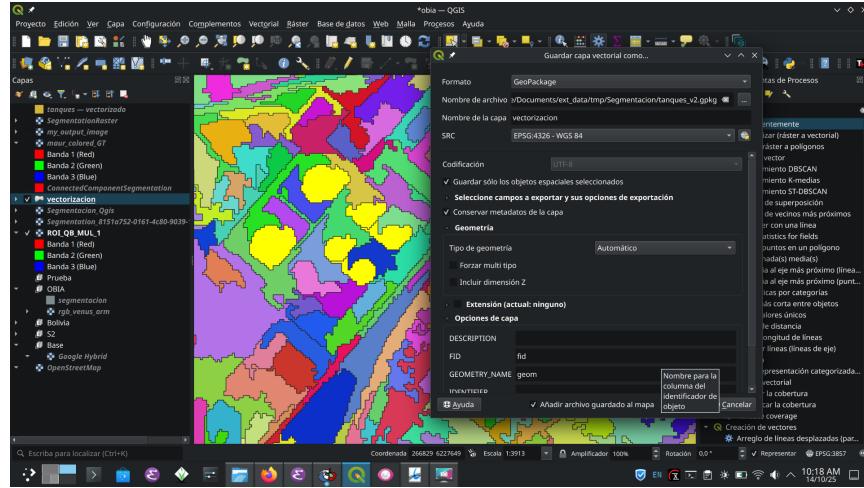


Figure 10: Selección y exportacion de segmentos

Finalmente se superponen los objetos extraídos con la imagen para validar que tan correctos son:

1.4 Automatización con python

Como bonus podemos automatizar la búsqueda de parámetros usando una estrategia de “fuerza bruta”, a través del siguiente ejemplo.

Para replicarlo necesitas una instalación de python con acceso a Orfeo Toolbox, puedes encontrar mas información en el siguiente enlace

```
#!/usr/bin/env python3

import os
from tqdm import tqdm
import otbApplication

os.chdir("/home/juanse/Documents/ext_data/tmp/Segmentacion")

def variacion_meanshift(spatial_radius: int):
    """
    varia los parametros del meanshift
    """

```

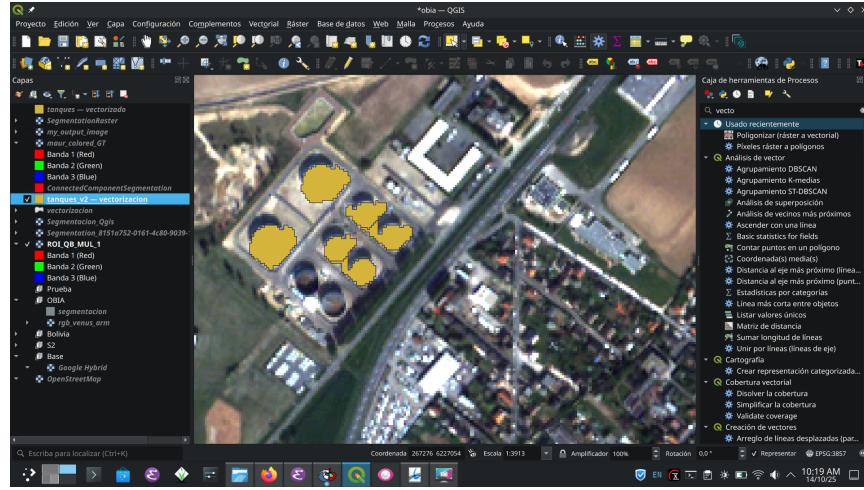


Figure 11: Vectorización Segmentación

```

app = otbApplication.Registry.CreateApplication("Segmentation")
app.SetParameterString("in",
    "/home/juanse/Documents/ext_data/tmp/Segmentacion/Copia de Sobel.tif")
app.SetParameterString("mode",
                      "raster")
app.SetParameterString("filter.meanshift.spatialr",
                      f"{spatial_radius}")
app.SetParameterString("mode.raster.out",
                      f"/home/juanse/Documents/ext_data/"+
                      "tmp/Segmentacion/variacion_parametros/SegmentationRaster_{spatial_radius}.tif")
app.SetParameterOutputImagePixelType("mode.raster.out", 3)
app.SetParameterString("filter","meanshift")
app.Execute()
# app.ExecuteAndWriteOutput()

return app

def main():

    lista_valores = list(range(5, 100, 5))

    for i in tqdm(lista_valores):

```

```
variacion_meanshift(i)

if __name__ == "__main__":
    main()
```

1.5 Tarea

Replique el ejercicio de comparación de segmentaciones, basado en la siguiente guía