



以HSV色彩與影像處理技術 於即時火警預警系統架構之研析

報告者:資管四 11112112 呂若瑄

指導教授:廖國良 教授

摘要

本研究旨在開發一個基於影像處理技術與環境溫度感測的即時火災檢測Web平台。該系統透過OpenCV進行火焰偵測，並結合DHT11溫度感測器，以提高火災預警的準確性。當系統偵測到火焰時，將於Web介面即時顯示影像，並彈出火災警報通知。本研究探討影像處理技術與感測器數據結合的可行性，並透過Flask架設Web伺服器，實現即時影像串流與火災警報系統。本系統可應用於小型監控場域，提供即時火災預警，提高安全防範能力。

目錄

1. 研究動機與背景
2. 研究目的與系統目標
3. 系統整體架構
4. 系統架構與技術方法
5. 火焰偵測技術說明（HSV）
6. 實驗設計與結果
7. 系統限制與挑戰
8. 未來展望
9. 結語與心得
10. 平台展示

研究動機與背景

火災為全球最常見且毀滅性極高的災害之一，不僅對人員安全構成威脅，也造成巨大的財產損失。根據內政部消防署統計，近年來火災發生頻率逐年上升，其中**工業廠房火災**特別嚴重，造成多起重大事故與人命傷亡。2024 年 12 月，台中市大肚區全聯福利中心生鮮倉儲施工現場發生嚴重火警，造成 9 人死亡、多人受傷。事故起因於焊接作業時產生的火花引燃易燃塗料，顯示施工工地在**火源控管與安全監測**方面的不足。2023 年 9 月，屏東科技產業園區某工廠發生爆炸與火災，導致重大人員傷亡與財產損失，凸顯了高風險環境中火災預警系統的必要性。

研究動機與背景

台中全聯倉儲大火9人不治、8人送醫 初判電焊火星掉落地下室引發



Yahoo奇摩（即時新聞）

更新時間：2024年12月19日



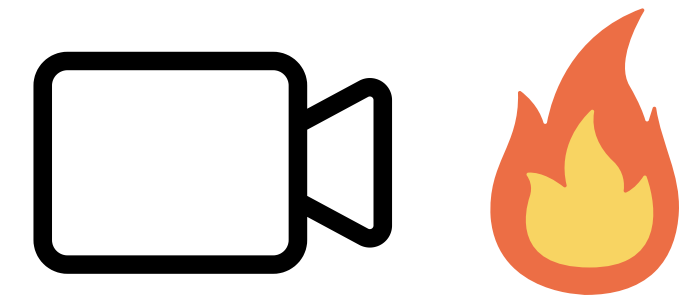
台中大肚區全聯倉儲今天大火，經消防搶救，有9人不治、8人送醫；台中市消防局長孫福佑晚間接受媒體聯訪時表示，根據掌握跡證，初步研判3樓施作電焊樓梯工程時，火星從樓梯間隙掉落地下室，地下室正進行地板工程，引燃油漆混合物引發火警，目前已掌握相關照片與事證，明天會進一步確認起火原因。

屏東明揚工廠爆炸24小時，4消防員殉職、98傷，揭廠房工安與消防職安缺失



研究動機與背景

傳統火災警報器多依賴**煙霧與高溫感測器**，雖具備基本功能，但容易因環境通風、溫度變化不穩定或感測死角導致偵測延遲或失靈，**無法在火災初期即時反應**。因此，僅依賴傳統偵測方式已難以滿足現代消防安全需求。隨著物聯網（IoT）和影像處理技術的快速發展，現代火災偵測系統已逐漸從單一感測器模式，轉向**多元感測器與即時影像分析**的整合方案。本研究提出以即時影像辨識為核心的火災預警平台，透過**電腦視覺技術**即時分析火焰特徵，提升火災反應的速度與準確度，降低災害損失風險。



研究目的與系統目標

本研究的主要目標是設計一個以HSV色彩模型為基礎的火焰偵測平台

- 即時辨識火焰的色彩與輪廓特徵
- 以Web平台即時顯示畫面與警報訊息
- 提供使用者友善的視覺介面與警報互動
- 降低誤判與漏判機率，提升反應效率
- 建構可擴充、低成本的火警預警架構

此平台結合了Python、OpenCV與Flask技術，並原預計整合DHT11溫度感測器，但因模組故障未能實作，最終專注於影像處理層面。

系統架構與技術方法

1. 影像擷取模組

使用OpenCV調用攝影機，並轉換畫面為HSV 色彩空間

2. 影像處理模組

套用色彩遮罩，判斷紅橙色區域是否達到火焰比例門檻。

3. 輪廓分析模組

藉由findContours()尋找不規則輪廓形狀，排除規則物體。

4. Web顯示與通知模組

使用Flask提供串流與紀錄，JavaScript監控火災狀態並觸發警示。

系統架構與技術方法

1. HSV色彩分析

選擇 H 色相範圍 0 - 50
S飽和度與V明度設定為中高範圍

2. 遮罩比例判斷

若火焰像素比例超過總畫面 2% ,
視為可能火災。

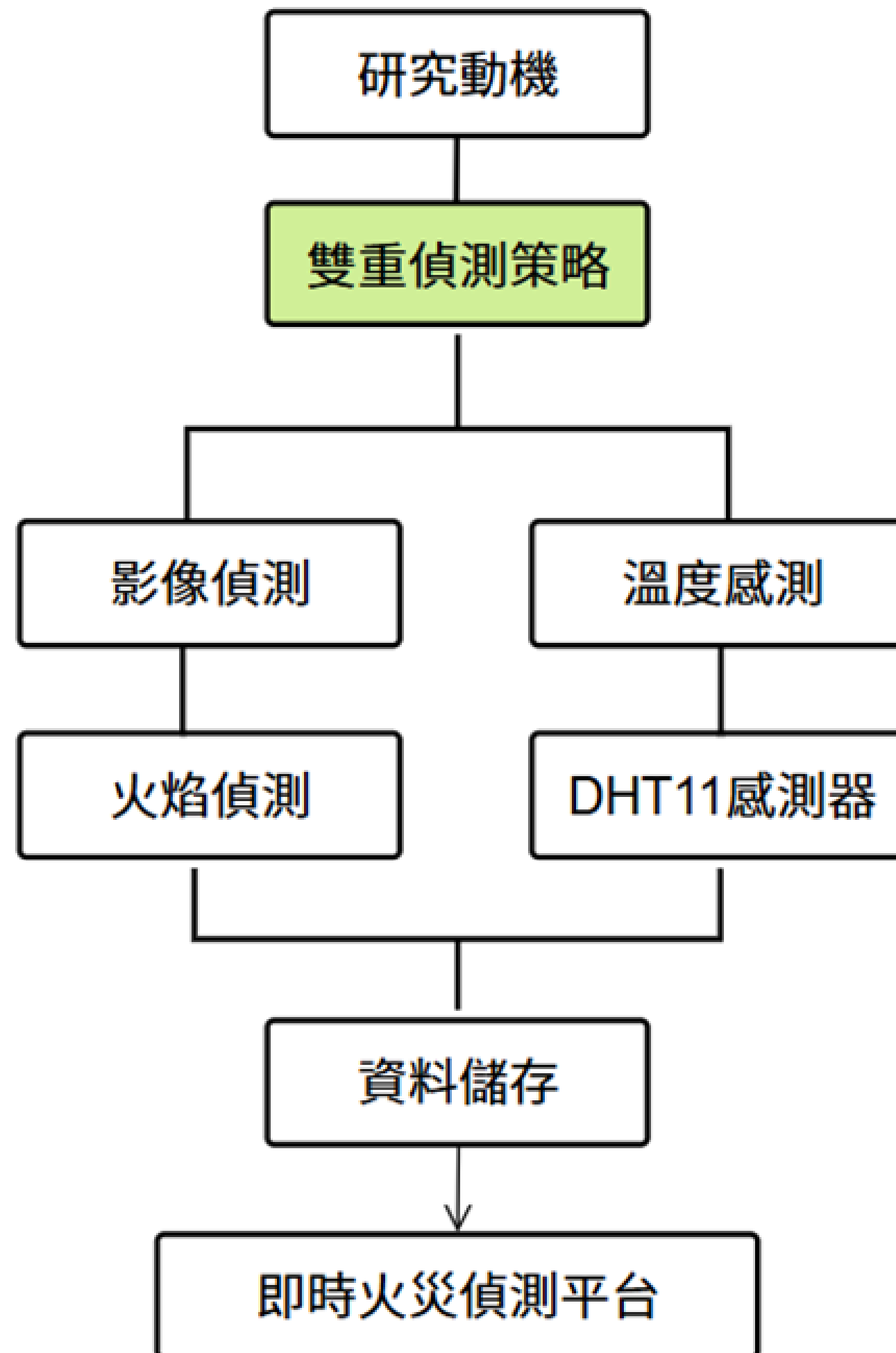
3. 輪廓形狀排除

過濾面積小、形狀固定的干擾物，
例如燈具或紅標誌。

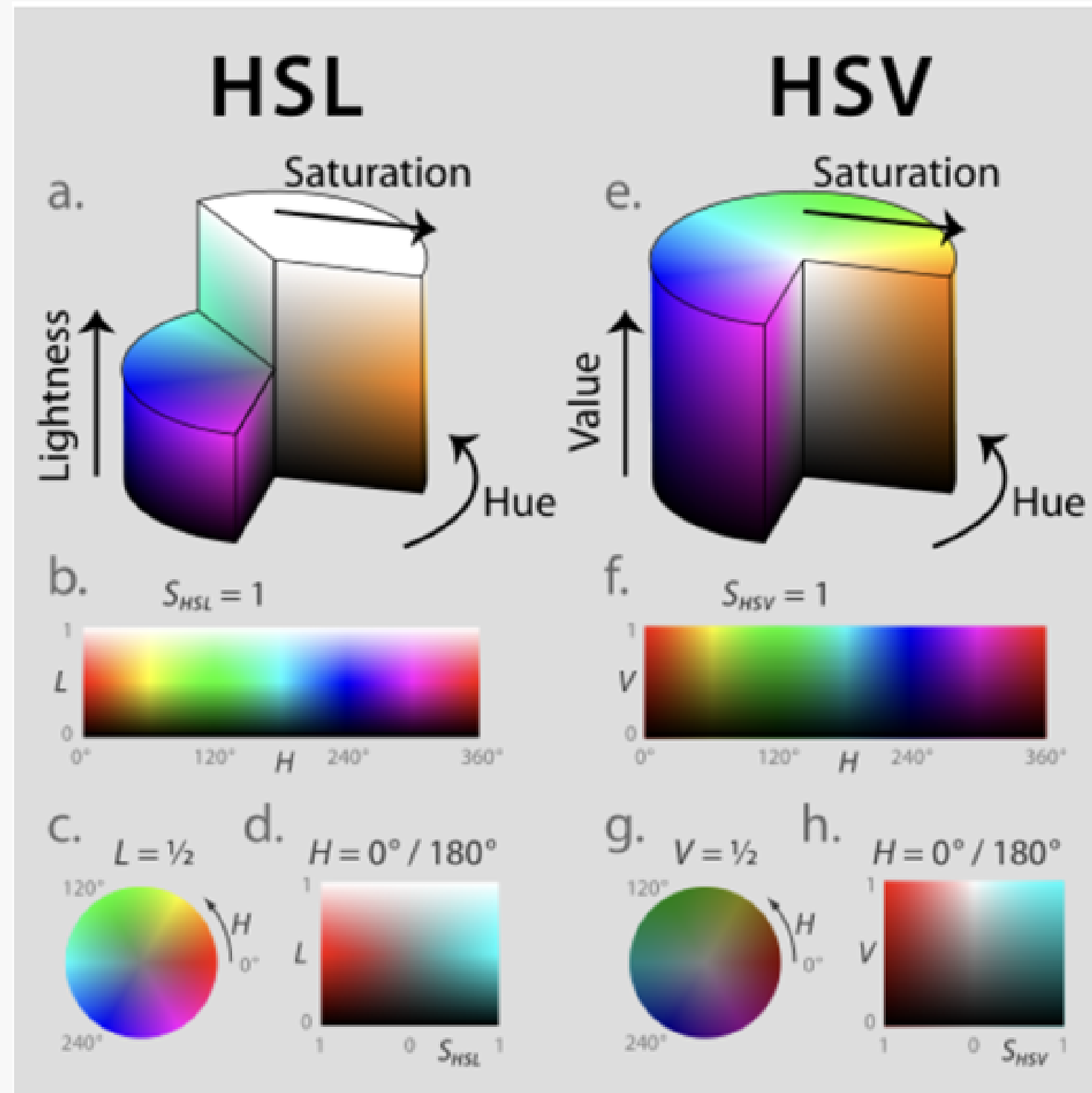
4. 幀間動態分析

透過 `cv2.absdiff()` 判定區域變化程度，排除穩定光源。

系統流程



影像辨識技術：HSV色彩模型



1. 色調 (Hue)

表示顏色的**種類**，例如紅色、藍色或黃色。

色調值範圍為 0° 至 360° ，但在 OpenCV 的 HSV 模型中，色調值被歸一化為 0 至 179。

2. 飽和度 (Saturation)

表示顏色的**純度或強度**，範圍為 0 至 255。

值越高，顏色越鮮豔，值越低，顏色則越趨於灰階。

3. 亮度 (Value)

表示顏色的**亮度或明暗程度**，範圍為 0 至 255。

值越高表示顏色越明亮，值越低表示顏色越暗淡。

1.設定顏色範圍

在 HSV 色彩空間中，紅色和橙色是火焰常見的顏色範圍。程式通過設定紅色和橙色的最小和最大 HSV 範圍來精確選擇火焰顏色：

紅色範圍

```
lower_red = np.array([0, 150, 150])
```

```
upper_red = np.array([10, 255, 255])
```

橘色範圍

```
lower_orange = np.array([10, 150, 150])
```

```
upper_orange = np.array([25, 255, 255])
```



2.二值化 (THRESHOLDING)

- 說明

二值化是將影像中的像素分類為兩類：**前景和背景**。在火焰偵測中，二值化處理有助於簡單且有效地分離火焰區域與背景，進一步提升火焰的識別精度。

- 實際操作

生成遮罩
(mask)

+

遮罩合併

+

輪廓檢測

2.二值化 (THRESHOLDING)

生成遮罩 (mask)

`cv2.inRange` 創建了兩個顏色範圍遮罩：**紅色和橙色**。這些遮罩會將火焰的顏色區域顯示為白色，而其他區域則顯示為黑色。

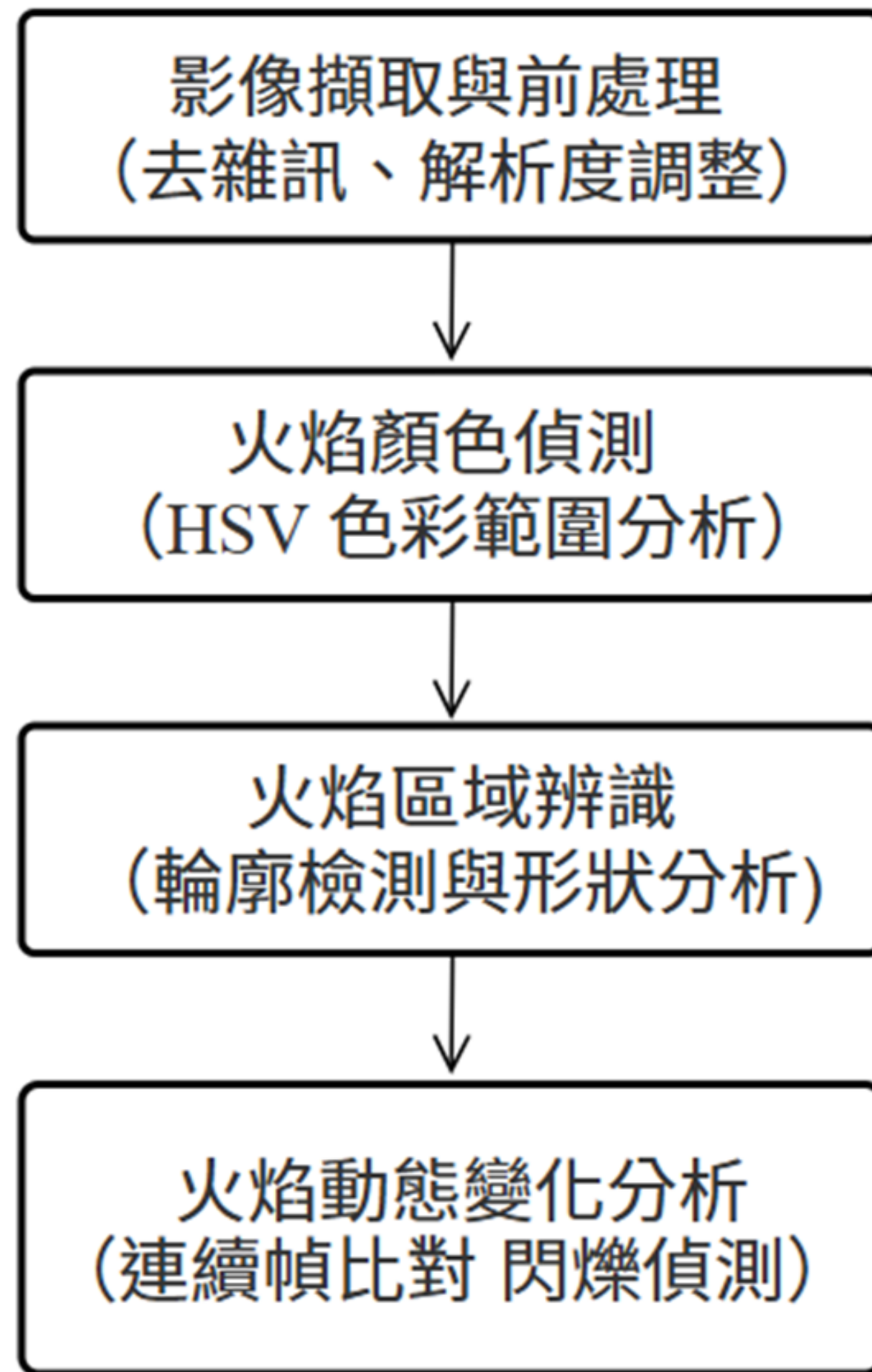
輪廓檢測

由於火焰的顏色可能跨越紅色和橙色範圍，程式將這兩個遮罩使用 `cv2.bitwise_or` **合併成一個遮罩**。這樣可以確保紅色和橙色的火焰都能被提取出來。合併後的遮罩中，火焰的區域將是白色（代表前景），背景則為黑色（代表背景）。

遮罩合併

最後，程式利用 `cv2.findContours` 函數檢測二值化後的影像中的輪廓。**通過輪廓檢測**，程式可以識別火焰區域並進行**面積過濾**，排除小面積的誤報，確保只有真正的火焰被偵測出來。

火焰影像 辨識流程圖



實驗一說明



測試目的

探討在**光線強弱、環境背景不同**的情況下，火焰辨識的穩定度與準確率。

定義誤判：系統在無火焰場景下顯示 FIRE DETECTED!
就記為 FP (FALSE POSITIVE)

不同光照條件測試

• 測試結果統計

• 實驗光照條件與定義：

條件編號	測試情境	描述
C1	自然日光	白天靠窗、有足夠日光照射
C2	昏暗環境	晚上或關燈，低光源情境
C3	逆光場景	火源背後有強光（例如靠窗）

光罩條件	測試次數	正確辨識	漏判次數	誤判次數	準確率
自然日光	10	0	10	10	0%
昏暗環境	10	9	1	1	90%
逆光場景	10	0	10	10	0%

結果：昏暗環境表現最佳，火焰顏色對比明顯

逆光與日光場景中，強光干擾火焰輪廓辨識，準確率大幅下降

實驗二說明

測試目的

測試系統面對**非火焰但具備類似色彩/亮度物體**時的誤判情形。

定義誤判：系統在無火焰場景下顯示 FIRE DETECTED!
就記為 FP (FALSE POSITIVE)

非火焰干擾測試（誤判測試）

• 實驗光照條件與定義：

干擾項目	描述	難度
黃光燈泡	色溫接近火焰，具暖色光源但缺乏動態與輪廓特徵	中
金屬反光 / 水面反射	強烈亮度變化、閃爍感類似火焰光斑，但缺乏不規則輪廓	高
紅橙色靜態物體	色彩與火焰接近，具高誤判潛力但位置穩定無變化	中
紅布隨風擺動	在風力下具備動態與輪廓變化，模擬火焰擺動，難以區分	高
火焰影片	顏色、亮度變動與不規則輪廓相似，極易誤判為火焰	高

非火焰干擾測試（誤判測試）

- 測試結果統計

測試干擾項目	測試次數	誤判次數FP	誤判率
黃光燈泡	10	4	40%
金屬反光 / 水面反射	10	2	20%
紅色靜態物體	10	3	30%
紅布隨風擺動	10	6	60%
火焰影片	10	8	80%

結果：火焰模擬影片與紅布擺動會大幅提升誤判率，
這些對象具有動態與類似火焰的不規則邊緣與色彩，容易混淆。

實驗總結與觀察

本系統在**昏暗環境**下可準確辨識火焰（準確率 90%），但在自然日光與逆光條件下，因強光干擾導致誤判率升高，辨識失準。對於非火焰物體測試中，靜態紅色物與燈泡具備一定排除能力，但對**模擬火焰影片與動態紅布**誤判率仍高（最高達 80%）。顯示目前系統雖可應對部分干擾，但在強光場景與動態假火源仍有辨識盲點。

系統限制與挑戰

- 溫度模組DHT11因故障未實作，僅以影像進行判斷
- 火焰模擬影片、紅布等動態物誤判率仍偏高
- 系統為單鏡頭設計，無多視角、多點支援能力
- 無遠端推播功能，無法即時發送警報至手機或通訊軟體

未來展望

1.技術升級

導入深度學習模型如YOLOv5、CNN進行分類強化
結合紅外線、煙霧、CO感測器進行多模態火災偵測

2.平台升級

將Web平台部署至雲端（如Firebase、AWS）
整合LINE、Telegram、手機APP進行即時推播

3.系統擴充

多鏡頭支援，應用於大型場所
火災影像記錄與AI自動生成通報、統計報表功能

結語與心得

這次做這個即時火災預警系統的專題，從一開始的發想，到實際寫程式、測試、調整，過程中真的遇到很多挑戰。一開始原本還想加上溫度感測器一起做，但後來模組壞掉，就決定專注把影像辨識做好。使用HSV色彩模型搭配OpenCV來判斷火焰，雖然會遇到一些誤判問題，但也學到怎麼透過輪廓分析跟動態偵測去改善。整體下來，我不只學會怎麼寫出一個完整的系統，也更有耐心去處理bug和測試結果。這次專題真的讓我成長很多，也對影像處理這塊有更深入的了解。

VSCODE程式碼 登入介面.html

```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>登入 - 火災即時偵測平台</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      background-color: #fef0ef;
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .login-box {
      background-color: white;
      padding: 40px 50px;
      border: 2px solid #e74c3c;
      border-radius: 15px;
      box-shadow: 0 0 20px rgba(231, 76, 60, 0.3);
      text-align: center;
      width: 400px;
    }
    .login-box h2 {
      color: #c0392b;
      margin-bottom: 30px;
      font-size: 28px;
    }
    .login-box input[type="text"],
    .login-box input[type="password"] {
      width: 90%;
      padding: 15px;
      margin: 15px 0;
      border: 1px solid #ccc;
      border-radius: 8px;
      font-size: 18px;
    }
    .login-box input[type="submit"] {
      width: 100%;
      padding: 15px;
      background-color: #e74c3c;
      color: white;
      border: none;
      border-radius: 8px;
      font-size: 20px;
      cursor: pointer;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="login-box">
    <h2>火災即時偵測平台登入</h2>
    <form method="POST">
      <input type="text" name="username" placeholder="使用者名稱" required><br>
      <input type="password" name="password" placeholder="密碼" required><br>
      <input type="submit" value="登入">
    </form>
    <div class="error">{{ error }}</div>
  </div>
</body>
</html>
```

```

    .login-box input[type="submit"]:hover {
      background-color: #c0392b;
    }
    .error {
      color: red;
      margin-top: 10px;
      font-size: 16px;
    }
  </style>
</head>
<body>
  <div class="login-box">
    <h2>火災即時偵測平台登入</h2>
    <form method="POST">
      <input type="text" name="username" placeholder="使用者名稱" required><br>
      <input type="password" name="password" placeholder="密碼" required><br>
      <input type="submit" value="登入">
    </form>
    <div class="error">{{ error }}</div>
  </div>
</body>
</html>

t tableBody = document.querySelector('.record-box table');
tableBody.innerHTML = `
  <tr>
    <th>時間</th>
    <th>狀態</th>
  </tr>
`;
data.forEach(record => {
  const row = document.createElement('tr');
  row.innerHTML = `<td>${record.time}</td><td>${record.status}</td>`;
  tableBody.appendChild(row);
});

setInterval(fetchRecords, 3000); // 每3秒刷新一次
window.onload = fetchRecords;
</script>
</body>
</html>
```

VSCODE程式碼 平台介面.html

```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>火災即時偵測平台</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    .header {
      background-color: #c0392b;
      color: white;
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 15px 25px;
    }
    .header h1 {
      margin: 0;
      font-size: 28px;
    }
    .header a {
      color: white;
      text-decoration: none;
      font-size: 18px;
      font-weight: bold;
    }
    .main {
      display: flex;
      height: calc(100vh - 70px); /* 扣掉 header */
    }
    .video-box {
      flex: 2;
      padding: 10px;
    }
```

```
.video-box img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  border: 2px solid #c0392b;
}
.record-box {
  flex: 1;
  padding: 10px;
  background-color: #f9f9f9;
  overflow-y: auto;
  display: flex;
  flex-direction: column;
}
.record-box h3 {
  font-size: 22px;
  margin-bottom: 10px;
  text-align: center;
  color: #c0392b;
}
table {
  width: 100%;
  border-collapse: collapse;
  flex-grow: 1;
}
th, td {
  border: 1px solid #aaa;
  padding: 12px;
  font-size: 16px;
  text-align: center;
}
th {
  background-color: #e74c3c;
  color: white;
  font-size: 18px;
}
</style>
</head>
```

```
<body>
  <div class="header">
    <h1>🔥 火災即時偵測平台</h1>
    <a href="{{ url_for('logout') }}">登出</a>
  </div>
  <div class="main">
    <div class="video-box">
      
    </div>
    <div class="record-box">
      <h3>🕒 偵測紀錄</h3>
      <table>
        <tr>
          <th>時間</th>
          <th>狀態</th>
        </tr>
        {% for record in records %}
        <tr>
          <td>{{ record.time }}</td>
          <td>{{ record.status }}</td>
        </tr>
        {% endfor %}
      </table>
    </div>
  </div>
  <script>
    function fetchRecords() {
      fetch('/api/records')
        .then(response => response.json())
        .then(data => {
          const tableBody = document.querySelector('.record-box table');
          tableBody.innerHTML = `
            <tr>
              <th>時間</th>
              <th>狀態</th>
            </tr>
          `;
          data.forEach(record => {
            const row = document.createElement('tr');
            row.innerHTML = `<td>${record.time}</td><td>${record.status}</td>`;
            tableBody.appendChild(row);
          });
        });
    }

    setInterval(fetchRecords, 3000); // 每3秒刷新一次
    window.onload = fetchRecords;
  </script>
</body>
</html>
```

PYTHON程式碼

app.py

```
from flask import Flask, render_template, Response, jsonify, request, redirect, url_for
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user
import cv2
from datetime import datetime
import time
import numpy as np

app = Flask(__name__)
app.secret_key = 'secret'

# 登入設定
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

USERS = {'admin': '1234'}

class User(UserMixin):
    def __init__(self, username):
        self.id = username

@login_manager.user_loader
def load_user(user_id):
    if user_id in USERS:
        return User(user_id)
    return None

# 相機設定
camera = cv2.VideoCapture(0)
detection_records = []

# 儲存上一幀火焰遮罩
prev_mask = None

# 火焰偵測（加強版）
def detect_fire(frame):
    global prev_mask
    fire_detected = False

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # 紅橙色範圍（改進色域）
    lower1 = np.array([0, 100, 100])
    upper1 = np.array([10, 255, 255])
    lower2 = np.array([160, 100, 100])
    upper2 = np.array([180, 255, 255])
    mask1 = cv2.inRange(hsv, lower1, upper1)
    mask2 = cv2.inRange(hsv, lower2, upper2)
    mask = cv2.bitwise_or(mask1, mask2)
```

```
# 移除雜訊
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

# 動態變化分析
dynamic_score = 0
if prev_mask is not None:
    diff = cv2.absdiff(mask, prev_mask)
    dynamic_score = cv2.countNonZero(diff) / (frame.shape[0] * frame.shape[1])
prev_mask = mask.copy()

# 輪廓分析
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for contour in contours:
    area = cv2.contourArea(contour)
    if area > 1000 and dynamic_score > 0.005: # 面積 + 動態變化
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
        fire_detected = True

return fire_detected

# 串流畫面
def gen_frames():
    last_detect_time = 0
    last_status = "Safe"

    while True:
        success, frame = camera.read()
        if not success:
            break

        now_time = time.time()
        now_str = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        if now_time - last_detect_time >= 3:
            is_fire = detect_fire(frame)
            last_status = "Fire Detected!" if is_fire else "Safe"
            detection_records.append({"time": now_str, "status": last_status})
            last_detect_time = now_time

        cv2.putText(frame, f"{now_str} - {last_status}", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7,
                    (0, 0, 255) if last_status == "Fire Detected!" else (0, 255, 0), 2)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

```
# 登入頁面
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        u, p = request.form['username'], request.form['password']
        if u in USERS and USERS[u] == p:
            login_user(User(u))
            return redirect(url_for('index'))
        return '登入失敗'
    return render_template('login.html')

# 登出
@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

# 首頁
@app.route('/')
@login_required
def index():
    return render_template('index3.html', records=detection_records[-10:])

# 串流影像
@app.route('/video_feed')
@login_required
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

# JSON API
@app.route('/api/records')
@login_required
def api_records():
    return jsonify(detection_records[-10:])

if __name__ == '__main__':
    app.run(debug=True)
```

平台實際畫面展示

使用者登入：系統透過 Flask-Login 實作帳號驗證機制，避免未授權使用。



The screenshot displays a login form titled "火災即時偵測平台登入" (Fire Disaster Real-time Detection Platform Login). It features two input fields: the first contains the username "admin", and the second contains masked characters "....". Below these fields is a prominent red button labeled "登入" (Login).

火災即時偵測平台登入

admin

....

登入

平台實際畫面展示

即時影像串流，顯示攝影機畫面

每三秒偵測一次

🔍 偵測紀錄

時間	狀態
2025-05-19 12:14:12	Safe
2025-05-19 12:14:15	Fire Detected!
2025-05-19 12:14:18	Fire Detected!
2025-05-19 12:14:21	Fire Detected!
2025-05-19 12:14:24	Fire Detected!
2025-05-19 12:14:27	Fire Detected!
2025-05-19 12:14:30	Fire Detected!
2025-05-19 12:14:33	Fire Detected!
2025-05-19 12:14:37	Fire Detected!
2025-05-19 12:14:40	Fire Detected!

平台畫面與功能展示



火災即時偵測平台登入

登入

THANK YOU!

感謝各位的聆聽