ชื่อ Water image processing

ผู้ร่วมงาน

1) ภัทรชนน เมธาวุฒิยาภรณ์ รหัสนักศึกษา 65010814
2) สรยุทธ เปี่ยมรอด รหัสนักศึกษา 65011076

อาจารย์ที่ปรึกษา

1) ผศ. ไพศาล สิทธิโยภาสกุล
2) บุณย์ชนะ ภู่ระหงษ์

ภาควิชาวิศวกรรมคอมพิวเตอร์ หลักสูตรระบบไอโอทีและสารสนเทศ

รายงานนี้เป็นส่วนหนึ่งของวิชา 01236256 MICROCONTROLLER AND EMBEDDED SYSTEMS

ปีการศึกษา 2566

บทนำ

ที่มาและความสำคัญ

เนื่องจากปัญหาน้ำป่าไหลหลากที่เกิดขึ้นในประเทศไทย เกิดขึ้นในทุกๆปีในช่วงฤดูฝนส่งผลกระทบต่อประชากรไทยในหลายพื้นที่ โดยเฉพาะพื้นที่ราบสูงติดป่าหรือน้ำตกซึ่งน้ำป่าไหลหลาก ส่งผลต่อชีวิตการเป็นอยู่ส่งผลต่อ พืชผลทางการเกษตรและชีวิตของประชากรในบริเวณโดยรอบ ทางผู้จัดทำจึงได้คำนึงถึงปัญหาและอยากจะช่วยเพิ่มการป้องการ เพื่อให้ประชากรโดยรอบได้รับรู้ถึงน้ำป่าไหลหลากจากต้นทางหรือเพื่อทำให้ทราบว่าน้ำที่ไหลทางต้นทางมีโอกาสทำให้เกิดน้ำป่าไหลหลาก เพื่อทำการรับมือป้องกันหรือทำการนำประชากรที่ดูแลตัวเองไม่ได้หรือ ลำบากให้อพยบออกจากพื้นที่เหล่านั้นไปก่อน

โดยเครื่องมือที่ทางผู้จัดทำได้คิดขึ้นมาคือ กล้องตรวจจับการเคลื่อนไหวของน้ำ เพื่อนำลักษณะคลื่น และทิศทางการไหลของน้ำเพื่อหาความเร็วและทิศทาง แจ้งเตือนไปที่ศูนย์ในชุมชน เพื่อให้นำมาวิเคราะห์ว่า มีโอกาสเกิดน้ำป่าไหลหลากหรือไหมเพื่อทำการรับมือได้ทันทวงที

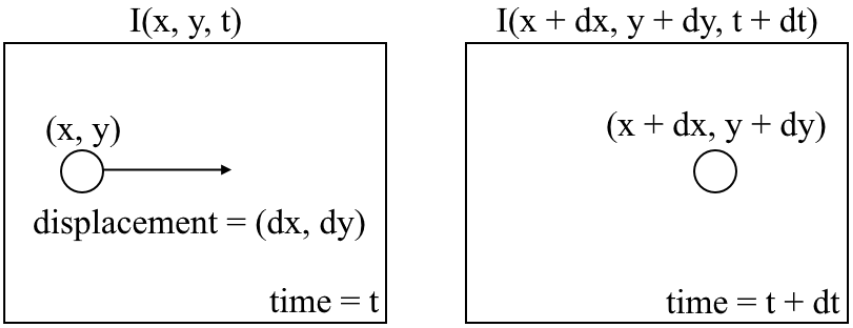
วัตถุประสงค์

- เพื่อสร้างเครื่องมือช่วยเพิ่มการป้องกันและลดการได้รับผลกระทบจากการเกิดอุทกภัย
- เพื่อศึกษาการวิเคราะห์กระแสน้ำไหลโดยวิธีการ image processing
- เพื่อหาแนวทางตรวจวัดลักษณะการไหลของน้ำอย่างครอบคลุมพื้นที่วงกว้าง

สมมติฐาน

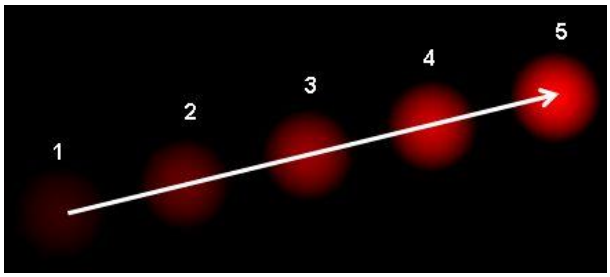
- จะสามารถวิเคราะห์การไหลของน้ำด้วยวิธี Optical Flow ใน image processing ได้
- ช่วยเพิ่มการป้องกันและลดการได้รับผลกระทบจากการเกิดอุทกภัยได้
- สามารถสร้างเครื่องมือที่เหมาะสมสำหรับการตรวจวัดลักษณะการไหลของน้ำได้

หลักการทำงาน



Dense Optical Flow

เป็นรูปแบบของการเคลื่อนไหวที่ชัดเจนของวัตถุภาพระหว่างสองเฟรมติดต่อกันที่เกิดจากการเคลื่อนไหวของวัตถุหรือกล้อง เป็นฟิลด์เวกเตอร์ 2 มิติที่แต่ละเวกเตอร์เป็นเวกเตอร์การกระจัดที่แสดงการเคลื่อนไหวของจุดจากเฟรมแรกไปยังเฟรมที่สอง



ในการทำงานสิ่งที่ผู้ทำการทดลองได้ใช้คือ หลักการ image processing โดยใช้หลักการเรื่อง Dense Optical Flow in OpenCV โดย Lucas-Kanade คำนวณการไหลของแสงสำหรับชุดคุณลักษณะแบบกระจัดกระจาย (ในตัวอย่างของเรา มุมที่ตรวจพบโดยใช้อัลกอริทึม Shi-Tomasi) OpenCV มีอัลกอริธึมอื่นในการค้นหาการไหลของแสงที่หนาแน่น โดยจะคำนวณการไหลของแสงสำหรับจุดทั้งหมดในเฟรม ขึ้นอยู่กับอัลกอริธึมของกุนนาร์ ฟาร์เนแบ็ก ซึ่งอธิบายไว้ใน "การประมาณค่าการเคลื่อนที่แบบสองเฟรมตามการขยายตัวพหุนาม" โดยกุนนาร์ ฟาร์เนแบ็ก ในปี พ.ศ. 2546 ขนาดและทิศทางของมัน เราใส่รหัสสีให้กับผลลัพธ์เพื่อให้เห็นภาพได้ดีขึ้น ทิศทางสอดคล้องกับค่าฮิวของรูปภาพ ขนาดสอดคล้องกับระนาบค่า

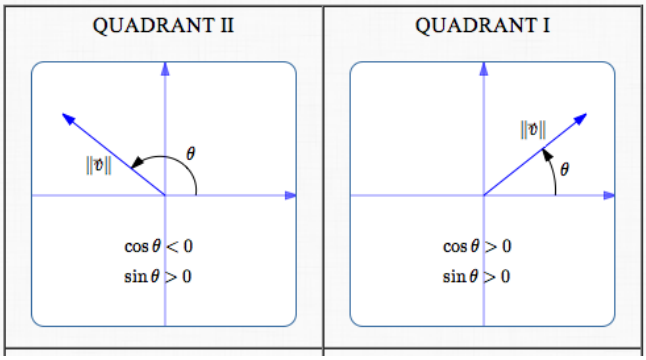
การคำนวณหาความเร็วและทิศทางการไหลของน้ำ สามารถคำนวณได้จากการหาขนาดและมุมเฉลี่ยของแต่ละเวกเตอร์จากสนามเวกเตอร์ที่ประมวลผลผ่านฟังก์ชั่นสำเร็จรูปจาก module opencv นั้นคือ cv2.calcOpticalFlowFarneback() ฟังก์ชั่นนี้จะรีเทิร์นค่าเป็น 2D array ของคู่อันดับ x1,y1 กับ x2,y2 ทำให้สามารถคำนวณได้จากการหาขนาดด้วยสมการ magnitude vector formula ดังต่อไปนี้

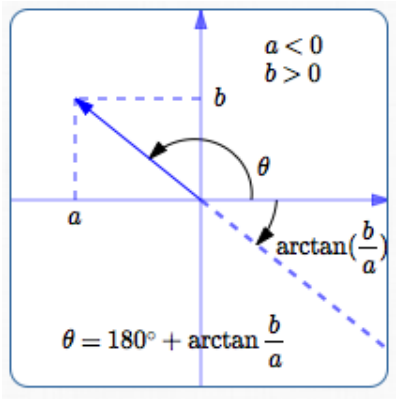
The **magnitude** or **length** of the vector $\mathbf{v} = \overrightarrow{PQ}$ is the nonnegative number

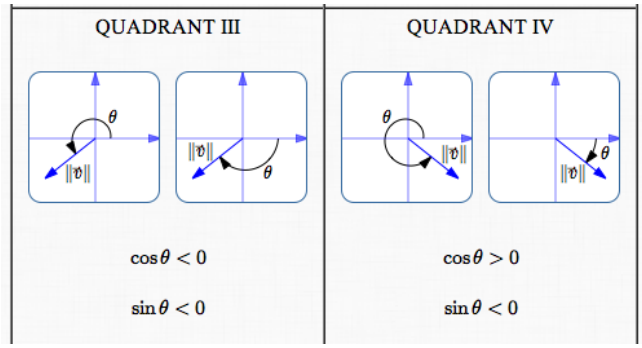
$$|\mathbf{v}| = \sqrt{v_1^2 + v_2^2 + v_3^2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

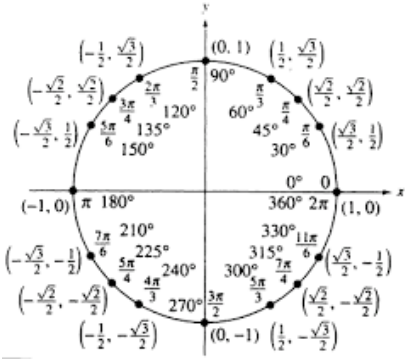
(See Figure 12.10.)

และคำนวณหาแนวโน้มของทิศทางน้ำได้โดยอาศัยหลักการของตรีโกรณมิติและวงกลม 1 หน่วยดังภาพเมื่อ a และ เป็นขนาดของเวกเตอร์เฉลี่ยที่หาได้ และนำไปพล็อตแบบเชิงขั้วโดย module matplotlib









การวิเคราะห์ความแรงของคลื่นน้ำและความรุนแรงของการไหล จะอาศัยหลักการทางสถิติทำการวิเคราะห์จากค่าความแปรปรวนของขนาดเวกเตอร์ในสนามเวกเตอร์ ที่มีค่าที่กระจัดกระจายไม่เป็นระเบียบว่ามีค่ามากหรือน้อยเพียงใด ดังนี้

	Population	Sample
Variance	$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$	$S^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}$
Standard deviation	$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}}$	$S = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}}$

เปรียบทำการคำนวณกับโปรแกรมที่เขียนจริงดังนี้

การหา magnitude vector

```
#vector
U = np.array([lines[t][0][0]-lines[t][1][0] for t in range(len(y))])
V = np.array([lines[t][0][1]-lines[t][1][1] for t in range(len(y))])
ang = []
for i in range(len(V)):
    agcal = V[i] / U[i]
    er = ["-inf","inf","nan"]
    if str(agcal) in er :agcal = 0
    ang.append(agcal)

qd[0] = np.mean(U)
qd[1] = np.mean(V)
mgnitue = (U**2 + V**2)**0.5
angl = np.array([np.arctan(ag) for ag in ang])
MeanAngle = np.mean(angl)* 57.2957795
MeanVelocity = np.mean(mgnitue)
```

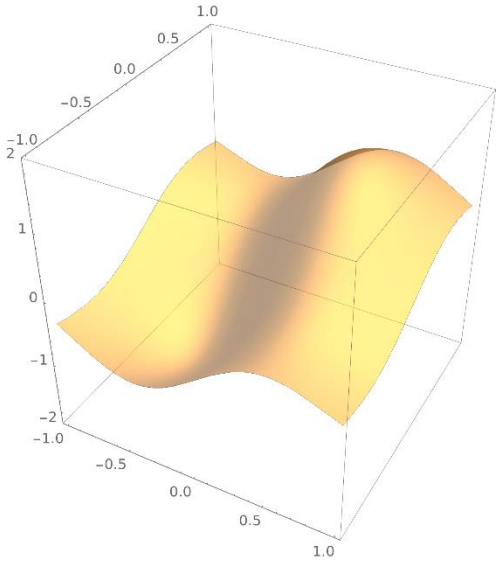
การหา องศาและทิศทางน้ำ

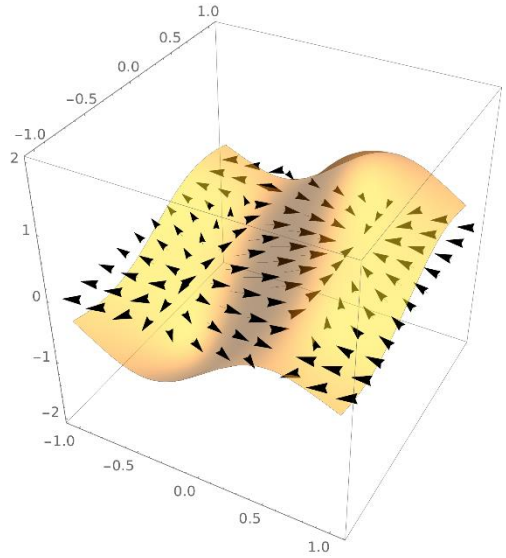
```
icons = ['right',
         'up',
         'left',
         'down']
icon = 'w:direction-'
deg = abs(MeanAngle)

if qd[0] >= 0 and qd[1] >= 0:    # (+,+) quadrant 1
    deg = 0 + deg
    id = int(qd[0]<qd[1])
    icon += icons[id]
elif qd[0] <= 0 and qd[1] >= 0:  # (-,+) quadrant 2
    deg = 180 - deg
    id = int(abs(qd[0]) > qd[1])
    icon += icons[id+1]
elif qd[0] <= 0 and qd[1] <= 0:  # (-,-) quadrant 3
    deg = 180 + deg
    id = int(abs(qd[0]) < abs(qd[1]))
    icon += icons[id + 2]
elif qd[0] >= 0 and qd[1] <= 0:  # (+,-) quadrant 4
    deg = 360 - deg
    id = int(qd[0] > abs(qd[1]))
    icon += icons[id - 1]
```

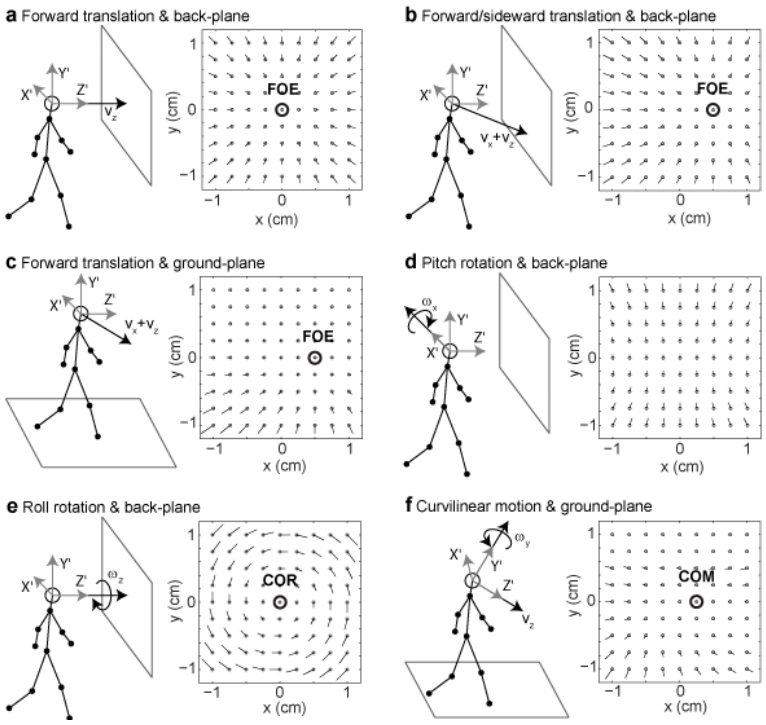
การหาความรุนแรงการไหลของน้ำในที่นี้จะใช้ฟังก์ชั้นสำเร็จรูปของ module เพื่อการวิเคราะห์ทางคณิตศาสตร์และวิทยาศาสตร์อย่าง numpy ด้วยฟังก์ชั่น nump.var() ดังนี้

```
MeanVelocity = np.mean(mgnitue)
VarianFlow = np.var(U)
```

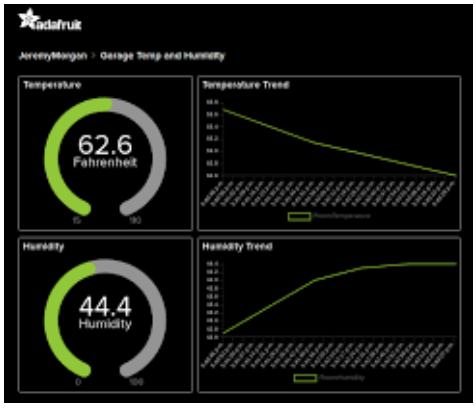
นอกจากนี้ยังใช้ความรู้เรื่อง vector field วิเคราะห์ในระดับที่สูงขึ้นไปเพื่อให้สามารถประยุกต์ใช้งานได้หลากหลาย วิเคราะห์ในหลายมุมมองทำให้มีประสิทธิภาพมากขึ้น เช่น อาจสามารถจำลองพื้นผิวของน้ำและการเคลื่อนไหวในแบบสามิติได้ เป็นต้น







เมื่อคำนวณค่าต่างๆแล้วจึงทำการส่งข้อมูลขึ้น could platform เพื่อเก็บข้อมูลและแสดงผลแบบเรียลไทม์ใน ที่นี้ใช้เป็น IoT platform ของ adafruit สื่อสารด้วย MQTT protocol



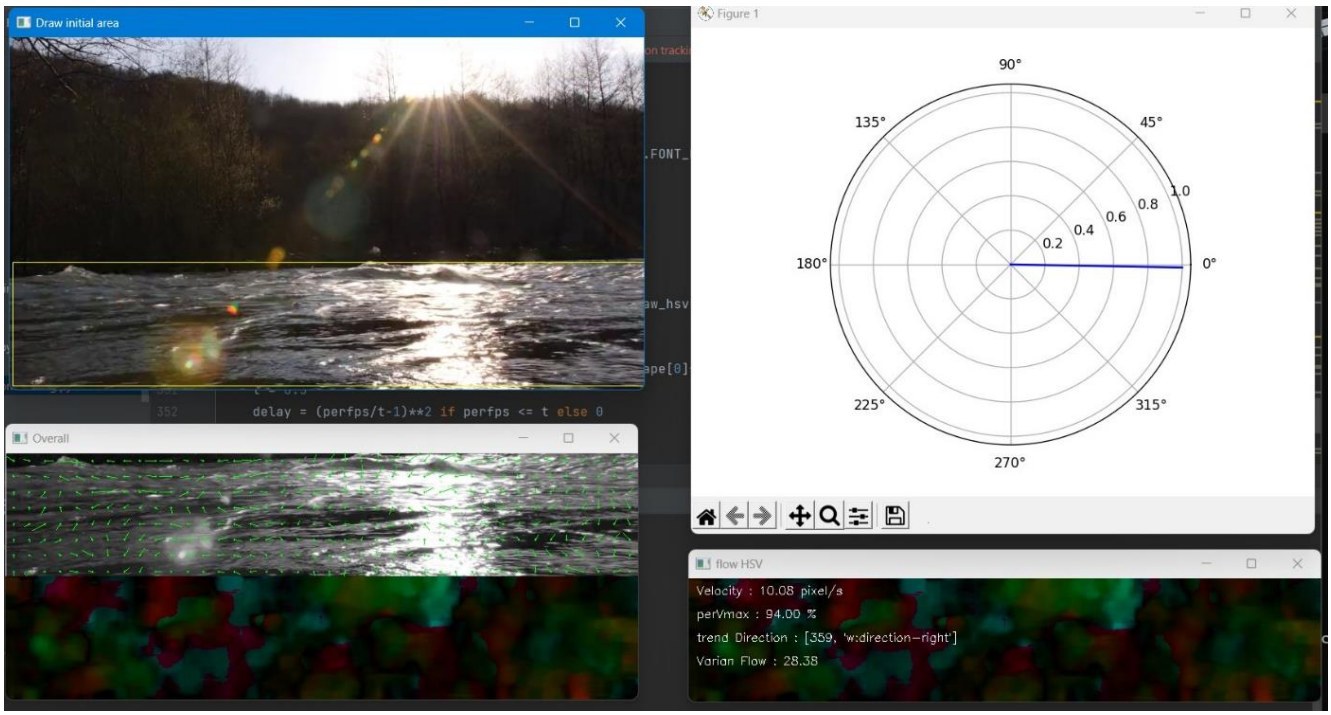
โค้ดส่วนการทำงานเป็นดังนี้

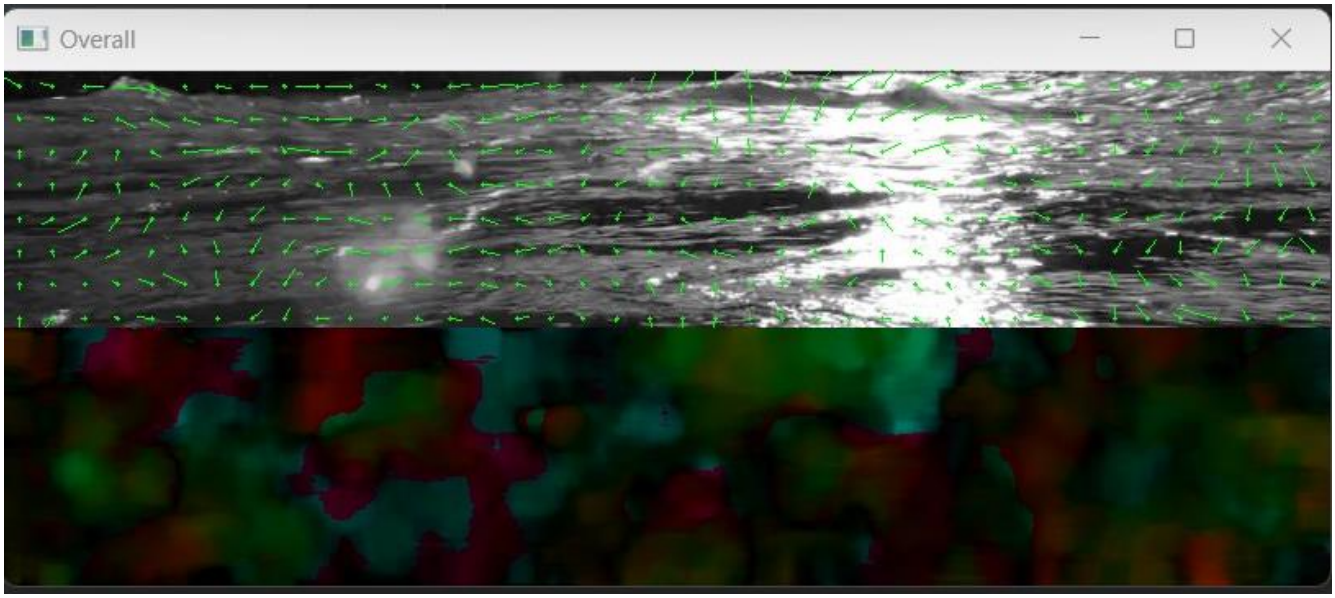
```python
def Up2Adafruit(i):
    feedsname = ['velocity-percent','varian-flow','velocity','trend-direction']
    value = [get_VperMax(),VarianFlow,MeanVelocity,get_direction()[1]]

    #for i in range(k):
    feed = aio.feeds(feedsname[i])
    # aio.send_data(feed.key, value[i])
    aio.append(feed.key, value[i])

    print('\n---------status-------------')
    print(datetime.datetime.now().strftime("%d %b %Y %I:%M:%S%p"))
    print(f'\nVector mean : {qd[0]:.2f} i + {qd[1]:.2f} j')
    print(f'MeanAng : {MeanAngle:.2f}')

    print(f'\nMeanV : {MeanVelocity:.2f} pixel/s')
    print(f'perVmax : {get_VperMax():.2f} %')
    print(f'trend Direction : {get_direction()}')
    print(f'Var Flow : {VarianFlow:.2f}')
```

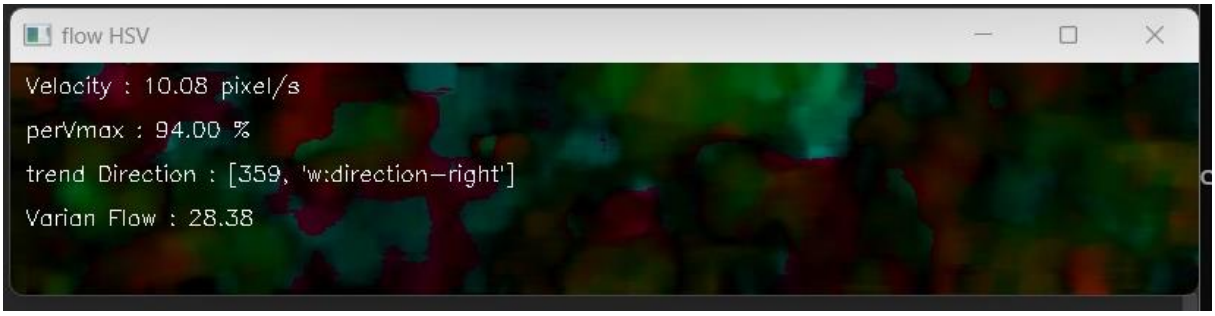
ผลการทำงาน

ทดลองการทำงานโดยใช้คลิปวิดีโอตัวอย่างจาก อินเทอร์เน็ต 1 วิดีโอ และบันทึกภาพจากสถานที่จริง บริเวณคลองประเวศคณะสถาปัตยกรรมสถาบันเทคโนโลยีเจ้าคุณทหารบาดกระบัง มี 2 วิดีโอ มีผลลับดังนี้

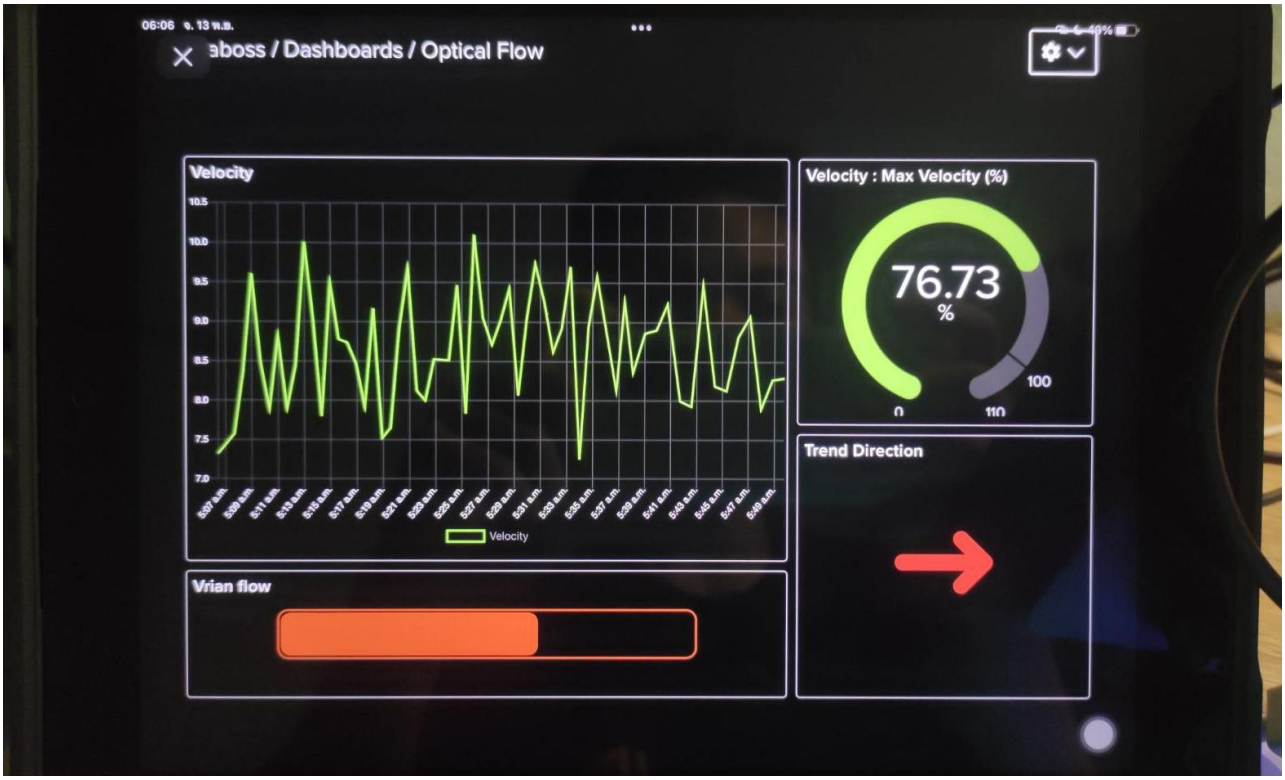
1. sample.mp4 (น้ำไหลแรง ทิศทางไปทางขวา)



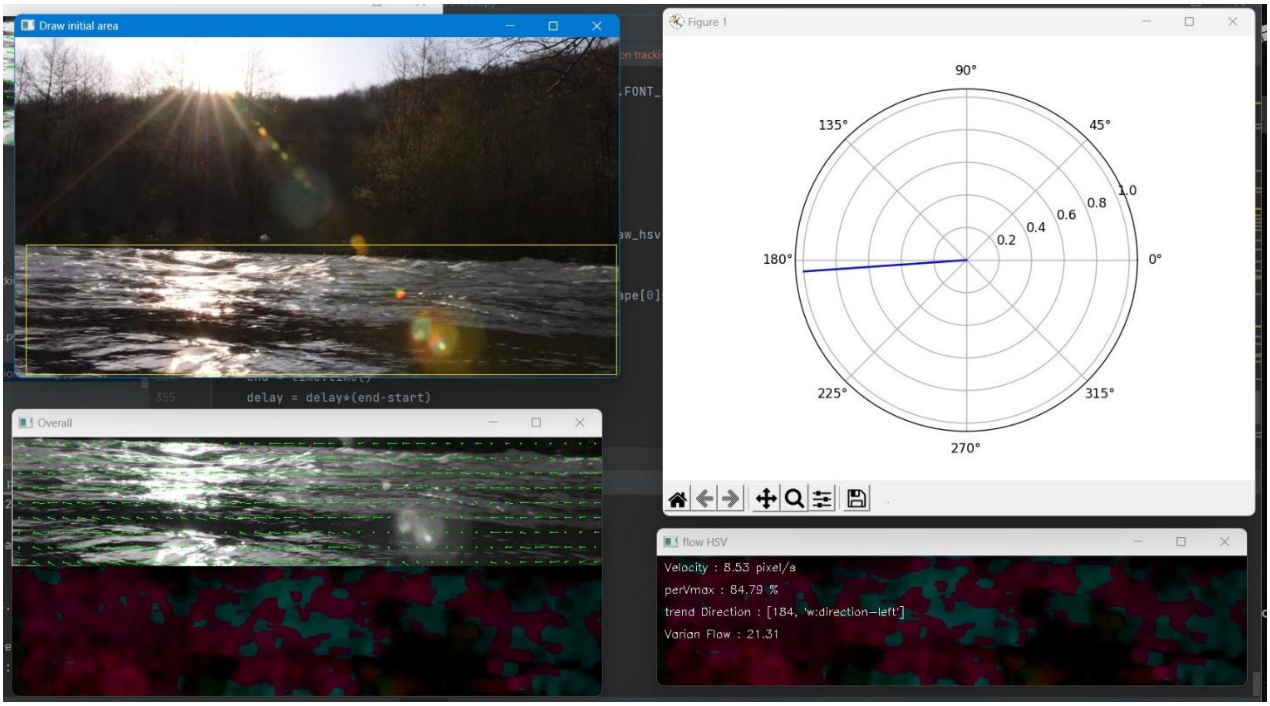


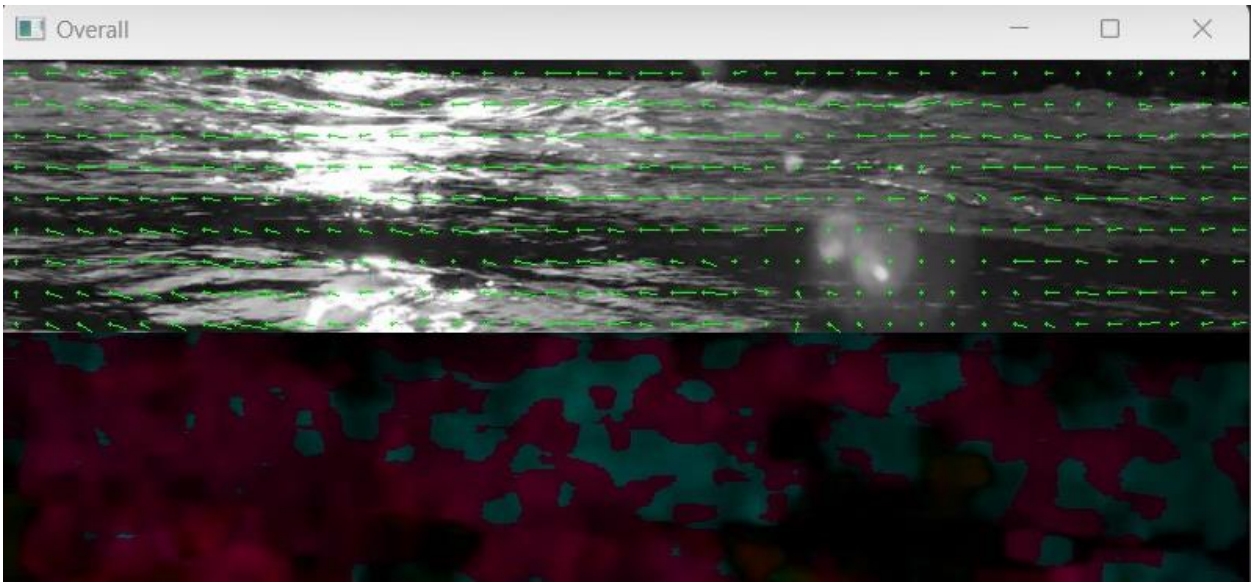


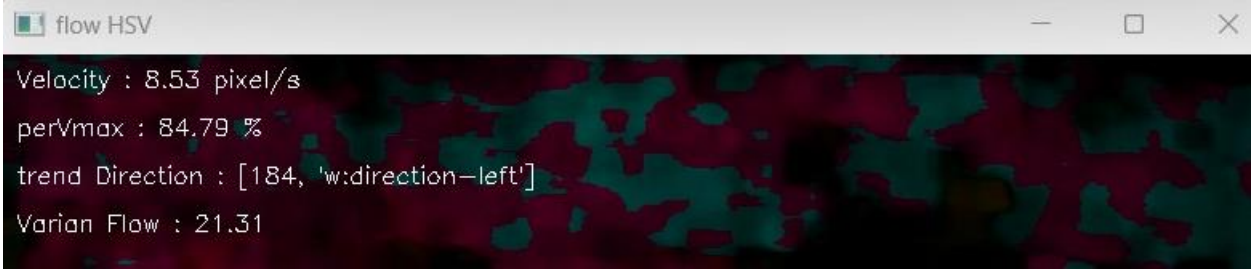
แดชบอร์ดแสดงสถานะต่างๆ



2. sample.mp4 (น้ำไหลแรง ทิศทางไปทางซ้าย) flip Video

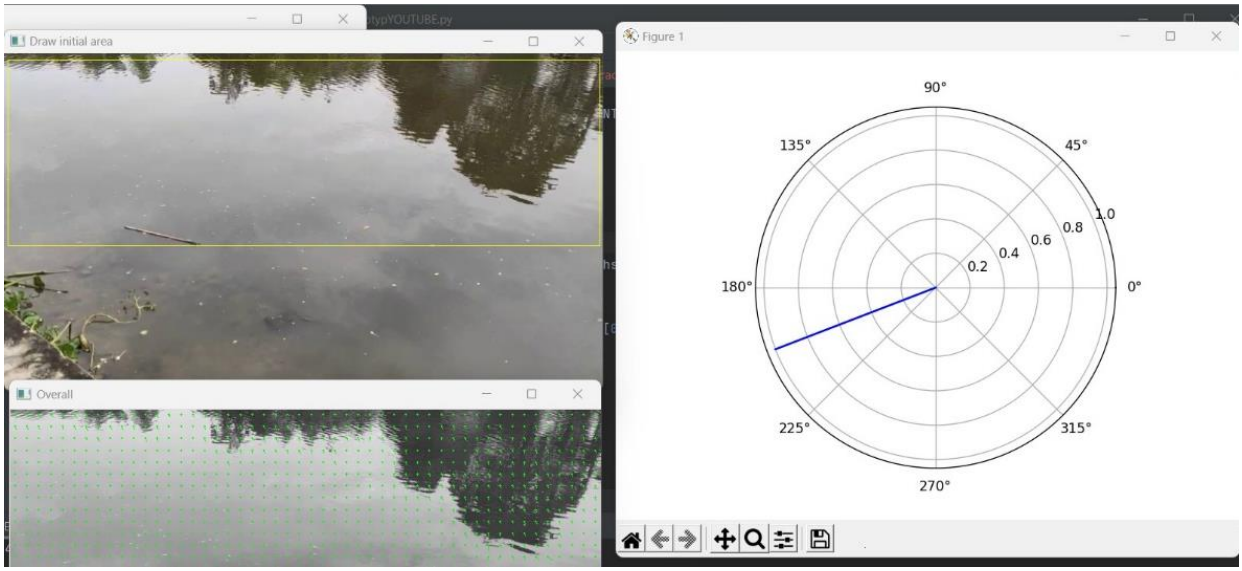


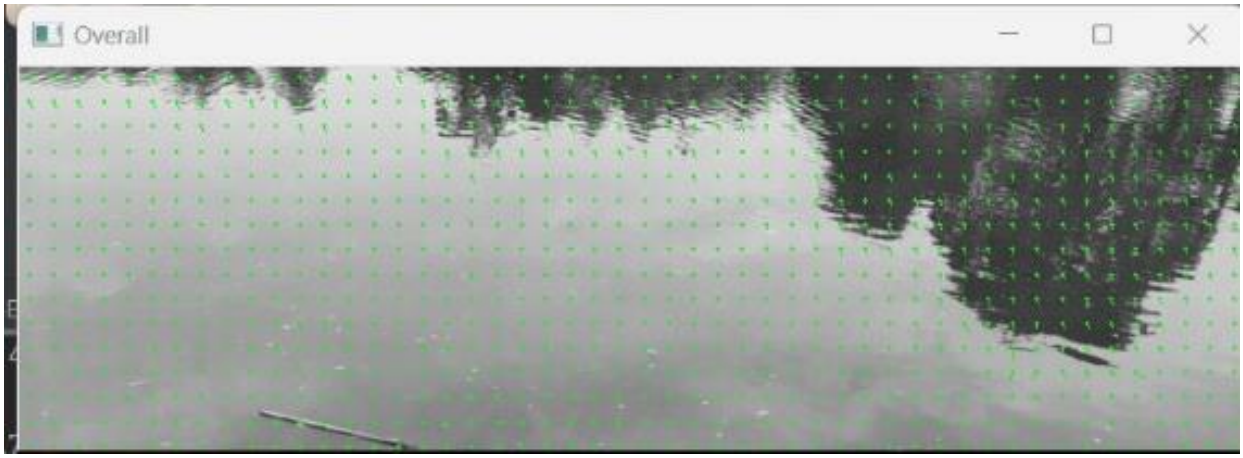


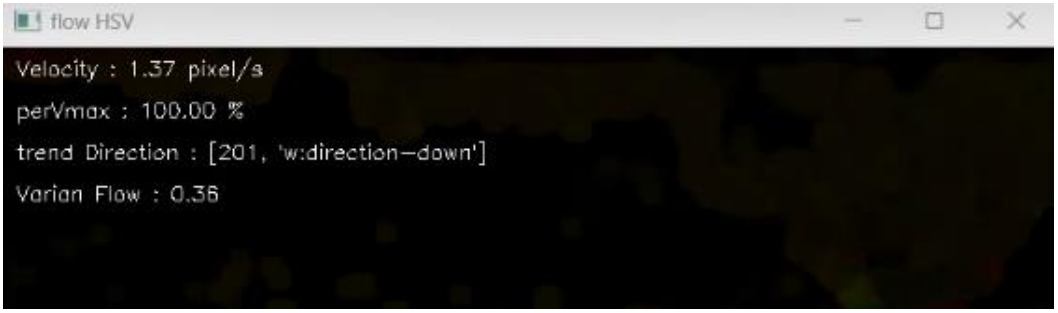


แดชบอร์ดแสดงสถานะต่างๆ

3. sample2.mp4 (น้ำค่อนข้างนิ่ง ไหลไปทางซ้าย) (คลองประเวศ)

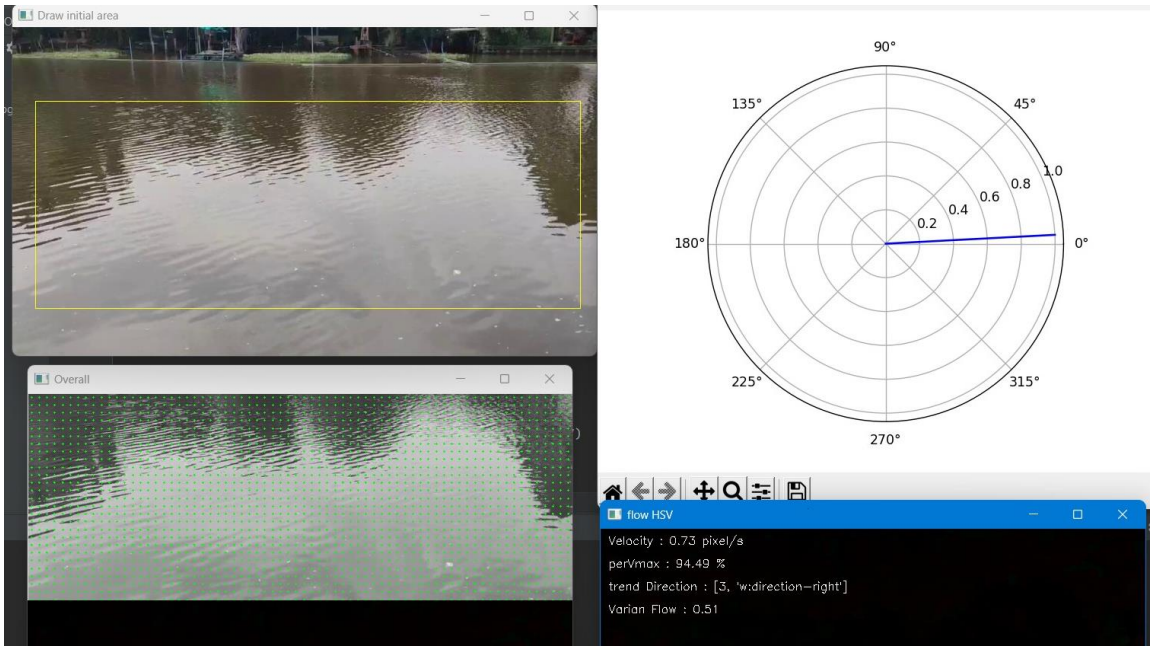


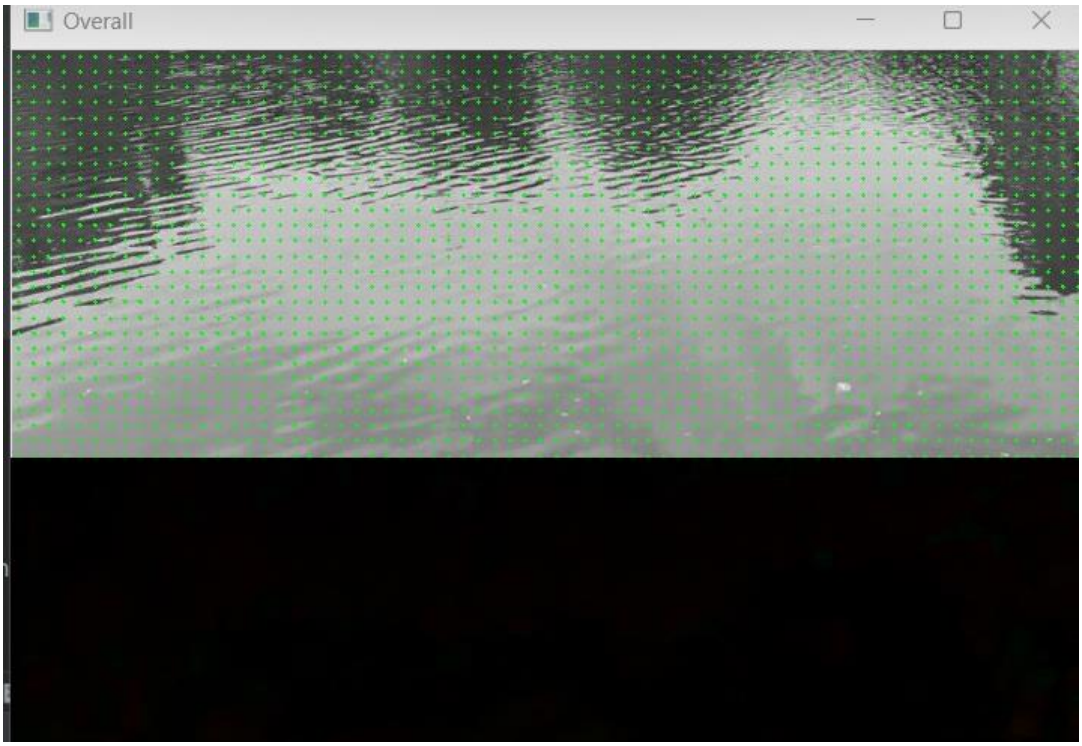


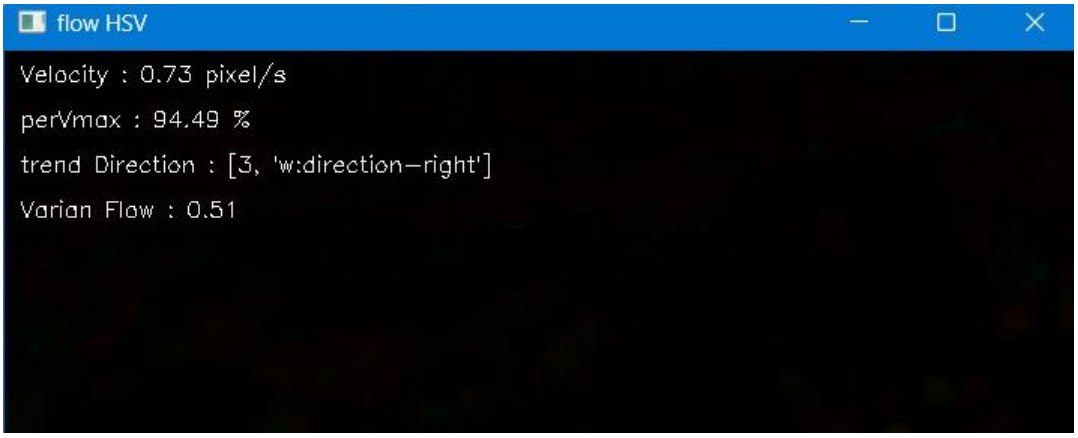


แดชบอร์ดแสดงสถานะต่างๆ

4. sample3.mp4 (น้ำค่อนข้างนิ่ง ไหลไปทางขวา) (คลองประเวศ)







แดชบอร์ดแสดงสถานะต่างๆ

ปัญหาที่พบ

จากการดำเนินการพบว่า การทำงานของโปรแกรมไม่ได้มีการปรับเปลี่ยน node ของ vector ให้มีจำนวนเหมาะสมต่อการวิเคราะห์กระแสน้ำไหลให้สอดคล้องตามขนาดของ Frame ที่ทำการครอปเพื่อวิเคราะห์เฉพาะพื้นที่ เพราะแม้ยังเป็นแม่น้ำเดิมแต่บริเวณที่วิเคราะห์มีพื้นที่ไม่เท่ากันจะส่งผลให้ขนาดของเวกเตอร์และองศาเปลี่ยนแปลงไปตามขนาดพื้นที่ที่วิเคราะห์หากยังคงจำนวน node ของ vector ไว้เท่าเดิม ทำให้ความเร็วหรือสถานะต่างๆที่คำนวณได้ผิดเพี้ยนไป

ในส่วนของ FPS พบว่าเมื่อ Frame มีขนาดที่ใหญ่จะทำให้การทำงานของโปรแกรมช้าในทางกลับกันหาก Frame ที่ทำการครอปเพื่อวิเคราะห์มีขนาดที่เล็กเกินกว่า Frame จริงมากจะทำให้การทำงานของโปรแกรมเร็วมาก ทางผู้จัดทำสังเกตว่า หากลองคำนวณอัตราส่วนของ ขนาดเฟรมที่ครอป : ขนาดเฟรมที่จริง และทำการ หน่วงเวลาต่อ frame โดยให้ระยะเวลาการหน่วงมีค่าเป็น ไปตามสมการพาราโบลาร์หงาย ดังนี้

delay = (อัตราส่วนของ ขนาดเฟรมที่ครอป : ขนาดเฟรมที่จริง) /0.5 -1)**2

พบว่า FPS ของภาพค่อนข้างคงที่มากกว่าเดิมในทุกๆอัตราส่วนเฟรม

และในส่วนของการอัปโหลดข้อมูลไปที่แพล็ตฟอร์มไอโอทียังคงใช้เวลานานและทำให้โปรแกรมส่วนอื่นหยุดทำงาน จึงควรพัฒนาโปรแกรมให้ทำงานได้แบบ multithreading

สรุปผลการทดลอง

จากการทดลองทั้ง 4 กรณีพบว่าการใช้หลักการหรืออัลกริทึม Dense Optical Flow มาวิเคราะห์กระแสน้ำไหลนั้นค่อนข้างมีประสิทธิภาพที่ดีในระดับที่ยอมรับได้ เมื่อพิจารณาจากค่าสถานะต่างๆ ทั้งในเรื่องของความเร็ว ความสั่นไหวของคลื่นผิวน้ำ และทิศทาง สอดคล้องกับความเป็นจริง และค่าข้างมีความเสถียร มีความผิดพลาดที่น้อย

ข้อเสนอแนะ

หากพัฒนากระบวนการและวิธีการตรวจวัดให้มีประสิทธิภาพมากขึ้นทางผู้จัดทำคาดว่าจะสามารถนำไปประยุกต์ใช้งานได้จริง และบรรลุวัตถุประสงค์ ที่จะสร้างเครื่องมือช่วยเพิ่มการป้องกันและลดการได้รับผลกระทบจากการเกิดอุทกภัย อีกทั้งยังมีแนวทางตรวจวัดลักษณะการไหลของน้ำอย่างครอบคลุมพื้นที่วงกว้าง เพื่อประยุกต์ใช้งานกับปัญหาอื่นๆได้อีกมากมาย

เอกสารอ้างอิง

- https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html

- https://nanonets.com/blog/optical-flow/

- https://www.google.com/search?q=Dense-Optical-Flow&sca_esv=581821413&hl=th&tbm=vid&sxsrf=AM9HkKm64ST6AN_Qy6n3kvWOUDJME7iUg:1699850297402&source=lnms&sa=X&ved=2ahUKEwjtUfpk8CCAxXKTGwGHUnbDSEQ_AUoAnoECAIQBA&biw=1488&bih=742&dpr=1.25#

- https://github.com/niconielsen32/ComputerVision/tree/master/opticalFlow

- https://ximera.osu.edu/mooculus/calculus3/vectorFields/digInVectorFields