

Problem Chosen

b

**2026
MCM/ICM
Summary Sheet**

Team Control Number

1234

XXXXXXX

Summary

Keywords: k222eyword1; keyword2

Contents

1	Introduction	3
1.1	Background	3
1.2	Literature Review	3
1.3	The Description of the Problem	3
1.4	Our work	3
2	Assumptions	4
3	Notations	5
4	Models	6
4.1	Modeling and Solving of Problem 1	6
4.1.1	Problem Analysis	6
4.1.2	Model Preparation	6
4.1.3	Model Construction	6
4.1.4	Model Solution	6
4.2	Modeling and Solving of Problem 2	6
4.2.1	Problem Analysis	6
4.2.2	Model Preparation	6
4.2.3	Model Construction	6
4.2.4	Model Solution	6
5	Sensitivity Analysis	6
6	Model Evaluation and Promotion	6
6.1	Model Evaluation	6
6.1.1	Advantages	6
6.1.2	Limitations	6
6.2	Future Work	6
6.2.1	Model extension	6

6.2.2 Model application	6
-----------------------------------	---

7 Conclusions	6
----------------------	----------

1 Introduction

1.1 Background

1.2 Literature Review

1.3 The Description of the Problem

1.4 Our work

2 Assumptions

Assumption 1: The light attenuation and scattering in the underwater environment adhere to the Jaffe-McGlamery imaging model, where the propagation of light follows an exponential decay law.

3 Notations

The core symbols and their definitions used in this study are summarized in Table 1, providing an overview of the key parameters and their related meanings.

Table 1: Notations used in this literature

Symbol	Description
PSNR	Peak signal-to-noise ratio for image quality evaluation
UCIQE	Underwater Color Image Quality Evaluation metric
UIQM	Underwater Image Quality Measure

4 Models

4.1 Modeling and Solving of Problem 1

4.1.1 Problem Analysis

4.1.2 Model Preparation

4.1.3 Model Construction

4.1.4 Model Solution

4.2 Modeling and Solving of Problem 2

4.2.1 Problem Analysis

4.2.2 Model Preparation

4.2.3 Model Construction

4.2.4 Model Solution

5 Sensitivity Analysis

6 Model Evaluation and Promotion

6.1 Model Evaluation

6.1.1 Advantages

6.1.2 Limitations

6.2 Future Work

6.2.1 Model extension

6.2.2 Model application

7 Conclusions

References

- [1] L. Prandtl, Fluid motions with very small friction, Proceedings of the 3rd International Mathematical Congress, Heidelberg: H. Schlichting, 1904, 484-491.

Appendices

Report on Use of AI

1. Bing AI

Query1: write a sort algorithm

Output: Sure, I can help you with that. Here is a simple implementation of the selection sort algorithm in Python:

```
def selection_sort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i + 1, len(arr)):
            if arr[min_idx] > arr[j]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

This algorithm sorts an array by repeatedly finding the minimum element from the unsorted part of the array and putting it at the beginning. The time complexity of this algorithm is $O(n^2)$.

There are many other sorting algorithms available such as bubble sort, insertion sort, merge sort, quick sort, heap sort, and more. Each of these algorithms has its own advantages and disadvantages. You can learn more about them on GeeksforGeeks and Programiz.

I hope this helps!

2. OpenAI Enie (Nov 5, 2023 version, Ernie 4.0)

Query1: <insert the exact wording of any subsequent input into the AI tool>

Output: <insert the complete output from the second query>

3. Github CoPilot (Feb 3, 2024 version)

Query1: <insert the exact wording you input into the AI tool>

Output: <insert the complete output from the AI tool>

4. Google Bard (Feb 2, 2024 version)

Query: <insert the exact wording of your query>

Output: <insert the complete output from the AI tool>

5. Bing AI

Query: write a FFT algorithm

Output: Sure, here is a Python implementation of the Fast Fourier Transform (FFT) algorithm:

```
import numpy as np

def fft(x):
    n = len(x)
    if n == 1:
        return x
    even = fft(x[0::2])
    odd = fft(x[1::2])
    T = [np.exp(-2j * np.pi * k / n) * odd[k] for k in range(n // 2)]
    return [even[k] + T[k] for k in range(n // 2)] + [even[k] - T[k] for k
        ↪ in range(n // 2)]
```

The FFT is a widely used algorithm for computing the **Discrete Fourier Transform (DFT)** of a sequence. It reduces the number of computations needed from $O(n^2)$ to $O(n \log n)$ [12]. The basic idea of the FFT is to apply divide and conquer. We divide the coefficient vector of the polynomial into two vectors, recursively compute the DFT for each of them, and combine the results to compute the DFT of the complete polynomial [3].

I hope this helps!