



---

# TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

## DETECCIÓN DE ANOMALÍAS EN CARTOGRAFÍA DIGITAL

---

**Autor**

Víctor Díaz Bustos

**Tutora**

Rocío Celeste Romero Zaliz



July 3, 2022



# Detección de anomalías en cartografía digital

Víctor Díaz Bustos

**Palabras clave:** Keypoint, Descriptor, Detector, Gradiente, Algoritmo SIFT, Pirámide de Lowe, Espacio de Escalas, Octava, Diferencia de Gaussianas, Algoritmos Genéticos, Convolución, Sensibilidad, F1 SCORE

## Resumen

Hoy en día, el campo de la inteligencia artificial y la visión por computador está comenzando a vivir una época de auge y gran expansión, de forma que se busca unificar ambas tecnologías para obtener resultados sorprendentes y con multitud de aplicaciones, como vehículos de conducción autónoma, reconocimiento facial, diagnósticos en el campo de la salud, etc.

Este proyecto está enfocado en una colaboración con el proyecto FQ4DEM de la Universidad de Jaén cuyo objetivo es detectar elementos inusuales en imágenes (mapas de sombras) que faciliten la combinación de varios mapas de sombras.

Para ello se ha utilizado un algoritmo de detección de puntos como es el algoritmo *SIFT* sobre distintos Modelos Digitales de Elevación (MDE) de una misma imagen y se ha tratado de obtener la mejor combinación posible con estos resultados respecto de una solución real, denominada *ground truth* (generada manualmente como prueba de concepto).

Para obtener la mejor combinación, dado que no es posible calcularlo mediante fuerza bruta debido al elevado número de datos, se ha optado por un algoritmo de optimización como el *Algoritmo Genético Elitista Uniforme* (AGG\_UN) y se ha probado a obtener la mejor solución con tres tipos diferentes de combinaciones: utilizando uniones de MDEs, utilizando intersecciones, y combinando uniones e intersecciones de MDEs.



# Detection of anomalies in digital cartography

Víctor Díaz Bustos

**Keywords:** Keypoint, Descriptor, Detector, Gradient, SIFT algorithm, Lowe's Pyramid, Scale-Space, Octave, Difference of Gaussians, Genetic Algorithms, Convolution, Sensibility, F1 SCORE

## Abstract

Nowadays, artificial intelligence and computer vision are beginning to experience a period of boom and great expansion, so that it seeks to unify both technologies to obtain surprising results and with many applications, such as autonomous driving vehicles, facial recognition, diagnoses in the field of health, etc.

This project is focused on a collaboration with the FQ4DEM project of the University of Jaén whose objective is to detect unusual elements in images (shadow maps) that facilitate the combination of several shadow maps.

For this, a point detection algorithm has been used, such as the *SIFT* algorithm on different Digital Elevation Models (DEM) of the same image and an attempt has been made to obtain the best possible combination with these results regarding a real solution, called *ground truth* (generated manually as a proof of concept).

To obtain the best combination, since it is not possible to calculate it by brute force due to the large number of data, an optimization algorithm such as the *Uniform Elitist Genetic Algorithm* (AGG\_UN) has been chosen and has tried to obtain the best solution with three different types of combinations: using unions of DEMs, using intersections, and combining unions and intersections of DEMs.



---

Yo, **Víctor Díaz Bustos**, alumno de la titulacion Grado en Ingeniería Informatica de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 80227576B, autorizo la ubicacion de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Víctor Díaz Bustos

Granada a July 3, 2022



# Agradecimientos

Con la entrega de este trabajo de fin de grado pongo punto y final a una intensa etapa en mi vida. 4 duros años de gran trabajo y sacrificio que por fin dan su fruto.

Desde aquí quiero dar las gracias a la Universidad de Granada por todo lo aprendido en estos años, por esa capacidad de esfuerzo, superación, trabajo en equipo y capacidad de aprender también de forma autodidacta que me ha hecho desarrollar en estos 4 años.

Gracias también por cada profesor que siempre estuvo dispuesto a la enseñanza y a resolver dudas, incluso en los complicados tiempos de pandemia, donde todos tuvimos que aprender a trabajar de un modo completamente diferente al que estábamos acostumbrados. En especial, gracias a Rocío, mi tutora de TFG, por su disponibilidad y ayuda a la hora de planificar este proyecto.

Gracias por la oportunidad de poder realizar una prácticas de empresa donde estoy pudiendo tener mi primera experiencia laboral, aprendiendo como se trabaja fuera de la universidad y que, seguro, marcará un antes y un después en el futuro de mi carrera profesional.

Por último, quería agradecer por cada persona que la universidad a puesto en mi camino en estos años, bien sea para quedarse, o bien simplemente estaba de paso, ya que también han contribuido a formar la persona que soy hoy en día.



# Contents

<b>1 INTRODUCCIÓN</b>	<b>18</b>
1.1 Motivación y objetivo . . . . .	18
1.2 Descripción del proyecto . . . . .	18
1.3 Estructura de la memoria . . . . .	18
<b>2 GESTIÓN Y PLANIFICACIÓN</b>	<b>21</b>
2.1 Objetivos . . . . .	21
2.2 Tareas . . . . .	21
2.3 Metodología de desarrollo . . . . .	22
2.3.1 SCRUM . . . . .	23
2.3.2 Aplicación de SCRUM a este proyecto . . . . .	25
2.4 Gestión de la configuración . . . . .	27
2.4.1 Gestión del código . . . . .	27
2.4.2 Gestión de la documentación . . . . .	27
2.5 Gestión del tiempo . . . . .	27
2.5.1 Planificación temporal . . . . .	27
2.5.2 Planificación de los sprints . . . . .	27
2.6 Gestión de riesgos . . . . .	27
2.6.1 Identificación de riesgos . . . . .	28
2.6.2 Riesgos materializados . . . . .	28
2.7 Gestión de recursos . . . . .	28
2.7.1 Recursos humanos . . . . .	28
2.7.2 Recursos materiales . . . . .	29
2.7.3 Recursos software . . . . .	29
2.7.4 Recursos de comunicación y documentación . . . . .	29
2.8 Gestión de costes . . . . .	29
2.8.1 Costes de recursos humanos . . . . .	30
2.8.2 Costes de recursos materiales . . . . .	30
2.8.3 Costes de recursos software . . . . .	31
2.8.4 Costes adicionales . . . . .	31
2.9 Presupuesto total . . . . .	32
<b>3 ESPECIFICACIÓN DE REQUISITOS</b>	<b>34</b>
3.1 Historias de usuario . . . . .	34
3.2 Tarjetas de historias de usuario . . . . .	35
<b>4 ANÁLISIS</b>	<b>38</b>
4.1 Análisis de riesgos . . . . .	38
<b>5 MODELO IMPLEMENTADO</b>	<b>41</b>
5.1 SIFT . . . . .	41
5.1.1 Pirámide de Lowe . . . . .	41
5.1.2 cv2.SIFT_create . . . . .	43

<b>6 COMPARACIÓN DE MODELOS</b>	<b>46</b>
<b>7 COMBINACIÓN Y OPTIMIZACIÓN DE RESULTADOS</b>	<b>52</b>
7.1 Algoritmos Genéticos . . . . .	52
7.1.1 Algoritmo Genético Elitista Uniforme . . . . .	53
7.1.2 Adaptación de AGG.UN a unión de MDEs . . . . .	54
7.1.3 Adaptación de AGG.UN a intersección de MDEs . . . . .	56
7.1.4 Adaptación de AGG.UN a combinación de uniones e intersecciones de MDEs . . . . .	58
<b>8 CONCLUSIONES Y POSIBLES AMPLIACIONES</b>	<b>68</b>
8.1 Conclusiones . . . . .	68
8.2 Posibles ampliaciones . . . . .	69
<b>9 BIBLIOGRAFÍA</b>	<b>71</b>
<b>A MDEs</b>	<b>73</b>
<b>B SOLUCIONES SIFT</b>	<b>83</b>
<b>C MEJORES COMBINACIONES</b>	<b>93</b>



## List of Figures

1	Fases en una metodología ágil . . . . .	22
2	Marco de trabajo SCRUM . . . . .	23
3	Roles de SCRUM . . . . .	24
4	Ciclo de trabajo de un sprint . . . . .	25
5	Pirámide Gaussiana . . . . .	42
6	Escala . . . . .	42
7	Ejemplo de SIFT con $nfeatures = 500$ . . . . .	44
8	Mapa de calor de los distintos MDEs . . . . .	46
9	Verdaderos positivos + falsos positivos . . . . .	48
10	Verdaderos positivos + falsos negativos . . . . .	49
11	Sensibilidad y precision de MDEs . . . . .	50
12	F1 SCORES . . . . .	51
13	Cruce AGG_UN . . . . .	53
14	F1 SCORE - unión . . . . .	55
15	Evaluación puntos - unión . . . . .	56
16	F1 SCORE - intersección . . . . .	57
17	Evaluación puntos - intersección . . . . .	58
18	Ejemplo de árbol binario . . . . .	59
19	Diagrama de clases . . . . .	61
20	Generación de hojas . . . . .	61
21	Generación aleatoria de subramas . . . . .	62
22	Generación de árbol binario aleatorio . . . . .	62
23	Generación de hijos . . . . .	63
24	Ejemplo de mutación de árbol . . . . .	64
25	Mejor árbol obtenido . . . . .	65
26	F1 SCORE - unión e intersección . . . . .	65
27	Evaluación puntos - unión e intersección . . . . .	66
28	MDT05-ETRS89-HU30-0172_Sombreado (imagen original) . . . . .	73
29	Cross-Sectional_Curvature_MDE05 . . . . .	73
30	Downslope_Curvature_DEM05_1 . . . . .	73
31	Gradient_DEM05 . . . . .	74
32	Hill_Index_MDE05 . . . . .	74
33	Horizontal-Overland-Flow-Distance_MDE05 . . . . .	74
34	Longitudinal_Curvature_MDE05 . . . . .	75
35	Mass_Balance_Index_MDE05 . . . . .	75
36	MDE05_surf_texture . . . . .	75
37	Mid-Slope_Positon_DEM05 . . . . .	76
38	Minimum_Curvature_MDE05 . . . . .	76
39	MRRTF_DEM05 . . . . .	76
40	MRVBF_DEM05 . . . . .	77
41	Normalized_Height_DEM05_1 . . . . .	77
42	Overland-Flow-Distance_MDE05 . . . . .	77
43	Profile_Curvature_MDE05 . . . . .	78
44	Protection_Index_DEM05 . . . . .	78

45	Slope_Height_DEM05 . . . . .	78
46	Slope_Height_DEM05_1 . . . . .	79
47	Slope_MDE05 . . . . .	79
48	Terrain_Ruggedness_Index_(TRI)_DEM05 . . . . .	79
49	Terrain_Ruggedness_Index_(TRI)_DEM05_1 . . . . .	80
50	TPI_MDT05_5_100_75 . . . . .	80
51	Upslope_Curvature_DEM05 . . . . .	80
52	Valley_Index_MDE05 . . . . .	81
53	Vertical_Distance_to_Channel_Network_MDE05 . . . . .	81
54	Vertical_Overland_Flow_Distance_MDE05 . . . . .	81
55	Ground_truth-10kp . . . . .	83
56	Cross-Sectional_Curvature_MDE05-SIFT-10kp . . . . .	83
57	Downslope_Curvature_DEM05_1-SIFT-10kp . . . . .	83
58	Gradient_DEM05-SIFT-10kp . . . . .	84
59	Hill_Index_MDE05-SIFT-10kp . . . . .	84
60	Horizontal_Overland_Flow_Distance_MDE05-SIFT-10kp . . . . .	84
61	Longitudinal_Curvature_MDE05-SIFT-10kp . . . . .	85
62	Mass_Balance_Index_MDE05-SIFT-10kp . . . . .	85
63	MDE05_surf_texture-SIFT-10kp . . . . .	85
64	Mid-Slope_Positon_DEM05-SIFT-10kp . . . . .	86
65	Minimum_Curvature_MDE05-SIFT-10kp . . . . .	86
66	MRRTF_DEM05-SIFT-10kp . . . . .	86
67	MRVBF_DEM05-SIFT-10kp . . . . .	87
68	Normalized_Height_DEM05_1-SIFT-10kp . . . . .	87
69	Overland_Flow_Distance_MDE05-SIFT-10kp . . . . .	87
70	Profile_Curvature_MDE05-SIFT-10kp . . . . .	88
71	Protection_Index_DEM05-SIFT-10kp . . . . .	88
72	Slope_Height_DEM05-SIFT-10kp . . . . .	88
73	Slope_Height_DEM05_1-SIFT-10kp . . . . .	89
74	Slope_MDE05-SIFT-10kp . . . . .	89
75	Terrain_Ruggedness_Index_(TRI)_DEM05-SIFT-10kp . . . . .	89
76	Terrain_Ruggedness_Index_(TRI)_DEM05_1 - SIFT - 10kp . . . . .	90
77	TPI_MDT05_5_100_75-SIFT-10kp . . . . .	90
78	Upslope_Curvature_DEM05-SIFT-10kp . . . . .	90
79	Valley_Index_MDE05-SIFT-10kp . . . . .	91
80	Vertical_Distance_to_Channel_Network_MDE05-SIFT-10kp . . . . .	91
81	Vertical_Overland_Flow_Distance_MDE05-SIFT-10kp . . . . .	91
82	Mejor unión . . . . .	93
83	Mejor intersección . . . . .	93
84	Mejor combinación de uniones e intersecciones . . . . .	93



## List of Tables

1	Costes de recursos humanos . . . . .	30
2	Costes de recursos materiales . . . . .	31
3	Costes adicionales . . . . .	32
4	Presupuesto total del proyecto . . . . .	32
5	Documento guía. Especificación de las historias de usuario. . . . .	35
6	Historia de Usuario - HU.1. . . . .	35
7	Historia de Usuario - HU.2. . . . .	35
8	Historia de Usuario - HU.3. . . . .	36
9	Historia de Usuario - HU.4. . . . .	36
10	Historia de Usuario - HU.5. . . . .	36
11	Historia de Usuario - HU.6. . . . .	37
12	Historia de Usuario - HU.7. . . . .	37
13	Riesgos del proyecto . . . . .	39



# 1 INTRODUCCIÓN

Hoy en día, el campo de la inteligencia artificial y la visión por computador está comenzando a vivir una época de auge y gran expansión, de forma que se busca unificar ambas tecnologías para obtener resultados sorprendentes y con multitud de aplicaciones, como vehículos de conducción autónoma, reconocimiento facial, diagnósticos en el campo de la salud, etc.

Un gran futuro está por venir en la integración de estas dos innovadoras tecnologías, pero todo parte de una base: la detección de puntos claves en una imagen.

## 1.1 Motivación y objetivo

Tener claras las bases de algo tan grande como la aplicación de inteligencia artificial en visión por computador es una tarea tan necesaria para su avance como interesante, pues simplemente con esta base se pueden lograr grandes cosas como detectar bordes, esquinas y objetos en una imagen o, incluso, formar panoramas uniendo distintas imágenes de una misma escena.

El principal objetivo de este programa es utilizar estas bases para paliar con algunos de los diferentes problemas que nos podemos encontrar a la hora de digitalizar mapas.

## 1.2 Descripción del proyecto

Este proyecto se centrará en la detección de elementos inusuales en imágenes de mapas que faciliten la combinación de varios mapas de sombras, los cuales cuentan con diferentes tamaños y calidad, haciendo mucho más compleja la tarea de unificarlos.

Para ello, se van a aplicar distintos métodos de extracción de características relevantes de las imágenes que posteriormente pueden usarse en reconocimiento de objetos y detección de movimiento. Finalmente se tratará combinar de estos extractores de puntos, buscando la mejor combinación de puntos, para obtener resultados lo mejor posible.

## 1.3 Estructura de la memoria

1. **Introducción:** una breve descripción del proyecto, donde se explica también la motivación y el principal objetivo de este.
2. **Gestión y planificación:** se enumeran todos los pequeños objetivos y tareas a cumplir y se define la metodología de desarrollo elegida para este proyecto, así como la planificación temporal, gestión de recursos y riesgos y costes del proyecto.
3. **Especificación de requisitos:** descripción de requisitos e historias de usuario del proyecto.

4. **Análisis:** análisis de riesgos.
5. **Modelos implementados:** se detalla la implementación y funcionamiento de los distintos modelos de detección de características de imágenes utilizados.
6. **Comparación de modelos:** en este apartado se comparan los resultados obtenidos por los diferentes modelos descritos en el apartado anterior.
7. **Conclusiones y posibles ampliaciones:** conclusión final del proyecto y posibles ampliaciones y propuestas de mejoras.
8. **Bibliografía:** artículos estudiados y/o citados y guías utilizadas.



## 2 GESTIÓN Y PLANIFICACIÓN

### 2.1 Objetivos

**Objetivo General:** detectar elementos inusuales en imágenes con diferentes métodos de extracción de características en imágenes y comparar sus resultados.

\*Estos mapas cuentan con diferentes tamaños y calidad, haciendo mucho más compleja la tarea de unificarlos.

#### Objetivos específicos

- **OBJ 1** - Implementar modelos de extracción de características en imágenes.
- **OBJ 2** - Obtener los elementos detectados por los diferentes modelos implementados y visualizarlos.
- **OBJ 3** - Comparar los resultados entre los distintos modelos. Para ello, generar un mapa de calor (*heatmap*) comparando los resultados de cada uno.
- **OBJ 4** - Combinar los modelos para obtener los mejores resultados posibles.

### 2.2 Tareas

Tareas alineadas con objetivos. Cada objetivo debe tener una o varias tareas que lo implementen.

- **T1** - Investigación y selección de los diferentes algoritmos de extracción de características en imágenes que podemos utilizar (OBJ 1).
  - **T1.1** - Investigación de lenguajes de programación y librerías que puedan ser útiles.
  - **T1.2** - Investigación de qué modelos vienen ya implementados en las librerías seleccionadas.
  - **T1.3** - Selección de los modelos a utilizar.
- **T2** - Implementación de un modelo básico del algoritmo SIFT (OBJ 1).
  - **T2.1** - Utilizar la librería OpenCV de python para utilizar el algoritmo SIFT.
- **T3** - Aplicar SIFT a distintos MDEs de una misma imagen (OBJ 2).
- **T4** - Visualizar los puntos detectados por SIFT sobre cada MDE de la imagen (OBJ 2).

- **T5** - Crear un mapa de calor para comparar la relación entre los distintos resultados (OBJ 3).
- **T6** - Calcular métricas para valoración de resultados (OBJ 3).
  - **T6.1** - Calcular F1 score.
- **T7** - Implementar Algoritmo Genético para obtener la mejor combinación posible (OBJ 4).

### 2.3 Metodología de desarrollo

Antes de comenzar con el proyecto, es imprescindible plantearse la metodología a utilizar en el transcurso de este, y para ello vamos a definir una **metodología** como el conjunto de métodos que se siguen en una investigación o estudio, estructurando y organizando la forma de trabajar, con el fin de reducir el riesgo de que surjan inconvenientes que retrasen el desarrollo, sobrecostes, o un mal planteamiento inicial que vuelva inservible todo el trabajo realizado.

Así, vamos a diferenciar dos grandes grupos de metodologías:

- **Metodologías tradicionales:** con procesos muy controlados, con numerosas políticas y normas, utilizadas en proyectos de duración media o elevada, con respuesta lenta a cambios, gestión de equipos grandes (generalmente distribuida) y con una curva de aprendizaje media o larga.
- **Metodologías ágiles:** con procesos menos controlados y con mayor flexibilidad, utilizadas en proyectos de corta duración, con respuesta rápida a cambios, gestión de equipos pequeños y con una curva de aprendizaje corta.

Por tanto, para el desarrollo de este proyecto se va a seguir una metodología ágil, ya que este busca construir un modelo básico que sirva para comparar de forma sencilla los resultados de los distintos algoritmos de extracción de características en imágenes previamente mencionados, para lo cual no podemos realizar una definición de requisitos inicial demasiado detallada, debe plantearse la metodología de desarrollo de forma que podamos agregar cambios de manera incremental y evolutiva.



**Figure 1:** Fases en una metodología ágil

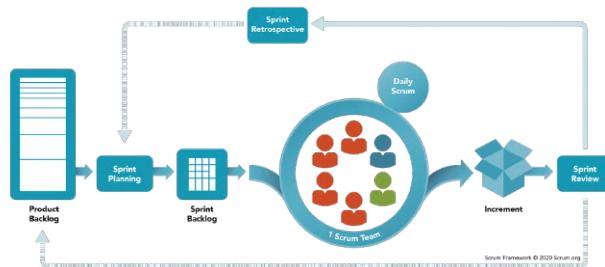
### 2.3.1 SCRUM

Existen diferentes metodologías agiles, como Extreme Programming (XP), Scrum, Crystal Clear, Lean Development, Kanban, etc.

En este caso, se va a utilizar la metodología de Scrum como herramienta de trabajo, ya que es una metodología muy utilizada en el desarrollo de software, permitiendo un desarrollo iterativo e incremental, con el fin de mejorar el producto.

Esta metodología está basada en tres pilares, que son:

1. El primero de ellos es el **equipo**, ya que una de las principales características de Scrum es contar con un equipo multidisciplinario, ya que cada uno de los integrantes del mismo tendrá una función específica y principal, pero a la vez siendo flexibles y adaptándose a las necesidades del proyecto en cada momento.
2. El segundo es el **producto**. Este pilar se divide en dos partes, ya que en la primera se deberá definir el producto a desarrollar, y en la segunda se tendrá que realizar el desarrollo e iteraciones del mismo.
3. Y por último, el tercer pilar de Scrum es el **proceso**. Este se divide en cuatro fases, siendo las dos primeras fases preparatorias, y las dos últimas de desarrollo e iteraciones.



**Figure 2:** Marco de trabajo SCRUM

## Roles

Un equipo Scrum suele tener distintos roles, cada uno con diferentes responsabilidades:

1. **Product Owner:** Es el encargado de optimizar y maximizar el valor del producto. Suele tener una labor de interlocutor con las partes interesadas y los patrocinadores del proyecto, así como de intermediario entre el cliente y el equipo de desarrollo.

2. **Scrum Master:** Es el líder y mayor responsable del equipo. Tiene dos funciones principales: gestionar el proceso Scrum de forma que se siga la metodología y cumplimiento de valores y ayudar a eliminar impedimentos que puedan afectar a la entrega del producto.
3. **Equipo de desarrollo:** Esta formado por un grupo de profesionales que se encargan de desarrollar el producto. Ellos se organizan y gestionan de forma autónoma para conseguir un incremento de software al final del ciclo de desarrollo.

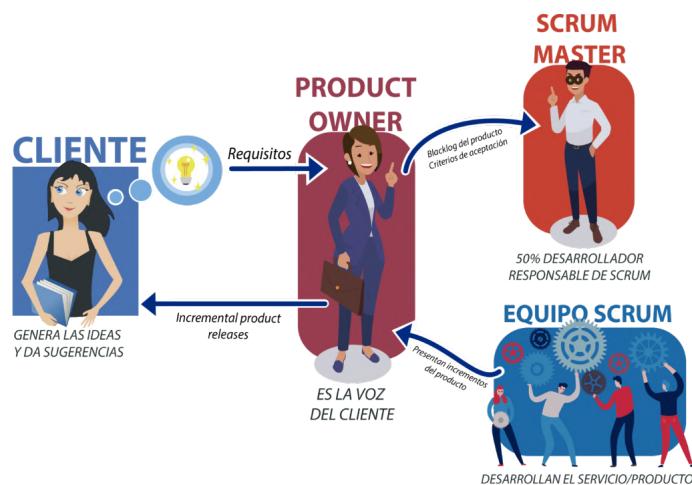


Figure 3: Roles de SCRUM

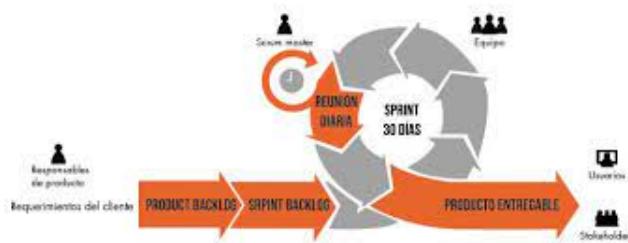
### Ciclo de trabajo

El desarrollo del proyecto se hará a través de los Sprints o diferentes partes en las que éste se dividirá, lo que permitirá abordarlo de forma más rápida y eficiente.

Cada Sprint lleva asociada una serie de fases o etapas que permiten definir que se va a desarrollar, cual va a ser su diseño y un plan flexible que guiará el trabajo y el producto final resultante. Las 5 fases de los Sprints son:

1. **Reunión de planificación (Sprint planning):** Se prevee el trabajo a realizar para sprint determinado. En esta reunión se define el plan para el desarrollo del sprint, así como que va a entregarse al final del mismo.
2. **Scrum Diario (Daily meeting):** Tiene como objetivo evaluar el progreso del sprint y detectar si hay algún retraso en el desarrollo del mismo. Es una reunión diaria de no más de 15 minutos en la que cada miembro del equipo expone la situación de su trabajo y posteriormente se crea un plan de trabajo para ese día.

3. **Trabajo de desarrollo durante el Sprint (Sprint):** Durante el desarrollo de un sprint no se deben realizar cambios que afecten a los objetivos del mismo y no se disminuye los objetivos de calidad. Si es demasiado largo, se puede redefinir para evitar retrasos.
4. **Revisión del Sprint (Sprint review):** Se lleva a cabo al final del sprint con el objetivo de revisar lo que se hizo, detectar problemas y redefinir los items del Product Backlog en su caso.
5. **Retrospectiva del Sprint (Sprint retrospective):** Tiene como objetivo revisar el desarrollo del sprint en lo que respecta a personas, relaciones, procesos y herramientas, de forma que se detecten los puntos a mejorar.



**Figure 4:** Ciclo de trabajo de un sprint

### 2.3.2 Aplicación de SCRUM a este proyecto

A pesar de que SCRUM es una metodología de trabajo enfocada principalmente en proyectos en grupo, se va a adaptar para este proyecto individual. Para ello, en lugar de realizar una reunión diaria con el equipo (*daily meeting*), se va a tener un encuentro cada dos semanas con el tutor asignado para mantener un seguimiento del desarrollo del trabajo y poder resolver las dudas que vayan surgiendo. De esta forma, definimos los *sprints* como períodos de trabajo fijos en los que se tiene que terminar una serie de tareas, según la prioridad.

Además se utilizará la técnica *product backlog*, la cual consiste en una lista de tareas por hacer y su prioridad. En esta lista, se irán añadiendo las tareas que se vayan necesitando para ir haciendo el proyecto. La prioridad de las tareas se irá definiendo, teniendo en cuenta los requisitos, la complejidad y el tiempo necesario para terminar cada tarea.

Así, cada sprint estará formado por 4 etapas:

1. Determinación de los product backlog
2. Documentación
3. Implementación
4. Evaluación

El desarrollo de este proyecto se va a dividir en 3 fases principales:

1. Fase de investigación
2. Fase de diseño
3. Fase de desarrollo

A continuación se detallan las actividades que se van a llevar a cabo en cada una de las fases:

**Fase de investigación:**

1. Selección del tema
2. Investigación bibliográfica sobre el tema seleccionado
3. Definición de objetivos
4. Identificación de herramientas y frameworks necesarios para desarrollar el proyecto
5. Diseño de una arquitectura de software básica

**Fase de diseño:**

1. Diseño de un modelo de datos adecuado
2. Diseño de diferentes modelos de extracción de características en imágenes

**Fase de desarrollo:**

1. Implementación de las diferentes partes del proyecto
2. Validación de resultados
3. Evaluación de resultados

La validación de resultados se realizará de la mano del tutor, donde se estudiarán los resultados obtenidos para verificar que son correctos.

La evaluación de este proyecto se va a realizar mediante un informe escrito, una presentación y una defensa oral. El informe debe describir el proceso seguido para el desarrollo del proyecto, así como los resultados obtenidos. La presentación será un resumen de lo contenido en el informe, mientras que la defensa oral servirá para explicar el proyecto y resolver cualquier duda que pudiera surgir sobre él.

## **2.4 Gestión de la configuración**

En esta sección se define la gestión de la configuración bajo la cual se desarrollarán todas las actividades que impliquen la creación, modificación o eliminación de alguno de los elementos de trabajo que conforman este proyecto, asegurando así que todo lo que concierne al proyecto sea válido durante la vida del mismo.

Teniendo en cuenta que los elementos de trabajo de este proyecto son los ficheros de código fuente y todos los archivos que conforman la documentación, abordaremos esta sección especificando, de forma independiente, cómo se ha tratado cada tipo de elemento de trabajo.

### **2.4.1 Gestión del código**

Para la gestión del código se creara un repositorio local con un sistema de control de versiones, que a su vez se sincronizara con un repositorio remoto en *GitHub*, y por lo tanto almacenado en la nube. GitHub es un servicio web de control de versiones y desarrollo de software basado en Git, publicado bajo una Licencia de código abierto. Esta copia remota previene el riesgo de pérdida del código fuente.

Dicho código será implementado en el lenguaje python, en concreto, utilizando un *cuaderno jupyter*, de forma que se pueda ejecutar secciones concretas del programa, sin tener que ejecutar todo el código completo. Esto será muy útil, ya que en análisis de imágenes, los tiempos de cómputo pueden llegar a ser muy elevados, y ejecutar un programa completo cuando solo interesa una parte de este puede llegar a ser una tarea muy tediosa.

### **2.4.2 Gestión de la documentación**

Para almacenar la documentación de este proyecto se utilizara un repositorio en *OverLeaf*, un editor colaborativo de *LaTeX* basado en la nube que se utiliza para escribir, editar y publicar documentos. Este dispone de control de versiones y, al estar en almacenado en la nube, se previene el riesgo de pérdida.

## **2.5 Gestión del tiempo**

### **2.5.1 Planificación temporal**

### **2.5.2 Planificación de los sprints**

## **2.6 Gestión de riesgos**

Un riesgo de un proyecto es un evento o condición que podría tener un efecto positivo o negativo sobre alguno de los recursos u objetivos del proyecto, como podría ser tiempo, coste, alcance o calidad. La gestión de riesgos es un proceso continuo que debe considerar tanto fuentes internas como externas de dichos riesgos. Por norma general, la materialización de un riesgo implica consecuencias negativas, por lo que la gestión de riesgos tiene un papel muy importante dentro de la gestión del proyecto.

En esta sección se identificaran los posibles riesgos del proyecto, estimando la probabilidad de aparición y su impacto potencial. Posteriormente se definirán las estrategias y pasos a seguir para prevenirlos o minimizar su impacto en caso de que se materialicen.

#### 2.6.1 Identificación de riesgos

- **R.01:** No identificar correctamente los requisitos del proyecto.
- **R.02:** Aparición de nuevos requisitos.
- **R.03:** No conseguir cumplir con la planificación.
- **R.04:** Cambios en la planificación.
- **R.05:** Errores eligiendo las tecnologías para el desarrollo del proyecto.
- **R.06:** Problemas con diferentes versiones de las tecnologías utilizadas.
- **R.07:** Deficiente disposición insuficiente disponibilidad del experto.
- **R.08:** Detección de fallos importantes en el software tras la validación.
- **R.09:** Pérdida de parte de proyecto (documentación o código).
- **R.10:** Fallos imprevistos en el ordenador.
- **R.11:** Problemas de formato entre las distintas tecnologías utilizadas.

#### 2.6.2 Riesgos materializados

- **R.01:** No identificar correctamente los requisitos del proyecto.
- **R.04:** Cambios en la planificación.
- **R.05:** Errores eligiendo las tecnologías para el desarrollo del proyecto.
- **R.06:** Problemas con diferentes versiones de las tecnologías utilizadas.
- **R.08:** Detección de fallos importantes en el software tras la validación.

### 2.7 Gestión de recursos

#### 2.7.1 Recursos humanos

- Dna. Rocío Celeste Romero Zaliz, profesora del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, en calidad de tutora del proyecto presente.
- Víctor Díaz Bustos, alumno en el grado de Ingeniería Informática en la Escuela Técnica Superior de Ingenierías Informática y Telecomunicación.

### 2.7.2 Recursos materiales

A continuación se listan los recursos materiales o hardware empleados para la realización del proyecto:

- **Ordenador portátil:** HP con procesador Intel Core i5 8th Gen, arquitectura de 64 bits, con 8 GB de memoria RAM sobre el que se programara y documentara todo el proyecto.
- **Monitor:** Acer 19 pulgadas LCD, 60Hz, para ampliar la visibilidad a la hora de hacer tareas concurrentemente.

### 2.7.3 Recursos software

Con el objetivo de reducir los costes el proyecto, se tratará de utilizar, en la medida de lo posible, programas y herramientas de software libre. A continuación se han un listado de los recursos software utilizados en el proyecto:

- **Sistema Operativo:** Se utilizará Ubuntu Linux, concretamente la versión 20.04 LTS.
- **Jupyter Notebook:** entorno informático gratuito interactivo para crear documentos de Jupyter Notebook. Un documento de Jupyter Notebook es un documento JSON, que sigue un esquema versionado y que contiene una lista ordenada de celdas de entrada/salida que pueden contener código, texto (usando Markdown), matemáticas, gráficos y texto enriquecidos, generalmente terminado con la extensión ".ipynb".

### 2.7.4 Recursos de comunicación y documentación

- **Google Meet:** plataforma para la comunicación vía vídeo, voz y texto a través de Internet de manera gratuita.
- **Git:** sistema de control de versiones de código abierto.
- **Github:** plataforma para el alojamiento y control de versiones del repositorio del proyecto.
- **Overleaf :** herramienta online para la creación, edición y revisión colaborativa de documentos en LaTeX.

## 2.8 Gestión de costes

En esta sección se hará una estimación de los costes del proyecto teniendo en cuenta todos los recursos anteriormente citados (humanos, materiales, software y de comunicación/documentación), además de unos costes indirectos o adicionales.

### 2.8.1 Costes de recursos humanos

En primer lugar tendremos en cuenta los costes asociados con el equipo de desarrollo. Dado que este equipo esta formado únicamente por una persona, se considerara el rol de Analista Programador.

Este proyecto se ha planteado de manera que su desarrollo dure un total de 5 meses (de febrero a junio, ambos incluidos). Esto da un total de 22 semanas.

Una jornada laboral habitual tiene una duración de 40 horas semanales, pero, debido a que este proyecto se desarrollará en paralelo a unas prácticas de empresa a media jornada (25 horas semanales) y al estudio de 2 asignaturas, el número de horas dedicado a este proyecto se verá reducido a unas 15 horas semanales para hacer un cálculo más realista. Por tanto tenemos un total de unas 330 horas de trabajo.

El precio de cada hora asociado al trabajo autonomo por parte del alumno, se ha calculado de acuerdo al salario de un ingeniero informático recién egresado, que suele rondar los 22.000€ anuales según la Universidad Europea. Si dividimos entre 12 meses que tiene un año, una media de 20 días laborales al mes y una jornada completa de 8 horas diarias, el precio hora resultantes es de 11,5€/hora.

El cálculo del coste por recursos humanos final de acuerdo a los precios hora establecidos se puede ver en la Tabla 1

Concepto	€/hora	NºHoras	Importe (€)
Trabajo autónomo	11,50	330	3795,00
<b>Total</b>			<b>3795,00</b>

Table 1: Costes de recursos humanos

### 2.8.2 Costes de recursos materiales

Para calcular el precio de los recursos materiales, tendremos que tener en cuenta que el precio del ordenador portátil con el que se ha desarrollado el proyecto, así como el del monitor, los cuáles van devaluándose con el paso del tiempo, debido a lo que se conoce como *depreciación del producto*.

En el caso del portátil, este fue adquirido por un precio de unos 600€. Suponemos que la vida útil de un portátil es de 5 años, por lo que tiene un *coeficiente de amortización lineal* igual al 20% de su valor, que es el máximo permitido para equipos electrónicos según la agencia Agencia Tributaria de España. Además, suponemos un *valor residual* de 50€, esto es, el valor que tiene al final de su vida útil.

Así, el **coste de amortización** viene dado por la siguiente fórmula:

$$(P_c - V_r)Ca$$

Donde  $P_c$  es el precio de compra del producto,  $V_r$  es el valor residual y  $C_a$  es el coeficiente lineal de amortización.

Por lo tanto, aplicando la fórmula, nos queda que el valor del portátil en un año es de  $(600\text{€}-50\text{€}) \cdot 0.2 = 110\text{€}$ .

Ahora se debe estimar la proporción de horas de uso del portátil durante el proyecto respecto al número de horas de uso anuales (5 horas diarias, durante 5 días a la semana, las 52 semanas que contiene un año) para obtener el coste de dicho uso.  $110\text{€} \cdot (330 / (5 \cdot 5 \cdot 52)) = 27,92\text{€}$

Se realiza el mismo cálculo para el monitor, el cuál tiene un precio de compra de unos 90€, un valor residual de 15€ y un coeficiente lineal de amortización del 15%, ya que se estima una durabilidad superior a los 5 años, con el mismo número de horas de uso que el portátil. Así se obtiene que el precio valor del monitor en un año es de  $(90\text{€}-15\text{€}) \cdot 0,15 = 11,25\text{€}$  y el coste del uso de este es de  $11,25\text{€} \cdot (330 / (5 \cdot 5 \cdot 52)) = 2,86\text{€}$

El coste total por recursos materiales tras la depreciación de los mismos se puede ver en la Tabla 2

Recurso	Coste individual (€)	Cantidad	Importe (€)
Portátil	600,00	1	27,92
Monitor	90,00	1	2,86
<b>Total</b>			<b>30,78</b>

**Table 2:** Costes de recursos materiales

### 2.8.3 Costes de recursos software

En ese caso, todas las herramientas de software utilizadas, tanto el sistema operativo como los programas para el desarrollo del proyecto y la documentación y comunicación con la tutora, han sido herramientas gratuitas, por lo que no correrán gastos por el uso de recursos software.

### 2.8.4 Costes adicionales

Se considerarán costes adicionales o indirectos aquellos gastos derivados del proyecto que son necesarios para el desarrollo de este como: las facturas de internet y luz.

En el caso del autor de este proyecto, éste vive en un piso de estudiantes junto a 3 compañeros de piso, por lo que se tendrán en cuenta únicamente una cuarta parte de dichas facturas, generando un gasto promedio de 6,25€/mes en el caso de internet y de 21,67€/mes en el caso de la luz. Por lo que el coste total en 5 meses será de 31,25€ para internet y de 108,34€ para la luz.

El coste total por dichos recursos adicionales se puede ver en la Tabla 3

Recurso	€/mes	Nºmeses	Importe (€)
Internet	6,25	5	31,25
Luz	21,67	5	108,34
<b>Total</b>			139,59

**Table 3:** Costes adicionales

## 2.9 Presupuesto total

El presupuesto total estimado del proyecto queda reflejado y desglosado como muestra la Tabla 4

Concepto	Importe (€)
<b>Recursos humanos:</b> Trabajo autónomo	3795,00
<b>Recursos materiales</b>	
Portátil	27,92
Monitor	2,86
<b>Recursos software:</b>	
Sistema operativo	0,00
Jupyter Notebook	0,00
Google Meet	0,00
Github	0,00
Overleaf	0,00
<b>Costes adicionales:</b>	
Internet	31,25
Luz	108,34
<b>Total:</b>	3965,37

**Table 4:** Presupuesto total del proyecto



## 3 ESPECIFICACIÓN DE REQUISITOS

Los requisitos se pueden definir como aquellas características esperables y/o deseables que debe cumplir un sistema para satisfacer unas necesidades específicas. Esta tarea es imprescindible para asegurar el éxito del proyecto.

En esta sección se describirán los requisitos del proyecto desde una perspectiva menos tradicional que encaja mejor con la metodología de desarrollo elegida: *historias de usuario*.

### 3.1 Historias de usuario

Las historias de usuario describen de forma breve aquello que un determinado usuario (en este caso desarrollador) espera al final del desarrollo. Es necesario que cumplan una serie de características:

- Deben ser **independientes**.
- Deben ser **negociables**. No actúan como un contrato, sino como una serie de pautas de acción.
- Deben aportar **valor** al proyecto.
- Se debe poder realizar una **estimación** sobre ellas (temporal o de carga de trabajo).
- Deben tener un **tamaño manejable**.
- Deben ser **verificables**, pues es necesario saber si una historia se ha cumplido.

Cada historia se expresará siguiendo el siguiente patrón:

Como <*ROL*>, quiero/espero <*OBJETIVO*> para  
<*FUNCIONALIDAD*>.

Además a cada una de ellas se le acompañará de un nivel de **prioridad** (alta(1), media(2) o baja(3)), una estimación de **tiempo** requerido para su desarrollo y un **identificador** de historia de usuario.

Se definen los 3 niveles de prioridad como:

- **Alta** (Cumplimiento obligatorio): la historia es imprescindible para el desarrollo y éxito del proyecto. Es necesario que esté en la versión final para aceptar el proyecto.
- **Media** (Cumplimiento deseable): es deseable la presencia de esta historia en la versión final del proyecto, pero no supone un aumento significativo del valor del producto.
- **Baja** (Cumplimiento opcional): su presencia no es crucial para el éxito del proyecto, pero aporta cierto valor funcional.

A continuación se especificaran todas las historias de usuario del proyecto teniendo en cuenta el documento guía de la Tabla 5 para la especificación de historias de usuario.

<b>Identificador</b>	Código - Nombre
<b>Descripción</b>	Breve descripción de la historia de usuario.
<b>Estimación</b>	Estimación temporal de la historia.
<b>Prioridad</b>	Prioridad según la importancia que tiene esta historia para el desarrollo satisfactorio del proyecto.
<b>Aceptación</b>	Criterios a cumplir para considerar que esta historia de usuario se ha cubierto satisfactoriamente.
<b>Notas</b>	Observaciones varias.

**Table 5:** Documento guía. Especificación de las historias de usuario.

### 3.2 Tarjetas de historias de usuario

A continuación vamos a detallar cada una de las historias de usuario planificadas para el desarrollo del proyecto siguiendo la estructura indicada en la Tabla 5.

<b>Identificador</b>	HU.1 - Añadir/Eliminar MDEs a la imagen
<b>Descripción</b>	Como desarrollador, quiero poder añadir o eliminar MDEs de la imagen para poder tener una mayor variedad de resultados.
<b>Estimación</b>	1 semana
<b>Prioridad</b>	Alta
<b>Aceptación</b>	Cuando el desarrollador añada o elimine una imagen del MDE deseado a la carpeta ' <i>imagenes/filtros/</i> ', el programa lo detectará automáticamente
<b>Notas</b>	

**Table 6:** Historia de Usuario - HU.1.

<b>Identificador</b>	HU.2 - Crear heatmap
<b>Descripción</b>	Como desarrollador, quiero poder crear un heatmap para poder ver la relación entre resultados.
<b>Estimación</b>	2 semanas
<b>Prioridad</b>	Alta
<b>Aceptación</b>	Tras obtener los keypoints de cada MDE, se debe obtener un heatmap que muestre la relación entre estos
<b>Notas</b>	

**Table 7:** Historia de Usuario - HU.2.

<b>Identificador</b>	HU.3 - Calcular F1 score
<b>Descripción</b>	Como desarrollador, quiero poder calcular el F1 score para poder valorar cada resultado.
<b>Estimación</b>	1 semana
<b>Prioridad</b>	Alta
<b>Aceptación</b>	Tras obtener los keypoints de cada MDE, se debe calcular el F1 score de cada uno de estos
<b>Notas</b>	

**Table 8:** Historia de Usuario - HU.3.

<b>Identificador</b>	HU.4 - Identificar MDEs que dan resultados muy similares
<b>Descripción</b>	Como desarrollador, quiero poder identificar MDEs que dan resultados muy similares, para las cuales puedes quedarte con solo una de ellas y reducir la complejidad de todo el sistema.
<b>Estimación</b>	1 semana
<b>Prioridad</b>	Media
<b>Aceptación</b>	Tras obtener el heatmap, eliminar aquellos MDEs que presenten una relación muy alta
<b>Notas</b>	

**Table 9:** Historia de Usuario - HU.4.

<b>Identificador</b>	HU.5 - Obtener la combinación óptima de MDEs
<b>Descripción</b>	Como desarrollador, quiero obtener la combinación óptima de MDEs para tener el mejor detector posible.
<b>Estimación</b>	3 semanas
<b>Prioridad</b>	Alta
<b>Aceptación</b>	Tras obtener los resultados y F1 scores, se aplicarán algoritmos genéticos que combinen MDEs para obtener la mejor combinación.
<b>Notas</b>	

**Table 10:** Historia de Usuario - HU.5.

<b>Identificador</b>	HU.6 - Visualización de gráficas
<b>Descripción</b>	Como desarrollador, quiero poder visualizar gráficas de los resultados obtenidos por SIFT así como del rendimiento del algoritmo genético para comprender mejor los resultados.
<b>Estimación</b>	1 semana
<b>Prioridad</b>	Baja
<b>Aceptación</b>	Tras obtener los resultados, F1 scores y mejor combinación, mostrar gráficas para visualizar y comprender mejor los resultados.
<b>Notas</b>	

**Table 11:** Historia de Usuario - HU.6.

<b>Identificador</b>	HU.7 - Añadir/modificar ground truth
<b>Descripción</b>	Como desarrollador, quiero poder añadir o modificar el ground truth para poder comparar los modelos con distintas imágenes.
<b>Estimación</b>	1 semana
<b>Prioridad</b>	Media
<b>Aceptación</b>	Tras pasar imagen y una lista con los keypoints que componen el ground truth, este se tomará como la verdad a tomar como referencia.
<b>Notas</b>	

**Table 12:** Historia de Usuario - HU.7.

## **4 ANÁLISIS**

### **4.1 Análisis de riesgos**

En este apartado se pretende analizar los riesgos identificados en el apartado 2.6.1, y que pueden aparecer y afectar a los objetivos del proyecto que se está realizando.

Con el objetivo de actuar de forma rápida ante la aparición de alguno de éstos, a continuación se enumerarán los posibles riesgos y a identificar su causa y el efecto que pueden producir. Además, se definirá una estrategia o plan de actuación para minimizar el impacto de éste.

A continuación se presentan en la Tabla 13 los riesgos considerados, siguiendo el esquema *Riesgo-Causa-Actuación*.

ID	Riesgo	Causa	Plan de actuación
R.01	No identificar correctamente los requisitos del proyecto	Desconocimiento o inexperience en el sector	Aumento de tiempo para estudiar
R.02	Aparición de nuevos requisitos	Análisis pobre o planificación incorrecta	Replanificar tareas y actualizar backlog
R.03	No conseguir cumplir con la planificación	Mala estimación temporal o trabajo poco productivo	Replanificar tareas y actualizar backlog
R.04	Cambios en la planificación	Planificación demasiado ambiciosa	Incremento de tiempo para replanificar
R.05	Errores eligiendo las tecnologías para el desarrollo del proyecto	Pobre estudio previo	Selección de nuevas tecnologías y replanificación de tareas
R.06	Problemas con diferentes versiones de las tecnologías utilizadas	Versiones incompatibles o funciones no compatibles con una misma versión	Crear entornos virtuales con distintas versiones
R.07	Deficiente disposición insuficiente disponibilidad del experto	Enfermedad, problemas técnicos, pérdidas de conexión o caídas de servidores	Aumentar trabajo autónomo
R.08	Detección de fallos importantes en el software tras la validación	Comprensión errónea de algoritmos y revisión pobre de resultados	Replanificar tareas y resolver fallos
R.09	Pérdida de parte de proyecto (documentación o código)	Fallo hardware o robo del portátil	Utilizar herramientas de sincronización en la nube
R.10	Fallos imprevistos en el ordenador	Fallos hardware imprevisibles	Reajuste de estimaciones y avanzar con estudio e investigación en caso de contar con algún dispositivo alternativo
R.11	Problemas de formato entre las distintas tecnologías utilizadas	Distintos algoritmos dan resultados con diferentes formatos	Formatear los resultados de los algoritmos

**Table 13:** Riesgos del proyecto



## 5 MODELO IMPLEMENTADO

### 5.1 SIFT

El algoritmo SIFT (Scale Invariant Feature Transform) permite detectar puntos característicos en una imagen y luego describirlos a través de un histograma orientado de gradientes. Este método es invariante a la orientación, posición y escala de la imagen.

Para obtener el conjunto de descriptores SIFT de una imagen es necesario obtener los puntos característicos y, posteriormente, para cada keypoint, calcular su vector descriptor a partir de la información de los píxeles de su alrededor, por lo que se diferencian dos fases del algoritmo: **detector** (obtención de puntos característicos) y **descriptor** (descripción de la región alrededor de cada punto característico).

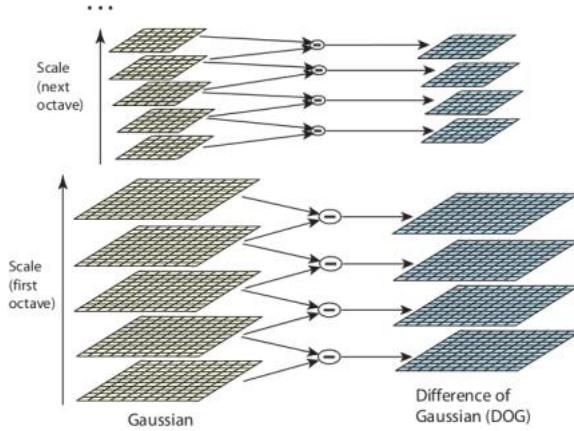
Esta segunda fase consiste en detectar aquellas regiones de la imagen en las que se producen diferencias de gradiente significativas a ambos lados de dicho punto y, de esta forma, se consiguen detectar esquinas. Sin embargo, SIFT va más allá y aplica la pirámide de Lowe, la cual permite detectar blobs (manchas) mediante la detección de puntos característicos en diferentes escalas. La detección de esos puntos característicos a diferentes escalas requiere crear un espacio de escalas, detectar en él puntos de interés, y eliminar aquellos que se consideren poco estables.

#### 5.1.1 Pirámide de Lowe

SIFT aplica la pirámide de Lowe para detectar blobs mediante la detección de puntos característicos en diferentes escalas. Para ello se crean varios espacios de escala reduciendo sucesivamente el tamaño de la imagen, recibiendo cada espacio de escalas el nombre de *octava*. Dicho proceso recibe el nombre de **obtención de puntos característicos** a partir de los extremos del espacio de escalas generado a partir de diferencias de Gaussianas (DoG)<sup>1</sup> dentro de una pirámide de diferencia de Gaussianas.

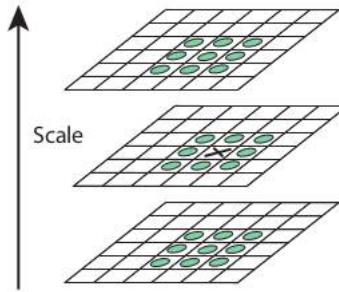
---

<sup>1</sup>Diferencia de Gaussianas: procesamiento aplicado a una imagen que consiste en restar una versión borrosa de la imagen original a otra versión de esta misma imagen menos borrosa. Dichas imágenes borrosas se obtienen mediante la convolución de la imagen original.



**Figure 5:** Pirámide Gaussiana

Dentro de cada octava se generan imágenes progresivamente más suavizadas mediante la *convolución*<sup>2</sup> con una gaussiana con un  $\sigma$  cada vez mayor. Este sigma es calculado como  $\sigma_s = \sigma_0 * \sqrt{2^{\frac{2s}{n_s}} - 2^{\frac{2(s-1)}{n_s}}}$ , donde  $\sigma_0$  es el sigma inicial,  $\sigma_s$  es el sigma a aplicar en cada escala,  $s$  es la escala actual y  $n_s$  es el número de escalas en la octava (normalmente 3, ya que se ha demostrado que más no aporta ninguna información extra), de manera que el  $\sigma$  al final de la octava es 2 veces el  $\sigma$  al comienzo de esta. Así, al reducir la imagen a la mitad para la siguiente octava volvemos a tener el mismo sigma inicial.



**Figure 6:** Escala

Para detectar los máximos y mínimos locales, cada punto de la muestra se compara con los 8 puntos de su alrededor de la imagen en la que se encuentra y con los 9 vecinos de la imagen que se encuentra una escala superior e inferior. Se comprueba si el punto del centro es un máximo o un mínimo local y, en dicho caso, se guarda como un extremo. Finalmente se devuelven los n mejores extremos detectados.

<sup>2</sup>Convolución: operador matemático que transforma dos funciones  $f$  y  $g$  en una tercera función que en cierto sentido representa la magnitud en la que se superponen  $f$  y una versión trasladada e invertida de  $g$ .

### 5.1.2 cv2.SIFT\_create

La librería OpenCV de python nos ofrece el método *SIFT\_create()* que nos permite crear un objeto extractor de keypoints mediante el algoritmo SIFT. Dicho método contiene los siguientes parámetros:

- **nfeatures** (int=0): número de mejores puntos característicos a obtener.
- **nOctaveLayers** (int=3): número de capas en cada octava. 3 es el valor utilizado en el paper de D. Lowe. El número de octavas se calcula automáticamente a partir de la resolución de la imagen.
- **contrastThreshold** (double=0.04): umbral de contraste utilizado para filtrar características débiles en regiones semiuniformes (de bajo contraste). Cuanto mayor sea el umbral, menos características producirá el detector.
- **edgeThreshold** (double=10): umbral utilizado para filtrar entidades similares a bordes. Cuanto mayor sea el umbral, menos características se filtran (se retienen más características).
- **sigma** (double=1.6): sigma de la Gaussiana aplicada a la imagen de entrada en la octava 0. Si la imagen se captura con una cámara débil con lentes suaves, es posible que se desee reducir el número.

Para el desarrollo de este proyecto, se van a mantener todos los parámetros por su valor predeterminado, salvo nfeatures que sí será necesario darle un valor superior a 0. Esto se hará así ya que el objetivo no es refinrar al máximo este algoritmo para un caso concreto, sino ver como se comporta en distintas situaciones.

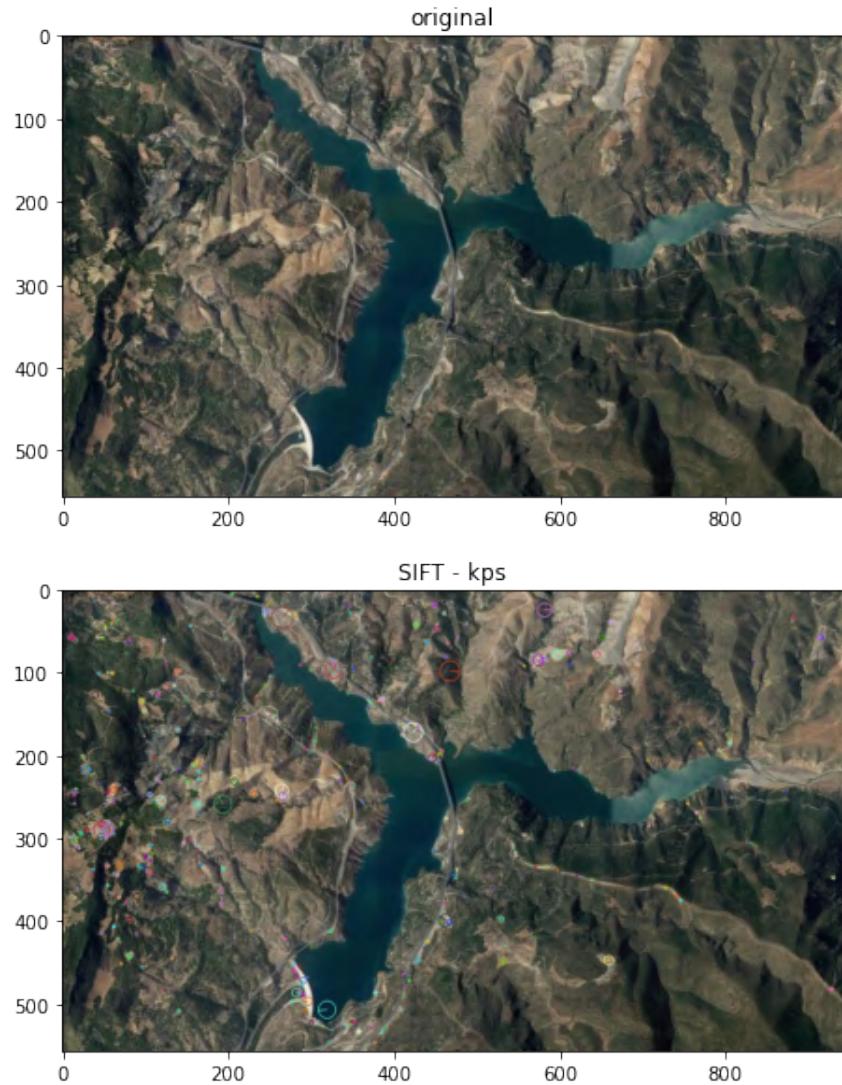
Posteriormente, se utilizará el método *detectAndCompute(imagen, máscara)* del objeto creado anteriormente con SIFT\_create para encontrar los keypoints de la imagen. Se puede pasar una máscara si desea buscar solo una parte de la imagen, aunque en este caso no se le pasará ninguna. Cada keypoint es una estructura especial que tiene los siguientes atributos atributos:

- **pt**: coordenadas (x,y) del punto
- **size**: diámetro del punto
- **angle**: orientación del punto
- **response**: fuerza del punto, es decir, cómo de importante es dicho punto para el descriptor.
- **octave**: octava de la pirámide en la que se ha detectado el punto.
- **class\_id**: identificación del objeto

Dicho método devolverá una lista de los  $n_{features}$  mejores keypoints detectados.

Finalmente, para visualizar los puntos obtenidos, basta con utilizar la función `drawKeypoints(imagen_entrada, keypoints, imagen_salida)` de OpenCV. Dicha imagen pinta los keypoints obtenidos sobre la imagen de entrada, guardándolo en la imagen de salida

Haciendo uso de la librería matplotlib, se muestra el siguiente ejemplo:



**Figure 7:** Ejemplo de SIFT con  $n_{features} = 500$



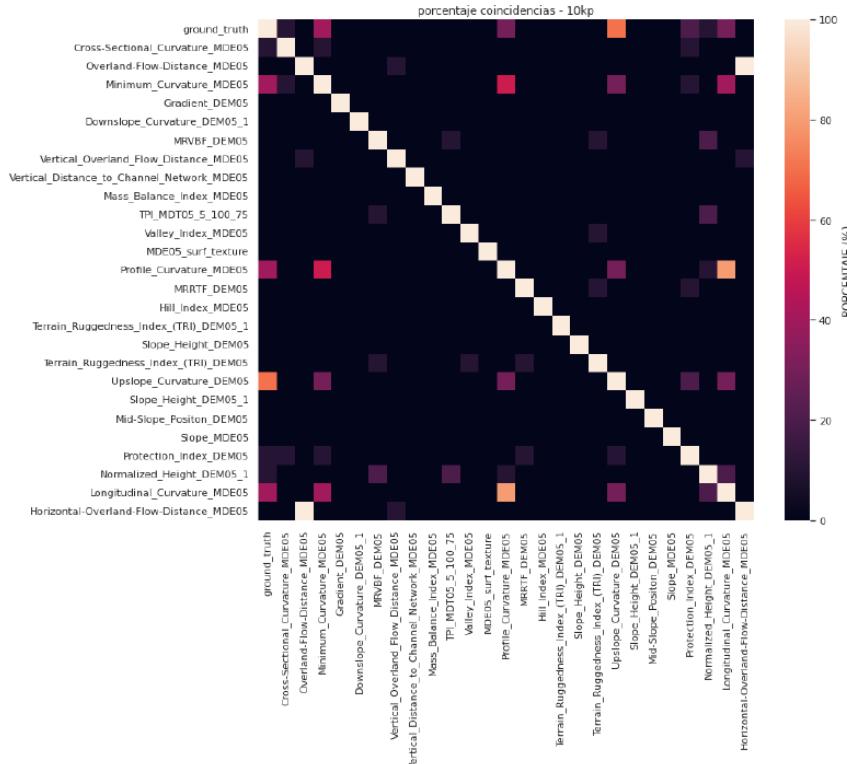
## 6 COMPARACIÓN DE MODELOS

Para este estudio, se ha trabajado en colaboración con el proyecto **FQ4DEM** de la Universidad de Jaén, a través del cuál se han obtenido una serie de filtros u operaciones sobre **Modelos Digitales de Elevación (MDE)** de una misma imagen (Apéndice A).

A continuación se aplica el algoritmo SIFT a los distintos MDEs sobre una misma imagen, donde se obtendrán los 10 puntos más significativos de cada MDE, con el objetivo de ver qué puntos se detecta en cada uno de ellos y poder compararlos, viendo la correlación que existe entre estos, así como su similitud al *ground truth* (solución real de puntos más destacados de la imagen original, figura 55), generado manualmente como prueba de concepto.

Para ello, una vez obtenidos los 10 mejores keypoints de cada MDE (Apéndice B), se va a ver cuántos puntos coinciden entre cada par de soluciones, es decir, cuán grande es la intersección de puntos obtenidos por cada par de MDEs, incluyendo además la intersección con el *ground truth*.

Para su visualización se genera el siguiente mapa de calor o *heatmap*, el cual representa el porcentaje de zonas en común que considera importante cada par de MDEs.



**Figure 8:** Mapa de calor de los distintos MDEs

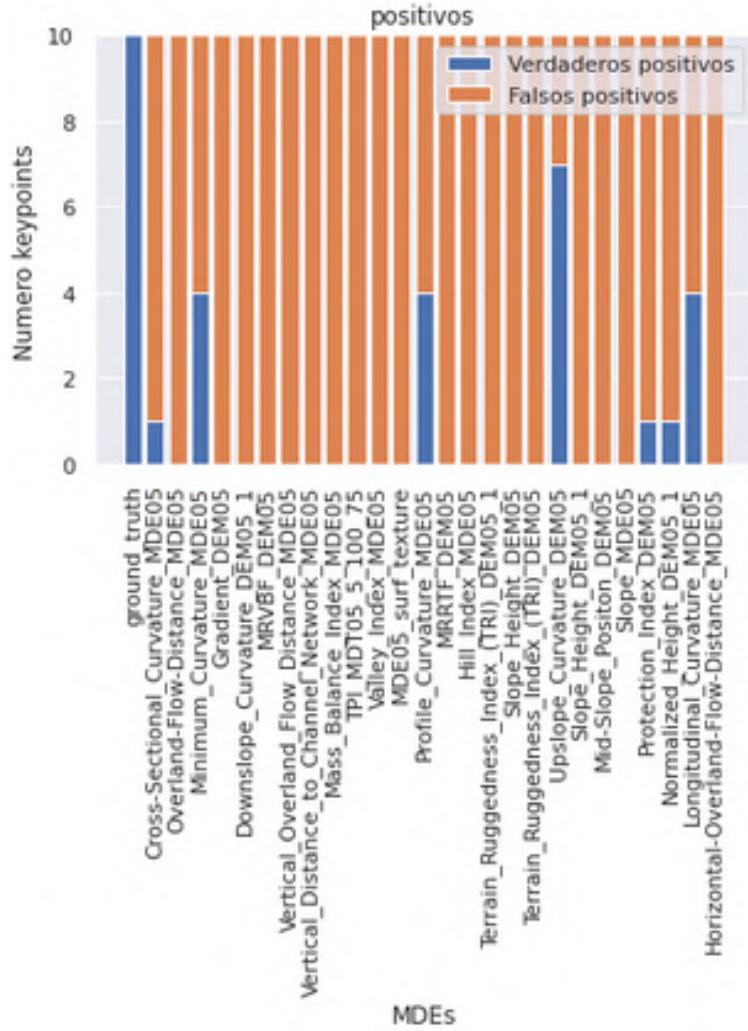
Observando este mapa de calor (figura 8) se pueden obtener dos conclusiones:

1. Los resultados obtenidos son por lo general muy independientes entre sí, por lo que no se puede reemplazar ninguno de ellos por otro, es decir, no hay información redundante.
2. Ninguno de los MDEs tiene una alta correlación con el ground truth, por lo que, según este ground truth, ningún MDE es lo suficientemente preciso.

Posteriormente, se procede a evaluar el resultado de cada MDE. Para lo cuál, se debe definir en primer lugar las siguientes métricas:

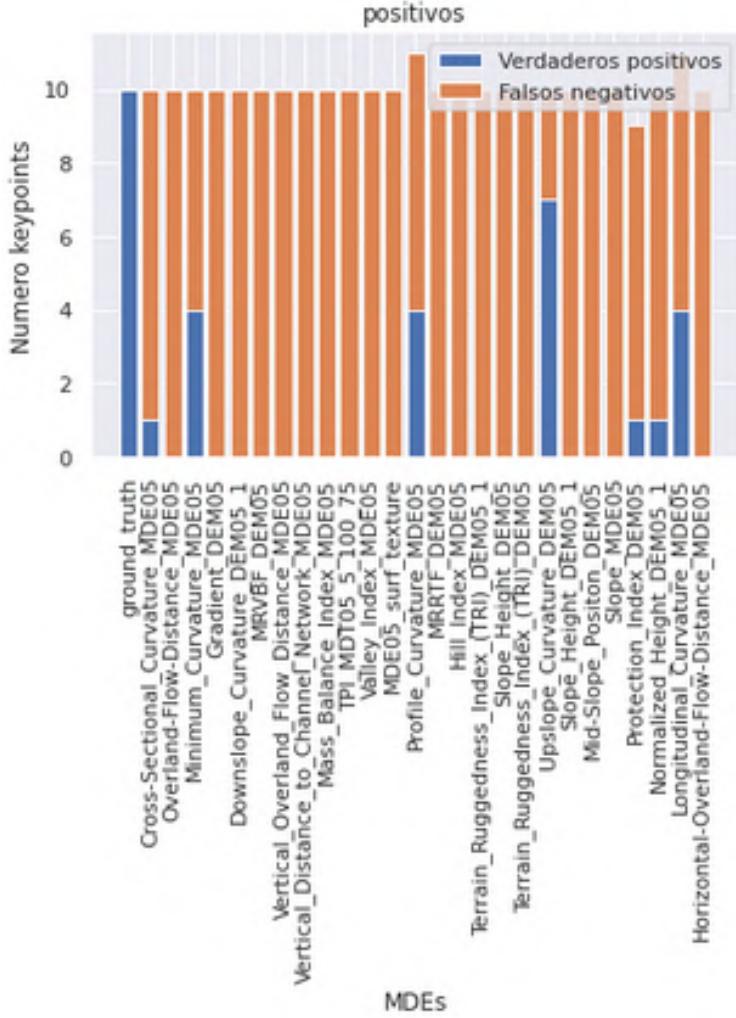
- **Verdaderos Positivos (VP):** puntos detectados por el MDE que están en ground truth. Se calculará de la siguiente forma: cada keypoint de un MDE se va comparando con cada keypoint del ground truth, si este coincide, se aumenta en 1 el número de VP y se dejan de buscar coincidencias para ese keypoint del MDE.
- **Falsos Positivos (FP):** puntos detectados por el MDE que NO están en ground truth. Se calculará de la siguiente forma: cada keypoint de un MDE se va comparando con cada keypoint del ground truth, si tras compararlo con todos no se ha encontrado ninguna coincidencia, se aumenta en 1 el número de FP.
- **Falsos Negativos (FN):** puntos del ground truth que NO han sido detectados por el MDE. Se calculará de la siguiente forma: cada keypoint del ground truth se va comparando con cada keypoint del MDE, si tras compararlo con todos no se ha encontrado ninguna coincidencia, se aumenta en 1 el número de FN.
- **Sensibilidad:** También se conoce como Tasa de Verdaderos Positivos (True Positive Rate) ó TP. Es la proporción de casos positivos que fueron correctamente identificados por el algoritmo. Se calcula como:  $\frac{VP}{VP+FN}$ .
- **Precision:** Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión. Se representa por la proporción de verdaderos positivos dividido entre todos los resultados positivos:  $\frac{VP}{VP+FP}$ .
- **F1 SCORE:** Resume la precisión y sensibilidad en una sola métrica. Será la métrica utilizada para evaluar los resultados obtenidos por cada MDE. Se calcula como:  $\frac{2*Precision*Sensibilidad}{Precision+Sensibilidad}$

A continuación se muestran los resultados obtenidos para cada métrica por cada MDE respecto del ground truth.



**Figure 9:** Verdaderos positivos + falsos positivos

En esta gráfica se puede apreciar que, para cada MDE, el número de verdaderos positivos más el número de falsos positivos es igual al número de puntos que componen el ground truth. Esto nos sirve para verificar que el cálculo es correcto y que no se ha quedado ningún punto del MDE sin ser valorado como positivo o negativo, ya que todas las soluciones están formadas por el mismo número de keypoints (10).



**Figure 10:** Verdaderos positivos + falsos negativos

Aquí se puede observar cómo no en todos los MDEs el número de verdaderos positivos mas el número de falsos negativos es igual al número de puntos que componen el ground truth. Esto se debe a que, partiendo que todas las soluciones están formadas por el mismo número de keypoints, se puede dar el caso de que dos o más keypoints de un MDE coincidan en posición con un mismo keypoint del ground truth, contando esto como varios verdaderos positivos, cuando realmente lo que hay es redundancia, mientras, el número de falsos negativos no disminuye, motivo por el que la suma da un resultado superior (p.ej. Profile\_Curvature\_MDE05, figura 70). Por otro lado está el caso contrario, un keypoint del MDE coincide con dos o más keypoints del ground truth, lo cual tan solo aumentará en 1 el número de verdaderos positivos (es únicamente un

punto del MDE el que se considera válido), ya que, como se ha explicado anteriormente, cuando se detecta una coincidencia para un keypoints de un MDE, se deja de buscar coincidencias para dicho keypoint. Por su lado, el número de falsos negativos si que disminuirá, ya que a la hora de iterar sobre los keypoints de ground truth, estos nos distinguirán entre los keypoints del MDE, detectando esas coincidencias que antes se obviaron, y disminuyendo así el número de falsos positivos. Por tanto, la suma de estos dará inferior al número de keypoints que componen la solución (p.ej. Protection\_Index\_DEM05, figura 71).

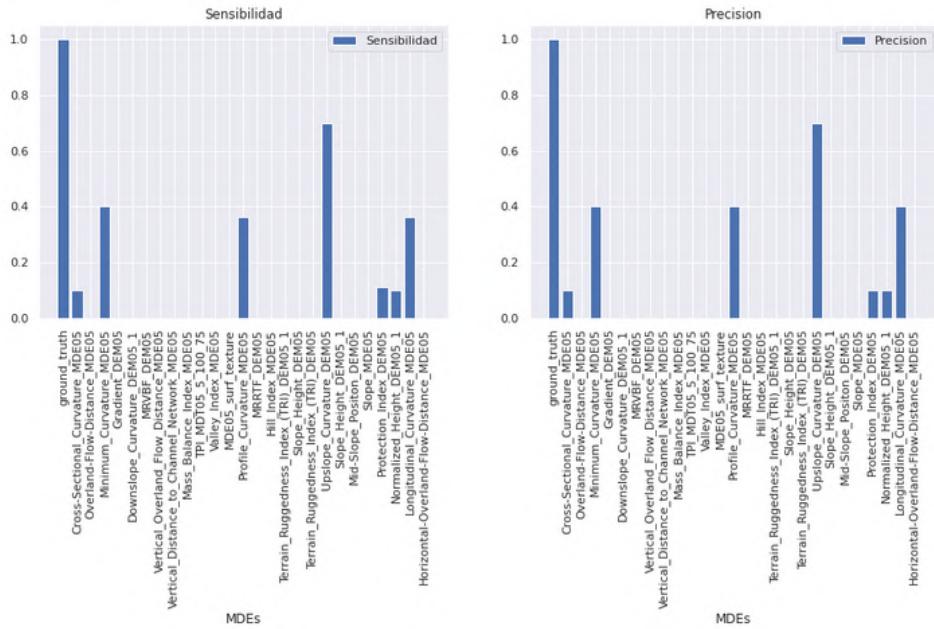
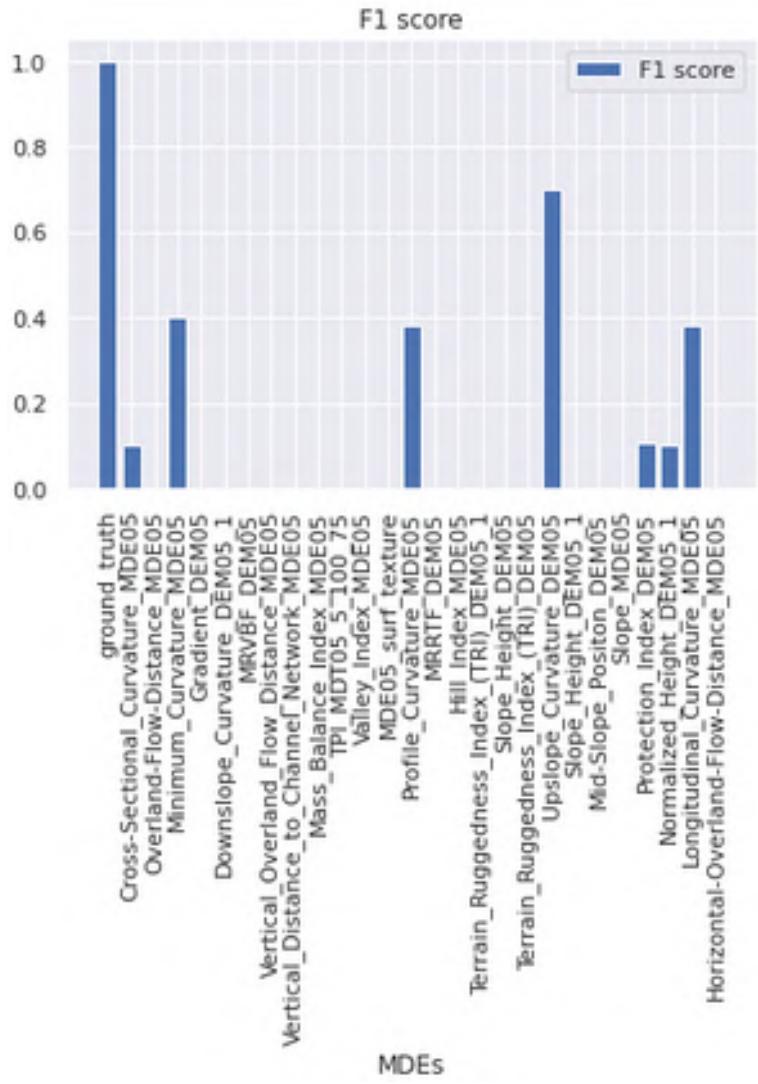


Figure 11: Sensibilidad y precision de MDEs

En este caso sensibilidad y precisión están altamente correlacionados. Esto se debe a que el número de falsos positivos y de falsos negativos son muy similares en todos los casos, ya que, como se ha explicado anteriormente, en la mayoría de casos la suma del número de verdaderos positivos mas el número de falsos negativos es igual al número de puntos que componen la solución, por lo que el número de falsos negativos es equivalente al número de falsos positivos. En los MDEs en los que esta propiedad no se cumplía, el desvío era mínimo, por lo que, incluso en estos casos, sensibilidad y precisión son muy similares.

Como consecuencia de esto, en el cálculo del F1 score se obtendrán resultados muy similares a estos, teniendo como MDE mejor valorado a Upslope\_Curvature-DEM05 (figura 78), tal y como se puede apreciar en la figura 12.



**Figure 12:** F1 SCORES

## 7 COMBINACIÓN Y OPTIMIZACIÓN DE RESULTADOS

Dado que ningún MDE es lo suficientemente preciso como para ser autosuficiente por sí mismo, se va a plantear la opción de combinar MDEs, de forma que podamos tener como solución, bien la unión, o bien la intersección, de dos o más de estos MDEs.

Obtener la mejor combinación posible por fuerza bruta sería altamente costoso, ya que, como se cuenta con 26 posibles MDEs, el número total de posibles combinaciones sería igual a las posibles combinaciones con 2 MDEs más posibles combinaciones con 3 MDEs más ... más posibles combinaciones con 26 MDEs, esto es igual a  $26 * 25 + 26 * 25 * 24 + \dots + 26! = \frac{26!}{(26-2)!} + \frac{26!}{(26-3)!} + \dots + 26! = \sum_{i=2}^{26} \frac{26!}{(26-i)!} \simeq 1,096 * 10^{27}$  posibles combinaciones.

Esto provoca que haya que recurrir a algún algoritmo de optimización para tratar de obtener la mejor combinación posible. Para ello, se ha optado por implementar un Algoritmo Genético Elitista Uniforme.

### 7.1 Algoritmos Genéticos

Los Algoritmos Genéticos (AG) consisten en, a partir de una población inicial aleatoria de cromosomas, generar nuevas generaciones que sustituyan a la anterior mediante la selección de unos padres, el cruce de estos, las mutaciones y el reemplazo de una generación. En función de cómo se lleve a cabo este proceso se pueden distinguir cuatro tipos de AGs: Elitista Uniforme (AGG\_UN), Elitista con Segmento Fijo (AGG\_SF), Estacionario Uniforme (AGE\_UN) y Estacionario con Segmento Fijo (AGE\_SF). Para este proyecto se implementará un AGG\_UN.

Lo más costoso en tiempo de ejecución de un Algoritmo Genético es la generación de números aleatorios para la selección, emparejamiento de padres, decidir si una pareja cruza y decidir si cada gen muta.

Para ello vamos a implementar una serie de alternativas. A la hora de emparejar las parejas de padres, se va a realizar un emparejamiento fijo, emparejando el primero con el segundo, el tercero con el cuarto, etc. Para decidir si una pareja de padres cruza, se ha estimado a priori el número de cruces esperados en cada generación como  $N^{\text{o}}\text{cruces} = P_c \frac{M}{2}$ , donde  $P_c$  es la probabilidad de cruce y  $M$  es el número de cromosomas de la población, y se cruzarán las primeras parejas de la población, donde cada pareja de padres generará 2 hijos, comprobando siempre que ambos son válidos.

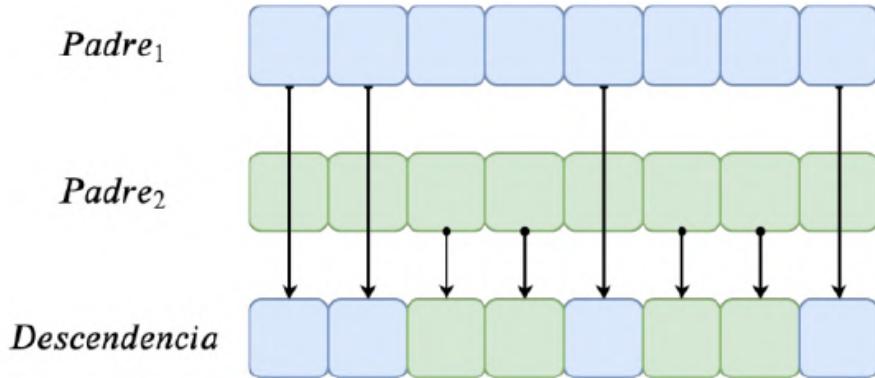
Para decidir si cada gen muta, será similar al caso anterior, se ha calculado a priori el número de mutaciones esperadas en cada generación como  $N^{\text{o}}\text{mutaciones} = P_m * N^{\text{o}}\text{genes población}$ , donde  $P_m$  es la probabilidad de mutación y el  $N^{\text{o}}\text{genes}$  de la población es igual al  $N^{\text{o}}\text{genes}$  de cada cromosoma \* el número de cromosomas de la población. Posteriormente, se generan números aleatorios para ver qué cromosomas mutan y otros para ver qué gen de cada cromosoma modificar.

### 7.1.1 Algoritmo Genético Elitista Uniforme

La principal diferencia entre un AG elitista y un AG estacionario es que el elitista siempre mantiene al mejor cromosoma de cada generación en la siguiente y reemplaza al resto, mientras que el estacionario únicamente cruza dos padres y genera dos hijos que reemplazaran a los dos peores cromosomas de la generación anterior, en caso de ser mejores que estos.

El AGG-UN consiste en seleccionar una población intermedia del mismo tamaño que la generación anterior, mediante un torneo binario, donde en cada emparejamiento se queda con el cromosoma con mejor fitness, pudiendo aparecer el mismo cromosoma en diferentes emparejamientos y, por ende, pudiendo ser padre varias veces en una misma generación.

Una vez seleccionados los padres, cruzaran los “Nºcruces” primeros, con una  $P_c = 0,7$ . Los dos hijos generados por cada pareja sustituirán a dichos padres en la población intermedia, por lo que esta contendrá a los padres que no han sido seleccionados para el cruce y a los hijos que han sustituido a los padres que sí han cruzado. Para ello, al ser uniforme, para cada gen se selecciona de forma aleatoria el parente del que procederá dicho gen, tal y como se refleja en la siguiente imagen.



**Figure 13:** Cruce AGG-UN

Una vez finalizado el cruce, generados todos los hijos y sustituidos los padres, se procede a realizar las mutaciones, con una  $P_m = 0,2$ , donde en cada generación mutarán 4 cromosomas de la población intermedia. Para ello se seleccionarán 4 cromosomas al azar y, posteriormente, seleccionamos también de forma aleatoria qué gen muta y qué nuevo valor se le da a dicho gen, comprobando que el cromosoma resultante es válido.

Llegados a este punto tan solo queda reemplazar la generación anterior por la generación intermedia. Para ello, se realiza una asignación donde se iguala la generación anterior a la intermedia y, ya que es elitista, se sustituye un cromosoma cualquiera (en este caso será siempre el último), por el mejor de la generación anterior. Una vez se tiene la nueva generación, se procede a evaluarla, buscando además cuál es el nuevo mejor cromosoma de la nueva generación.

Tras el reemplazo, se repite el mismo proceso tantas veces como poblaciones se deseen generar, en este caso, dado que el algoritmo converge muy rápido, con 100 generaciones será más que de sobra.

### Pseudocódigo

```
1: Generar población inicial aleatoriamente
2: Evaluar población y almacenar mejor cromosoma
3: for 1:num_generaciones do
4:   Seleccionar padres
5:   Generar hijos
6:   Mutar cromosomas
7:   Reemplazar población manteniendo mejor cromosoma
8:   Evaluar población y almacenar mejor cromosoma
9: end for
10: Devolver mejor cromosoma
```

#### 7.1.2 Adaptación de AGG\_UN a unión de MDEs

Para la combinación de MDEs mediante la unión de estos se va a definir una posible combinación o cromosoma de la población como un conjunto de 2 o mas MDEs, sin repetir, y un conjunto de keypoints. Así, para el algoritmo genético, la población inicial estará formada por 50 conjuntos de 2 o mas MDEs aleatorios, siempre sin repetir ninguno dentro de un mismo conjunto.

Con el AGG\_UN se tratará de obtener la mejor combinación posible. Así, a la hora de evaluar cada cromosoma, se obtendrán todos los keypoints detectados por cada MDE perteniente a dicho cromosoma, y se calculará su F1 score, tal y como se hizo en el apartado 6.

Para generar cada hijo, este estará formado por un número aleatorio de MDEs,  $x$ . Dicho número estará comprendido entre los dos números de MDEs que componen cada parente. Posteriormente, se creará un conjunto de MDEs formado por los MDEs de cada parente, y, de dicho conjunto, se seleccionarán los  $x$  MDEs aleatorios que contendrá el hijo, de forma que este únicamente podrá contener MDEs pertenecientes a sus padres. Por ejemplo: supongamos dos padres, uno con 4 MDEs (1,2,3,4) y otro con 7 MDEs (3,4,5,7,8,9,10). El hijo estará por un número aleatorio de entre 4 y 7 métricas, por ejemplo 5. Para evitar tener MDEs repetidos, se meten todos en un conjunto,

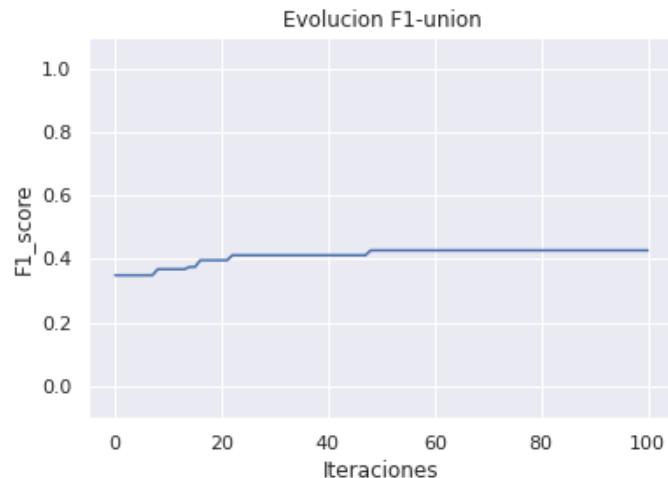
de forma que no estén repetidos, en este caso (1,2,3,4,5,7,8,9,10). Así, el hijo estará formado por 5 MDEs aleatorios de dicho conjunto, por ejemplo (2,3,5,7,8).

A la hora de mutar un cromosoma, bastará con reemplazar uno de los MDEs que contiene, por uno nuevo que no contenga.

## Resultados obtenidos

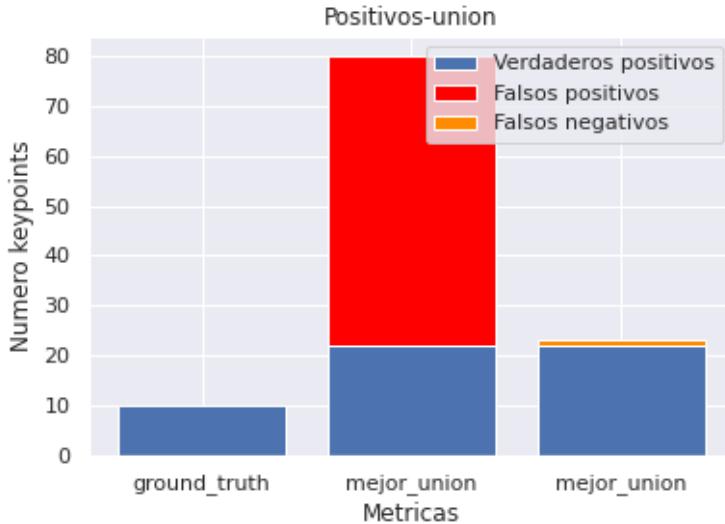
Mejor combinación obtenida (figura 82):

Profile\_Curvature\_MDE05 (figura 70)  $\cup$   
 Protection\_Index\_DEM05 (figura 71)  $\cup$   
 Cross-Sectional\_Curvature\_MDE05 (figura 56)  $\cup$   
 Longitudinal\_Curvature\_MDE05 (figura 61)  $\cup$   
 Upslope\_Curvature\_DEM05 (figura 78)  $\cup$   
 Minimum\_Curvature\_MDE05 (figura 65)  $\cup$   
 Mid-Slope\_Positon\_DEM05 (figura 64)  $\cup$   
 Normalized\_Height\_DEM05\_1 (figura 68)



**Figure 14:** F1 SCORE - unión

Como se puede apreciar, el resultado obtenido no es muy bueno. Esto se debe a que, como los MDEs por sí mismo no son muy precisos, al unir varios de estos el número de falsos positivos aumenta considerablemente, mientras que el de verdaderos positivos no lo hace tanto.



**Figure 15:** Evaluación puntos - unión

En esta gráfica se puede ver como, efectivamente, el número de falsos positivos aumenta en gran medida. Sin embargo, el número de falsos negativos no lo hace tanto. Esto se debe a que, a través de la unión, el conjunto de keypoints obtenido es muy grande y, por tanto, es más probable que algún punto coincida con otro keypoint del ground truth. Además, el número de verdaderos positivos es superior al número de keypoints del ground truth, debido a que, con un conjunto tan grande de puntos, hay distintos keypoints de la solución que coinciden con un mismo keypoint del ground truth.

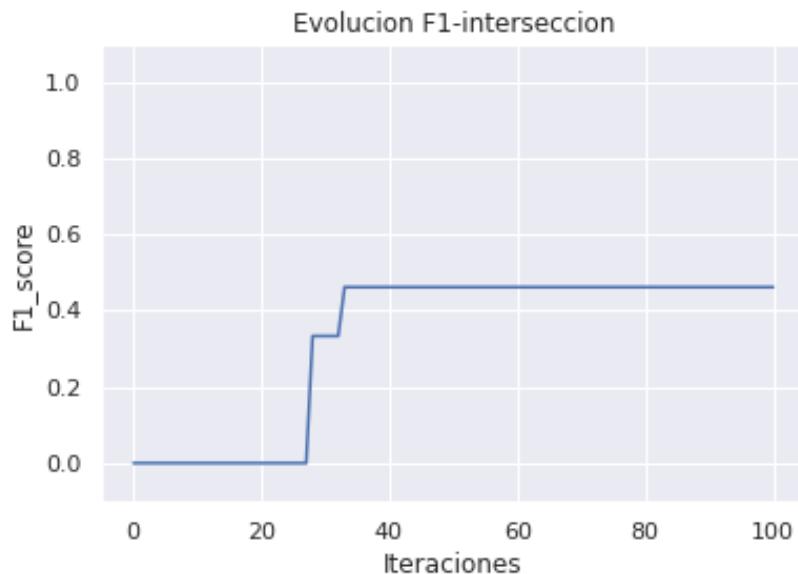
### 7.1.3 Adaptación de AGG\_UN a intersección de MDEs

Para la combinación de MDEs mediante la intersección de estos, el proceso es idéntico al caso anterior, con la única diferencia de que, a la hora de obtener los keypoints que componen el cromosoma, en lugar de estar compuesto por todos los keypoints de todos los MDEs, este estará compuesto únicamente por aquellos keypoints que sean detectados por todos los MDEs que forman el cromosoma, de forma que este conjunto será mucho menor, evitando además que la solución esté formada por un gran número de MDEs, ya que en ese caso, es mucho menos probable que un keypoint sea detectado por todos.

## Resultados obtenidos

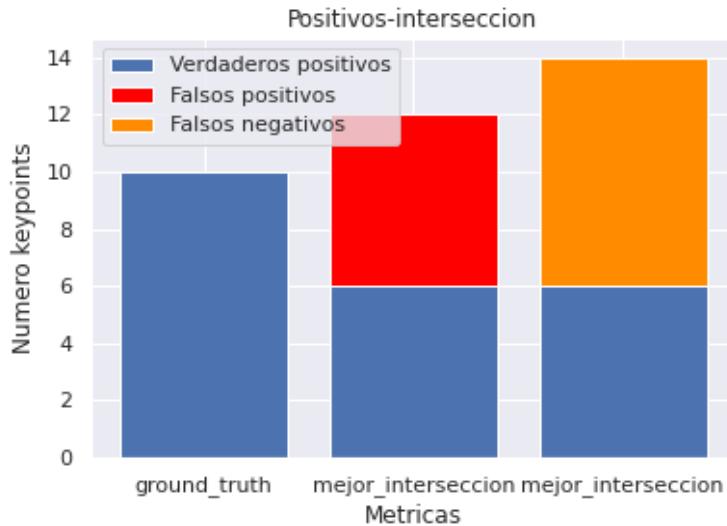
Mejor combinación obtenida (figura 83):

Profile\_Curvature\_MDE05 (figura 70)  $\cap$   
Longitudinal\_Curvature\_MDE05 (figura 61)  $\cap$   
Minimum\_Curvature\_MDE05 (figura 65)



**Figure 16:** F1 SCORE - intersección

En este caso, la solución tampoco da un resultado mucho mejor al del caso anterior (la unión de MDEs). Esto se debe a que, tal y como se vio en la figura 8, la correlación entre los distintos MDEs es por lo general muy baja o nula. Esto provoca que la intersección de los distintos MDEs de como resultado un conjunto de keypoints muy pequeño, haciendo que aumente en gran medida el número de falsos negativos.



**Figure 17:** Evaluación puntos - intersección

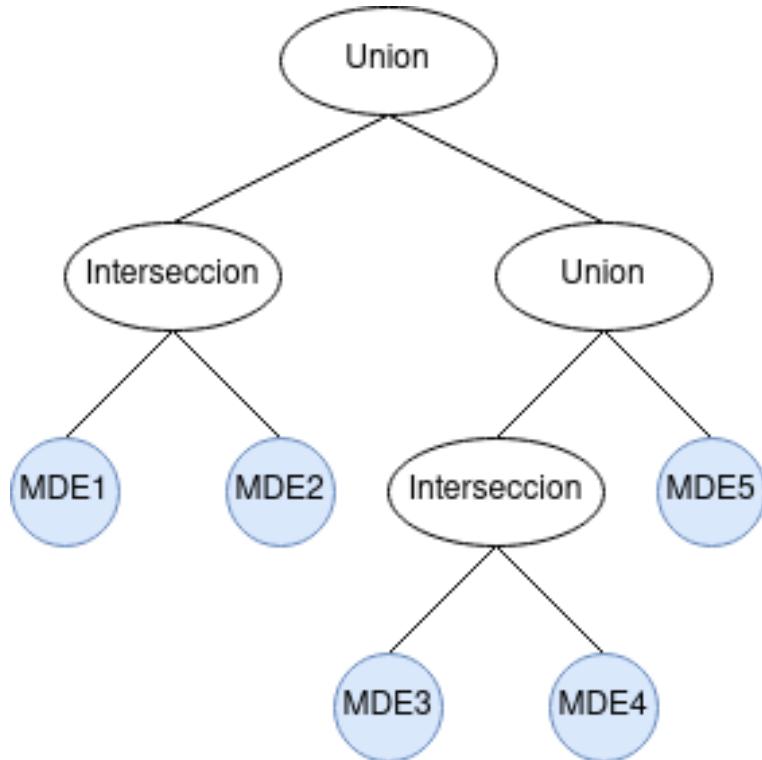
En esta gráfica se aprecia que, efectivamente, el número de falsos positivos aumenta considerablemente. No obstante, el número de verdaderos y de falsos positivos no aumenta tanto. Esto se debe a que, como se acaba de mencionar, el conjunto de keypoints solución es muy pequeño y, por tanto, quedan muchos keypoints del ground truth por detectar. Solo en el caso de que los MDEs fuesen muy precisos, sería posible tener buenos resultados con tan pocos puntos, ya que, con menos falsos positivos, el número de verdaderos positivos aumentaría y, si estos no son redundantes, el número de falsos negativos disminuiría.

#### 7.1.4 Adaptación de AGG\_UN a combinación de uniones e intersecciones de MDEs

Dado que ninguno de los dos casos ha logrado alcanzar resultados lo suficientemente buenos, se va a probar a obtener soluciones que combinen tanto uniones como intersecciones de MDEs, lo cual complica mucho el problema, ya que, no solo habría que elegir qué acción se realiza entre dos MDEs (union o intersección), sino también el orden en el que se realiza dicha acción, ya que el resultado final se verá altamente afectado por dicho orden.

Para ello, se utilizará una estructura de **árbol binario**, donde cada nodo es una acción y cada hoja es el MDE sobre el que se

aplica la acción del nodo superior a este, de forma que cada acción se realizará sobre bien dos nodos, bien dos hojas, o bien un nodo y una hoja.



**Figure 18:** Ejemplo de árbol binario

Para su implementación, se han diseñado dos clases: Nodo y Hoja.

### Clase Nodo

La clase **Nodo** representará a todo nodo del árbol, desde la raíz, que no sea una hoja. Este está formado por un nodo padre (que será nulo en el caso de la raíz), dos nodos hijos ( $n_1$  y  $n_2$ ) y una acción entre sus dos nodos hijos (unión o intersección).

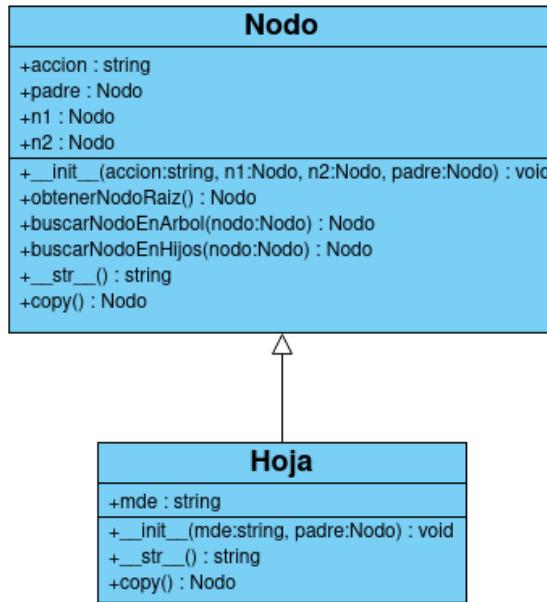
Además, se han implementado los siguientes métodos de la clase:

- `__init__(accion:string, n1:Nodo, n2:Nodo, padre:Nodo)`: es el constructor de la clase, recibe el valor de los distintos atributos y los inicializa.

- **obtenerNodoRaiz()**: devuelve el nodo raíz del árbol. Para ello, partiendo del nodo actual, va ascendiendo por los padres de los distintos nodos hasta dar con un nodo cuyo padre sea nulo.
- **buscarNodoEnArbol(nodo:Nodo)**: busca si un nodo está en el árbol. Devuelve dicho nodo en caso de encontrarlo y un valor nulo en caso contrario. Para ello, el método asciende al nodo raíz y, posteriormente, busca recursivamente por las ramas inferiores hasta, bien encontrar el nodo, o bien haber recorrido todos los nodos del árbol.
- **buscarNodoEnHijos(nodo:Nodo)**: es el método utilizado por el método buscarNodoEnArbol para recorrer las distintas ramas y subramas que divergen del nodo actual. Devuelve el nodo pasado como parámetro en caso de haber sido encontrado y un valor nulo en caso de que no queden más nodos por explorar y este no haya sido encontrado.
- **\_\_str\_\_()**: método para dar formato de *string* al objeto de cara a poder ser mostrado por un output.
- **copy()**: método que devuelve una copia del nodo actual. Crea un nuevo nodo con los mismos atributos, pero con un identificador diferente.

## Clase Hoja

La clase **Hoja** representará únicamente las hojas del árbol. Es una clase **heredera** de la clase **Nodo**, de forma que esta posee los mismos atributos (aunque manteniendo *n1* y *n2* a nulo) y los mismos métodos, con la debida sobrecarga de los métodos **\_\_str\_\_()**, **\_\_init\_\_(mde:string, parent:Nodo)** y **copy()**. Además, esta clase posee un atributo *mde* de tipo *string*, que contiene el nombre del MDE que representa la hoja.



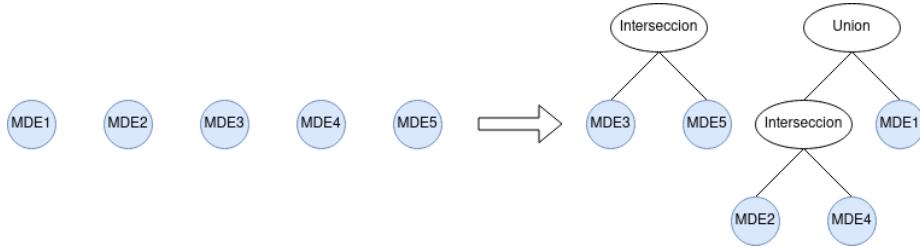
**Figure 19:** Diagrama de clases

Una vez definidas las estructuras de datos a utilizar, se adaptan los distintos pasos del algoritmo genético a este árbol.

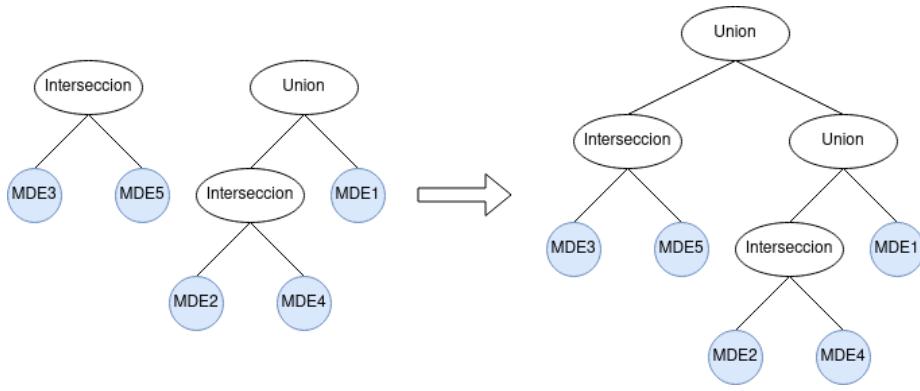
Para ello, comenzando por la generación de la población aleatoria inicial, se ha implementado una función para generar árboles binarios aleatorios, a la cual se le pasa como parámetro el conjunto de MDEs que tiene el árbol como hojas hojas, el cual se genera aleatoriamente para cada árbol inicial. Así, conociendo los MDEs que lo componen, la función crea una hoja para cada MDE y posteriormente comienza a crear nodos, emparejando aleatoriamente las hojas que quedan libres con, bien otra hoja libre, o bien un nodo ya generado, con una acción aleatoria entre 'union' e 'intersección', permitiendo así generar subramas con distintas profundidades. Una vez están todas las hojas adheridas a alguna rama, se emparejan aleatoriamente las raíces de las distintas subramas generadas, hasta que estén todas unidas en un único árbol.



**Figure 20:** Generación de hojas



**Figure 21:** Generación aleatoria de subramas

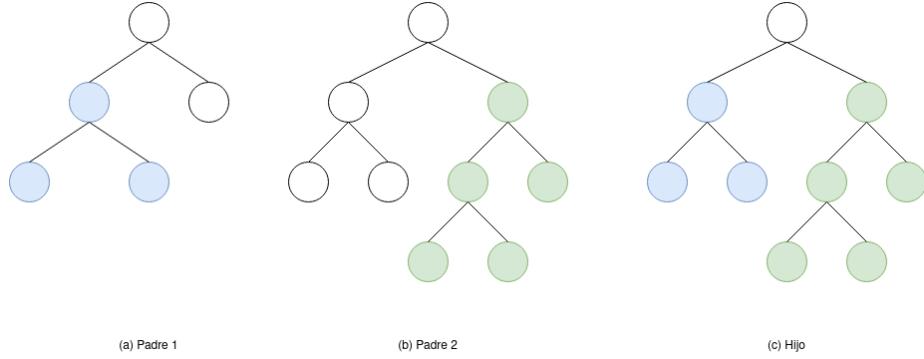


**Figure 22:** Generación de árbol binario aleatorio

Una vez generada la población inicial de  $M$  árboles binarios aleatorios, se procede a evaluarla. Para su evaluación, en primer lugar se obtiene el conjunto de keypoints resultante del árbol. Este se obtiene a través de una función recursiva, la cual tiene como parámetros un nodo y el conjunto de keypoints detectados por cada MDE. Así, si el nodo pasado como parámetro es un nodo hoja, la función devolverá los puntos del conjunto asociado a su MDE. En caso contrario, para cada hijo,  $n_1$  y  $n_2$ , se pueden dar dos casos: que el hijo sea un nodo, o que sea una hoja. Si el hijo es una hoja se obtiene el conjunto de keypoints asociados a su MDE. Si es un nodo no hoja, se llamará recursivamente a esta función hasta llegar a sus nodo hoja. Finalmente, una vez obtenidos los keypoints de cada uno de sus hijos, se aplica la acción que contenta el nodo (unión o intersección). Cuando se ha logrado obtener los keypoints del árbol, este pasa a evaluar su F1 score de la misma manera que se hizo en el apartado 6.

El siguiente paso es iterar sobre tantas generaciones de árboles como se deseé. La selección de padres se realiza de la misma manera que en los apartados 7.1.2 y 7.1.3, mediante un torneo binario.

Para la generación de hijos, el proceso es bastante más complejo, pues tomar los MDEs de los dos padres y generar un conjunto aleatorio, como en los dos problemas de optimización anteriores, es inviable, ya que, para ello haría que generar un nuevo árbol de forma aleatoria a partir de este conjunto de MDEs, lo cual distaría en gran medida de los padres, perdiendo todo tipo de similitud con estos. Para solucionar este problema, se ha planteado la siguiente solución: cada hijo estará formado por la mejor rama de cada uno de sus padres, manteniendo la información mas relevante de cada uno de ellos, para lo cual hay que evaluar cada una de las ramas de los padres, de forma similar a como se evalúan los árboles, solo que partiendo de un nodo inferior. Una vez seleccionadas las dos ramas que compondrán al hijo se selecciona la acción del nodo raíz, es decir, la relación entre estas dos ramas (unión o intersección), de forma aleatoria entre la acción de las dos raíces de los padres. De esta forma, si los dos padres tienen la misma acción en su nodo raíz, esta se conservará en el nodo raíz del hijo, mientras que si estas acciones son diferentes, se seleccionará una de ellas de forma aleatoria.

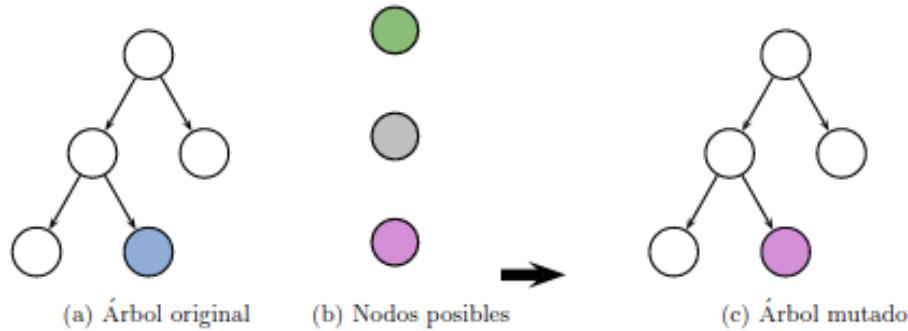


**Figure 23:** Generación de hijos

En este caso, por la propia implementación del algoritmo, puede darse que un MDE esté repetido, ya que este podría aparecer en cada una de las mejores ramas de los padres. Para lidiar esto de forma que no lleve un gran coste computacional, se ha optado por, a la hora de evaluar un árbol, añadir una penalización al f1 score de aquellos árboles que contengan algún nodo repetido. De esta forma, el f1 score de cada árbol quedará como el valor del f1 calculado de forma normal (sin penalización) dividido por el

índice de penalización ( $PEN = 1.5$ ) elevado al número de MDEs repetidos:  $nuevo\_F1\_SCORE = \frac{F1SCORE}{PEN^{N^oMDEs\_repetidos}}$ , de forma que, a medida que aumenten el número de MDEs repetidos, la penalización aumente de forma exponencial, reduciendo drásticamente el F1 SCORE de una solución, evitando así dar por válidas aquellas soluciones que repitan muchos MDEs.

Tras la generación de los hijos, se procede a la mutación. Para lo cual se seleccionan los árboles que van a mutar con el mismo  $P_m$  (0.2) y se modifica el MDE de una de sus hojas, seleccionada de forma aleatoria, por un nuevo MDE que no esté contenido en el árbol.



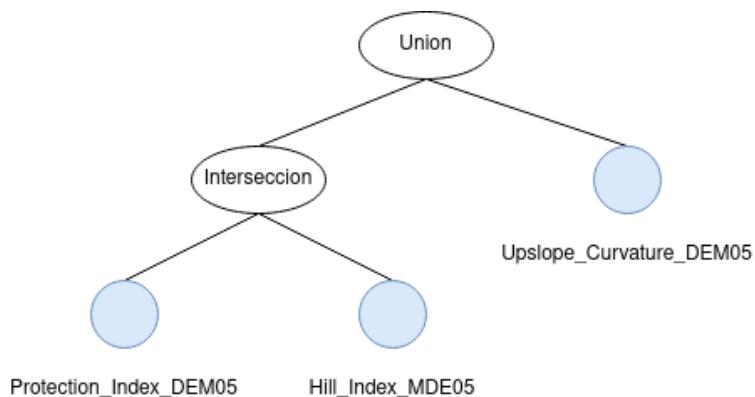
**Figure 24:** Ejemplo de mutación de árbol

Por último se realiza el reemplazo de la población, que no varía con respecto a los apartados anteriores, manteniendo el mejor árbol de cada población, y la evaluación de la nueva población, explicada anteriormente.

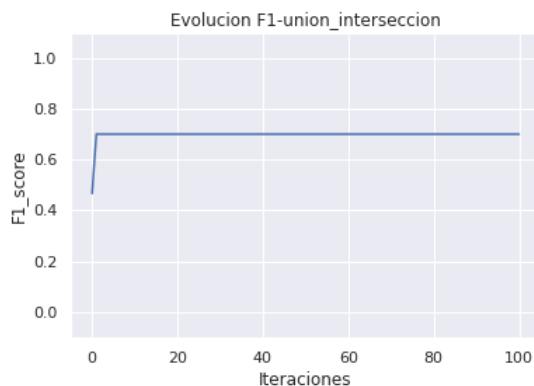
## Resultados obtenidos

Mejor combinación obtenida (figura 84):

( Upslope\_Curvature\_DEM05 (figura 78)  $\cap$   
 ( Protection\_Index\_DEM05 (figura 71)  $\cup$   
 Hill\_Index\_MDE05 (figura 59) ))

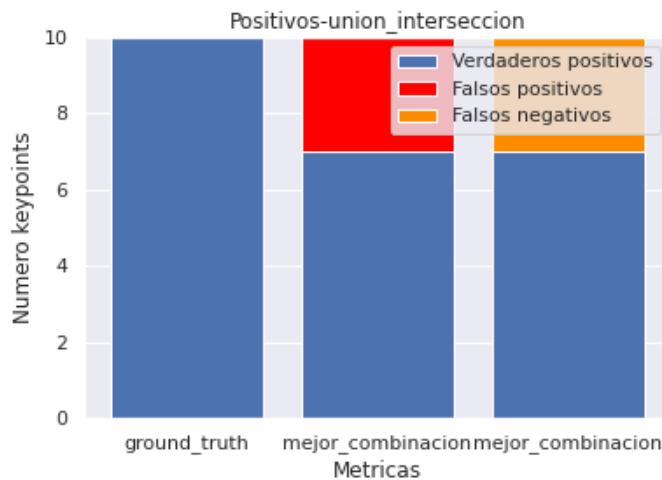


**Figure 25:** Mejor árbol obtenido



**Figure 26:** F1 SCORE - unión e intersección

Esta vez el resultado obtenido es muy bueno, mejorando considerablemente los resultados obtenidos hasta ahora. Este se debe a que combinado uniones e intersecciones se evita que se dispare el número de falsos negativos mientras se controla que no aumente demasiado el número de falsos positivos, logrando tener un buen equilibrio entre verdadero positivos, falsos negativos y falsos positivos. Además, se puede ver que este algoritmo converge mucho más rápido. Esto se debe a la forma por la que se generan los hijos, manteniendo la mejor rama de cada padre, eliminando una gran cantidad de datos irrelevantes, alcanzando óptimos en muchas menos generaciones.



**Figure 27:** Evaluación puntos - unión e intersección

Aquí se puede apreciar como, efectivamente, equilibrio entre verdadero positivos, falsos negativos y falsos positivos es perfecto. En la barra central se puede apreciar cómo la suma de verdaderos positivos más la de falsos positivos es igual al número de keypoints del ground truth, ya que con la intersección de MDEs se evita que el conjunto de keypoints sea demasiado grande, evitando que se dispare el número de falsos positivos. Además, en la barra de la derecha se ve cómo la suma de verdaderos positivos más la de falsos negativos es también igual al número de keypoints del ground truth. Esto se debe a que el conjunto de keypoints tiene un tamaño lo suficientemente grande y sin redundancia, por lo que a la solución sólo le falta un poco de precisión.



## 8 CONCLUSIONES Y POSIBLES AMPLIACIONES

### 8.1 Conclusiones

Este proyecto está enfocado a una colaboración con el proyecto FQ4DEM de la Universidad de Jaén cuyo objetivo es detectar elementos inusuales en imágenes (mapas de sombras) que faciliten la combinación de varios mapas de sombras.

Para ello se ha utilizado un algoritmo de detección de puntos como es el algoritmo *SIFT* sobre distintos Modelos Digitales de Elevación (MDE) de una misma imagen y se ha tratado de obtener la mejor combinación posible con estos resultados respecto de una solución real, denominada *ground truth* (generada manualmente como prueba de concepto).

Para obtener la mejor combinación, dado que no es posible calcularlo mediante fuerza bruta debido al elevado número de datos, se ha optado por un algoritmo de optimización como el *Algoritmo Genético Elitista Uniforme* (AGG\_UN) y se ha probado a obtener la mejor solución con tres tipos diferentes de combinaciones: utilizando uniones de MDEs, utilizando intersecciones, y combinando uniones e intersecciones de MDEs.

Los dos primeros no han logrado resultados muy buenos, ya que el número de puntos obtenidos por las combinaciones era muy elevado y muy pequeño, respectivamente. Este problema se ha solventado con la tercera opción, combinando uniones e intersecciones, la cual ha alcanzado un gran equilibrio entre verdaderos positivos, falsos positivos y falsos negativos, convergiendo además a una velocidad mayor que los dos anteriores.

De esta forma, se logra sacar un gran potencial al combinar dos importantes campos de la informática: la *visión por computados* y la *inteligencia artificial*, que juntos, pueden alcanzar logros tan importantes como los vehículos autónomos que, sin estas dos grandes ciencias, sería inviable lograr.

## 8.2 Posibles ampliaciones

Como se ha mencionado anteriormente, se ha estado trabajando con un ground truth generado manualmente, por lo que los resultados obtenidos son ficticios, no tienen por qué ser reales. Trabajar en el futuro con un ground truth real produciría combinaciones mucho más fiables.

Otra posible ampliación podría ser aplicar este estudio a varios conjuntos de datos más grandes, teniendo más MDEs, con ground truths reales, con mas keypoints y trabajando todo esto sobre distintas imágenes originales, teniendo varios conjuntos de MDEs. Esto ayudaría a generar un modelo mucho más polivalente, capaz de encontrar la mejor combinación, no solo para un caso concreto, sino que esta pueda servir para muchas más situaciones.

Por otro lado, en este proyecto se ha utilizado únicamente el algoritmo SIFT para detectar puntos. Una posible ampliación a esto podría ser trabajar con varios algoritmos detectores de puntos, viendo cual de ellos ofrece mejores de resultados, y tratando de combinar, no solo MDEs, sino también buscar combinaciones de distintos algoritmos y ver qué resultados ofrecen. Esto, al igual que la ampliación anterior, beneficiaría también a la hora de generar un modelo más consistente.

Por último, se podrían utilizar también diferentes algoritmos de optimización como pueden ser otros tipos de algoritmos genéticos, algoritmos de búsqueda local, o, además, aplicar redes neuronales para obtener las mejores combinaciones posibles.



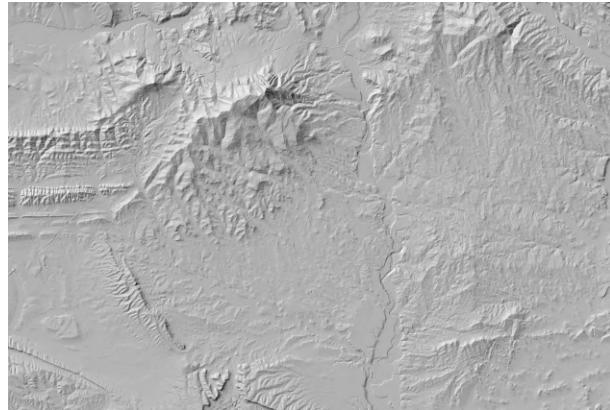
## 9 BIBLIOGRAFÍA

### References

- [1] Sijin Li, Liyang Xiong, Guoan Tang, Josef Strobl. "Deep learning-based approach for landform classification from integrated data sources of digital elevation model and imagery". En *Geomorphology*, págs 2-14
- [2] Tatjana Bürgmann, Wolfgang Koppe, Michael Schmitt. "Matching of TerraSAR-X derived ground control points to optical image patches using deep learning". En *ISPRS Journal of Photogrammetry and Remote Sensing*, págs 241-248
- [3] Amin Sedaghat, Nazila Mohammadi. "Illumination-Robust remote sensing image matching based on oriented self-similarity". En *ISPRS Journal of Photogrammetry and Remote Sensing*, págs 21-35
- [4] Hu Ding, Kai Liu, Xiaozheng Chen, Liyang Xiong, Guoan Tang, Fang Qiu and Josef Strobl . "Optimized Segmentation Based on the Weighted Aggregation Method for Loess Bank Gully Mapping"
- [5] Jung Ok Kim, Kiyun Yu, Joon Heo, Won Hee Lee. "A new method for matching objects in two different geospatial datasets based on the geographic context". En *Computers Geosciences*, págs 1115-1122
- [6] <https://pysource.com/2018/03/21/feature-detection-sift-surf-opencv-3-4-with-python-3/>
- [7] [https://docs.opencv.org/4.x/da/df5/tutorial\\_py\\_sift\\_intro.html](https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html)
- [8] <https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/>
- [9] <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas>
- [10] <https://ccia.ugr.es/~rocio/tesis.pdf>
- [11] CÓDIGO FUENTE:  
[https://github.com/victordiazb00/TFG\\_Deteccion\\_anomalias\\_cartografia\\_digital](https://github.com/victordiazb00/TFG_Deteccion_anomalias_cartografia_digital)



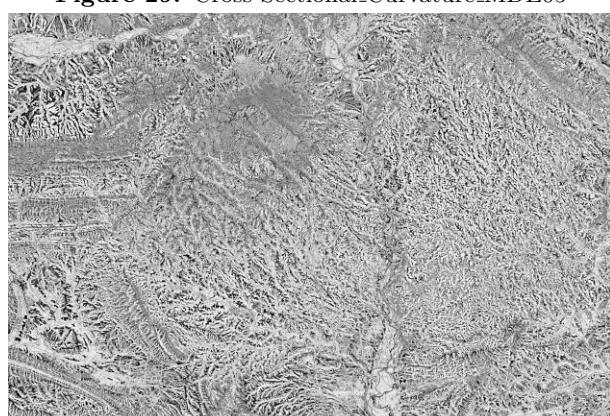
## A MDEs



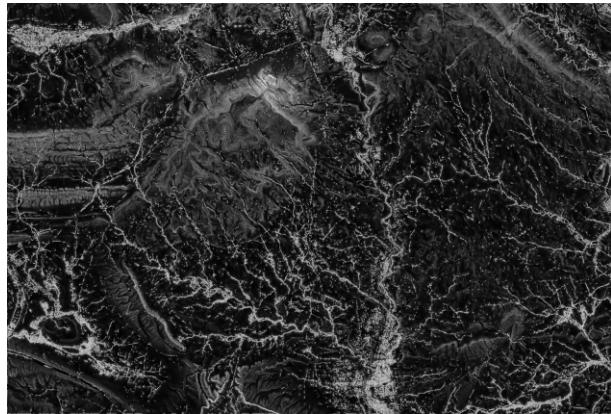
**Figure 28:** MDT05-ETRS89-HU30-0172\_Sombreado (imagen original)



**Figure 29:** Cross-Sectional\_Curvature\_MDE05



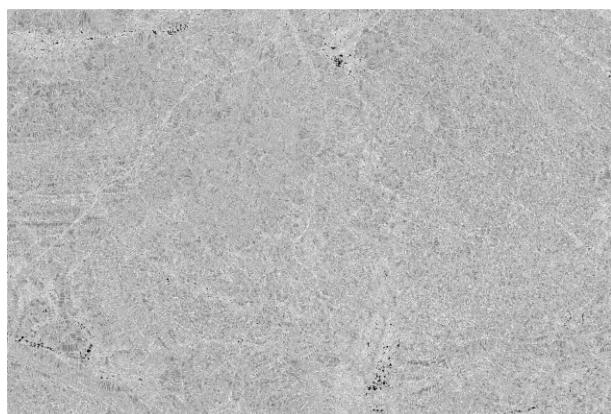
**Figure 30:** Downslope\_Curvature\_DEM05\_1



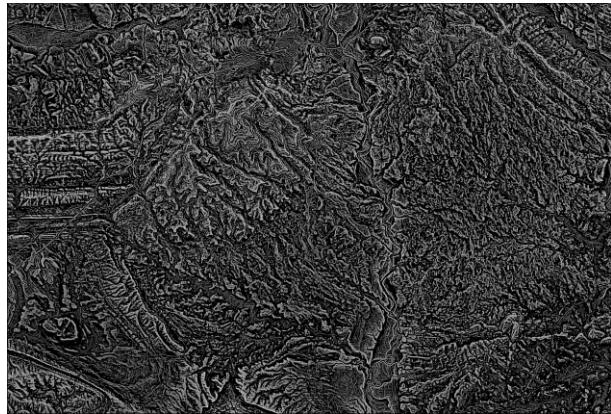
**Figure 31:** Gradient\_DEM05



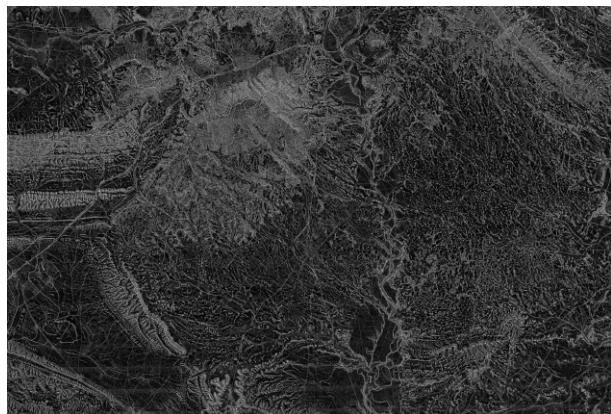
**Figure 32:** Hill\_Index\_MDE05



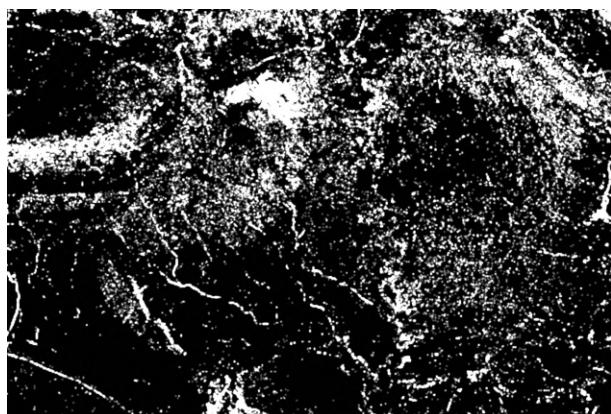
**Figure 33:** Horizontal-Overland-Flow-Distance\_MDE05



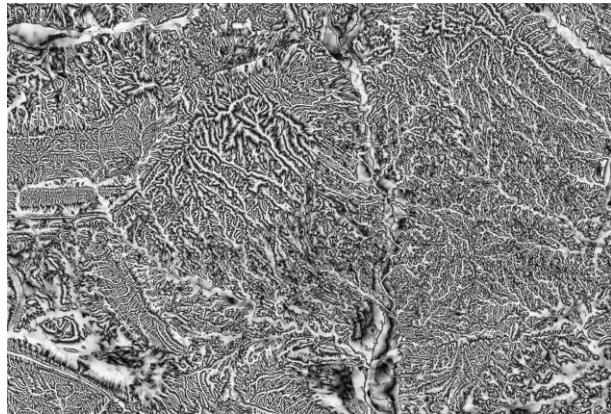
**Figure 34:** Longitudinal\_Curvature\_MDE05



**Figure 35:** Mass\_Balance\_Index\_MDE05



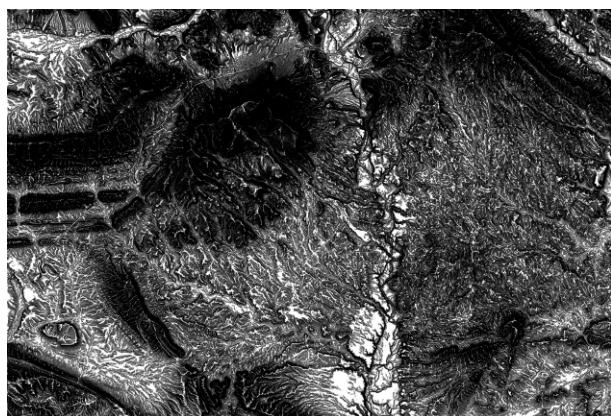
**Figure 36:** MDE05\_surf\_texture



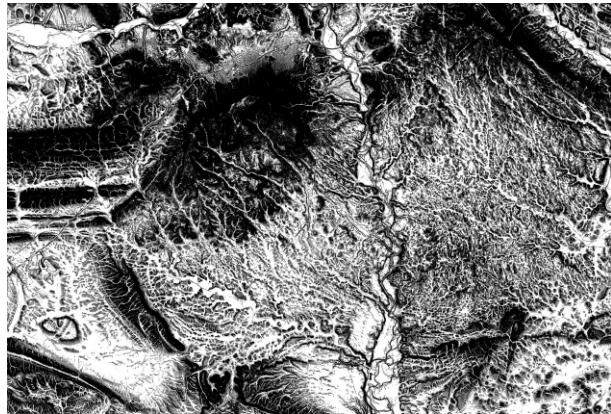
**Figure 37:** Mid-Slope\_Positon\_DEM05



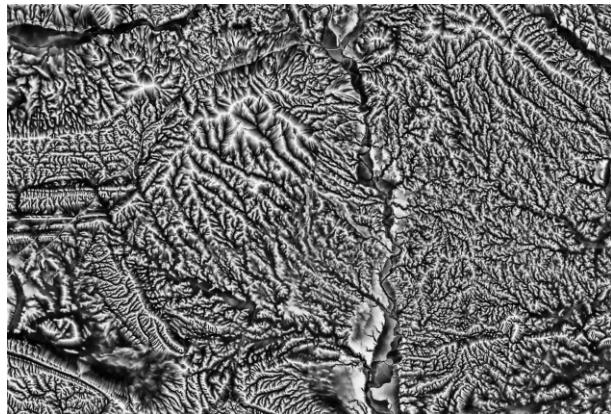
**Figure 38:** Minimum\_Curvature\_MDE05



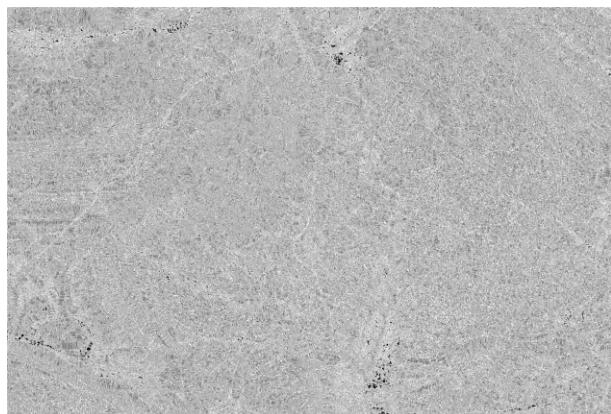
**Figure 39:** MRRTF\_DEM05



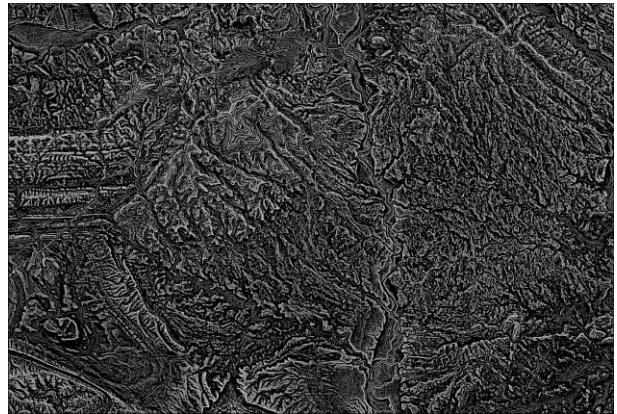
**Figure 40:** MRVBF\_DEM05



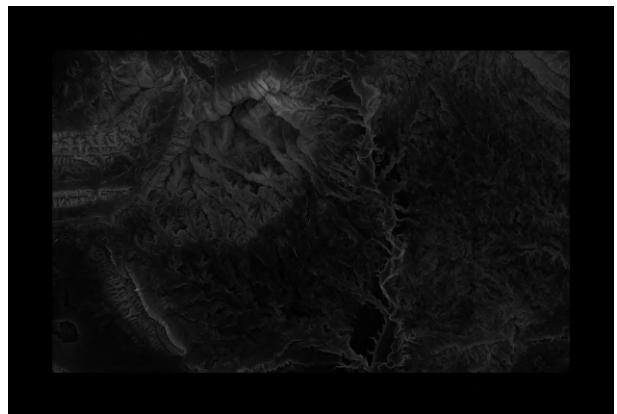
**Figure 41:** Normalized\_Height\_DEM05\_1



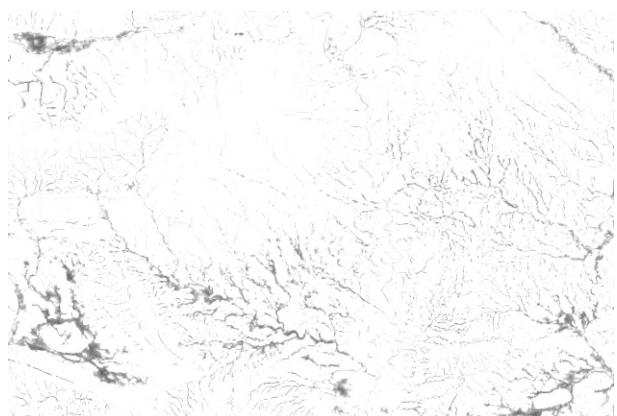
**Figure 42:** Overland-Flow-Distance\_MDE05



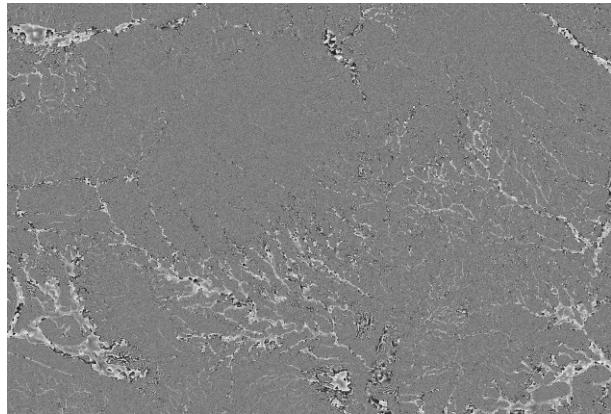
**Figure 43:** Profile\_Curvature\_MDE05



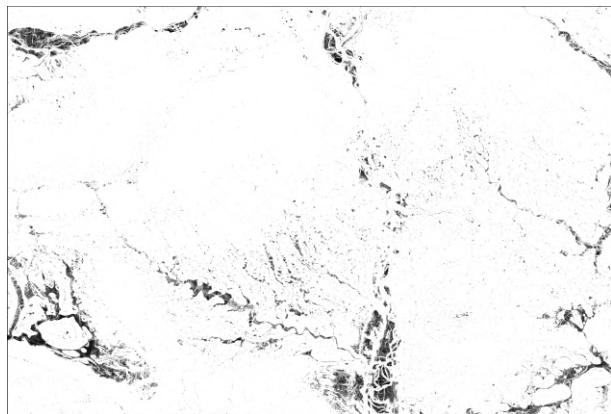
**Figure 44:** Protection\_Index\_DEM05



**Figure 45:** Slope\_Height\_DEM05



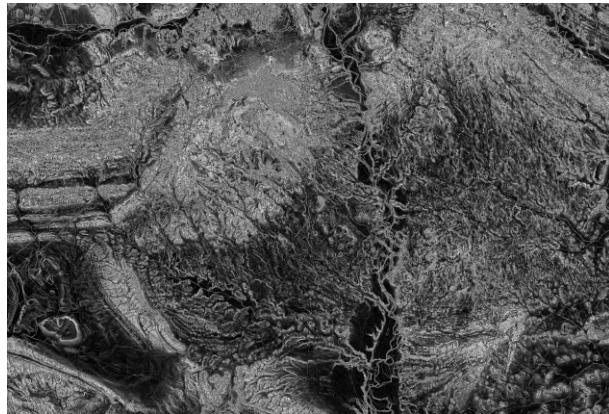
**Figure 46:** Slope\_Height\_DEM05\_1



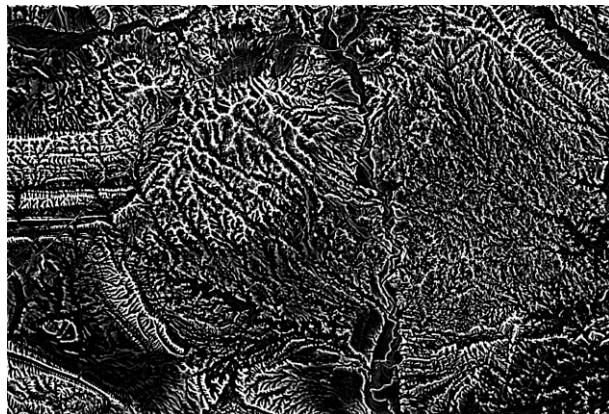
**Figure 47:** Slope\_MDE05



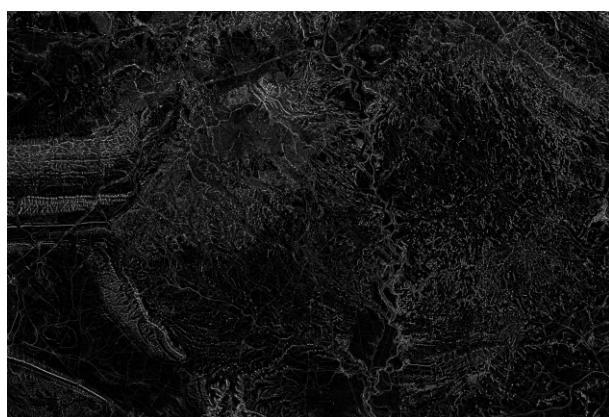
**Figure 48:** Terrain\_Ruggedness\_Index\_(TRI)\_DEM05



**Figure 49:** Terrain\_Ruggedness\_Index\_(TRI)\_DEM051



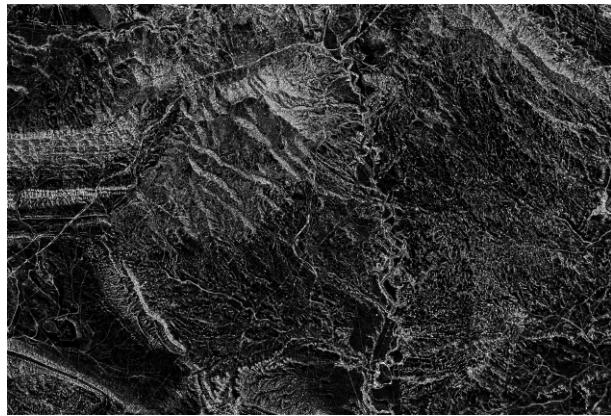
**Figure 50:** TPI\_MDT05\_5\_100\_75



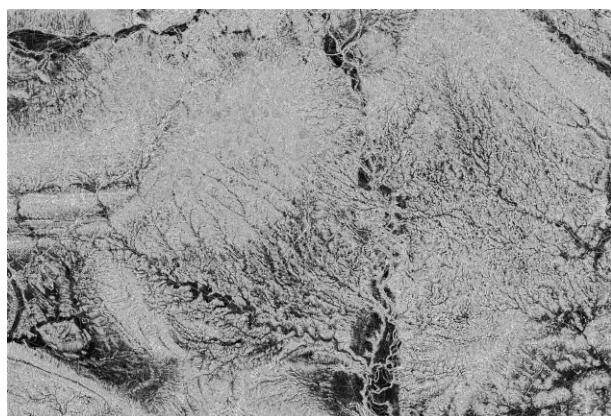
**Figure 51:** Upslope\_Curvature(DEM05



**Figure 52:** Valley\_Index\_MDE05



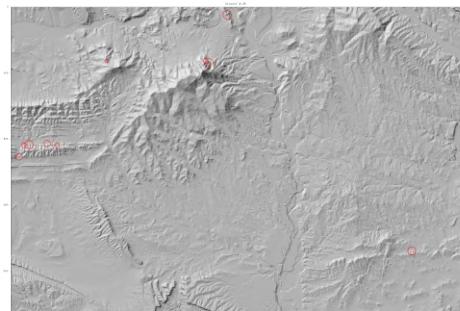
**Figure 53:** Vertical\_Distance\_to\_Channel\_Network\_MDE05



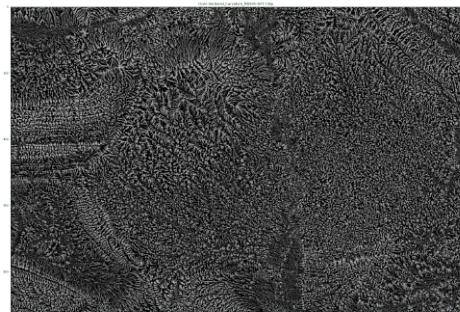
**Figure 54:** Vertical\_Overland\_Flow\_Distance\_MDE05



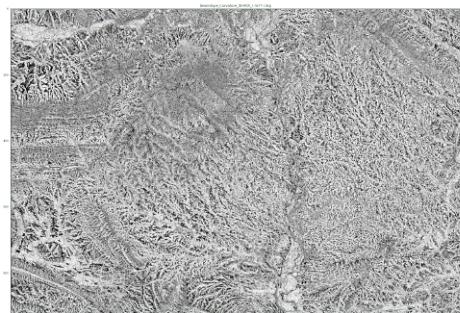
## B SOLUCIONES SIFT



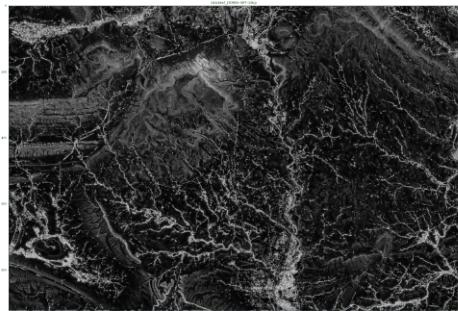
**Figure 55:** Ground\_truth-10kp



**Figure 56:** Cross-Sectional\_Curvature\_MDE05-SIFT-10kp



**Figure 57:** Downslope\_Curvature\_DEM05\_1-SIFT-10kp



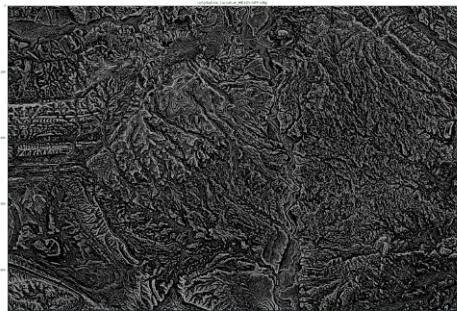
**Figure 58:** Gradient\_DEM05-SIFT-10kp



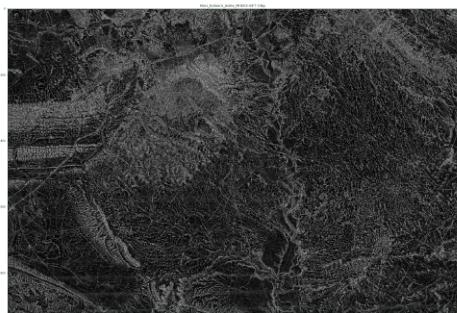
**Figure 59:** Hill\_Index\_MDE05-SIFT-10kp



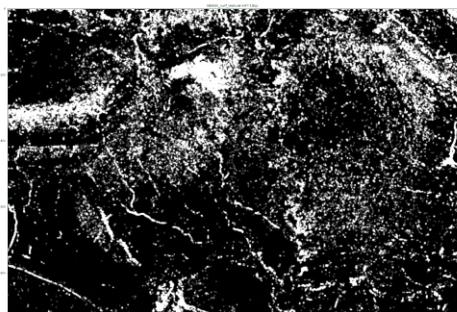
**Figure 60:** Horizontal-Overland-Flow-Distance\_MDE05-SIFT-10kp



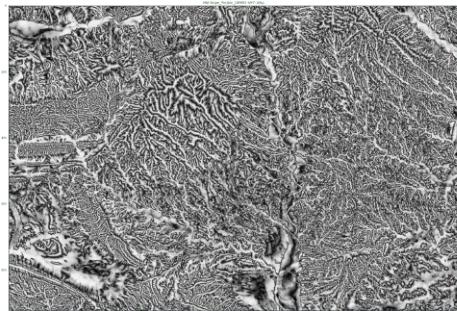
**Figure 61:** Longitudinal\_Curvature\_MDE05-SIFT-10kp



**Figure 62:** Mass\_Balance\_Index\_MDE05-SIFT-10kp



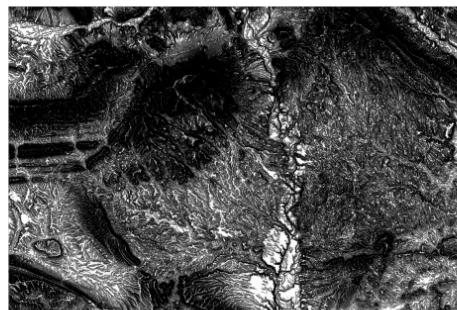
**Figure 63:** MDE05\_surf\_texture-SIFT-10kp



**Figure 64:** Mid-Slope\_Positon\_DEM05-SIFT-10kp



**Figure 65:** Minimum\_Curvature\_MDE05-SIFT-10kp



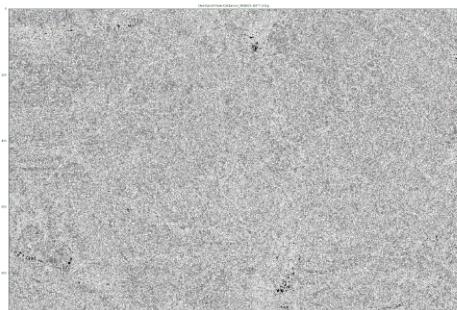
**Figure 66:** MRRTF\_DEM05-SIFT-10kp



**Figure 67:** MRVBF\_DEM05-SIFT-10kp



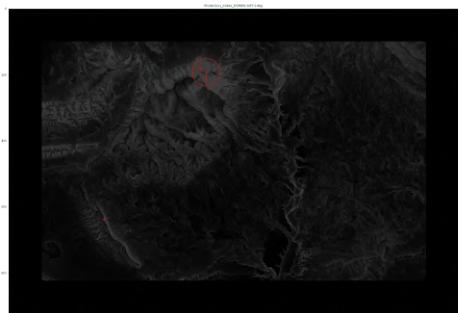
**Figure 68:** Normalized\_Height\_DEM05\_1-SIFT-10kp



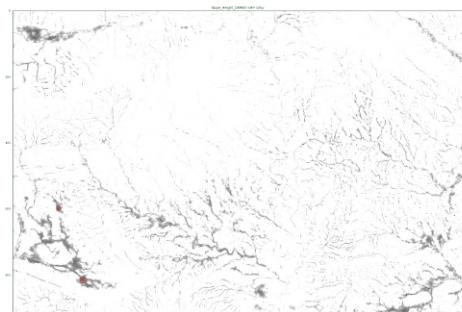
**Figure 69:** Overland-Flow-Distance\_MDE05-SIFT-10kp



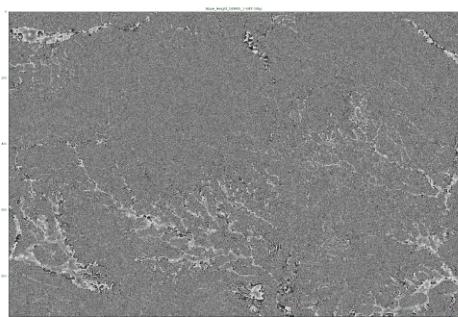
**Figure 70:** Profile\_Curvature\_MDE05-SIFT-10kp



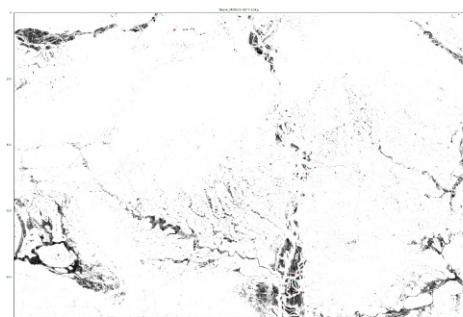
**Figure 71:** Protection\_Index\_DEM05-SIFT-10kp



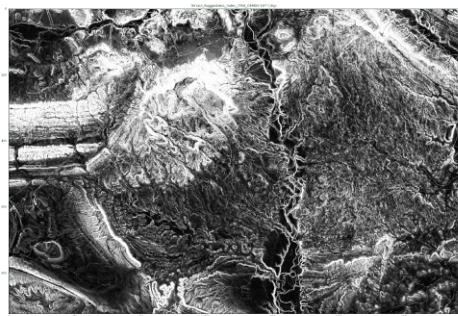
**Figure 72:** Slope\_Height\_DEM05-SIFT-10kp



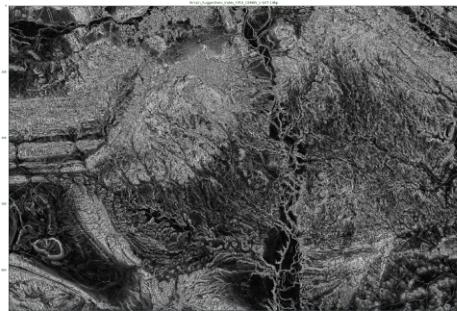
**Figure 73:** Slope\_Height\_DEM05\_1-SIFT-10kp



**Figure 74:** Slope\_MDE05-SIFT-10kp



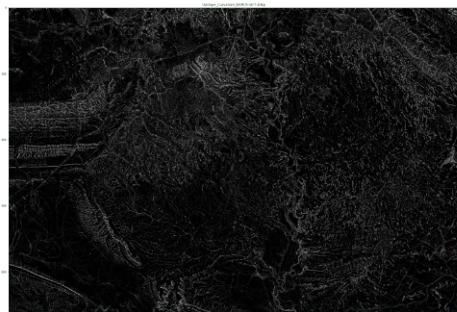
**Figure 75:** Terrain\_Ruggedness\_Index\_(TRI)\_DEM05-SIFT-10kp



**Figure 76:** Terrain\_Ruggedness\_Index\_(TRI)\_DEM05<sub>1</sub> – SIFT – 10kp



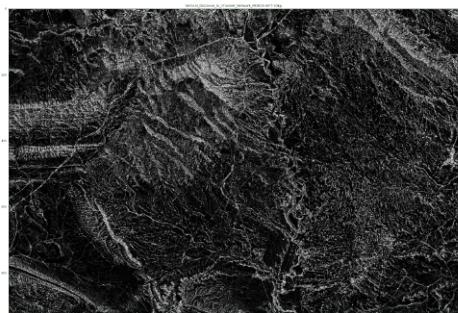
**Figure 77:** TPI\_MDT05\_5\_100\_75-SIFT-10kp



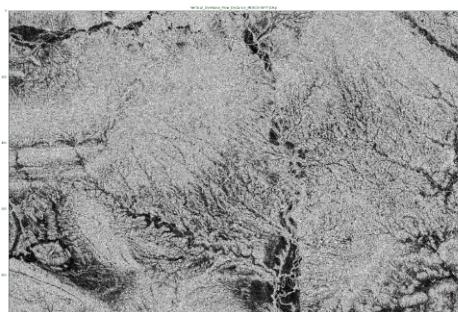
**Figure 78:** Upslope\_Curvature\_DEM05-SIFT-10kp



**Figure 79:** Valley\_Index\_MDE05-SIFT-10kp



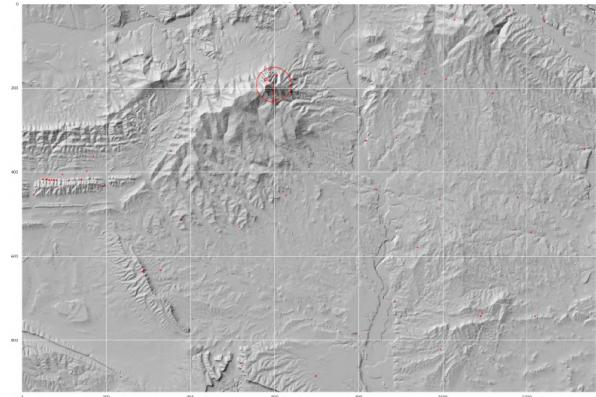
**Figure 80:** Vertical\_Distance\_to\_Channel\_Network\_MDE05-SIFT-10kp



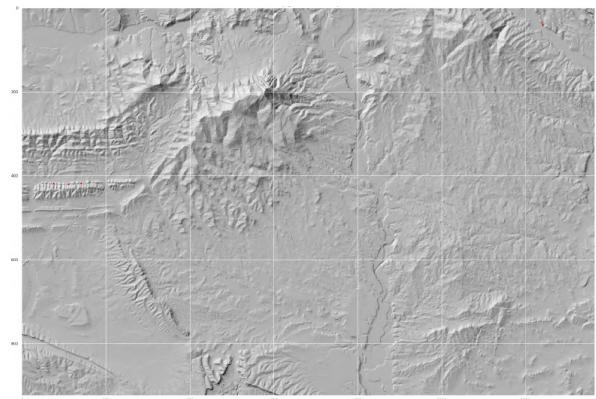
**Figure 81:** Vertical\_Overland\_Flow\_Distance\_MDE05-SIFT-10kp



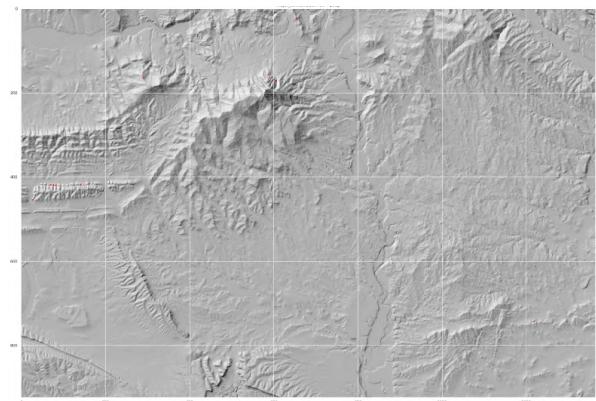
## C MEJORES COMBINACIONES



**Figure 82:** Mejor unión



**Figure 83:** Mejor intersección



**Figure 84:** Mejor combinación de uniones e intersecciones

