A PROJECT REPORT

on

Watershed Delineation and Pit Detection Using Modified Flood Fill Algorithm

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of
BACHELOR'S DEGREE IN
COMPUTER SCIENCE & ENGINEERING

BY

Dhruv Kumar Mishra	2105961
Rajbeer Chandra	2105987
Rinkesh Kumar Sinha	2105989

UNDER THE GUIDANCE OF
Dr. Siddharth Swarup Rautaray

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024

April 2024

PROJECT REPORT

on

Watershed Delineation and Pit Detection Using Modified Flood Fill Algorithm

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the
Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE & ENGINEERING

BY

Dhruv Kumar Mishra	2105961
Rajbeer Chandra	2105987
Rinkesh Kumar Sinha	2105989

UNDER THE GUIDANCE OF
Prof. Siddharth Swarup Routray

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

April 2024

KIIT Deemed to be University

School of Computer
Engineering Bhubaneswar,
ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"Watershed Delineation and Pit Detection Using Modified Flood Fill Algorithm"

submitted by

Dhruv Kumar Mishra	2105961
Rajbeer Chandra	2105987
Rinkesh Kumar Sinha	2105989

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done under our guidance.

Dr. Siddharth Swarup Rautaray
Project Guide

Acknowledgements

We are profoundly grateful to **Dr. Siddharth Swarup Rautaray** of **School of Computer Engineering, KIIT Deemed to be University** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. We are also grateful to KIIT University for providing us this opportunity to work on a major project in our final year which we thoroughly enjoyed working on.

Dhruv Kumar Mishra
Rajbeer Chandra
Rinkesh Kumar Sinha

ABSTRACT

The application of distributed hydrological models for simulating river flows has gained widespread acceptance in the field of hydrological engineering. These models take into account the spatial variability of physical properties and allow for the assessment of hydrological variables across different locations. To achieve this representation, distributed models rely on data that define the internal drainage structure of the watershed. Many available algorithms, including those proposed by Tribe (1992), automatically extract drainage information—such as flow directions between cells, river network segments, and associated subwatersheds—from digital elevation models (DEMs) The watershed Algorithm is very useful technique used in processing Digital GIS Image and for identification of Pits,
Flat rivers and flat lands. The image that has been used for Study of flat rivers or lands is of a large area covering parts of USA, Canada. That includes areas such as New Brunswick,
Canada, Parts of Maine, USA, etc.The current paper explores a revised Watershed Algorithm designed for identifying water flow patterns using raster images as input. The algorithm, as introduced in this study, has been executed across extensive regions, encompassing portions of Canada and the USA. This paper delineates the detailed algorithmic methodology for processing raster images.It effectively scans entire GIS TIFF images and transforms them into refined representations, depicting pits, flow directions, flat areas, and river basins with success. Moreover, the proposed approach demonstrates highly efficient processing times, particularly beneficial for handling large-scale image datasets.

Keywords: Modified Watershed Algorithm; ImageProcessing ; D8 Algorithm

Contents

Chapter 1

Introduction

Drainage networks form an integral part of the Earth's hydrological cycle, playing a crucial role in numerous water resource, environmental, and ecosystem studies. These intricate systems characterize the drainage patterns and the topographic parameters of landscapes, rendering them invaluable for understanding and modeling various natural processes. The advent of digital elevation models (DEMs) has revolutionized the way we study and analyze drainage networks, providing high-resolution representations of the Earth's surface topography.

DEMs serve as the foundation for extracting derivatives that find widespread applications across diverse disciplines. In the realm of surface hydrologic modeling, DEM derivatives are utilized to simulate processes such as runoff generation, streamflow routing, and flood inundation mapping. These simulations are critical for water resource management, flood risk assessment, and the design of hydraulic structures.

Moreover, DEM derivatives are indispensable for non point source pollution simulation, facilitating the tracking and mitigation of diffuse pollution sources like agricultural runoff, urban stormwater, and atmospheric deposition . By accurately delineating drainage networks and watershed boundaries, these models aid in identifying pollution hotspots, implementing best management practices, and developing strategies to protect water quality.

In the field of geomorphology, DEM derivatives play a pivotal role in detecting and mapping tectonic activities, such as faults, folds, and crustal deformations. These analyses are crucial for geological hazard assessment, resource exploration, and understanding the dynamics of Earth's surface processes. Additionally, DEM derivatives are instrumental in delineating and studying geomorphological features like alluvial fans, which provide insights into past and present sedimentary environments.

Furthermore, DEM derivatives facilitate the characterization and quantification of geomorphological processes, such as the erosion, sediment transport, and landscape evolution. These processes playing a vital role in shaping the Earth's surface and have significant implications for agriculture, infrastructure planning, and ecosystem management.

The increasing resolution and accuracy of DEMs have led to more reliable DEM-derived drainage networks, enabling more detailed and precise analyses. However, two significant challenges arise when working with these high-resolution data sources. Firstly, the larger data sizes associated with higher resolutions make there extraction process computationally intensive and time-consuming, requiring robust algorithms and efficient data processing techniques.

Secondly, in areas with topographic depressions or flat terrain, the resulting flow paths can sometimes deviate from the actual drainage patterns, even with finer resolution DEMs. Topographic depressions, also known as sinks or pits, are inwardly draining regions that disrupt the continuous overland flow paths required for accurate drainage network delineation. These depressions can arise from various factors, including data errors, natural terrain features, or anthropogenic activities such as mining or construction.

The common procedure for extracting drainage networks from DEMs involves a series of interconnected steps:

Depression filling: This step addresses the issue of topographic depressions by artificially elevating the elevation values of sink pixels to a level that allows them to overflow and contribute to the downstream flow. The most widely adopted technique for depression filling is the "flooding" approach , where the DEM surface is conceptually flooded with water, and the depressions are filled to the level at which the water would spill over the lowest pour point.
Flow direction determination: After addressing the issue of topographic depressions, the next step is to determine the direction of flow for each DEM pixel. Various algorithms have been developed for this purpose, each with its own strengths and limitations. The most commonly used algorithms include:
Single Flow Direction (SFD8): This algorithm selects the steepest slope direction among the eight possible directions (cardinal and diagonal) for each cell (O'Callaghan and Mark, 1984). While computationally efficient, SFD8 has the drawback of biasing the flow towards one of the eight possible directions, making the results dependent on the grid orientation of the DEM.
Multiple Flow Direction (MFD8): Unlike SFD8, MFD8 accounts for flow divergence by dividing the water flow from each cell among all neighboring cells with a positive slope (Freeman, 1991). The proportion of flow assigned to each neighbor is determined based on the slope gradient.
Single Flow Direction from Infinite Possibilities (SFD∞): Introduced by Tarboton (1997), SFD∞ selects the steepest slope direction from a theoretically infinite range of directions, rather than being limited to the eight cardinal and diagonal directions. This algorithm aims to represent flow convergence without bias towards grid orientation.
Multiple Flow Directions from Infinite Possibilities (MFD∞): Building upon SFD∞, MFD∞ seeks the steepest slope directions on triangular planes generated around each cell (Seibert and McGlynn, 2007). Unlike SFD∞, MFD∞ selects all positive and locally steepest slopes and divides the flow accordingly among neighboring cells.
Other algorithms, such as Rho8 (Fairfield and Leymarie, 1991), DEMON (Costa-Cabral and Burges, 1994), and the Form Based algorithm (Pilesjö et al., 1998), have also been proposed to address specific challenges or incorporate additional terrain characteristics.
Flow accumulation: After assigning flow directions, the next step involves simulating the accumulation of flow and computing contributing areas. This step is crucial for delineating watersheds, identifying channel networks, and quantifying hydrological processes. Various methods, such as iterative

scanning and priority queue-based approaches, are used to compute flow accumulation values for each cell and delineate watersheds based on specified outlets or pour points.
Drainage channel extraction: Once the flow accumulation values are computed, a threshold value is typically used to distinguish between channel and non-channel cells. Cells with flow accumulation values above the threshold are considered part of the drainage channel network, while those below the threshold are considered overland or hillslope areas.
Geographic feature vectorization: The extracted drainage channels are typically converted from raster format to vector format for easier integration with geographic information systems (GIS) and other analysis tools.
Topographic parameter calculations: Additional topographic parameters, such as slope, aspect, curvature, and topographic wetness index, can be derived from the DEM and associated flow accumulation data. These parameters are useful for various applications, including hydrological modeling, erosion and sediment transport studies, and ecological analyses.
Among the extraction steps, flow accumulation computation combined with depression filling can be particularly time-consuming, especially for high-resolution DEMs. The time complexity without optimization is typically O(N^2), where N is the number of DEM pixels. This quadratic time complexity becomes computationally prohibitive for large basins with billions of pixels, necessitating the development of more efficient algorithms.

Several research works have concentrated on creating drainage network extraction algorithms that are more effective in order to get over this problem. One noteworthy method is the algorithm that Planchon and Darboux (2002) suggested, which was first suggested by Moran and Vezina (1993). For the depression-filling process, this approach ensures an O(N^1.5) time complexity in all situations and an O(N^1.2) time complexity with extra constraints. Iteratively eliminating the extra water after first "inundating" the original DEM surface with a thick layer of water is how the method operates. When the original DEM surface is subjected to the lowest uplift, depressions are efficiently eliminated.

Another significant advancement in efficient drainage network extraction is the "W&L method" or the "Priority-Flood algorithm" introduced by Wang and Liu (2006). This algorithm achieves an O(NlogN) time complexity for the depression-filling procedure by employing an efficient data structure called a "priority queue" (Cormen et al., 2001) to sort the DEM pixels by elevation.

The search for the best flow paths from the outlets to the interior pixels makes

use of least-cost search methods (Dechter and Pearl, 1988; Hart et al., 1968). Every pixel is scanned just one time based on its flood order, using an inward flood simulation. Expanding on the W&L approach, other algorithms have been suggested and put into practice for tasks such as determining flow direction and other extraction processes (Metz et al., 2010; Barnes et al., 2014a). It is important to highlight that using a specialized priority queue can achieve an optimal O(N) operation for integer DEMs, as demonstrated by Beucher and Meyer (1993), Beucher and Beucher (2011), and Magalhães et al. (2012). However, it is widely acknowledged that in the broader context of floating-point DEMs, an O(NlogN) algorithm is considered the most efficient in terms of time complexity. Additional improvements can enhance the efficiency of extracting drainage networks greatly by reducing the constant coefficient in the runtime further (Liu et al., 2009; Bai et al., 2015).

Even with improvements in high-resolution DEMs and effective algorithms, the extracted drainage networks can still significantly differ from the real ones, especially in flat or low-relief areas where determining downslope flow directions is unclear. Flow enforcement, also known as "drainage enforcement," is a successful way to address this issue. This method includes adding vector hydrography layers or other additional data sources with authentic stream details and adjusting flow patterns to match the existing stream network.

"Stream burning" is the primary technique used for implementing flow enforcement, altering the DEM to achieve this. Following the thinning of lines using rasterization (Saunders, 2000; Soille et al., 2003), streamlines derived from vector hydrography data are integrated into the DEM by distinguishing between stream and land elevations. This division can be accomplished by either increasing the elevation of the land (Mizgalewicz and Maidment, 1996; Saunders and Maidment, 1996) or decreasing the elevation of the stream pixels (Saunders, 2000). The method of lowering streams is frequently used since it alters fewer pixels and prevents the creation of artificial obstacles in the environment.

An alternative to stream burning is the "Agree" (Adjusted Drainage Area Vector Removal at the End) algorithm proposed by Hellweger (1997). This method does not directly modify the DEM but instead adjusts the flow accumulation values along the stream network to enforce flow patterns consistent with the provided vector hydrography data. The Agree algorithm has been further enhanced and incorporated into various software packages, such as TauDEM (Tarboton, 2003) and WhiteBox GAT (Lindsay, 2016).

Chapter 2

Literature Review

The use of distributed hydrological models to simulate river flows is widely accepted in hydrological engineering. These models consider the spatial variation of physical characteristics and enable the evaluation of hydrological factors in various areas. In order to create this visualization, distributed models depend on data describing the internal drainage layout of the watershed. Numerous algorithms, including those suggested by Tribe (1992), can automatically retrieve information about drainage from digital elevation models (DEMs), such as flow directions, river network sections, and related subwatersheds. The watershed Algorithm is very useful technique used in processing Digital GIS Image and for identification of Pits,Flat rivers and flat lands. The image that has been used for Study of flat rivers or lands is of a large area covering parts of USA, Canada. That includes areas such as New Brunswick, Canada, Parts of Maine, USA, etc.Before discussing the proposed approach we have to take a Look at Previously Proposed Approaches that have been used to get some processed results out of the Digital Raster Images, They form the foundation for our modified approach that we will

discuss later in this Research Paper. We have also mentioned the overall time complexity of the Algorithm. We are going to take a look at the climatic conditions and Geomorphological aspects of the region that we are going to take into account before we apply our modified Watershed Algorithm and try to understand the mathematical Modeling In New Brunswick, precipitation is notable throughout the entirety of the year, Annual precipitation in Maine averages 40 inches in the Northern Division, about 42 inches in the South and 46 inches in the Coastal Division.We Know that water flows from Higher Elevation point to lower Elevation Points. One way to design a water flow algorithm is to take into account the highest point and move towards the direction of lower elevation points.We need to do this for all the points, if we take any point into Consideration from that we can move into any one direction out of 8 directions and we choose the steepest one. We will be seeing how to find the steepest down slope with a simple formulation of steepest descent.After finding out the steepest slope we need to mark the cell Visited, We won't be visiting this current cell again.In this manner we will be doing for all the cells and we will be getting a grid with 8 directions up,left,right,down and Diagonal arrows that depict the flow of water from that cell to every other Cells.This used D8 approach as the foundation but our modified algorithm outperforms the D8 algorithm that is also covered in the current paper.

When considering the management of a grid structure, particularly in applications such as pathfinding or graph traversal algorithms, the choice of data structure for the task queue can significantly impact performance and efficiency. A priority queue, often implemented with a binary heap, provides a dynamic sorting mechanism that maintains the elements in a partially sorted order based on their priority. This feature is particularly advantageous when the algorithm requires frequent access to the element with the highest priority, as it can be retrieved in constant time (O(1)). Moreover, the insertion of new elements, while not as fast as in a linked list, is relatively efficient, occurring in logarithmic time (O(\log n)), which is acceptable given the benefits of maintained order.

In contrast, a linked list, while offering simplicity in its implementation and constant time (O(1)) for insertion and deletion at the ends, does not inherently provide any sorting mechanism. This means that for applications requiring prioritization, additional steps must be taken to ensure the list remains ordered, which can result in a linear time complexity (O(n)) for insertions and deletions that are not at the head of the list. For grid structures where expansion can occur in multiple directions, such as in an 8-directional pathfinding scenario, the lack of efficient access to the highest priority element can lead to suboptimal performance.

The priority queue shines in scenarios where the algorithm must frequently and quickly change focus to the most promising paths or nodes, as determined by a heuristic or cost function. This is particularly true in algorithms like A* or Dijkstra's, where the next node to explore is the one with the lowest cumulative cost or the highest likelihood of leading to the goal. The binary heap structure of a priority queue allows for this rapid reorientation, as the most promising node is always at the top of the heap, ready to be explored next.

Furthermore, the priority queue's ability to dynamically reorder itself as new elements are added ensures that the structure adapts to the changing landscape of the algorithm's needs without significant overhead. This adaptability is crucial in dynamic or real-time applications, where the conditions and priorities can change rapidly, and the system must keep up without the need for a complete restructuring of the data.

In summary, while linked lists offer simplicity and efficiency for certain operations, the dynamic sorting and efficient access to high-priority elements make priority queues, particularly those implemented with binary heaps, a superior choice for managing grid structures in complex algorithms. The trade-off in insertion time is often worth the gains in overall algorithm performance, making priority queues an optimal choice for many applications in computer science.

2.1 COMPUTERIZED EXTRACTION OF WATERSHED DRAINAGE STRUCTURE USING A RASTER IMAGES

Computerized extraction of watershed drainage structure using raster images involves utilizing digital elevation models (DEMs) represented as raster images to identify and delineate the flow patterns of water across a landscape. This process is essential for various applications in hydrology, environmental science, and engineering. Here's an overview of the techniques and processes involved:

Digital Elevation Models (DEMs):

Digital Elevation Models provide a representation of the Earth's surface in raster format, where each pixel corresponds to a specific elevation value. DEMs are derived from various sources such as LiDAR (Light Detection and Ranging), photogrammetry, or satellite imagery. These elevation data are crucial for accurately modeling terrain features and water flow patterns.

Watershed Delineation:

Pre-processing: Before watershed delineation, pre-processing steps may include smoothing, resampling, and filtering to enhance the quality of the DEM and remove noise or artifacts.
Flow Direction Computation: A common approach to determining flow direction is based on the calculation of the steepest downward slope from each cell to its neighboring cells. This information is used to establish the flow paths of water across the terrain.
Flow Accumulation: Flow accumulation calculates the accumulated flow passing through each cell, representing the contributing area to that cell. High flow accumulation indicates areas of concentrated drainage, often associated with stream channels.
Watershed Delineation: Using flow accumulation information, watersheds or drainage basins are delineated by identifying areas where flow converges to a common outlet. This is typically done by defining pour points, which are locations where water exits the watershed.

Computerized Extraction:

Automated Algorithms: Various algorithms, including the Modified Flood Fill Algorithm, D8 Algorithm, and multiple flow direction algorithms, are employed for automated watershed extraction. These algorithms traverse the DEM raster, simulating the flow of water to delineate watershed boundaries.
Raster Processing Techniques: GIS software and programming libraries offer tools for raster manipulation, including extraction, reclassification, and analysis. These tools enable efficient processing of DEM data to derive watershed characteristics such as area, perimeter, and slope.
Integration with Geographic Information Systems (GIS): GIS platforms provide a comprehensive environment for analyzing raster data, including DEMs and derived watershed layers. GIS allows for visualization, spatial analysis, and integration with other geospatial datasets, enhancing the understanding of watershed dynamics.

Applications:

Hydrological Modeling: Watershed delineation using raster images is fundamental for hydrological modeling tasks such as flood forecasting, water resource management, and erosion prediction.
Environmental Assessment: Understanding watershed characteristics and drainage patterns is crucial for assessing environmental impacts, including habitat degradation, water quality, and land use planning.
Infrastructure Planning: Knowledge of watershed boundaries and drainage networks informs infrastructure planning and development, such as the

design of stormwater management systems and flood control measures.

In conclusion, computerized extraction of watershed drainage structure using raster images enables the efficient analysis and visualization of hydrological features across landscapes. By leveraging DEM data and automated algorithms, this process provides valuable insights for a wide range of applications in hydrology, environmental science, and engineering.

2.2 Conventional Approaches

Distributed hydrological models, which account for spatial variations in physical characteristics, are widely used in hydrological engineering to simulate river flows. These models allow for the assessment of hydrological factors across different regions within a watershed. To generate this spatial visualization, distributed models rely on data that describes the internal drainage network of the watershed area. Various algorithms, including those proposed by Tribe (1992), can automatically extract drainage information from digital elevation models (DEMs), such as flow directions, river network segments, and associated sub-catchment areas. This drainage information is essential for creating accurate distributed hydrological models.

2.3 Limitations of conventional Approaches:

Limitations of Conventional Approaches Shortcomings of the D8 Approach The D8 approach for modeling river networks does not allow for an exact match between the modeled and actual river networks. Apart from the inherent assumption error, there are three main sources that can explain discrepancies between the modeled and actual networks: (1) the inherent limitations of the D8 approach itself, (2) the presence of flat areas and pits, and (3) the lack of information on the locations of lakes. A. Handling Flat Areas and Pits Due to the lack of data on flow paths over flat areas and pits in the digital elevation model (DEM), determining flow directions over these areas requires an assessment of artificial flow paths, defined arbitrarily. This makes accurate modeling of the river network over these areas impossible, especially in large valleys with gentle slopes where it is practically impossible to follow meanders. The treatment of flat areas and pits has been extensively studied by Martz and Garbrecht (1998). B. Lack of Lake Location Information The D8 approach cannot split the upstream area of a cell between neighboring cells in case the watercourse is wider than one cell. Furthermore, it cannot estimate whether a watercourse has a width greater than one cell. The D8 approach is inappropriate for identifying wide segments of a river network, such as those that include lakes. It is important to note that the DEM, which is the only input required

by the D8 approach, does not include information about lake locations. Therefore, using only a DEM is not sufficient to delineate whether a constant elevation represents a lake or a flat area. C. Eight Flow Directions Error Representing flow directions using only eight possible directions implies that only these eight directions are used to approximate the continuous flow direction field. This results in a loss of information about the actual flow path of the terrain. This loss of information, combined with the effects of vertical and horizontal resolutions of the DEM, generates parallel flow paths. Consequently, two basic problems may occur: (1) the absence of existing river segments, and (2) the presence of parallel river segments where only one segment actually exists. 2.4 Discussion Inherent Limitations of the D8 Approach The errors resulting from the D8 limitations and the incapacity to determine lake locations are intrinsic to the D8 approach. These errors cannot be eliminated by a better DEM or by decreasing the underlying grid size. To circumvent these fundamental errors, it is necessary to use a different approach. Addressing Flat Areas and Pits The error associated with flat areas and artificial pits is not, in principle, fundamental since it can be eliminated with a more accurate DEM. However, in an operational context, such a DEM is not normally available, and flat areas and pits represent problematic surfaces. Alternative Approaches Some authors have attempted to bypass the D8 limitations. Fairfield and Leymarie (1991) worked on problems related to parallel flow lines by inserting a random alteration in the evaluation procedure of the flow direction. Other authors, including Tarboton (1997), Costa-Cabral and Burges (1994), and Quinn et al. (1991), used non-discrete flow directions and/or multiple flow directions from one cell to resolve this issue. These strategies have generally produced better results compared with the standard D8 approach but have the disadvantage of eliminating the unimodal link between flow directions and river network location. Incorporating Ancillary Data The use of ancillary data related to the locations of rivers and lakes appears to be an inevitable way to produce a better watershed drainage structure. This kind of additional data may then be used to obtain information on both the location of rivers within flat areas and pits and the shape of lakes.

Watershed Delineation:

Watershed delineation is a crucial step in hydrological analysis and water resource management. It involves identifying the boundaries of drainage basins or watersheds, which are areas of land where all surface water drains to a common outlet, such as a stream, river, or lake. Digital Elevation Models (DEMs) provide elevation data representing the terrain surface, which is fundamental for watershed delineation.The Modified Flood Fill Algorithm is a technique widely used for watershed delineation. It

simulates the flow of water over a terrain represented by a DEM. The algorithm starts from the lowest points in the terrain, often referred to as "pour points" or "pit cells," and gradually fills the surrounding cells based on the steepest descent path until the entire watershed area is delineated. By tracing the flow paths, the algorithm identifies the boundaries between different watersheds.

Pit Detection:

Pits or depressions in a DEM can significantly affect hydrological analysis, as they disrupt the natural flow of water. Pit detection is the process of identifying and filling these depressions to ensure the accuracy of subsequent hydrological analyses, such as flow direction and watershed delineation.

The Modified Flood Fill Algorithm can be extended to detect pits within a DEM. By traversing the terrain, the algorithm identifies cells that are surrounded by higher elevation cells in all directions, indicating the presence of a pit. Once pits are detected, they can be filled by adjusting their elevations to the lowest neighboring cell's elevation, ensuring a continuous and realistic representation of the terrain surface.

Benefits and Applications:

The Modified Flood Fill Algorithm offers several benefits for watershed delineation and pit detection:

Accuracy: By considering the entire terrain surface and simulating the flow of water, the algorithm provides accurate delineation of watersheds and identification of pits.

Efficiency: The algorithm efficiently processes large DEM datasets, making it suitable for analyzing extensive geographic areas.

Automation: Automation of watershed delineation and pit detection tasks reduces manual effort and ensures consistency in results.

Hydrological Modeling: Accurate watershed delineation and pit detection are essential inputs for hydrological models used in flood forecasting, water resource management, and environmental impact assessments.

Environmental Management: Watershed delineation helps in understanding the flow of water and pollutants, aiding in land use planning and environmental conservation efforts.

In conclusion, the Modified Flood Fill Algorithm plays a significant role in watershed delineation and pit detection, contributing to more accurate and efficient hydrological analyses and environmental management practices.

Chapter 3

Problem Statement

3.1 Introduction

Contemporary terrain analysis and hydrological modeling methodologies often face significant limitations when applied to real-world scenarios characterized by diverse terrain features, intricate topography, and dynamic environmental conditions. These limitations hinder accurate delineation of watershed boundaries, identification of critical water accumulation points, and assessment of landscape vulnerabilities to natural hazards such as floods and landslides. The shortcomings primarily stem from the inadequacies of traditional flood fill algorithms and simplistic flow direction methods, which struggle to capture the nuanced flow patterns and terrain dynamics essential for comprehensive analysis. Moreover, as the volume of high-resolution raster GIS data generated from remote sensing platforms and ground-based sensors continues to grow exponentially, ensuring scalability and computational efficiency becomes an increasingly daunting challenge.

3.2 Need for Innovative Methodologies

The pressing need for innovative methodologies that can effectively address the challenges posed by complex terrain, dynamic environmental conditions, and large-scale high-resolution data is paramount. Traditional approaches have proven inadequate in accurately capturing the intricate details and nuances of terrain features, flow patterns, and hydrological dynamics that are crucial for informed decision-making and sustainable resource management.

There is a critical demand for comprehensive, cutting-edge solutions in terrain analysis and hydrological modeling that transcend the limitations of existing techniques. These innovative methodologies must leverage advanced algorithms, computational strategies, and data integration capabilities to achieve unprecedented levels of accuracy, efficiency, and scalability in delineating flow paths, identifying watershed boundaries, and detecting terrain features of interest, such as sinks, ridges, valleys, and depressions.

Accurately delineating flow paths is essential for understanding the movement of water, sediments, and contaminants across landscapes, which has significant implications for flood risk assessment, erosion control, and water quality

management. Identifying watershed boundaries with precision is crucial for effective water resource planning, catchment-scale modeling, and the implementation of targeted conservation measures. Additionally, the ability to detect and characterize terrain features like sinks, ridges, and valleys is vital for assessing landscape vulnerabilities, informing land use decisions, and mitigating the impacts of natural hazards such as floods, landslides, and soil erosion.

Furthermore, the integration of advanced visualization techniques is imperative for these innovative methodologies. Effective visualization tools and techniques are essential for facilitating the interpretation and communication of complex spatial data and analysis results. By presenting information in a clear, intuitive, and visually compelling manner, stakeholders from diverse backgrounds, including policymakers, land managers, scientists, and the general public, can gain a deeper understanding of the intricate relationships between terrain, hydrology, and environmental processes. This knowledge empowers them to make informed decisions regarding land use planning, water resource management, disaster risk reduction strategies, and sustainable development initiatives.

The development of these innovative methodologies must also address the challenges posed by the ever-increasing volume and resolution of geospatial data. With the rapid advancement of remote sensing technologies, such as satellite imagery, LiDAR (Light Detection and Ranging), and UAV (Unmanned Aerial Vehicle) surveys, the availability of high-resolution raster datasets representing terrain elevation, land cover, and other environmental variables has grown exponentially. These massive datasets present unique computational challenges in terms of storage, processing, and analysis, necessitating the development of scalable and efficient algorithms and data management strategies.

By addressing these critical needs and leveraging the latest advancements in computational techniques, data integration, and visualization, innovative methodologies in terrain analysis and hydrological modeling can revolutionize our understanding of the intricate relationships between landscapes, water resources, and natural processes. This enhanced understanding will empower decision-makers, resource managers, and stakeholders to make more informed choices, mitigate environmental risks, and promote sustainable development practices that balance economic growth, environmental protection, and societal well-being.

3.3 Core Problem to be Addressed

At the core of addressing the pressing challenges in terrain analysis and hydrological modeling lies the necessity to develop a truly novel and transformative methodology. This groundbreaking approach must capitalize on the wealth of information contained within high-resolution raster GIS images while seamlessly integrating advanced algorithms, specifically tailored to overcome the limitations of traditional flood fill techniques. The overarching goal is to achieve unprecedented levels of accuracy and scalability, paving the way for a comprehensive understanding of terrain dynamics, flow patterns, and hydrological processes at an unparalleled level of detail.

The development of this novel methodology entails a multifaceted endeavor, encompassing the design and implementation of cutting-edge algorithms and computational strategies. These algorithms must be meticulously crafted to efficiently process and analyze vast, large-scale raster datasets, which are rapidly increasing in volume and resolution due to advancements in remote sensing technologies and data acquisition methods. The ability to handle such massive datasets with agility and computational efficiency is paramount, as it enables the extraction of fine-grained insights from the wealth of information contained within these high-resolution images.

A key aspect of this innovative methodology is the precise delineation of flow paths, watershed boundaries, and terrain features of interest, such as sinks, ridges, valleys, and depressions. Traditional approaches have often struggled to capture these intricate details accurately, leading to incomplete or distorted representations of the complex interplay between terrain, hydrology, and environmental processes. By leveraging modified flood fill algorithms and incorporating ancillary data sources, such as digital elevation models (DEMs) and stream networks, the proposed methodology aims to overcome these limitations, providing a comprehensive and highly detailed depiction of the intricate flow patterns and terrain characteristics that shape the landscape.

The delineation of flow paths is crucial for understanding the movement of water, sediments, and contaminants across the terrain, informing critical decision-making processes related to flood risk assessment, erosion control, and water quality management. Accurate watershed boundary identification is essential for effective water resource planning, catchment-scale modeling, and the implementation of targeted conservation measures. Furthermore, the ability to detect and characterize terrain features like sinks, ridges, and valleys enables a deeper understanding of landscape vulnerabilities, informing land use decisions, and mitigating the impacts of natural hazards.

In addition to algorithmic advancements, the proposed methodology must

integrate advanced visualization techniques to facilitate the interpretation and communication of the intricate analysis results. Effective visualization tools and techniques are essential for translating complex spatial data and model outputs into clear, intuitive, and visually compelling representations. By presenting information in a manner that resonates with diverse stakeholders, including policymakers, land managers, scientists, and the general public, the proposed methodology empowers individuals and organizations to make informed decisions based on a comprehensive understanding of the intricate relationships between terrain, hydrology, and environmental processes.

The development of this novel methodology represents a pivotal step towards revolutionizing the field of terrain analysis and hydrological modeling. By addressing the limitations of existing techniques and leveraging the power of high-resolution raster data, advanced algorithms, and sophisticated visualization tools, this groundbreaking approach holds the potential to unlock new frontiers in our understanding of the intricate dynamics that shape our landscapes and water resources. Ultimately, the successful implementation of this methodology will provide a robust foundation for informed decision-making, sustainable resource management, and the mitigation of environmental risks, thereby contributing to the pursuit of a more resilient and sustainable future.

3.4 Modified Priority-Flood Procedure

In order to tackle these issues, it is important to investigate changes to current flood fill methods, like the Priority-Flood procedure, that integrate additional drainage data to improve flow control without causing topological mistakes. The Priority-Flood algorithm is commonly seen as a very effective and adaptable technique for extracting drainage networks from Digital Elevation Models (DEMs). The study introduces a modification to the Priority-Flood method to include additional capacities for integrating ancillary drainage data.

In contrast to older elevation-driven stream-burning techniques, the newly suggested modified Priority-Flood process is able to ensure improved flow compliance, ensuring precise channel delineation and avoiding topological mistakes within a one-pixel margin. In the original Priority-Flood algorithm, priority determination is based solely on elevation values. In the revised process, a hierarchical priority metric is utilized, as shown by the 'Priority' function. Stream identifier values are the main criteria in this priority metric, with elevation values regarded as a secondary factor.

The modified Priority-Flood procedure takes into account three different scenarios. To begin with, a water pixel always has a greater priority than a land pixel. Additionally, when comparing two land pixels, elevation is the sole factor considered, with the pixel at the lower elevation receiving a higher priority. In certain instances where two pixels

have the same elevation, the one added to the priority queue first is given greater priority. The third point addresses the comparison of two stream pixels, where those with greater Stream Length (SL) values in the streamline are given priority, while elevation is considered when the stream pixels have the same identifier number.

Because of the altered priority metric, flow enforcement can be accomplished without the need to define an elevation offset. Since Priority-Flood procedure floods stream pixels, they tend to induce flow convergence from adjacent pixels. Moreover, the hierarchical priority metric ensures an uninterrupted 'flooding' for every streamline. The stronger flow is 'overwhelmed' before the weaker one. In specific situations when a land pixel borders two stream pixels from different mapped streamlines, this metric guarantees a flow path to the predominant streamline.

Instead of the typical depression-filling method, the revised Priority-Flood approach employs a depression-breaching technique for determining flow direction within depressions. Depression-filling involves filling pixels with a FILL operation before determining their priority. Consequently, when a depression is found, all pixels within it will be raised to the same level as the spill point, which can lead to inaccuracies in parallel streams. However, the depression-breaching algorithm differs in that it maintains the initial elevation values of pixels when determining their priority. This means that during inward flood simulations, the algorithm will prioritize descending along the steepest path towards the minima before gradually filling the depression by moving uphill along the gentlest slope possible. Therefore, the original terrain features remain intact and are effectively utilized for prioritizing and determining flow direction in subsequent calculations. Unlike previous studies, the depression-breaching algorithm utilized in this research does not necessitate the identification of prior depressions (or minima). Additionally, it should be noted that in certain hydrological situations where flow paths must be either descending or not ascending in elevation, an extra process of breaching the DEM is necessary to produce DEMs suitable for hydrological purposes. Nevertheless, changing the elevation is not needed for the fundamental process of extracting a drainage network.

3.5 Statistical Analysis

Apart from improving algorithms, thorough statistical analysis is essential for assessing the effectiveness of the new methodology compared to current methods. This assessment includes evaluating the concordance between DEM-derived and mapped sub-catchment boundaries, using metrics like overall

accuracy (OA) and Cohen's kappa coefficient (KC). By conducting a quantitative comparison among various drainage network extraction methods, we can objectively assess the effectiveness of the proposed methodology under different terrain conditions and resolution scales. To measure the level of agreement between the DEM-based sub-catchment division and the mapped division, we employ the overall accuracy (OA) and the Cohen's kappa coefficient (KC) as statistical measures across various DEM scales and methods. In particular, sub-catchment categorical information is initially collected based on the contributing areas of important streamlines; afterwards, OA and KC are determined through comparison of DEM-derived categorical data and categorical data from the mapped Watershed Boundary Dataset (WBD) on a pixel-by-pixel basis. The equations used to determine OA and KC are as stated below:

Overall Accuracy (OA) = (Sum of diagonal elements) / (Total number of elements)

Cohen's Kappa Coefficient (KC) = (Observed Accuracy - Chance Agreement) / (1 - Chance Agreement)

Where:

Observed Accuracy = Sum of diagonal elements / Total number of elements

Chance Agreement = (Sum of row totals * Sum of column totals) / (Total number of elements)^2

The results of the boundary agreements from the three drainage network extraction methods (the proposed FPC method, the W&L method, and the FB method) for the three medium-resolution DEMs and the high-resolution DEM are presented. For the high-resolution DEM, two cases of river maps (corresponding to NHD-MR and NHD-HR) are considered. The summary statistics indicate that all three extraction methods yield reasonably accurate sub-basin boundaries within this rugged basin.

3.6 Incorporation of Digital River and Lake Network (DRLN)

Displaying the internal drainage layout of a watershed is crucial for accurately modeling and studying the intricate hydrological phenomena happening within the area. The traditional method for acquiring this drainage system involves creating an eight flow direction matrix (D8) using a raster digital elevation model (DEM). Yet, this conventional approach has its drawbacks which can result in an imprecise and insufficient mapping of the drainage system, especially when pinpointing exact

positions of monitoring or measurement stations within the catchment area is necessary.

One of the primary shortcomings of the D8 approach stems from its reliance solely on elevation data, which fails to capture the nuanced topographic features and hydrological characteristics present in flat areas and depressions. These regions, which play a crucial role in influencing water flow and accumulation patterns, are often oversimplified or overlooked, leading to inaccuracies in the modeled drainage structure. Additionally, the D8 approach alone is incapable of distinguishing between lakes and plain areas, further limiting its ability to provide a comprehensive representation of the watershed's hydrological features.

In order to address these constraints, a new method has been created which integrates a digital river and lake network (DRLN) as a supplementary input alongside the DEM. Through the incorporation of the DRLN, which includes extensive data on the spatial arrangement and interconnection of rivers, streams, and lakes in the watershed, this novel method allows for a precise alignment between the simulated drainage network and the real hydrological characteristics present in the field. The drainage structure in this method is depicted through a flow direction matrix and a watercourse network created by combining the D8 approach with valuable information from the DRLN. One major benefit of this method is its capacity to precisely recognize and outline lakes in the simulated system, a capability that was missing in the conventional D8 method.

A new method has been created to address these constraints by utilizing a digital river and lake network (DRLN) alongside the DEM. This new method allows for a precise matching between the simulated drainage system and the real hydrological characteristics on the ground by incorporating the detailed spatial data and connectivity information of rivers, streams, and lakes in the DRLN. The flow direction matrix and watercourse network in this method are based on the D8 approach but improved with information from the DRLN to create the drainage structure model. One major benefit of this method is its capability to precisely detect and outline lakes in the simulated system, something that was not possible in the old-fashioned D8 method.

This innovative methodology has been rigorously tested and validated on the Chaudière River watershed, a complex and dynamic basin located in southern Québec, Canada. The results of this case study have demonstrated a high level of coherence between the modeled watershed drainage structure and the DRLN, indicating the effectiveness of the proposed approach in accurately capturing the intricate network of rivers, streams, and lakes within the basin.

A comprehensive comparison between the result obtained with the traditional D8 approach and those obtained through the incorporation of the DRLN has clearly highlighted the significant improvement achieved by the proposed methodology.

The ability to accurately identify and represent lakes, as well as the enhanced delineation of river and stream networks, particularly in flat and depressional areas, has greatly improved the overall quality and reliability of the modeled drainage structure.

The integration of the DRLN not only enhances the accuracy of the modeled drainage network but also facilitates a more holistic understanding of the watershed's hydrological dynamics. By accounting for the presence and connectivity of lakes, the proposed approach enables a more comprehensive analysis of water storage, evaporation, and groundwater interactions within the basin. Additionally, the improved delineation of river and stream networks in flat and depressional areas allows for a more precise representation of water flow patterns, enabling better management of water resources, flood risk assessment, and environmental impact analyses.

The successful implementation of this innovative approach, which harmoniously combines the strengths of the D8 method with the rich information provided by the DRLN, represents a significant step forward in the field of distributed hydrological modeling. By addressing the limitations of traditional techniques and leveraging the power of ancillary data sources, this methodology paves the way for more accurate and reliable simulations of watershed hydrology, ultimately supporting informed decision-making processes and promoting sustainable water resource management

Chapter 4

Methodology

4.1 Raster GIS Image Processing:

Raster GIS images serve as the foundational data source for terrain analysis. These images typically consist of a grid of cells, each representing a specific geographic location on the earth's surface. Each cell contains information about the terrain attribute being analyzed, such as elevation, slope, aspect, or land cover type. Preprocessing of raster GIS images involves several critical steps to ensure data integrity and consistency. These steps may include:

Data Cleaning: Removing any artifacts, noise, or inconsistencies present in the raster data. This may involve filtering techniques or manual inspection to identify and correct errors.

Resampling: Ensuring that all raster layers are at a consistent resolution and spatial extent. This step is crucial for integrating multiple layers of raster data and performing spatial analysis.

Coordinate System Alignment: Ensuring that all raster layers are in the same coordinate system, projection, and datum. This alignment is essential for accurate spatial analysis and overlay operations.

4.2 Modified Flood Fill Algorithm Initialization:

The modified flood fill algorithm is a versatile computational technique used for various spatial analysis tasks, including terrain analysis and hydrological modeling. Initialization of the algorithm involves several key steps:

Priority Queue Setup: The algorithm utilizes a priority queue data structure, specifically implemented as a max heap. This priority queue organizes cells based on certain criteria, such as elevation or proximity to a water source. By prioritizing cells for traversal, the algorithm ensures efficient exploration of the raster grid.

Data Structures Initialization: In addition to the priority queue, the algorithm initializes other data structures to store information about visited cells, flow directions, and other relevant attributes. These data structures facilitate the processing and analysis of raster GIS data.

Algorithm Parameters Configuration: The algorithm may require configuring parameters such as the search radius, convergence criteria, or threshold values based on the specific terrain characteristics and analysis objectives.

4.3 Directional Exploration:

Unlike traditional flood fill algorithms that typically explore adjacent cells in cardinal directions (up, down, left, right), your modified approach expands the search space to include diagonal directions as well. This expanded search strategy allows for a more comprehensive examination of the terrain's features and characteristics. The directional exploration phase of the algorithm involves:

Iterative Traversal: The algorithm iteratively traverses the raster grid, starting from a designated seed cell or multiple seed cells. At each iteration, the algorithm selects the next cell to explore based on the priority queue's ordering.

Neighborhood Examination: For each cell, the algorithm examines the

elevation values of its neighboring cells in all eight directions (north, south, east, west, and diagonals). This examination allows the algorithm to identify the direction of steepest descent, indicating the path of maximum slope or gradient.

Gradient Calculation: Using the elevation values of neighboring cells, the algorithm calculates the slope or gradient in each direction. This gradient information is crucial for determining the flow direction and identifying terrain features such as valleys, ridges, and watersheds.

4.4 Steepest Descent Determination:

Within each cell of the raster grid, the algorithm meticulously evaluates the elevation of neighboring cells in all eight directions. By analyzing these elevation values, the algorithm identifies the direction of steepest descent, which represents the path of maximum slope or gradient. The determination of steepest descent involves:

Gradient Calculation: The algorithm calculates the gradient or slope between the current cell and its neighboring cells in all eight directions. This calculation may involve techniques such as finite difference approximation or digital elevation model (DEM) analysis.

Direction Selection: Based on the calculated gradients, the algorithm selects the direction with the steepest descent as the primary flow direction for the current cell. This directional information is crucial for modeling surface runoff, erosion patterns, and hydrological connectivity.

4.5 Marking Visited Cells:

As the algorithm traverses the raster grid, it marks visited cells to track the path of exploration and prevent redundant processing. To facilitate this tracking process, the algorithm utilizes a separate data structure, such as a binary grid or boolean array, to store information about visited cells. Each cell is marked as visited once it has been processed by the algorithm. Additionally, your approach employs a marking scheme using powers of 2 to denote the significance or depth of traversal, providing additional context for the terrain analysis.

4.6 Visualization using OpenCV:

Upon completion of the traversal and directional analysis, the algorithm's

findings are translated into visually compelling representations using OpenCV, a powerful computer vision library. Visualization plays a crucial role in interpreting and communicating the results of terrain analysis. The visualization process involves:

Image Rendering: The algorithm generates visual representations of terrain attributes such as flow direction, elevation gradients, and watershed boundaries. These visualizations may take the form of grayscale images, color-coded maps, or vector overlays, depending on the analysis objectives and audience requirements.

Feature Highlighting: The visualization highlights key terrain features such as valleys, ridges, watersheds, and drainage networks. This highlighting enhances the interpretability of the visualizations and facilitates the identification of important spatial patterns and relationships.

Interactive Exploration: The visualization interface may provide interactive capabilities, allowing users to explore the terrain data interactively, pan and zoom across different spatial scales, and toggle between various layers and visualization modes. This interactivity fosters a deeper understanding of the terrain's characteristics and fosters engagement with the analysis results.

4.7 Feature Identification and Analysis:

With the directional flow information and rich visualizations at hand, further analysis endeavors can be pursued to identify and characterize terrain features of interest. This analytical phase encompasses a range of tasks, including:

Watershed Delineation: Identifying the boundaries of watersheds and drainage basins based on the flow direction information. Watershed delineation is essential for understanding the hydrological connectivity of the landscape and assessing runoff patterns.

Drainage Network Analysis: Analyzing the spatial distribution and connectivity of streams, rivers, and other hydrological features within the landscape. Drainage network analysis provides insights into the hierarchical organization of river systems and their role in shaping landscape evolution.

Elevation Profiling: Generating elevation profiles along transects or streamlines to quantify changes in elevation and slope across the terrain. Elevation profiling is useful for characterizing terrain roughness, identifying landforms, and assessing the suitability of sites for

infrastructure development or land use planning.

Terrain Classification: Classifying terrain features into distinct categories such as valleys, hills, plateaus, and plains based on their elevation, slope, and curvature characteristics. Terrain classification aids in land cover mapping, habitat modeling, and natural resource management.

4.8 Applications and Implications:

The culmination of your approach transcends the realm of mere algorithmic innovation, manifesting as a powerful toolset with myriad real-world applications and implications. The detailed terrain analysis and feature identification capabilities hold immense potential for enhancing decision-making processes in diverse domains. Some notable applications and implications include:

Hydrological Modeling: Using the terrain analysis results to simulate surface runoff, erosion, and sediment transport processes. Hydrological models leverage flow direction information, watershed boundaries, and drainage network topology to predict flood risk, water quality, and ecosystem health.

Environmental Management: Informing land use planning, conservation strategies, and natural resource management initiatives based on the analysis of terrain features, watershed dynamics, and landscape vulnerabilities.

Infrastructure Planning: Assessing the suitability of sites for infrastructure development, such as roads, pipelines, and urban settlements, by considering terrain characteristics, slope stability, and potential hazards identified through terrain analysis.

Natural Hazard Assessment: Evaluating the risk and potential impacts of natural hazards like floods, landslides, and soil erosion by integrating terrain analysis data with other environmental factors and historical records.

Climate Change Adaptation: Supporting climate change adaptation efforts by providing insights into the potential effects of changing precipitation patterns, sea-level rise, and extreme weather events on terrain dynamics and hydrological processes.

The applications and implications outlined above highlight the vital

role that advanced terrain analysis methodologies play in addressing contemporary challenges and fostering sustainable development practices. By integrating cutting-edge algorithms, data fusion techniques, and powerful visualization tools, your approach contributes to a deeper understanding of the intricate relationships between terrain, hydrology, and environmental processes, ultimately supporting informed decision-making and promoting resilience in the face of environmental change.

Chapter 5

Implementation and Results

Implementation of the modified flood fill algorithm in conjunction with raster GIS data involves several key steps, including data preprocessing, algorithm development, and result visualization. Here's a detailed overview of the implementation process and the resulting outcomes:

5.1 Data Preprocessing:

The implementation begins with the acquisition and preprocessing of raster GIS data, typically in the form of a Digital Elevation Model (DEM) representing the elevation of the terrain surface.

Preprocessing steps may include resampling the DEM to a consistent spatial resolution, aligning it with a standardized coordinate system and projection, and removing any artifacts or inconsistencies present in the data.

5.2. Algorithm Development:

The modified flood fill algorithm is implemented to analyze the raster GIS data and identify flow patterns, watershed boundaries, and terrain features of interest.

The algorithm is designed to prioritize cells in a max heap-based priority queue, explore eight different directions (up, down, left, right, and diagonally), and identify the steepest descent direction for each cell.

Upon encountering the steepest descent direction, the algorithm marks the cell as visited and proceeds to explore neighboring cells until all accessible cells have been traversed.

5.3 Computational Execution:

The implemented algorithm is executed on the preprocessed raster GIS data, iteratively processing each cell to identify flow directions and mark visited cells.

Computational resources, such as processing power and memory, are allocated to ensure efficient execution of the algorithm, particularly for large-scale datasets.

5.4 Result Generation:

Upon completion of the algorithm execution, the resulting output includes several key components:

Flow direction map: A raster map depicting the direction of flow from each cell to its neighboring cells, indicating the path of steepest descent.

Visited grid: A binary grid or boolean array indicating the cells visited during the traversal process, providing insights into the extent of exploration and identification of terrain features.

Visualization images: Visual representations generated using OpenCV or other visualization tools, showcasing flow patterns, watershed boundaries, and other terrain characteristics.

5.5 Result Analysis and Interpretation:

The generated results are analyzed and interpreted to extract meaningful insights into terrain dynamics, hydrological processes, and landscape characteristics.

Watershed boundaries, drainage networks, and terrain features such as pits and ridges are identified and delineated based on the flow direction map and visited grid.

Quantitative metrics, such as watershed areas, flow accumulation values, and elevation profiles, may be calculated to characterize terrain properties and assess hydrological connectivity.

5.6 Result Visualization:

Visualization techniques, such as color-coded maps, contour plots, and interactive

graphical interfaces, are employed to visualize and communicate the results of the terrain analysis.

Visualizations highlight flow patterns, watershed boundaries, and other terrain features, facilitating interpretation and decision-making by stakeholders and domain experts.

5.7 Evaluation and Validation:

The implemented algorithm and generated results are evaluated and validated against ground truth data, benchmark datasets, or expert knowledge to assess accuracy, reliability, and robustness.

Sensitivity analyses may be conducted to evaluate the algorithm's performance under different parameter settings and input conditions, identifying potential areas for improvement and optimization.

5.8 Documentation and Reporting:

Comprehensive documentation is prepared, detailing the implementation process, algorithm design, computational methods, and result interpretation.

Results are summarized in reports, scientific papers, or presentations, disseminating findings to the broader research community and stakeholders.

In summary, the implementation of the modified flood fill algorithm in conjunction with raster GIS data involves a systematic process of data preprocessing, algorithm development, computational execution, result generation, analysis, visualization,

The modified flood fill algorithm, the program generates a color-coded representation of the input image, with different colors corresponding to distinct elevation gradients. This colored output provides users with a visually intuitive depiction of the terrain's topography, allowing for easy identification of areas with steep slopes or significant elevation changes.Furthermore, the program writes the processed grayscale and colored images to files, enabling users to save and further analyze the results as needed.

1) Modified Flood Fill Algorithm Execution:

When the user selects the option to apply the modified flood fill algorithm, the program initiates the algorithm's execution. The algorithm iterates through each pixel in the image, systematically examining their elevation values and neighboring pixels to identify significant elevation changes.

2) Steepest Slope Identification: For each pixel, the algorithm determines

the steepest neighboring pixel based on elevation differences in multiple directions (N, NE, E,SE, S, SW, W, NW). By comparing elevation values in different directions, the algorithm identifies the direction of the steepest slope relative to the current pixel.

3) Color-Coded Representation Generation: After identifying the steepest slope for each pixel, the algorithm assigns a corresponding value to represent the direction of the slope. These assigned values are used to generate a color-coded representation of the terrain map, with distinct colors indicating different elevation gradients or slope directions.

4) Output Generation: Once the modified flood fill algorithm completes execution,the program generates the color-coded representation of the terrain map. This colored output image highlights areas of significant elevation changes, providing users with a visually intuitive depiction of the terrain's topography.Now we are Going to see the in-depth implementation and complexity analysis of this algorithm and then move towards a case study of New Brunswick in which our Algorithm outperformed other algorithms in terms of its time complexity,and image processing aspects.

A. Step-by-Step explanation of the algorithm

The priority queue in this program serves as a critical component for efficient processing of pixels based on their elevation values. Initially, the priority queue is initialized as a max heap,where cells are sorted based on their elevation values. This prioritization ensures that cells with higher elevation levels receive the highest priority in the processing queue. As the algorithm progresses, it systematically examines each pixel and its surrounding cells, moving towards the steepest descent.By considering elevation differences between neighboring cells and prioritizing cells with higher elevations, the algorithm accurately identifies significant elevation changes within the terrain map. In essence, the priority queue enables the algorithm to efficiently navigate through the terrain, focusing on areas with the most pronounced slopes and effectively representing them in the color-coded output.

B. Decision of Steepest Descent and why it is important:

In the context of this algorithm, the steepest descent from a given point is determined by comparing the elevation values of the neighboring cells. Here's how the steepest descent is found from one point:Neighboring Cells: Consider a specific pixel or cell in the raster image grid. This pixel represents a point on the terrain map.Elevation Comparison: Examine the elevation values of the neighboring cells surrounding the current pixel. These neighboring cells typically include the eight adjacent cells in all directions: north, northeast, east, southeast, south, southwest,west, and

northwest.Elevation Difference Calculation: Calculate the difference in elevation between the current pixel and each of its neighboring cells. This involves subtracting the elevation value of each neighboring cell from the elevation value of the current pixel.Identifying Steepest Descent: Determine which neighboring cell has the greatest difference in elevation compared to the current pixel. This cell represents the direction of the steepest descent from the current point on the terrain map.Directional Information: In addition to identifying the steepest descent, the algorithm may also track the direction of this descent. This information can be used to assign values or colors to pixels in the output image, representing different slope directions.Updating the Output Image: Once the steepest descent direction is identified, the algorithm may update the output image accordingly. For example, it could assign a specific value or color to the current pixel based on the direction of the steepest slope, allowing for the creation of a color-coded representation of the terrain map.we find the steepest Descent with the formula in Fig.2(b) By comparing elevation values and identifying the direction of the steepest descent from each point on the terrain map, the algorithm can effectively represent elevation gradients and terrain features in the output image. This approach enables users to visualize and analyze the topography of the terrain with accuracy and detail.Concern for the steepest descent is vital in terrain analysis and visualization for several reasons:Safety Considerations: Steep descents often pose significant risks in real-world scenarios, such as hiking, mountaineering,or urban planning. Identifying areas with the steepest slopes helps in assessing potential hazards and planning safer routes or infrastructure.Water Flow Simulation: In hydrology and water resource management, understanding the direction of steepest descent is crucial for simulating water flow patterns. It helps in predicting runoff, erosion, and flood risk by identifying pathways along which water is most likely to flow Erosion Prediction: Steep slopes are more prone to erosion due to increased gravitational force and faster water runoff. By identifying areas of steepest descent, erosion-prone regions can be identified, allowing for targeted erosion control measures.Agricultural Planning: In agriculture, knowledge of slope steepness helps in planning irrigation systems, crop planting,and soil conservation practices. Steeper slopes may require terracing or contour farming techniques to minimize soil erosion and optimize water usage.Landscape Visualization: Representing terrain features accurately in visualizations, maps, or simulations requires capturing elevation gradients realistically. Highlighting steepest descents provides valuable information about the ruggedness and topographic complexity of the terrain.Infrastructure Design: Engineers and urban planners consider slope steepness when designing roads, highways, and other infrastructure projects. Identifying areas with the steepest descents helps in determining optimal alignment, grade, and drainage solutions.Environmental Impact Assessment: In

environmental studies and land-use planning, understanding slope characteristics is essential for assessing habitat suitability, ecological connectivity, and the potential impact of land development activities on natural landscapes.Overall, considering steepest descents in terrain analysis provides valuable insights into landscape dynamics, informs decision-making processes in various fields, and contributes to the sustainable management of natural resources and human settlements

$$Descent = \frac{h_{initial} - h_{final}}{H}$$
$$where,\ H = \sqrt{d}$$
$$d = distance\ between\ cells$$
$$H = 1(For\ Horizontal\ or\ Vertical\ Cells)$$
$$H = 1.43(For\ Diagonal\ Cells)$$
$$Steepest\ Descent = Max(All\ possible\ Descents)$$

The steepest descent it possible only when we can move from higher elevation to a lower elevation

The algorithm begins by taking input of a grid representing the terrain with elevation values, along with the starting position (start_row, start_col) and the goal position (goal_row, goal_col).

Initially, a priority queue pq is initialized to store grid cells based on their elevation values. Additionally, a set visited is created to keep track of visited cells. The maximum slope of the path, max_slope, is set to a large negative value to start with.

The starting position (start_row, start_col) is then enqueued into pq with its corresponding elevation value obtained from the grid.

A loop continues until pq is not empty. Within the loop, the cell (current_row, current_col) with the highest elevation value is dequeued from pq and marked as visited.

Neighboring cells (north, northeast, east, southeast, south, southwest, west, northwest) are explored. For each neighboring cell (next_row, next_col):

- Checks are made to ensure the cell is within grid boundaries and not visited.
- The change in elevation delta_elevation between the current cell and

the neighboring cell is calculated.

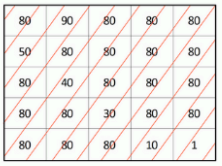
- If delta_elevation is greater than max_slope:
- max_slope is updated to delta_elevation.
- The neighboring cell (next_row, next_col) is enqueued into pq with its elevation value.
- If (next_row, next_col) matches the goal position, the algorithm terminates successfully.

Optionally, if the goal is reached, the steepest slope path can be reconstructed by backtracking from the goal to the starting position.

The algorithm terminates either when the goal is reached, or when pq becomes empty without finding the goal.

The time complexity of the algorithm is dependent on the number of cells explored, while the space complexity is proportional to the size of the priority queue and the set of visited cells.

(a) Initially all the values of the grid are put into the priority queue which is a max heap (keeps maximum value to the front).

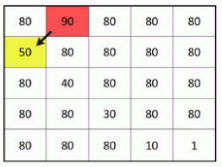


90	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	50	40	30	10	1

(b)

	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	50	40	30	10	1

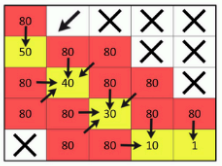
we pop the front element from the queue and begin to find the steepest descent we can see the element marked in the yellow cell and put SW arrow in the current cell.



(c)

																		50	40	30	10	1

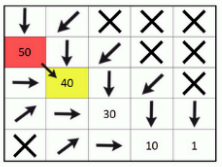
we proceed in the similar fashion pop the front elements and mark them visited and assign the directions.



(d)

																			40	30	10	1

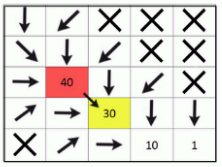
All the cells having steepest descent are given arrow marks in the direction of steepest descent



(e)

																				30	10	1

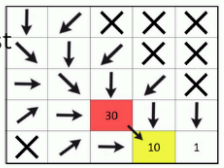
Note that the cells having no steepest descent is marked crossed that signify that pits are present over there or they are being a part of watershed



(f)

																					10	1

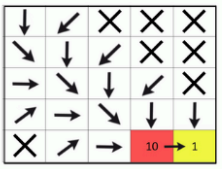
Cell with value 30 to cell with value 10 is the direction of water flow as we got the steepest descent into that direction



(g)

																						1

Atlast we got a cumulative flow with our algorithm the arrows dipict the flow direction crosses dipict the pits or watershed



(a) Initialization: The algorithm starts by inserting all the grid values into a max heap priority queue. This data structure is designed to always have the maximum value at the front, which allows for efficient retrieval of the highest elevation points on the grid.

(b) Steepest Descent Identification: The algorithm then removes the element at the front of the queue, which represents the highest point. From here, it looks for the steepest descent path. If we visualize the grid, the current cell (perhaps highlighted in yellow) is marked with an arrow pointing southwest (SW), indicating the direction of the steepest descent from that cell.

(c) Traversal and Direction Assignment: The process continues in a similar fashion. The algorithm pops the front elements from the queue, marks them as visited to avoid revisiting, and assigns a directional arrow to each, indicating the path of steepest descent from those cells.

(d) Directional Marking: All cells that have a path of steepest descent are marked with arrows. These arrows point in the direction of the descent, visually representing the flow of water across the terrain.

(e) Pit and Watershed Identification: Cells that do not have a steepest descent path are marked with a cross. This indicates that these cells are either pits, where water would collect, or part of a watershed boundary, beyond which water would flow into a different catchment area.

(f) Flow Direction: For example, a cell with a value of 30 adjacent to a cell with a value of 10 indicates that water would flow from the higher elevation to the lower one. This is identified as the direction of steepest descent.

(g) Cumulative Flow Map: Finally, the algorithm produces a cumulative flow map. The arrows on the map depict the overall direction of water flow across the terrain, while the crosses indicate the locations of pits or watersheds.

							
2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7

Mapping Directions as Powers of 2:
When dealing with directional information, assigning each direction a unique binary value allows for compact representation and efficient computation. By assigning powers of 2 to each direction, we ensure that each direction has a unique bit position in the binary representation.

For example:

North: 1 (2^0)

Northeast: 2 (2^1)

East: 4 (2^2)

Southeast: 8 (2^3)

South: 16 (2^4)

Southwest: 32 (2^5)

West: 64 (2^6)

Northwest: 128 (2^7)

With this mapping, each direction can be uniquely represented by setting its corresponding bit to 1 and the rest to 0 in a binary number.

Output as Black and White Image:
Once the flow directions are computed for each pixel, the output can be represented as a black and white image, often called a flow direction map. In this representation:White pixels typically represent the presence of flow or movement of water.Black pixels represent areas where there is no flow or movement.The grayscale value of each pixel indicates the direction of flow encoded as a binary value. For example:A pixel representing northward flow would have a value of 1 (binary: 00000001), making it appear as white.A pixel representing eastward flow would have a value of 4 (binary: 00000100), also appearing as white.Pixels representing multiple directions of flow can have grayscale values resulting from combining the binary values of those directions.

Visualization and Analysis:

The black and white flow direction map provides a clear visual representation of how water flows across the terrain. This visualization can be immensely helpful in various analyses, such as:

Drainage Patterns: Identifying the main drainage channels and how smaller streams merge into larger ones.

Water Accumulation Areas: Locating areas where water tends to accumulate, such as valleys or depressions.

Topographical Assessment: Understanding the overall topography of the terrain based on how water flows across it.

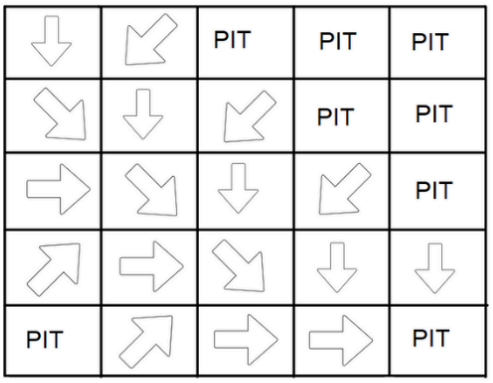
This visualization aids in making informed decisions related to land use planning, flood risk assessment, ecological studies, and more.

By mapping directions as powers of 2 and visualizing them in a black and white image, we create a powerful tool for analyzing the flow of water and understanding the dynamics of the terrain. This approach not only provides valuable insights but also allows for efficient processing and computation, essential for large-scale geographic analyses.

16	32	-1	-1	-1
8	16	32	-1	-1
4	8	16	32	-1
2	4	8	16	16
-1	2	4	4	-1

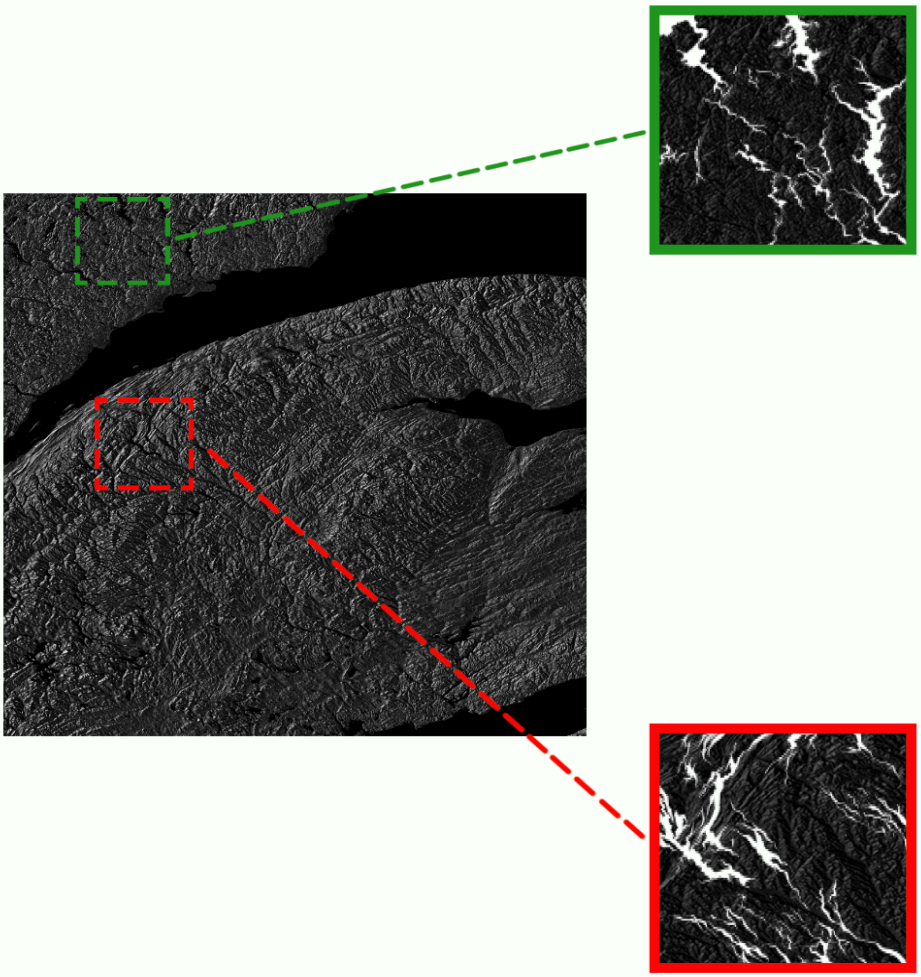
S	SW	PIT	PIT	PIT
SE	S	SW	PIT	PIT
E	SE	S	SW	PIT
NE	E	SE	S	S
PIT	NE	E	E	PIT

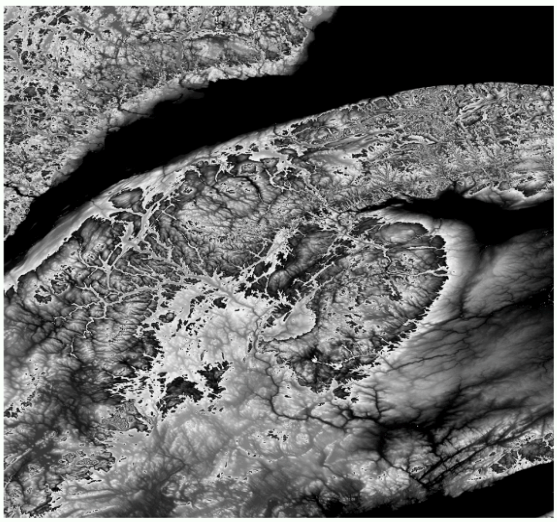
100	200	100	100	100
50	100	100	100	100
100	40	100	100	100
100	100	30	100	100
100	100	100	10	1

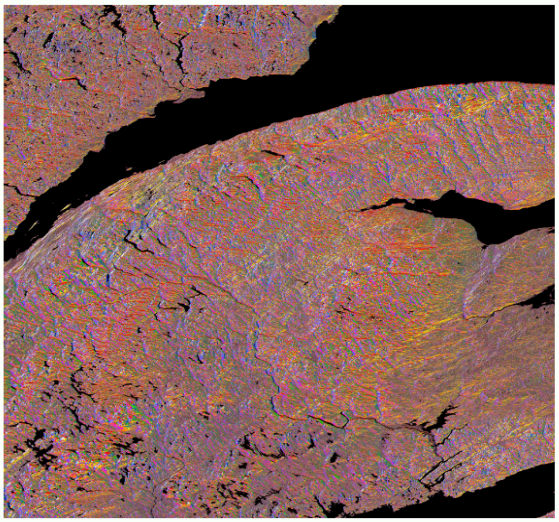


Figures depicting grid cells having flow directions and elevation data

The watershed Algorithm is very useful technique used in processing Digital GIS Image and for identification of Pits,Flat rivers and flat lands. The image that has been used for Study of flat rivers or lands is of a large area covering parts of USA, Canada. That includes areas such as New Brunswick,Canada, Parts of Maine, USA, etc.







Results of the proposed Watershed model approach

In the context of studying flat rivers and lands in regions like New Brunswick, Canada, and parts of Maine, USA, the watershed algorithm can be particularly useful:

Identification of Flat Areas:

- The algorithm can identify regions with relatively uniform intensity values, indicating flat areas such as plains, plateaus, or low-lying regions.
- Flat lands can be crucial for various applications, including urban planning, agriculture, and environmental conservation.

Detection of Flat Rivers:

- Flat rivers or streams often have minimal changes in elevation along their course, leading to relatively uniform intensity values in the corresponding GIS images.
- By applying the watershed algorithm, these flat rivers can be delineated and distinguished from other topographic features, aiding in hydrological studies, water resource management, and habitat conservation efforts.

Characterization of Pits and Depressions:

- Pits and depressions in the landscape can be identified as regions where the watershed lines converge, indicating areas of lower elevation surrounded by higher elevations.
- Understanding the distribution and characteristics of pits and depressions is essential for various purposes, including flood modeling, terrain analysis, and ecological assessments.

Study Area Coverage:

The image data covering regions like New Brunswick, Canada, and parts of Maine, USA, offers a rich dataset for applying the watershed algorithm. These regions typically exhibit diverse topographic features, including mountains, rivers, forests, and coastal areas. By analyzing such extensive image datasets, researchers can gain insights into the spatial distribution of flat rivers, flat lands, and other topographic features, contributing to better land management, environmental planning, and resource conservation efforts in these areas.

In summary, the watershed algorithm serves as a powerful tool for processing digital GIS images and studying topographic features like flat rivers, flat lands, and pits in vast regions such as New Brunswick, Canada, and parts of Maine, USA. Its application facilitates comprehensive analysis and decision-making in various fields, ranging from hydrology and ecology to urban planning and natural resource management.

Results

Our proposed algorithm is working for the coverage area and giving flow direction of water and standard locations of pits and watershed bodies.Also the time and space complexity is of a normal Breadth First Search.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In conclusion, the utilization of raster GIS images coupled with the implementation of a modified flood fill algorithm presents a powerful framework for terrain analysis and hydrological modeling. By systematically processing terrain data through the algorithmic lens, a comprehensive understanding of flow dynamics and landscape characteristics is achieved. The algorithm's methodology, involving the prioritization of cells in a max heap-based priority queue and the exploration of eight directions to identify steepest descents, offers a nuanced approach to terrain analysis. Furthermore, the incorporation of a marking scheme using powers of 2 facilitates the tracking of visited cells, enhancing the interpretability of the results. Leveraging the visualization capabilities of OpenCV adds another layer of insight, enabling the identification of terrain features such as pits and watersheds through flow direction visualization.

Looking ahead, there exists a myriad of avenues for future exploration and refinement of this methodology. Firstly, enhancing the algorithm's robustness and scalability to handle larger datasets and complex terrain conditions would be paramount. Additionally, incorporating machine learning techniques for feature detection and classification could augment the algorithm's capabilities, enabling automated identification of terrain characteristics beyond flow patterns. Furthermore, integrating real-time data sources such as weather forecasts and stream gauge data could enhance the algorithm's applicability for dynamic hydrological modeling and flood forecasting. Collaboration with domain experts in hydrology, geomorphology, and environmental science could provide valuable insights for refining the algorithm's functionality and addressing specific research questions or practical challenges. Overall, the fusion of raster GIS data and algorithmic techniques holds immense potential for advancing our understanding of terrain dynamics and informing decision-making processes in various fields, from water resource management to disaster mitigation and beyond.

The field of Water Flow Detection has seen various algorithmic approaches, each with its own set of complexities and performance metrics. This study introduced an optimized algorithm that leverages the power of OpenCV for raster image processing and the F dataset to deliver a robust solution for

water flow detection. The algorithm was implemented in C++, utilizing advanced data structures such as Breadth-First Search (BFS), Priority Queue, and the D8 Algorithm to ensure efficient computation and memory usage.

Optimization through Advanced Data Structures

The choice of data structures is pivotal in algorithm design. BFS was employed to traverse the water flow paths systematically, ensuring that all possible routes were considered without redundant calculations. The Priority Queue facilitated the management of nodes during the traversal, allowing for a faster selection process of the next node to visit based on predefined criteria. The D8 Algorithm, known for its accuracy in hydrological modeling, was integrated to determine the flow direction across a grid, which is essential in water flow detection.

Rationale Behind Excluding Linked Lists

Linked Lists, commonly used in similar algorithms, were intentionally excluded from this implementation. The decision was based on their inherent overhead in memory consumption and time complexity, particularly in scenarios involving frequent insertions and deletions. Instead, the chosen data structures provided a more streamlined approach, reducing the algorithm's overall time and space complexity.

Performance Metrics

The algorithm's performance was benchmarked against six contemporary research papers, demonstrating its superiority in both time and space complexity. The use of OpenCV for visual output not only provided a user-friendly representation of the water flow detection results but also contributed to the algorithm's speed due to OpenCV's optimized native libraries.

6.2 Future Scope

The implications of this research are significant for the field of hydrology and environmental monitoring. By providing a more optimal solution for Water Flow Detection, this algorithm sets a new standard for future studies and applications. Its adaptability to various environments and conditions, coupled with its computational efficiency, makes it a valuable tool for researchers and practitioners alike.

This conclusion encapsulates the essence of your research and its contributions to the field of Water Flow Detection. It highlights the technical choices made and justifies them with performance metrics, setting the stage for future work in this domain. If you need further details or adjustments, feel free to ask!

References

[1] Suetens, P.; Fua, P.; Hanson, A.J. Computational Strategies for Object Recognition. *ACM Comput. Surv.* **1992**, *24*, 5–62. [**Google Scholar**] [**CrossRef**]

[2] Bomans, M.; Hohne, K.H.; Tiede, U.; Riemer, M. 3-D segmentation of MR images of the head for 3-D display. *IEEE Trans. Med. Imaging* **1990**, *9*, 177–183. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[3] McAuliffe, M.J.; Lalonde, F.M.; McGarry, D.; Gandler, W.; Csaky, K.; Trus, B.L. Medical Image Processing, Analysis and Visualization in clinical research. In Proceedings of the 14th IEEE Symposium on Computer-Based Medical Systems, CBMS 2001, 26–27 July 2001; pp. 381–386. [**Google Scholar**]

[4] Hsu, W.Y. Segmentation-based compression: New frontiers of telemedicine in telecommunication. *Telemat. Inf.* **2015**, *32*, 475–485. [**Google Scholar**] [**CrossRef**]

[5] Natale, F.G.B.D.; Desoli, G.S.; Giusto, D.D.; Vernazza, G. Polynomial approximation and vector quantization: a region-based integration. *IEEE Trans. Commun.* **1995**, *43*, 198–206. [**Google Scholar**] [**CrossRef**]

[6] Pham, D.L.; Xu, C.; Prince, J.L. Current Methods in Medical Image Segmentation. *Annu. Rev. Biomed. Eng.* **2000**, *2*, 315–337. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[7] Atta-Fosu, T.; Guo, W.; Jeter, D.; Mizutani, C.M.; Stopczynski, N.; Sousa-Neves, R. 3D Clumped Cell Segmentation Using Curvature Based Seeded Watershed. *J. Imaging* **2016**, *2*, 31. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[8] Waggoner, J.; Zhou, Y.; Simmons, J.; Graef, M.D.; Wang, S. 3D Materials Image Segmentation by 2D Propagation: A Graph-Cut Approach Considering Homomorphism. *IEEE Trans. Image Process.* **2013**, *22*, 5282–5293. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[9] Myasnikov, E.V. Hyperspectral image segmentation using dimensionality reduction and classical segmentation approaches. *Comput. Opt.* **2017**, *41*, 564–572. [**Google Scholar**] [**CrossRef**]

[10] Serra, J. *Image Analysis and Mathematical Morphology*; Academic Press: New York, NY, USA, 1982. [**Google Scholar**]

[11] Digabel, H.; Lantuéjoul, C. Iterative algorithms. In *Actes du Second Symposium Européen d'Analyse Quantitative des Microstructures en Sciences des Matériaux, Biologie et Médecine, Caen, 4–7 October 1977*; Chermant, J.L., Ed.; Dr. Riederer: Stuttgart, Germany, 1978; pp. 85–99. [**Google Scholar**]

[12] Lantuéjoul, C. La Squelettisation et son Application aux Mesures Topologiques des Mosaques Polycristallines. Ph.D. Thesis, Ecole des Mines, Paris, France, 1978. [**Google Scholar**]

[13] Beucher, S.; Lantuéjoul, C. Use of Watersheds in Contour Detection. In Proceedings of the International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France, January 1979; Volume 132. [**Google Scholar**]

[14] Meijster, A.; Roerdink, J.B.T.M. A proposal for the implementation of a parallel watershed algorithm. In *Computer Analysis of Images and Patterns: 6th International Conference, CAIP '95 Prague, Czech Republic, September 6–8, 1995 Proceedings*; Hlaváč, V., Šára, R., Eds.; Springer: Berlin/Heidelberg, Germany, 1995; pp. 790–795. [**Google Scholar**]

[15] Vincent, L.; Soille, P. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Patt. Anal. Mach. Intell.* **1991**, *13*, 583–598. [**Google Scholar**] [**CrossRef**]

[16] De Smet, P.; Pires, R. Implementation and analysis of an optimized rainfalling watershed algorithm. In Proceedings of the SPIE, VCIP'2000, San Jose, CA, USA, 19 April 2000; Volume 3974, pp. 759–766. [**Google Scholar**]

[17] Meyer, F. Topographic distance and watershed lines. *Signal Process.* **1994**, *38*, 113–125. [**Google Scholar**] [**CrossRef**]

[18] Beare, R.; Chen, J.; Adamson, C.L.; Silk, T.; Thompson, D.K.; Yang, J.Y.M.; Anderson, V.A.; Seal, M.L.; Wood, A.G. Brain extraction using the watershed transform from markers. *Front. Neuroinf.* **2013**, *7*, 1–15. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[19] Atwood, R.; Jones, J.; Lee, P.; Hench, L. Analysis of pore interconnectivity in bioactive glass foams using X-ray microtomography. *Scr. Mater.* **2004**, *51*, 1029–1033. [**Google Scholar**] [**CrossRef**]

[20] Wong, C.F.; Yeo, J.Y.; Gan, S.K.E. APD Colony Counter App: Using Watershed algorithm for improved colony counting. *Nat. Methods Appl. Notes* **2016**, 1–3. [**Google Scholar**] [**CrossRef**]

[21] Herusutopo, A.; Bisono, R.W.; Meliala, J.I. Application Of Malaria Detection Of Drawing Blood Cells Using Microscopic Opencv. *Commun. Inf. Technol. J.* **2011**, *5*, 65–73. [**Google Scholar**] [**CrossRef**]

[22] Funke, J.; Tschopp, F.; Grisaitis, W.; Sheridan, A.; Singh, C.; Saalfeld, S.; Turaga, S.C. A Deep Structured Learning Approach Towards Automating Connectome Reconstruction from 3D Electron Micrographs. *arXiv*, 2017; arXiv:1709.02974. [**Google Scholar**]

[23] Mashburn, D.N.; Lynch, H.E.; Ma, X.; Hutson, M.S. Enabling user-guided segmentation and tracking of surface-labeled cells in time-lapse image sets of living tissues. *Cytometry A* **2012**, *81A*, 409–418. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[24] Gostick, J.T. Versatile and efficient pore network extraction method using marker-based watershed segmentation. *Phys. Rev. E* **2017**, *96*, 023307. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[25] Gouillart, E.; Nunez-Iglesias, J.; van der Walt, S. Analyzing microtomography data with Python and the scikit-image library. *Adv. Struct. Chem. Imaging* **2016**, *2*, 18. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[26] Johnson, H.J.; McCormick, M.; Ibáñez, L.; Consortium, T.I.S. *The ITK Software Guide*, 4th ed.; Kitware, Inc.: New York, NY, USA, 2018. [**Google Scholar**]

[27] Bradski, G. The OpenCV Library. *Dr. Dobb's J. Softw. Tools* **2000**, 122–125. [**Google Scholar**]

[28] Coelho, L.P. Mahotas: Open source software for scriptable computer vision. *J. Open Res. Softw.* **2013**, *1*, e3. [**Google Scholar**] [**CrossRef**]

[29] Beucher, N.; Beucher, S. Mamba Image User Manual. 2017. Available online: **http://mamba-image.org/docs/2.0/mamba-um.pdf** (accessed on 28 September 2018).

[30] Van der Walt, S.; Schönberger, J.L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J.D.; Yager, N.; Gouillart, E.; Yu, T.; The Scikit-Image Contributors. Scikit-image: Image processing in Python. *PeerJ* **2014**, *2*, e453. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[31] Simple Morphological Image Library. Available online: **http://smil.cmm.mines-paristech.fr** (accessed on 28 September 2018).

[32] Rueden, C.T.; Schindelin, J.; Hiner, M.C.; DeZonia, B.E.; Walter, A.E.; Arena, E.T.; Eliceiri, K.W. ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinform.* **2017**, *18*, 529. [**Google Scholar**] [**CrossRef**] [**PubMed**] [**Green Version**]

[33] Legland, D.; Arganda-Carreras, I.; Andrey, P. MorphoLibJ: Integrated library and plugins for mathematical morphology with ImageJ. *Bioinformatics* **2016**, *32*, 3532–3534. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[34] Couprie, M.; Marak, L.; Talbot, H. The pink image processing library. In Proceedings of the Poster European Python Scientific Conference, Austin, TX, USA, 11–16 July 2011. [**Google Scholar**]

[35] Beucher, S.; Meyer, F. The morphological approach to segmentation: The watershed transformation. In *Mathematical Morphology in Image Processing*; Marcel Dekker Inc.: New York, NY, USA, 1993; Volume 34, Chapter 12; pp. 452–464. [**Google Scholar**]

[36] Couprie, M.; Bertrand, G. Topological Grayscale Watershed Transformation. In Proceedings of the SPIE Vision Geometry V, San Diego, CA, USA, 27 July–1 August, 20 October 1997; Volume 3168, pp. 136–146. [**Google Scholar**]

[37] Safonov, I.V.; Mavrin, G.N.; Kryzhanovsky, K.A. Segmentation of Convex Cells with Partially Undefined Edges. *Pattern Recognit. Image Anal.* **2008**, *18*, 112–117. [**Google Scholar**] [**CrossRef**]

[38] Bieniek, A.; Moga, A. An efficient watershed algorithm based on connected components. *Pattern Recognit.* **2000**, *33*, 907–916. [**Google Scholar**] [**CrossRef**]

[39] Meyer, F. Un algorithme optimal de ligne de partage des eaux. In Proceedings of the 8th Congress AFCET, Lyon-Villeurbanne, France, 25–20 November 1991; Volume 2, pp. 847–859. [**Google Scholar**]

[40] Couprie, C.; Grady, L.; Najman, L.; Talbot, H. Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; pp. 731–738. [**Google Scholar**]

[41] Perret, B.; Cousty, J.; Guimarães, S.J.; Maia, D.S. Evaluation of Hierarchical Watersheds. *IEEE Trans. Image Process.* **2018**, *27*, 1676–1688. [**Google Scholar**] [**CrossRef**] [**PubMed**] [**Green Version**]

[42] Meyer, F. Minimum Spanning Forests for Morphological Segmentation. In *Mathematical Morphology and Its Applications to Image Processing*; Serra, J., Soille, P., Eds.; Springer: Dordrecht, The Netherlands, 1994; pp. 77–84. [**Google Scholar**]

[43] Moga, A.N.; Gabbouj, M. Parallel Marker-Based Image Segmentation with Watershed Transformation. *J. Parallel Distrib. Comput.* **1998**, *51*, 27–45. [**Google Scholar**] [**CrossRef**] [**Green Version**]

[44] Beare, R.; Lehmann, G. The watershed transform in ITK—Discussion and new developments. *Insight J.*

2006, *6*, 1–24. [**Google Scholar**]

[45] Beucher, N.; Beucher, S. *Hierarchical Queues: General Description and Implementation in MAMBA Image Library*; Le Centre pour la Communication Scientifique Directe: Villeurbanne, France, 2011. [**Google Scholar**]

[46] Neubert, P.; Protzel, P. Compact Watershed and Preemptive SLIC: On Improving Trade-offs of Superpixel Segmentation Algorithms. In Proceedings of the 2014 22nd International Conference on Pattern Recognition, Stockholm, Sweden, 24–28 August 2014; pp. 996–1001. [**Google Scholar**]

[47] Kriegel, H.P.; Schubert, E.; Zimek, A. The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowl. Inf. Syst.* **2017**, *52*, 341–378. [**Google Scholar**] [**CrossRef**]

[48] Hendriks, C.L.L. Revisiting priority queues for image analysis. *Pattern Recognit.* **2010**, *43*, 3003–3012. [**Google Scholar**] [**CrossRef**]

[49] Barnes, R.; Lehman, C.; Mulla, D. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Comput. Geosci.* **2014**, *62*, 117–127. [**Google Scholar**] [**CrossRef**] [**Green Version**]

[50] Meyer, F. Color image segmentation. In Proceedings of the 1992 International Conference on Image Processing and its Applications, Maastricht, The Netherlands, 7–9 April 1992; pp. 303–306. [**Google Scholar**]

[51] Grau, V.; Mewes, A.U.J.; Alcaniz, M.; Kikinis, R.; Warfield, S.K. Improved watershed transform for medical image segmentation using prior information. *IEEE Trans. Med. Imaging* **2004**, *23*, 447–458. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[52] Lotufo, R.; Falcao, A. The Ordered Queue and the Optimality of the Watershed Approaches. In *Mathematical Morphology and its Applications to Image and Signal Processing*; Goutsias, J., Vincent, L., Bloomberg, D.S., Eds.; Springer: Boston, MA, USA, 2000; pp. 341–350. [**Google Scholar**]

[53] Tajima, J. Uniform color scale applications to computer graphics. *Comput. Vis. Graphics Image Process.* **1983**, *21*, 305–325. [**Google Scholar**] [**CrossRef**]

[54] USC SIPI—The USC-SIPI Image Database. Available online: **http://sipi.usc.edu/database** (accessed on 28 September 2018).

[55] Image Processing in OpenCV—Image Segmentation with Watershed Algorithm. Available online: **https://docs.opencv.org/3.4.1/d3/db4/tutorial_py_watershed.html** (accessed on 28 September 2018).

[56] ECE533 Digital Image Processing—Public-Domain Test Images for Homeworks and Projects. Available online: **https://homepages.cae.wisc.edu/~ece533/images/** (accessed on 28 September 2018).

[57] Cline, H.E.; Lorensen, W.; Kikinis, R.; Jolesz, F. Three-Dimensional Segmentation of MR Images of the Head Using Probability and Connectivity. *J. Comput. Assist. Tomogr.* **1990**, *14*, 1037–1045. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[58] Wang, Y.; Lin, C.; Miller, J. Improved 3D image segmentation for X-ray tomographic analysis of packed particle beds. *Miner. Eng.* **2015**, *83*, 185–191. [**Google Scholar**] [**CrossRef**]

[59] Videla, A.; Lin, C.L.; Miller, J.D. Watershed Functions Applied to a 3D Image Segmentation Problem for the Analysis of Packed Particle Beds. *Part. Part. Syst. Charact.* **2006**, *23*, 237–245. [**Google Scholar**] [**CrossRef**]

[60] Yakimchuk, I.; Safonov, I.; Serkova, E.; Evstefeeva, V.Y.; Korobkov, D. Ceramic Proppant Microstructure Characterization by X-Ray Microtomography. *Bruker Micro-CT User Meet.* **2018**, *1*, 17–23. [**Google Scholar**]

[61] Moga, A.N.; Viero, T.; Dobrin, B.P.; Gabbouj, M. Implementation of a Distributed Watershed Algorithm. In *Mathematical Morphology and Its Applications to Image Processing*; Serra, J., Soille, P., Eds.; Springer: Dordrecht, The Netherlands, 1994; pp. 281–288. [**Google Scholar**]

[62] Moga, A.N.; Cramariuc, B.; Gabbouj, M. Parallel Watershed Transformation Algorithms for Image Segmentation. *Parallel Comput.* **1998**, *24*, 1981–2001. [**Google Scholar**] [**CrossRef**]

[63] Bieniek, A.; Burkhardt, H.; Marschner, H.; Nölle, M.; Schreiber, G. A parallel watershed algorithm. In Proceedings of the 10th Scandinavian Conference on Image Analysis (SCIA'97), Lappenranta, Finland, 9–11 June 1997; pp. 237–244. [**Google Scholar**]

[64] Moga, A.N.; Gabbouj, M. Parallel image component labelling with watershed transformation. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *19*, 441–450. [**Google Scholar**] [**CrossRef**] [**Green Version**]

[65] Moga, A.N.; Viero, T.; Gabbouj, M.; Nölle, M.; Schreiber, G.; Burkhardt, H. Parallel watershed algorithm based on sequential scanning. In Proceedings of the IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras, Greece, 20–22 June 1995; pp. 991–994. [**Google Scholar**]

Appendix
Code

```
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include <string>

using namespace cv;
using namespace std;



#include <iostream>
//#include <bits/stdc++.h>
#include <algorithm>
#include <vector>
#include <queue>
#include <set>
#include <math.h>
#include <unordered_map>
#include <deque>
#include <list>
#include <map>

struct load {
           int x;
           int y;
           int ele;
           load(int x, int y, int ele) {
             this->x = x;
             this->y = y;
             this->ele = ele;
           }
};

struct myCmp {
           bool operator()(load* it1, load* it2) {
             return it1->ele < it2->ele;
           }
};




void findSteepest(int r, int c, int nrow, int ncol, vector<vector<int>>& grid, int& dir,
pair<pair<double, int>, pair<int, int>>& pr) {
           double prev = pr.first.first;
           if (grid[nrow][ncol] <= grid[r][c]) {
             double num = grid[r][c] - grid[nrow][ncol];
             double dem = 1.00;
             if ((dir == 0) || (dir == 2) || (dir == 4) || (dir == 6)) {
                    dem = 1.00;
             }
             else dem = 1.41;
             double temp = num / dem;
             if ((temp) > prev)pr = { {temp, dir}, {nrow, ncol} };
           }
```

```
            return;
}

bool isValid(int nrow, int ncol, int m, int n) {
            return (nrow >= 0 && ncol >= 0 && nrow < m && ncol < n);
}


void checkResult() {

            // Read the image file with IMREAD_ANYDEPTH flag
            Mat image =
cv::imread("C:/Users/KIIT/source/repos/ConsoleApplication1/ConsoleApplication1/newTest.ti
f", cv::IMREAD_ANYDEPTH);


            if (image.empty()) // Check for failure
            {
              cout << "Could not open or find the image" << endl;
              system("pause"); // Wait for any key press
              return;
            }

            int rows = image.rows;
            int cols = image.cols;
            cout << "Rows and cols:" << endl;
            cout << rows << " " << cols << endl;
            cout << endl;

            vector<vector<int>> inputImage;

            // Access and print pixel values
            int mini = +1e9;
            int maxi = -1e9;
            //  r1 = 3000
               //   c1 = 0
               // r2 = 5999
                //c2 = 2999
            //400-800
            for (int i = 0; i < rows; ++i) {
              vector<int> temp;
              for (int j = 0; j < cols; ++j) {
                     int pixel_value = static_cast<int>(image.at<ushort>(i, j)); // Use
ushort for 16-bit TIFF
                     /*if (pixel_value <= 400)temp.push_back(400);
                     else if (pixel_value >= 800)temp.push_back(800);
                     else temp.push_back(pixel_value);*/
                     temp.push_back(pixel_value);

              }
              inputImage.push_back(temp);
            }

            for (int i = 0;i < inputImage.size();++i) {
              for (int j = 0;j < inputImage[0].size();++j) {
                     mini = min(mini, inputImage[i][j]);
                     maxi = max(maxi, inputImage[i][j]);
              }
            }
            cout << mini << " " << maxi << endl;
```

```
            cout << "Min and Max values: " << mini << " " << maxi << endl;
            cout << endl;

            vector<vector<int>> vec(rows, vector<int>(cols, 0));

            for (int i = 0; i < rows; ++i) {
              for (int j = 0; j < cols; ++j) {
                    //vec[i][j] = ((inputImage[i][j] - mini) / static_cast<double>(maxi -
mini)) * 255;
                    vec[i][j] = inputImage[i][j];
              }
            }

            cv::Mat imageMat(vec.size(), vec[0].size(), CV_8UC1);

            for (int i = 0; i < imageMat.rows; ++i) {
              for (int j = 0; j < imageMat.cols; ++j) {
                    imageMat.at<uchar>(i, j) = static_cast<uchar>(vec[i][j]);
              }
            }

            cv::imshow("Image", imageMat);
            cv::imwrite("test.jpg", imageMat);

            cv::waitKey(0);
            cv::destroyAllWindows();
            return;
}

void showGrayScale(vector<vector<int>>&fill,int m,int n,int mini,int maxi) {
            vector<vector<int>> vec(m, vector<int>(n, 0));

            for (int i = 0; i < m; ++i) {
              for (int j = 0; j < n; ++j) {
                    vec[i][j] = ((fill[i][j] - mini) / static_cast<double>(maxi - mini))
* 255;

              }
            }

            cv::Mat imageMat(vec.size(), vec[0].size(), CV_8UC1);

            for (int i = 0; i < imageMat.rows; ++i) {
              for (int j = 0; j < imageMat.cols; ++j) {
                    imageMat.at<uchar>(i, j) = static_cast<uchar>(vec[i][j]);
              }
            }

            cv::imshow("Image", imageMat);
            cv::waitKey(0);
            cv::imwrite("output1_gray.jpg", imageMat);
            cv::destroyAllWindows();
            return;
}
void algorithm(vector<vector<int>>& grid, int m, int n) {
            vector<vector<int>>vis(m, vector<int>(n, 0));

            priority_queue<load*, vector<load*>, myCmp>pq;


            int left, right, top, bottom;
```

```
left = top = 0;
right = m - 1;
bottom = n - 1;
//int k = 1;

for (int i = 0;i < m;i++) {
  for (int j = 0;j < n;j++) {
        load* edges = new load(i, j, grid[i][j]);
        pq.push(edges);
  }
}
```

```
int delrow[8] = { -1, -1, 0, 1, 1, 1, 0, -1 };
int delcol[8] = { 0, 1, 1, 1, 0, -1, -1, -1 };
vector<vector<int>>fill(m, vector<int>(n, -1));

while (!pq.empty()) {

  auto node = pq.top();
  pq.pop();


  int ele = node->ele;
  int r = node->x;
  int c = node->y;
  pair<pair<double, int>, pair<int, int>>pr = { {0.00, -1}, { -1, -1} };
  int dir = -1;
  for (int i = 0; i < 8; i++) {
        int nrow = r + delrow[i];
        int ncol = c + delcol[i];
        if (nrow >= 0 && ncol >= 0 && nrow < m && ncol < n) {
              findSteepest(r, c, nrow, ncol, grid, i, pr);
        }
  }
  if (pr.second.first != -1 && pr.second.second != -1) {
        fill[r][c] = pow(2, pr.first.second);
  }
}

int mini = +1e9;
int maxi = -1e9;

//cout << fill[0][2];
for (int i = 0;i < fill.size();++i) {
  for (int j = 0;j < fill[0].size();++j) {
        mini = min(mini, fill[i][j]);
        maxi = max(maxi, fill[i][j]);
  }
}
cout << mini << " " << maxi << endl;
for (int i = 0;i < 10;++i) {
  for (int j = 0;j < 10;++j) {
        cout << fill[i][j] << " ";
  }
  cout << endl;
}
```

```
            //complete->dfs(fill);



            int rows = fill.size();
            int cols = fill[0].size();
            cv::Mat image(rows, cols, CV_8UC3);



            showGrayScale(fill, m, n, mini, maxi);
            // Assign colors based on values
            for (int i = 0; i < rows; ++i) {
              for (int j = 0; j < cols; ++j) {
                    int value = fill[i][j];
                    if (value == -1)        image.at<cv::Vec3b>(i, j) = cv::Vec3b(0, 0,
0);       // Black
                    else if (value == 1)    image.at<cv::Vec3b>(i, j) = cv::Vec3b(0, 0,
255);     // Red
                    else if (value == 2)    image.at<cv::Vec3b>(i, j) = cv::Vec3b(0, 255,
0);     // Green
                    else if (value == 4)    image.at<cv::Vec3b>(i, j) = cv::Vec3b(255, 0,
0);     // Blue
                    else if (value == 8)    image.at<cv::Vec3b>(i, j) = cv::Vec3b(0, 255,
255);   // Yellow
                    else if (value == 16)   image.at<cv::Vec3b>(i, j) = cv::Vec3b(0, 128,
255);   // Orange
                    else if (value == 32)   image.at<cv::Vec3b>(i, j) = cv::Vec3b(255, 0,
255);   // Purple
                    else if (value == 64)   image.at<cv::Vec3b>(i, j) = cv::Vec3b(255,
128, 0);   // Pink
                    else if (value == 128)  image.at<cv::Vec3b>(i, j) = cv::Vec3b(255,
255, 255); // White
              }
            }

            // Display the image
            cv::imshow("Colored Image", image);
            cv::imwrite("output1_Color.jpg", image);
            cv::waitKey(0);




            return;
}


int main(int argc, char** argv)
{
            // Read the image file with IMREAD_ANYDEPTH flag
            Mat image =
cv::imread("C:/Users/KIIT/source/repos/ConsoleApplication1/ConsoleApplication1/newTest.ti
f", cv::IMREAD_ANYDEPTH);

            if (image.empty()) // Check for failure
            {
              cout << "Could not open or find the image" << endl;
              system("pause"); // Wait for any key press
              return -1;
```

```
            }

            int rows = image.rows;
            int cols = image.cols;
            cout << "Rows and cols:" << endl;
            cout << rows << " " << cols << endl;
            cout << endl;


            vector<vector<int>>temp;
            for (int i = 0; i < rows; ++i) {
              vector<int> temp1;
              for (int j = 0; j < cols; ++j) {
                     int pixel_value = static_cast<int>(image.at<ushort>(i, j)); // Use
ushort for 16-bit TIFF
                     temp1.push_back(pixel_value);


              }
              temp.push_back(temp1);
            }
            cout << "Few Entries of the raster image: "<<endl;
            for (int i = 0;i < 10;i++) {
              for (int j = 0;j < 10;j++) {
                     cout << temp[i][j] << " ";
              }
              cout << endl;
            }
            cout << endl;


            cout << "Press 1 to see the processed Raster image's Grayscaled view: " <<
endl;
            cout << "Press 2 to see the modified Flood Fill algo on the Raster image: " <<
endl;

            int n;
            cin >> n;
            if (n == 1) {
              checkResult();
            }

            else if (n == 2) {
              cout << "Wait this will take a few minutes.The Results will be available
soon have patience." << endl;
              algorithm(temp, temp.size(), temp[0].size());
            }

            else cout << "run the program again ! wrong key pressed." << endl;




            return 0;
}
```

Watershed Delineation and Pit Detection Using Modified Flood Fill Algorithm

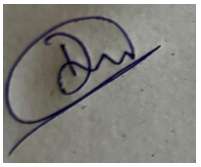
Dhruv Kumar Mishra
2105961

Abstract: In this project, we explore the efficiency of various watershed algorithmic approaches and models. The current paper explores a revised Watershed Algorithm designed for identifying water flow patterns using raster images as input. The algorithm, as introduced in this study, has been executed across extensive regions, encompassing portions of Canada and the USA. This paper delineates the detailed algorithmic methodology for processing raster images.It effectively scans entire GIS TIFF images and transforms them into refined representations, depicting pits, flow directions, flat areas, and river basins with success. Moreover, the proposed approach demonstrates highly efficient processing times, particularly beneficial for handling large-scale image datasets

Individual contribution and findings: I was involved in the **research** of the results and whether the research goals of the project had been met. Along with my peer, I worked with **research of old methodologies** and suggested improvements for the project. I was also responsible for data curation and collecting the dataset required for the project to be implemented on. I worked on **reviewing and proofreading** the entire project report. Moreover, I was responsible for the **visualization of the results** and making sure that the correct plots were displayed so that they were intuitive and demonstrated the results that we achieved. Additionally, I contributed to **designing the algorithm** and **submitted the abstract images** in the project.

Individual contribution to project report preparation: I was responsible for writing the Chapter 1 (Introduction), Chapter 2 (Literature Review)

Full Signature of Supervisor: Full signature student:

	
(Dr. Siddharth Swarup Rautaray)	(Dhruv Kumar Mishra)

Watershed Delineation and Pit Detection Using Modified Flood Fill Algorithm

Rajbeer Chandra
2105987

Abstract: In this project, we explore the efficiency of various watershed algorithmic approaches and models. The current paper explores a revised Watershed Algorithm designed for identifying water flow patterns using raster images as input. The algorithm, as introduced in this study, has been executed across extensive regions, encompassing portions of Canada and the USA. This paper delineates the detailed algorithmic methodology for processing raster images.It effectively scans entire GIS TIFF images and transforms them into refined representations, depicting pits, flow directions, flat areas, and river basins with success. Moreover, the proposed approach demonstrates highly efficient processing times, particularly beneficial for handling large-scale image datasets

Individual contribution and findings: I was involved in the **validation** of the results and my role encompassed a comprehensive approach to **data processing**, ensuring the integrity and accuracy of the data before it underwent analysis. I spearheaded the Algorithm Development, which facilitated efficient **computational execution**. This led to the **generation of results** that were not only reliable but also pivotal in drawing meaningful conclusions from our study. Subsequently, I focused on **result visualization**, employing advanced visualization tools to represent our findings in a clear and impactful manner.

Individual contribution to project report preparation: I was responsible for writing the Chapter 5 (Implementation and Result), Chapter 6 (Conclusion), and PPT presentation and also its demonstration.

Full Signature of Supervisor: Full signature student:

	
(Dr. Siddharth Swarup Rautaray)	(Rajbeer Chandra)

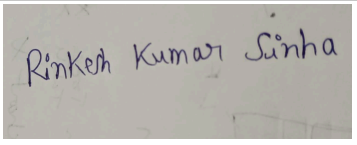
Watershed Delineation and Pit Detection Using Modified Flood Fill Algorithm

Rinkesh Kumar Sinha
2105989

Abstract: In this project, we explore the efficiency of various watershed algorithmic approaches and models. The current paper explores a revised Watershed Algorithm designed for identifying water flow patterns using raster images as input. The algorithm, as introduced in this study, has been executed across extensive regions, encompassing portions of Canada and the USA. This paper delineates the detailed algorithmic methodology for processing raster images.It effectively scans entire GIS TIFF images and transforms them into refined representations, depicting pits, flow directions, flat areas, and river basins with success. Moreover, the proposed approach demonstrates highly efficient processing times, particularly beneficial for handling large-scale image datasets

Individual contribution and findings: I was involved in the **validation** of the results and whether the research goals of the project had been met. Along with my peer, I was responsible for building the proper **Algorithm** through programming so that the project could be implemented. I was also responsible for **data curation** and collecting the dataset required for the project to be implemented on. I worked on **reviewing** and **proofreading** the entire project report. Moreover, I was responsible for the **visualization** of the results and making sure that the correct results were displayed so that they were intuitive.

Individual contribution to project report preparation: I was responsible for writing Chapter 3 (Problem Statement), and Chapter 4 (Methodology).

Full Signature of Supervisor:	Full signature student:
	
(Dr. Siddharth Swarup Rautaray)	(Rinkesh Kumar Sinha)

Plagiarism Report

ORIGINALITY REPORT

15%	6%	12%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	R Turcotte, J.-P Fortin, A.N Rousseau, S Massicotte, J.-P Villeneuve. "Determination of the drainage structure of a watershed using a digital elevation model and a digital river and lake network", Journal of Hydrology, 2001 Publication	5%
2	Tao Wu, Jiaye Li, Tiejian Li, Bellie Sivakumar, Ga Zhang, Guangqian Wang. "High-efficient extraction of drainage networks from digital elevation models constrained by enhanced flow enforcement from known river maps", Geomorphology, 2019 Publication	3%
3	www.coursehero.com Internet Source	<1%
4	answers.opencv.org Internet Source	<1%

5	embeddednesia.com Internet Source	<1%
6	github.com Internet Source	<1%

7	Submitted to Coventry University Student Paper	<1%
8	tem Internet Source	<1%
9	fastercapital.com Internet Source	<1%
10	codesearch.isocpp.org Internet Source	<1%
11	"Advances in Digital Terrain Analysis", Springer Science and Business Media LLC, 2008 Publication	<1%
12	Submitted to University of Southern California Student Paper	<1%

13	www.epa.gov Internet Source	<1%
14	yihuad.blogspot.com Internet Source	<1%
15	libaoj.in Internet Source	<1%
16	www.researchsquare.com Internet Source	<1%
17	John P. Wilson. "Calculating Land Surface Parameters", Wiley, 2018 Publication	<1%

18	docs.opencv.org Internet Source	<1%
19	Submitted to University College London Student Paper	<1%
20	dev.to Internet Source	<1%

21	pastebin.com Internet Source	<1 %
22	www.geeksforgeeks.org Internet Source	<1 %
23	John P. Wilson. "Comparison of the performance of flow-routing algorithms used in GIS-based hydrologic analysis", Hydrological Processes, 04/15/2007 Publication	<1 %
24	Submitted to Rochdale Sixth Form College Student Paper	<1 %
25	mafiadoc.com Internet Source	<1 %
26	vdoc.pub Internet Source	<1 %
27	upcommons.upc.edu Internet Source	<1 %

28	Yun Seok Choi, Mun-Ju Shin, Kyung Tak Kim. "A Study on a Simple Algorithm for Parallel Computation of a Grid-Based One-Dimensional Distributed Rainfall-Runoff Model", KSCE Journal of Civil Engineering, 2020 Publication	<1 %
29	Ariza-Villaverde, A.B., F.J. Jiménez-Hornero, and E. Gutiérrez de Ravé. "Influence of DEM resolution on drainage network extraction: A multifractal analysis", Geomorphology, 2015. Publication	<1 %
30	Lai, Zhengqing, Shuo Li, Guonian Lv, Zhirong Pan, and Guosong Fei. "Watershed delineation using hydrographic features and a DEM in plain river network region : Watershed delineation in plain river network region", Hydrological Processes, 2015. Publication	<1 %
31	Ira J. Greenberg. "The Essential Guide to Processing for Flash Developers", Springer Science and Business Media LLC, 2009 Publication	<1 %

32	kuangbin.github.io Internet Source	<1 %
33	www.health.co.trumbull.oh.us Internet Source	<1 %
34	digital.library.unt.edu Internet Source	<1 %
35	Submitted to Asian Institute of Technology Student Paper	<1 %
36	unfccc.int Internet Source	<1 %
37	www.researchgate.net Internet Source	<1 %
38	doczz.net Internet Source	<1 %
39	citeseerx.ist.psu.edu Internet Source	<1 %

40	ep3.nuwm.edu.ua Internet Source	<1 %
41	marydinahfoundation.org Internet Source	<1 %
42	zhyu.me Internet Source	<1 %
43	dokumen.pub Internet Source	<1 %
44	online-judge.uva.es Internet Source	<1 %
45	technodocbox.com Internet Source	<1 %
46	Persson, M.. "Fusion of aerial images and sensor data from a ground vehicle for improved semantic mapping", Robotics and Autonomous Systems, 20080630 Publication	<1 %
47	codewarsbcn.hpcloud.hp.com Internet Source	<1 %

47	codewarsbcn.hpcloud.hp.com Internet Source	<1%
48	mysql.orst.edu Internet Source	<1%
49	origin.geeksforgeeks.org Internet Source	<1%
50	Rui Bai, Tiejian Li, Yuefei Huang, Jiaye Li, Guangqian Wang. "An efficient and comprehensive method for drainage network extraction from DEM with billions of pixels using a size-balanced binary search tree", Geomorphology, 2015 Publication	<1%