ชื่อโครงงาน Motion Detection By Background Substraction

ผู้จัดทำ นาย สนั่น งานวิวัฒน์ถาวร รหัสนักศึกษา 4610533

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ปีการศึกษา 2551

อาจารย์ที่ปรึกษาโครงงาน

.....

(ดร. นิคม สุวรรณวร)

อาจารย์ที่ปรึกษาโครงงานร่วม

.....

(ดร. ธเนศ เคารพาพงศ์)

อาจารย์ที่ปรึกษาโครงงานร่วม

.....

(รศ.ดร. มนตรี กาญจนเดชะ)

โครงการนี้เป็นส่วนหนึ่งของรายวิชา Computer Engineering Project I-II ตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมศาสตร์คอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์

.....

ผศ.ดร. พิชญา ตัณฑัยย์

(หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์)

หนังสือรับรองความเป็นเอกลักษณ์

ผู้จัดทำรายงานฉบับนี้ขอรับรองว่ารายงานฉบับนี้เป็นรายงานที่มีความเป็นเอกลักษณ์ โดยที่ผู้จัดทำมิได้คัดลอกมาจากที่ใดที่หนึ่ง เนื้อหาทั้งหมดถูกรวบรวมตามขั้นตอนการดำเนินงานในการจัดทำโครงงาน และหากมีการคัดลอกข้อความหรือส่วนใดส่วนหนึ่งมาจากแหล่งที่มาหรือเอกสารอื่น ผู้จัดทำก็ได้มีการอ้างอิงถึงเอกสารต้นฉบับเหล่านั้นไว้อย่างเหมาะสมแล้วในส่วนของบรรณานุกรม และขอรับรองว่ารายงานฉบับนี้เป็นรายงานที่มิได้เสนอต่อสถาบันใดมาก่อน

.....

สนั่น งานวิวัฒน์ถาวร

ผู้จัดทำ

17 ตุลาคม พ.ศ. 2551

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

กิตติกรรมประกาศ

โครงงาน Motion Detection By Background Substraction สามารถดำเนินงานให้บรรลุวัตถุประสงค์ที่ตั้งไว้ เนื่องจากได้รับความสนับสนุนและช่วยเหลือในด้านต่างๆจากบุคคลต่อไปนี้

อาจารย์ดร. นิคม สุวรรณวร ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ได้ให้ความรู้ ข้อเสนอแนะและช่วยเหลือ ตลอดจนแนวคิดในการจัดทำโครงงาน

อาจารย์ธเนศ เคารพาพงศ์ อาจารย์มนตรี กาญจนเดชะ ซึ่งเป็นอาจารย์ที่ปรึกษาร่วมโครงงาน ที่ได้ให้ข้อแนะนำและตรวจสอบโครงงาน

และขอขอบคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้

ขอขอบคุณเจ้าหน้าที่ของวิศวกรรมคอมพิวเตอร์ทุกท่าน ที่ได้อำนวยความสะดวกในด้านต่างๆ

ขอขอบคุณเพื่อน ๆ และรุ่นพี่ทุกคน ที่ให้ความร่วมมือในการทดสอบโครงงาน และเป็นกำลังใจที่ดีในการจัดทำโครงงานนี้มาโดยตลอด

สุดท้ายนี้ ขอระลึกถึงพระคุณบิดามารดาที่ได้เลี้ยงดู อบรมสั่งสอนจนเติบใหญ่และยังคอยห่วงใยเรื่อยมา

สนั่น งานวิวัฒน์ถาวร

ผู้จัดทำ

บทคัดย่อ

ระบบ Motion Detection คือ ระบบที่มีไว้สำหรับตรวจสอบการเคลื่อนไหวของวัตถุ เพื่อใช้ในระบบการตรวจตรา จับการเคลื่อนไหว ซึ่งระบบดังกล่าวนี้แบ่งได้เทคนิคใหญ่ได้ 2 ประเภทคือ background subtraction เหมาะสำหรับกล้องแบบอยู่นิ่ง และ optical flow เหมาะสำหรับกล้องแบบเคลื่อนที่ ในโครงงานของข้าพเจ้าได้เน้นกรณีศึกษาที่ตัวกล้องอยู่นิ่ง โดยที่ข้อมูล input ของโครงงานนี้จะเป็นจวิดีโอสตรีมมาจากกล้อง IP Camera ที่ติดอยู่ทั่วทางเข้าออกของมหาวิทยาลัย สงขลานครินทร์ นั่นคือเราจะต้องตรวจสอบรถที่เข้าออกในมหาวิทยาลัยขณะนั้นโดยใช้ขั้นตอนวิธีในการตัดพื้นหลัง โดยหาการค่าเฉลี่ยของจุดสีของคลิปวิดีโอทั้งหมดเพื่อได้พื้นหลังแล้วนำไปตัดออกจากภาพต้นฉบับ เพื่อเหลือแค่ถาพวัตถุขณะนั้น ทำให้ output ของงานจะเป็นเฉพาะรถอย่างเดียว ซึ่งโดยปกติแล้วหน่วยรักษาความปลอดภัยจะต้องเก็บคลิปวิดีโอเหล่ามาทำการดูย้อนหลังทุกวินาที ทำให้เสียเวลาเป็นอย่างมาก ข้าพเจ้าหวังเป็นอย่างยิ่งว่าผลของโครงงานนี้จะมีส่วนสำคัญต่อการพัฒนาระบบรักษาความปลอดภัย

Abstract

Motion detection is a crutial step in the 3th generation of surveillance system. Two principal approches of detection techniques could be classified: background subtraction and optical flow, depending on the camera types (fixed or pan-tilt-zoom). We emphasize in this project on the background modeling which is the fundamental process in background subtraction method. We review some of the classical algorithms such as running average, guanssian and etc and propose an adaptive method that improves the quality of detection which suitable for surveillance context. The experimentation is demonstrated with the image sequences obtained from IP Cameras of the PSU's security system. The computational time of the proposed algorithm is also tested which is executed in a specified real time. The encouraged results show some avantages that may be used in the real situation of surveillance circomstances.

บทที่ 1 บทนำ

1.1 ความเป็นมา (Motivation)

โครงการนี้เป็นโครงการเพื่อรักษาความปลอดภัยต่อระบบการเข้าออกของรถบริเวณทางเข้าของมหาวิทยาลัยสงขลานครินทร์ ซึ่งจะมีการใช้กล้องวีดีโอวงจรปิดถ่ายเอาไว้ จุดประสงค์เพื่อหาว่าช่วงเวลาใดที่มีรถเข้ามาบ้างกี่คัน จากอดีตที่ต้องมาตรวจสอบวีดีโอย้อนหลังทุกๆคลิปของกล้องที่ถ่ายเอาไว้ ซึ่งเสียเวลามากในกรณีที่รถว่าง เราจะดูแค่ output ของโปรแกรมที่เป็นไฟล์คลิปที่มีเฉพาะส่วนของรถวิ่งจากไฟล์วิดีโอ ซึ่งจะลดเวลาที่จะต้องดูย้อนหลังได้เป็นอย่างมาก

ข้อมูล input ของโปรแกรมนี้เป็นไฟล์วิดีโอสตรีม หลังจากเอามาเข้ากระบวนการคำนวณผลลัพธ์จะได้ไฟล์ บิตแม็บ ที่ตัดเอาพื้นหลังออกจะได้เป็นภาพเคลื่อนไหวของวัตถุแต่ละตัว แต่ในกรณีที่วัตถุมาพร้อมกันหลายๆวัตถุก็ต้องสามารถแยกออกมาได้

1.2 วัตถุประสงค์ของโครงงาน

- นำเทคโนโลยีการวิเคราะห์ผลทางด้าน Image Processing เพื่อตรวจจับการเคลื่อนไหวของวัตถุจากข้อมูลภาพที่ได้มาจากกล้อง IP Camera ของระบบ Surveillance System ในมหาวิทยาลัยสงขลานครินทร์

ในโครงงานจะทำการนำเอาความรู้ของ Image processing มาประยุกต์การใช้งานในด้านประมวลผลภาพวิดีโอสตรีม เพื่อใช้เป็นตัวตรวจจับหาวัตถุที่เคลื่อนไหวในวิดีโอสตรีม และใช้ในการตัดสินใจเลือกอัลกอริธึมที่เหมาะสมสำหรับการตรวจหาวัตถุ รวมไปถึงการสร้างโปรแกรมจำลองขึ้นมาเพื่อให้ผู้ใช้ ศึกษา Algorithm ให้เข้าใจได้ง่ายขึ้น

1.3 ขอบเขตของโครงงาน

สามารถนำ Algorithm ที่ศึกษามาเปรียบเทียบคุณสมบัติกันเพื่อให้ได้มาซึ่ง Algorithm ที่ดีที่สุดและเหมาะสมที่สุด เพื่อนำมาประยุกต์ใช้กับการสร้างเป็นโปรแกรมจำลอง เพื่อการศึกษา Algorithm ต่อไป

1.4 ขั้นตอนการทำงาน

1. ศึกษาอัลกอริธึมที่ใช้สำหรับการตรวจจับการเคลื่อนไหว

2. พัฒนาโปรแกรมที่ได้จากการปรับปรุงแนวคิด

3. นำโปรแกรมที่พัฒนาไปใช้กับระบบความปลอดภัย IP Camera Network

4. ใช้อัลกอริธึมทางการตรวจจับวัตถุในภาพที่มีอยู่มาทำการการปรับปรุงแนวคิดและออกแบบให้ดีขึ้นสำหรับระบบ Surveillance System

1.5 แผนการดำเนินงานของโครงงาน

-ขั้นตอนที่ 1 ศึกษาปัญหาและแนวคิดของโครงงาน

-ขั้นตอนที่ 2 ทำการ Implement ทดสอบแนวคิด

-ขั้นตอนที่ 3 ทำการปรับปรุงแนวคิดและออกแบบให้ดีขึ้น

-ขั้นตอนที่ 4 ทำการ Implement ส่วนการปรับปรุงแนวคิด

-ขั้นตอนที่ 5 Classified and Tracking ส่วน Objects หลายชิ้นที่อยู่เฟรมเดียวกัน

-ขั้นตอนที่ 6 Implement ส่วนการ Classified and Tracking

-ขั้นตอนที่ 7 Integrate กับระบบกล้อง IP Camera และ Serverให้ใช้งานได้จริง

-ขั้นตอนที่ 8 จัดทำส่วนของ File Description ในแบบเอกสาร XML เช่นเดียวกับ log file

-ขั้นตอนที่ 9 ทดสอบ Implement ทั้งหมด

1.6 ตารางการดำเนินงานของโครงงาน

เดือน	มิถุนายน			กรกฎาคม			สิงหาคม			กันยายน			ตุลาคม			พฤศจิกายน			ธันวาคม			มกราคม			กุมภาพันธ์		
ขั้นตอนที่ 1																											
ขั้นตอนที่ 2																											
ขั้นตอนที่ 3																											
ขั้นตอนที่ 4																											
ขั้นตอนที่ 5																											
ขั้นตอนที่ 6																											
ขั้นตอนที่ 7																											
ขั้นตอนที่ 8																											
ขั้นตอนที่ 9																											

ตาราง 1-1 ขั้นตอนการทำงาน

1.7 เครื่องมือที่จำเป็นสำหรับการพัฒนา

1. ระบบปฏิบัติการ Microsoft Window

2. โปรแกรม Microsoft Visual C++ 6.0 สำหรับ Editor และ Complier

3. Library ของ OpenCV สำหรับการเรียกใช้ฟังก์ชันการทำงานต่างๆ

4. PC 1 เครื่องสำหรับการเขียนโปรแกรม

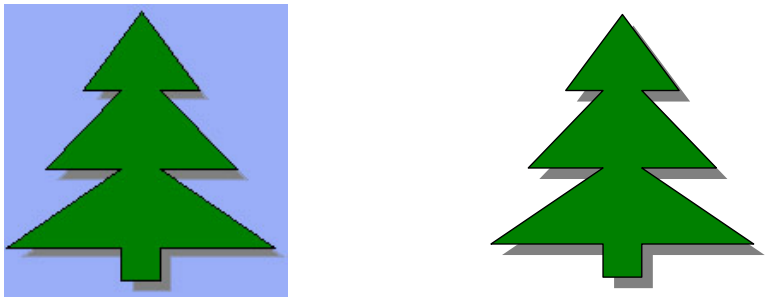
5. กล้อง IP Camera ที่ติดอยู่บริเวณทางเข้าออกของมหาวิทยาลัย

บทที่ 2 ความรู้พื้นฐาน

ในบทนี้จะกล่าวถึงความรู้พื้นฐานที่จำเป็นจะต้องใช้ในการพัฒนาโครงงานนี้ โดยจะประกอบไปด้วยหัวข้อดังต่อไปนี้ซึ่งนั่นคือ ความรู้ในด้านการวางแผนการเคลื่อนที่ Algorithm ตลอดจนเครื่องมือที่ใช้พัฒนาโปรแกรม

2.1 การเปรียบเทียบจุดสี

เนื่องจากภาพ bitmap จะประกอบไปด้วยจุดเล็กๆที่เรียกว่า pixel เราสามารถหาค่าสีแต่ละจุดของภาพได้ ซึ่งออกมาในรูปแบบ RGB (สีแดง สีเขียว สีน้ำเงิน) เราสามารถนำมาเปรียบเทียบกับจุดสีอื่นๆได้โดยวิธีการวนลูปแกน x และ แกน y ของภาพ bitmap นั้นๆ



รูปภาพ 2-1 ภาพต้นฉบับและหลังจากตัด background ออก

ทั้ง 2 รูปแสดงถึงการตัดสี background ออกโดยใช้ขั้นตอนวิธีเปรียบเทียบจุดสี

2.2 การเปรียบเทียบเฟรม

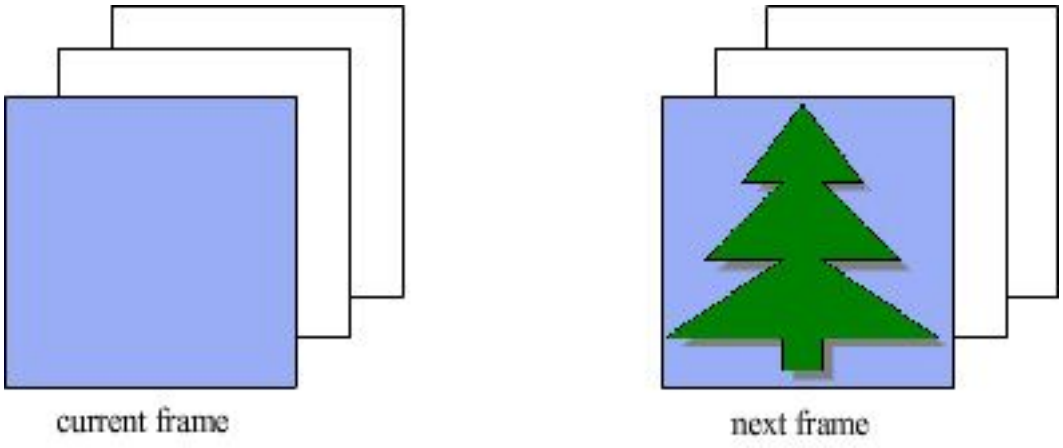
เนื่องจากคลิปวิดีโอประกอบด้วย bitmap หลายๆภาพมาแสดงผลเรียงกัน เราเรียก bitmap แต่ละตัวว่า **เฟรม (frame)** เราจำเป็นจะต้องเปรียบเทียบเฟรมหลายเฟรมเข้าด้วยกันเพื่อหาว่าช่วงเวลาใดที่มีรถเข้ามาโดยวิธีตามกรณีดังนี้

1. ถ้าเฟรมปัจจุบัน (current frame) เหมือนกับเฟรมถัดไป (next frame) ให้คิดว่าไม่มีรถเข้าออก

2. ถ้าเฟรมปัจจุบัน (current frame) ต่างกับเฟรมถัดไป (next frame) ให้คิดว่ามีรถเข้าออก และตรวจจับค่าจุดสีของเฟรมถัดไป เพื่อหาภาพที่เคลื่อนไหวโดยตัด background ออก

2.1 นำภาพแต่ละเฟรมที่เคลื่อนไหวและตัด background ออกมาทำการลด noise เพื่อมีคุณภาพดีขึ้น

2.2 นำภาพจากหัวข้อ 2.1 มาบันทึกเป็นไฟล์ทั้งหมด



รูปภาพ 2-2 การเปรียบเทียบเฟรม

2.3 เทคนิคการหาภาพวัตถุที่เคลื่อนไหว

Background Subtraction

เป็นขั้นตอนวิธีตัดพื้นหลังออกจากเฟรมวิดีโอ มีหลักการคือ เมื่อภาพของเฟรมหนึ่งๆ มาลบกับพื้นหลังจะได้เป็น object นั้นๆ ดังสมการ

$$O = I - B$$

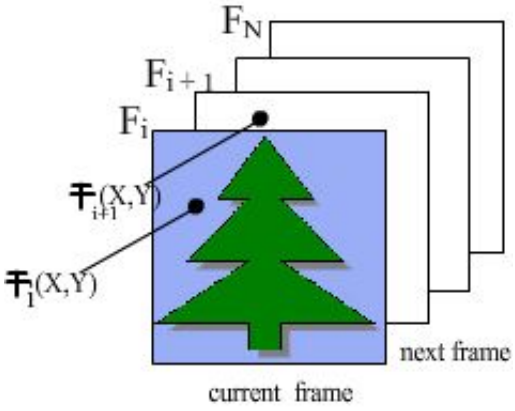
เมื่อ O คือ Object

I คือ ภาพของเฟรมหนึ่งๆ

B คือ ภาพพื้นหลัง

- สำหรับขั้นตอนการประมวลผลจำเป็นจะต้องทำให้ภาพเป็น Gray Scale (ขาวดำ) เพื่อลดหน่วยความจำที่ใช้ เพราะ Gray Scale 1 จุดใช้ 8 bit ในขณะที่ RGB ใช้ 24 bit

รูปขั้นตอนการประมวลผลโดยวิธี Background Substracttion



รูปภาพ 2-3 เทคนิคการหาภาพวัตถุที่เคลื่อนไหว

$\mp$ (X,Y) คือ ค่าจุดสีแบบ Gray Scale ของภาพที่จุด X ,Y

N คือ จำนวนเฟรมทั้งหมดของ buffer ที่นำมาจองหน่วยความจำเอาไว้

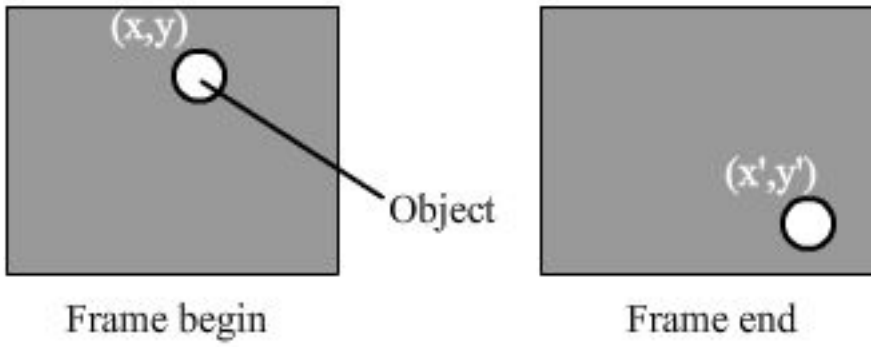
นำมาเข้าสมการได้ดังนี้

$$\frac{\sum_{i=0}^{N} \mp(x,y)}{N}$$

นั่นคือ ผลรวมของค่าสี Gray Scale ที่จุด X,Y ของภาพ หารด้วยจำนวนของเฟรมทั้งหมดของ buffer จะได้ค่า background เฉลี่ยทั้งหมด โดยต้องกำหนดให้ background คือส่วนของภาพในเฟรมที่ไม่เหมือนกัน

Optical Flow

เป็นวิธีการหาเว็กเตอร์ลัพธ์ของส่วนที่ภาพเคลื่อนไหวไปแล้ว ดังรูป



รูปภาพ 2-4 Optical Flow

รูปภาพจะมี Object ที่เคลื่อนไหวไปมาเราสามารถนำมาหาเวคเตอร์ลัพธ์ของการเคลื่อนที่ของ Object นั้นได้ โดยต้องรู้พิกัดที่ จุด X ,Y ของเฟรมเริ่มต้นและเฟรมสุดท้าย

2.4 วิธีการหาผลต่างของเฟรม

$$| frame_i - frame_{i-1} | > Th$$

$frame_i$ คือ เฟรมปัจจุบัน

$frame_{i-1}$ คือ เฟรมก่อนหน้า

Th คือ ค่า threshold

วิธีนี้จะสมมุติให้ background ที่ประมาณค่าได้เป็นเฟรมก่อนหน้าเพื่อง่ายต่อการประมวลผล และค่าความเร็วของวัตถุกับจำนวนเฟรมต่อวินาที(FPS) ก็ส่งผลต่อวิธีนี้เช่นกัน ส่วนวิธีนี้ไวต่อค่า Threshold เป็นอย่างมาก

2.5 วิธีการหาพื้นหลัง

- นิยมใช้วิธีการหา background ค่าเฉลี่ย(average) หรือ ฐานนิยม(median)
- วิธีนี้ค่อนข้างจะเร็วแต่ใช้ memory ค่อนข้างเยอะ เนื่องจากต้องการจอง memory เป็นจำนวนเฟรม x ขนาดภาพ

2.6 Motion Template

เป็นการทำงานเกี่ยวกับการออกแบบสร้างการเคลื่อนไหวจำลองต่อเนื่องกันจากจุดที่เกิดการเคลื่อนไหว ว่าเกิดขึ้นอย่างไร ทิศทางอย่างไร ซึ่งอัลกอริธึมเหล่านี้ถูกเขียนขึ้นโดย Mr. Davis and Bobick และ Bradski and Davis ส่วนการทำงานบนรูปภาพของ output ของเฟรม หรือ การหาผลต่างของ background หรือวิธีการทำ Image Seqmentation ทั้งหลายเหล่านั้นจะทำในรูปแบบภาพ Grayscale นั่นคือ 1 channel นั้นเอง

MHI (Motion History Image)

จากรูปทางด้านซ้ายแสดงภาพโครงร่างของวัตถุ(foreground silhouette)ของวัตถุที่เคลื่อนไหวในภาพ หรือคน ซึ่งวิธีที่ได้มาของภาพโครงร่างของวัตถุมาจากเทคนิคของการตัดภาพพื้นหลังในรูปแบบต่างๆ ซึ่งวิธีการตัดพื้นหลังสำหรับคนหรือการเคลื่อนไหวของวัตถุ จะทำโดยการคัดลอกภาพพื้นหลังที่มีการอัพเดทค่าล่าสุดใน Motion History Image ซึ่งจะทำการสร้างเลเยอร์ประวัติ (layered history) ของภาพสุดท้าย โดยปกติแล้วค่าล่าสุดจะเป็นแค่ค่า timestamp ของเวลาที่ผ่านไปซึ่งแสดงผลในหน่วย millisecond

รูปทางขวาแสดงผลลัพธ์ที่เรียกว่า MHI (Motion History Image) ซึ่งบอกถึงระดับของพิกเซล หรือ threshode ของเวลาที่เปลี่ยนแปลง

รูปภาพ 2-5 MHI (Motion History Image)

การเคลื่อนไหวที่อัพเดทล่าสุดจะมีค่าสูงสุด และการเคลื่อนไหวก่อนหน้าจะมีค่าน้อยไปเรื่อยๆ ซึ่งกระบวนการของ Motion Template จะกล่าวถัดไป

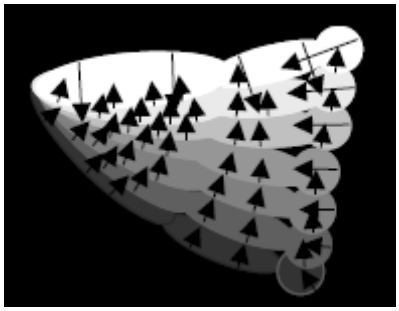
Motion Template Processing

A) Updating MHI Images

ปกติแล้วเราจะทำงานกับค่าจุดทศนิยมของรูปภาพตั้งแต่เราอ่านค่าผลต่างของเวลาในหน่วย millisecond จากการเริ่มของโปรแกรม และทำการคำนวณค่าของภาพโครงร่างของวัตถุล่าสุดที่จะนำมาหาทิศทางการเคลื่อนที่

B) Making Motion Gradient Image

เป็นการสร้างทิศทางการเคลื่อนไหวของภาพโครงร่างทั้งหมด แต่เนื่องจากทิศทางที่ได้จากการคำนวณจะค่าคลาดเคลื่อนไปบ้าง ดังนั้นจึงต้องมีการทำ Noramlize



รูปภาพ 2-6 ภาพโครงร่างที่ทำ Motion Gradian

รูปภาพ 2-7 ภาพโครงร่างที่ทำ Motion Gradian หลังจากทำ Nomalize

C) Finding Regional Orientation

ปกติแล้วไม่จำเป็นต้องคำนวณค่ามุมของการเคลื่อนที่ทั้งหมด ดังนั้นการกำหนดพื้นที่ของการเคลื่อนไหวจะกระทำโดยเคลื่อนที่ชิ้นส่วนของวัตถุทั้งหมด ซึ่งทุกครั้งที่ภาพ MHI ถูกสร้างขึ้น ภาพโครงร่างล่าสุดจำเป็นต้องใช้ค่าสูงสุดหลายๆตัวในการสแกน แล้วเข้าสู่ contour ของภาพโครงร่างเพื่อช่วยในการหาพื้นที่ของการเคลื่อนไหว เช่น time stamp ล่าสุด อัลกอริธึมสำหรับสร้าง mask ไปยังพื้นที่ segment motion เป็นดังนี้

1. สแกนรูป MHI จนกระทั่งเจอภาพโครงร่างล่าสุด(รูป a)

2. สำรวจไปยังพื้นที่ของภาพโครงร่างล่าสุดแล้วทำการลบจุดที่มี step ของ motion history เมื่อเจอ step ที่ดีที่สุด จะทำเครื่องหมายไว้ลงใน stack ถ้าขนาดของที่ใส่เพียงพอแล้วก็ zero out พื้นที่ (รูป b)

3. เพิ่มเติม

- บันทึก ตำแน่งที่น้อยที่สุดแล้วทำการ downfill (รูป c)

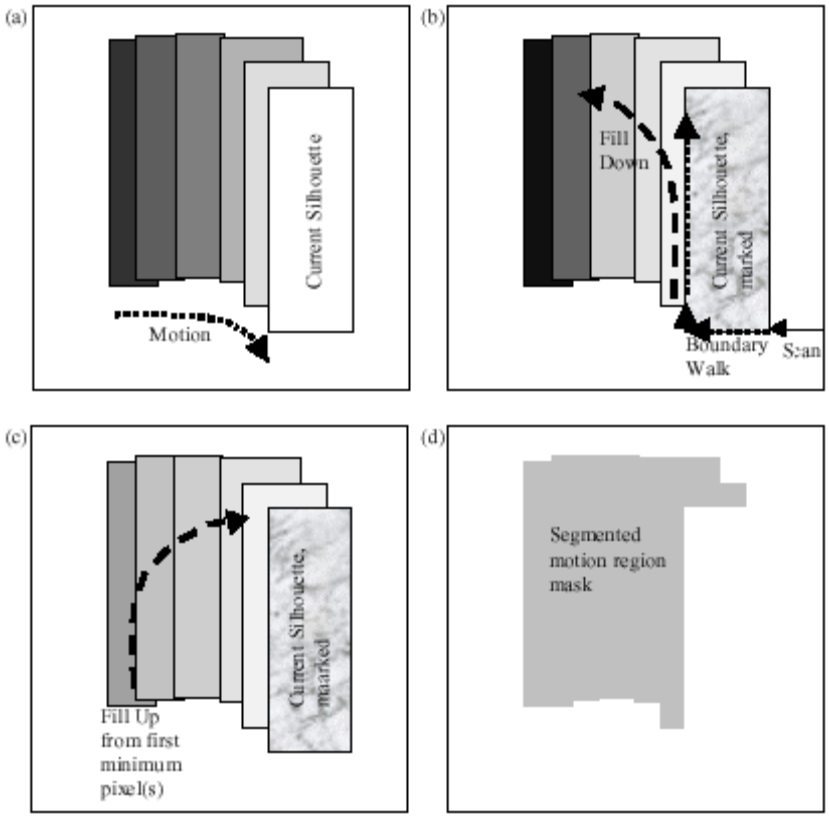
- แบ่ง floodfill up จากการตรวจสอบตำแหน่ง (รูป d)

- รวมส่วนย่อยโดย logic AND สำหรับ upfill และ downfill

4. จัดเก็บส่วนที่ตรวจเจอพื้นที่ segmented motion ไปสู่ mask

5.ต่อส่วนของการสำรวจจนกระทั่งหมดภาพโครงร่างทั้งหมด

6.เพิ่มเติม ไปยังข้อ 1 จนกระทั่งเจอพื้นที่ภาพโครงร่างทั้งหมด



รูปภาพ 2-8 Finding Regional Orientation

2.7 วิธีการและอัลกอริธึมทางด้าน Background Substraction

<u>ปัญหาที่พบ</u>

1. ด้านภาพ
   - วิถีเอียงของภาพ
   - กลุ่มควัน และ หมอก
2. การเคลื่อนไหวของภาพ
   - มุมกล้อง

- วัตถุพื้นหลังที่มีความถี่สูง เช่น กิ่งไม้ คลื่นทะเล

3. การเปลี่ยนแปลงของภาพพื้นหลัง
   - การจอดรถ

วิธีที่ 1 : running average โดยใช้สมการ

$$B_{i+1} = \alpha * F_i + (1 - \alpha) * B_i$$

B_{i+1} คือ background ของเฟรมถัดไป

B_i คือ background ของเฟรมปัจจุบัน

α คือ ค่าสัมประสิทธิ์ ปกติมีค่า 0.05

F_i คือ ค่าจุดสีของเฟรมปัจจุบัน

เนื่องจากวิธีนี้เป็นวิธีง่ายที่สุดของการคำนวณ โดยที่ไม่จำเป็นต้องใช้หน่วยความจำ เพียงแค่ทดค่าตัวแปรต่างๆเข้าสู่สมการแล้ววนลูปไปก็ได้ background ของเฟรมถัดไปมาใช้ได้

วิธีที่ 2 : running average แบบ selectivity

<u>หลักการ</u>

- แต่ละเฟรมใหม่ที่เข้ามา แต่ละจุดสีจะต้องมีการจำแนกเป็น foreground หรือ background
- สิ่งที่เป็นการจำแนกของ background model
  i. ถ้าจุดสีจำแนกเป็น foreground จะถูกตัดทิ้งใน backgroung model

สมการแนวคิดของวิธีนี้

กรณีที่ 1 ถ้า F_t เป็น background

$$B_{i+1}(x,y) = \alpha * F_t(x,y) + (1 - \alpha) * B_t(x,y)$$

กรณีที่ 2 ถ้า F_t เป็น foreground

$$B_{i+1}(x,y) = B_i(x,y)$$

วิธีนี้ให้คล้ายกับวิธีแรกเพียงแต่มีการแยกพื้นหลัง (background) กับ วัตถุ (foreground) เพื่อแยกประมวลผลแต่ละสมการ เห็นได้ว่าเมื่อ F_t เป็น foreground ค่าของ background ของเฟรม ถัดไปจะเท่ากับเฟรมก่อนหน้า

วิธีที่ 3 : ใช้วิธี Running Gaussian Average

<u>หลักการ</u>

- ให้ μ เป็นค่าการกระจายของ Gaussian และ σ เป็นค่าความแปรปรวนของ Gaussian

$$\mu_{t+1} = \alpha F + (1 - \alpha)\,\mu_t$$
$$\sigma^2{}_{t+1} = \alpha(F_t - \mu_t)^2 + (1 - \alpha)\,\sigma^2{}_t$$

วิธีที่ 4 : ใช้วิธี Mixture of Gaussian

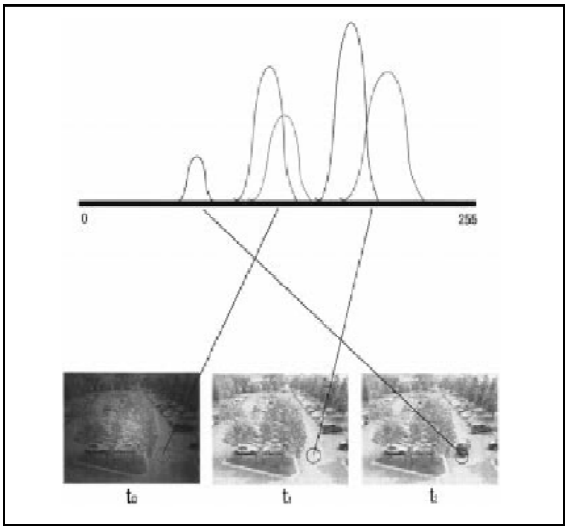
<u>หลักการ</u>

-ให้ค่า ω_i มีการเปลี่ยนแปลงตลอดเวลาทุกๆ เฟรมใหม่

- ในทุกๆเฟรมใหม่ค่าของ Gaussian จะตรงกับค่าปัจจุบันทำให้ค่า μ,σ มีการเปลี่ยนแปลงตลอด

- วิธีนี้จะต้องหาทั้งค่า background และ foreground เพื่อหาค่าความแปรปรวน

- จำเป็นจะใช้ค่าสถิติมาเปรียบเทียบ



รูปภาพ 2-9 Mixture of Gaussian

วิธีที่ 5 : Kernel Density Estimator

- หาค่า PDF (x)ใช้ขั้นตอนวิธี background PDF ซึ่งนำมาจาก histogram ที่มีค่าจุดสีของมากที่สุดของเฟรมล่าสุด
- ถ้า PDF (x) > Th , ที่ x pixel จะเป็น background
- มีปัญหาด้านการจอง memory (n * size(frame))

วิธีที่ 6 : Mean-shift Base Estimation

<u>หลักการ</u>

- ใช้วิธี หา gradian เพื่อหาค่าฐานนิยมจากการแปรปรวนของข้อมูล ตามสมการ

$$m(x) = \frac{\sum_{i=1}^{n} x_i g((x - x_i/h)^2)}{\sum_{i=1}^{n} g((x - x_i/h)^2)} - x$$

m (x) คือ ค่า mean shift vectori

x คือ ค่าจุดใดๆบนภาพ

x_i คือ เป็นเซ็ตของจุดข้อมูล

h คือ ค่าของ bandwidth ที่ได้รับวิเคราะห์มาแล้ว

g(u) คือ อนุพันธ์อันดับ 1 ของฟังก์ชันร่วม k(u) โดยที่ k(u) คือ kernel profile

$$k_E(x) = \begin{cases} 1 - x & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases}$$

ปัญหาของวิธีนี้

- การทำงานของสมการนี้ (วนรอป) จะช้า
- มีการจอง memory : (n * size(frame))

วิธีที่ 7 : Eigenbackground

<u>หลักการ</u>

1. จำนวนเฟรมจะถูกจัดเรียงใหม่ในรูปของแมทริกซ์ A
2. หาค่า covariance metrix โดย

$$C = AA^T$$

3. หาค่า eigenvalue L และ eigenvector metrix Φ จาก C
4. เก็บค่า eigen vector metrix M เอาไว้
5. ขณะที่ภาพใหม่เข้ามา (/)จะมีการทำ sub-space กับค่า M แล้วเก็บเป็นค่า /'
6. หาผลต่างของ / - /' จะได้ค่า foreground object

ข้อได้เปรียบของวิธีนี้คือมีความเที่ยงตรงสูง แต่ใช้ขั้นตอนการคำนวณสูง และใช้หน่วยความจำสำหรับเก็บค่าจำนวนเฟรมทั้งหมด

2.8 ข้อแตกต่างระหว่างอัลกอริธึม

เห็นได้ว่าแต่ละวิธีการแต่ละชนิดมีข้อได้เปรียบหรือเสียเปรียบอย่างเห็นได้ชัด โดยปกติจะเป็นปัญหาทางด้านการใช้หน่วยความจำมาก เพราะจำเป็นต้องใช้ในการเก็บค่าย้อนหลังเพื่อเปรียบเทียบ

อัลกอริธึม	ข้อเปรียบเทียบ

	ความเที่ยงตรง	การใช้หน่วยความจำ	ความซับซ้อน
running average	น้อย	ไม่ใช้	ง่าย
selectivity	น้อย	ไม่ใช้	ง่าย
Mixture of Gaussian	ปานกลาง	ใช้การเก็บแบบสถิติ	-
Kernel Density Estimator	-	ใช้มาก	-
Mean-shift Estimation	สูง	ใช้มาก	ยาก
Eigenbackground	สูง	ใช้เฉพาะสร้าง metrix	ยาก

ตาราง 2-1 เปรียบเทียบอัลกอริทึม

Running Gaussian Average Model

เริ่มต้น model อันนี้เป็น model ที่มาจากการหาค่า Gaussian Probably Density (pdf) ในค่าของ pixel สุดท้ายของภาพ ในแต่ละ pixel ให้ค่าเฉลี่ยของภาพ μ_t และ ค่าความแปรปรวน σ ในแต่ละ channel ของจุดสี ซึ่งการหาค่าเฉลี่ยทำได้จากการนำผลรวมของค่าเฟรมทั้งหมดหารด้วยจำนวนภาพ

$$\bar{x} = \frac{\sum x}{N}$$

และ ค่าความแปรปรวนหาได้จาก ค่า

$$S.D. = \sqrt{\frac{\sum (x - \bar{x})^2}{N}}$$

แต่ในความเป็นจริงการหาค่าเหล่านี้ไม่สามารถนำมาใช้จริงในรูปแบบงาน video stream ได้ เนื่องจากเราไม่ทราบค่าของ frame ทั้งหมดเพื่อแทนค่าในสมการนี้ได้ จึงได้มีการคิดสมการเพื่อใช้กับ video stream จริงได้โดย

$$\mu_{t+1} = \alpha F_t + (1 - \alpha)\, \mu_t \qquad \textit{ให้เป็นสมการที่ 1}$$

$$\sigma^2_{t+1} = \alpha(F_t - \mu_t)^2 + (1 - \alpha)\, \sigma^2_t \qquad \textit{ให้เป็นสมการที่ 2}$$

$$F_{out} = | F_t - \mu_t | > Th$$ *ให้เป็นสมการที่ 3*

สมการที่ 1 เป็นการหาค่าเฉลี่ยของภาพ (μ) ณ เวลาปัจจุบัน โดยมี

F_t คือ ค่า Input Image

μ_t คือ ค่าเฉลี่ยของภาพ ตัวก่อนหน้า มีค่าเริ่มต้นเป็น 0

μ_{t+1} คือ ค่า ค่าเฉลี่ยของภาพ ตัวปัจจุบัน

α คือ ค่าสัมประสิทธิ์

สมการที่ 2 เป็นการหาค่าความแปรปรวนของภาพ (σ) ณ เวลาปัจจุบัน โดยมี

F_t คือ ค่า Input Image

μ_t คือ ค่าเฉลี่ยของภาพ

σ_t คือ ค่า ความแปรปรวนของภาพ ตัวก่อนหน้า มีค่าเริ่มต้นเป็น 0

σ_{t+1} คือ ค่า ความแปรปรวนของภาพตัวปัจจุบัน

α คือ ค่าสัมประสิทธิ์

สมการที่ 3 เป็นการหาค่าภาพผลลัพธ์ (F_{out})

F_t คือ ค่า Input Image

μ_t คือ ค่าเฉลี่ยของภาพ

Th คือ ค่าของ threshold แทนด้วย $k * \sigma_t$

F_{out} ค่า ภาพผลลัพธ์มีค่า เป็น 0 หรือ 1 (จุดสีดำหรือขาว)

2.9 การประมวลผลภาพกับรูปร่างและโครงร่างของภาพ (Morphological Image Processing)

Morphological Image Processing เป็นการประมวลผลภาพโดยการเปลี่ยนแปลงลักษณะรูปร่างหรือโครงสร้างของภาพ โอเปอเรชั่นพื้นฐานโดยทั่วไปได้แก่ การ Dilation Erosion และ Skeleton โดยการ Dilation คือการขยายภาพโดยมีสัดส่วนเท่ากันทั่วทั้งภาพ(Uniform) การ Erosion คือการย่อภาพ ส่วนการทำ Skeleton เป็นการหาโครงสร้างหลักของวัตถุ นอกจากโอเปอเรชั่นพื้นฐานดังที่ได้กล่าวข้างต้นแล้วยังมีโอเปอเรชั่นอื่น ๆ อีกที่ได้แก่การ Opening และ Closing เป็นต้น

2.10 การขยายภาพ(Dilation)

การขยายภาพในที่นี้จะพิจารณาสำหรับข้อมูลภาพที่เป็นแบบไบนารีโดยการใช้เทคนิคการ Hit และ Miss ตามที่ได้กล่าวไว้ การขยายภาพจะทำได้โดยกำหนดTemplate (ซึ่งสามารถสร้างได้จาก * และ 1 โดยมีจุดเริ่มต้นที่กำหนดโดยวงกลม) และนำ Template นี้สแกนไปบนข้อมูลภาพตามลำดับตลอดทั้งภาพซึ่งในขณะที่จุดเริ่ม (Origin) ของ Template ตรงกับตำแหน่งข้อมูลภาพที่พิกเซลมีค่าเท่ากับ 1 นั้นก็จะทำการยูเนียน Template นี้เข้ากับข้อมูลภาพดังตัวอย่าง

ข้อมูลภาพ

*	*	*	*	*	*	1	*	*	1	*
*	*	*	*	*	*	1	*	*	*	1
*	*	*	*	*	1	1	*	1	1	*
*	*	*	*	1	1	1	1	1	1	1
*	*	*	*	1	1	1	1	1	*	1
*	*	*	*	1	1	1	1	1	1	1
*	*	*	*	1	1	1	1	1	1	1

Template

1	*
1	1

ข้อมูลภาพและTemplate

ข้อมูลแถวแรกของภาพเป็นดังนี้

*	*	*	*	*	*	1	*	*	1	*

ข้อมูลแถวแรกของภาพ

เมื่อทำการยูเนียนกับ Template ณ. ตำแหน่งข้อมูลภาพที่พิกเซลเท่ากับ 1 ในแถวแรก

* * * * * * 1 * * * *
* * * * * * 1 1 * 1 *

ยูเนียนกับ Template ณ. ตำแหน่งข้อมูลภาพที่พิกเซลเท่ากับ 1 ในแถวแรก

และเมื่อยูเนียนกับ Template เข้ากับพิกเซลที่มีค่าเท่ากับ 1 ณ. ตำแหน่งพิกเซลที่สองในแถวแรก

* * * * * * 1 * * * *
* * * * * * 1 1 * 1 1

ยูเนียนกับ Template เข้ากับพิกเซลที่มีค่าเท่ากับ 1 ณ. ตำแหน่งพิกเซลที่สองในแถวแรก

และเมื่อทำการยูเนียนทั้งภาพจะได้ภาพสุดท้ายดังนี้

* * * * * * 1 * * 1 * *
* * * * * * 1 1 * 1 1 *
* * * * * 1 1 1 1 1 1 1
* * * * 1 1 1 1 1 1 1 1
* * * * 1 1 1 1 1 1 1 1
* * * * 1 1 1 1 1 1 1 1
* * * * 1 1 1 1 1 1 1 1
* * * * 1 1 1 1 1 1 1 1

ยูเนียนทั้งภาพ

2.11 การย่อภาพ(Erosion)

การย่อภาพเป็นลักษณะของการลบข้อมูลภาพบริเวณขอบของภาพ การย่อภาพสามารถทำได้มีลักษณะคล้ายกับการขยายภาพโดยการสร้าง Template ขึ้นแล้วนำ Template ไปสแกนตามข้อมูลภาพ

สำหรับทุกตำแหน่งที่เลื่อน Template ไปบนภาพก็จะมีการเปรียบเทียบกับข้อมูลภาพ ถ้าข้อมูลภาพมีค่าเหมือนกับ Template จะทำการกำหนดค่าข้อมูลภาพในตำแหน่งที่ตรงกับจุดเริ่มต้น (Origin)ของ Template ถูกกำหนดให้มีค่าเท่ากับ 1

ข้อมูลภาพ Template

*	*	*	*	*	*	1	*	*	1	*
*	*	*	*	*	*	1	*	*	*	1
*	*	*	*	*	1	1	*	1	1	*
*	*	*	*	1	1	1	1	1	1	1
*	*	*	*	1	1	1	1	1	*	1
*	*	*	*	1	1	1	1	1	1	1
*	*	*	*	1	1	1	1	1	1	1

1	*
1	1

ข้อมูลภาพและTemplate

ผลที่ได้จะมีเพียง 3 ตำแหน่งเท่านั้นที่มีค่าเหมือนกับ Template

*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	1	*	*	1	*
*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	1	*	*
*	*	*	*	*	*	*	*	*	*	*

ตำแหน่งที่มีค่าเหมือนกับ Template

ผลที่ได้ตาม ข้อมูลภาพที่ผ่านการทำโอเปอเรชั่นกับ Template แล้วพบว่ามีข้อมูลของภาพเพียง 3 ตำแหน่งเท่านั้นที่เหมือนกับ Template ถ้ามีการเปลี่ยน Template เป็น $\begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix}$ ผลที่ได้มีลักษณะดังนี้คือ

*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	1	*	*	1	*	*
*	*	*	*	1	1	1	1	1	*	*
*	*	*	*	1	1	1	1	*	*	*
*	*	*	*	1	1	1	1	1	1	*

ข้อมูลภาพที่ผ่านการทำโอเปอเรชั่นกับ Template

ผลที่ได้ตามรูปที่ 5-13 จะเห็นว่าจะเป็นการย่อขนาดของภาพแต่สามารถย่อขนาดได้น้อยกว่าเมื่อใช้ Template $\begin{matrix} 1 & * \\ 1 & 1 \end{matrix}$ ซึ่งได้ผลเป็นที่น่ายอมรับมากกว่าดังนั้นในการเลือก Template เป็นสิ่งที่สำคัญอย่างหนึ่งในการย่อและขยายภาพ

2.12 Running Gaussian Average By Selectivity

เป็นการเลือกใช้อัลกอริธึม Running Gaussian Average ของในแต่ละกรณีที่ได้กำหนดไว้

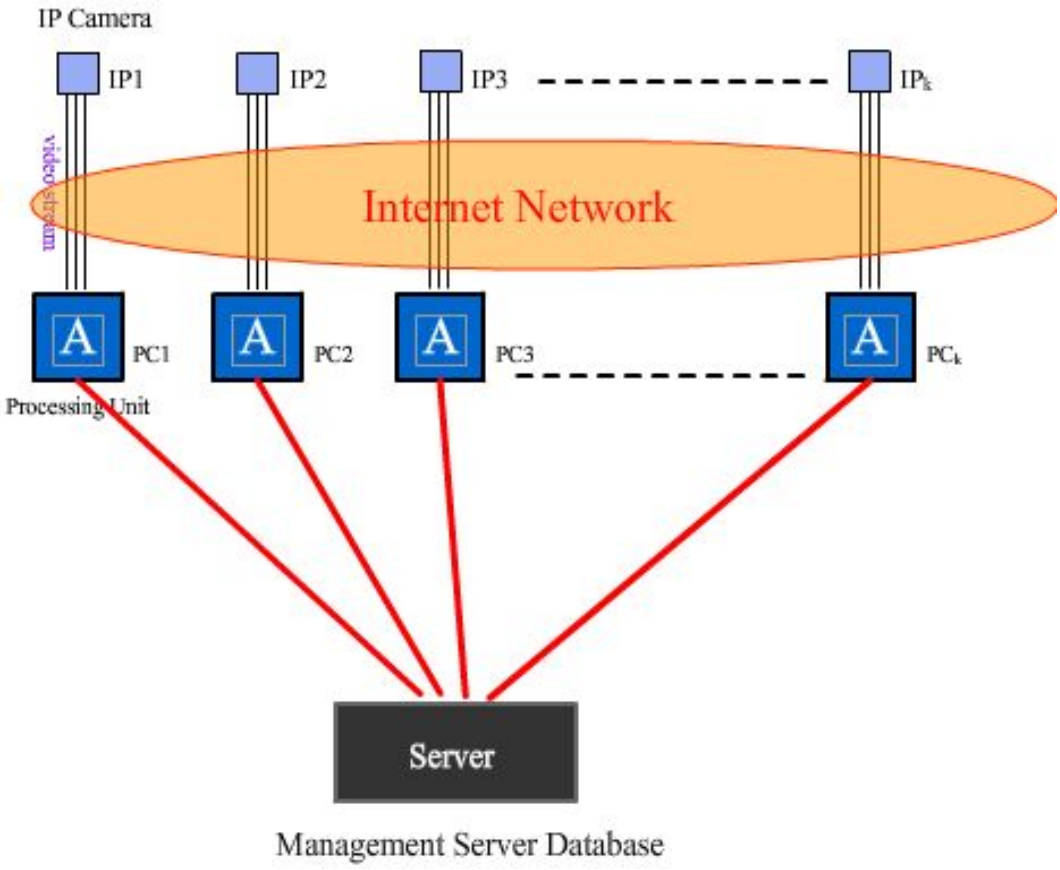
กรณีที่ 1 เมื่อภาพเคลื่อนไหวที่เป็น output > 90% ของพื้นที่ภาพทั้งหมดให้ใช้ค่าภาพเฉลี่ยแบบปกติ

กรณีที่ 2 เมื่อภาพเคลื่อนไหวที่เป็น output < 1% ของพื้นที่ภาพทั้งหมดให้ใช้ค่าภาพเฉลี่ยแบบตัดวัตถุออกไป

เมื่อทำผลที่ได้เสร็จแล้วนำไปแจกแจง output ตามค่าของ k

บทที่ 3 การออกแบบระบบ

3.1 โครงสร้างของระบบ IP Camera



รูปภาพ 3-1 โครงสร้างของระบบ IP Camera

IP1 … IPk คือ กล้องวีดีโอที่ตรวจจับรถทางเข้าตัวที่ 1…k

PC1 …PCk คือ คอมพิวเตอร์ที่นำมาประมวลผลตัวที่ 1…k

A คือ algorithm ที่ใช้ในการประมวลผล

Management Server Database คือ server ที่เก็บข้อมูลเฉพาะวิดีโอที่ประมวลผลมาจาก PC แล้ว

ข้อมูล input ของโปรแกรมนี้เป็นไฟล์วิดีโอสตรีมซึ่งรับค่ามาจาก IP Camera (IP1…IPk) ผ่านเครือข่ายอินเตอร์เน็ต ส่งข้อมูลเข้าสู่ PC หลังจากนั้นเอามาเข้ากระบวนการคำนวณผลลัพธ์โดยใช้

เนื่องจากขั้นตอนประมวลผล algorithm จะใช้ทรัพยากรเป็นอย่างสูงจึงจำเป็นจะต้องแบ่งการทำงานให้กับ PC อื่นๆ

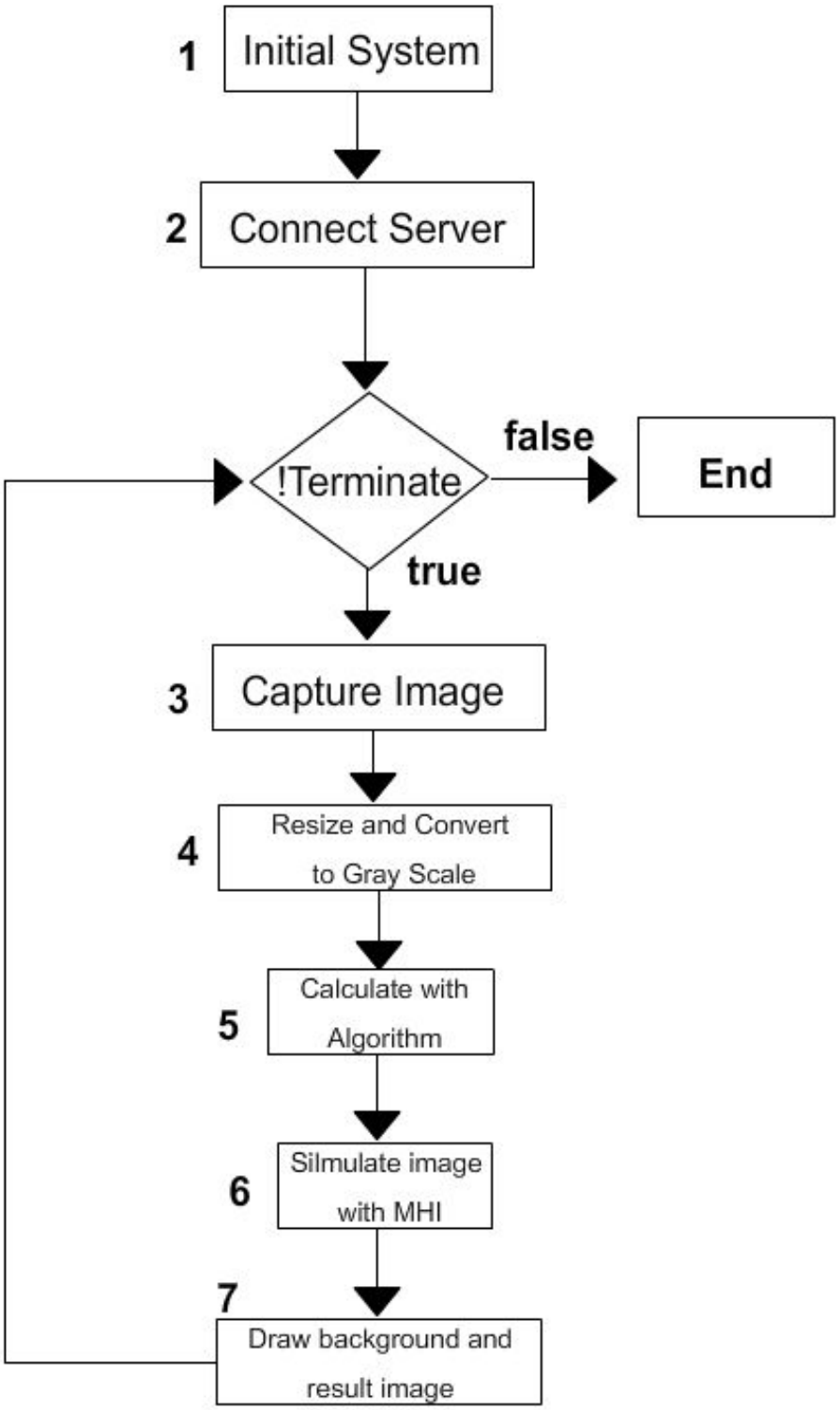
ในส่วนของอัลกอริธึมเราจำเป็นจะต้องทดสอบแต่ละวิธีให้ละเอียดเพื่อให้เราทำการ Implement จากอัลกอริธึมที่เราได้เลือกแล้วทำการปรับปรุงให้ดีขึ้น

เมื่อเสร็จสิ้นการประมวลผล PC แต่ละตัวจะส่ง output เข้าสู่ server ทำการจัดเก็บเป็นฐานข้อมูล ทำให้ เราจะดูแค่ output ของโปรแกรมที่เป็นไฟล์ clip ที่มีเฉพาะส่วนของรถวิ่งจากไฟล์วิดีโอ ซึ่งจะลดเวลาที่จะต้องดูย้อนหลังและลดทรัพยากรจัดเก็บได้เป็นอย่างมาก

3.2 Flow Diagram

จากการออกแบบสรุปได้เป็น Flow Diagram ของโปรแกรมดังนี้

ขั้นตอนการทำงานของโปรแกรม เมื่อโปรแกรมเริ่มทำงานจะมีการ Initial System ซึ่งประกาศตัวแปรที่จะใช้ในโปรแกรม กำหนดค่าเริ่มต้นให้แก่ตัวแปรต่างๆ ตามโค๊ดที่จะนำเสนอต่อไป หลังจากที่ผ่านการ initial System แล้ว โปรแกรมจะทำการติดต่อกับ sever เพื่อติดต่อไปยัง IP camera ด้วยหมายเลข IP ของกล้อง หลังจากนั้นโปรแกรมจะทำการตรวจสอบเงื่อนไขว่าต้องการจะหยุดการประมวลผลโปรแกรมต่อไปหรือไม่ ถ้าต้องการก็จะหยุดการทำงานของโปรแกรม แต่ถ้าไม่ต้องการโปรแกรมก็จะ ทำงานต่อไป โดยเริ่มจาก capture ภาพจากที่กล้องบันทึกได้มาเป็นภาพแบบไฟล์ bitmap เนื่องจาก input ข้อมูลเป็นวิดีโอสตรีมจำเป็นจะต้องบันทึกก่อนแล้วจึงจะนำมาใช้ จากนั้นทำการจะทำการลดขนาดของภาพ และแปลงเป็นภาพแบบขาวเทา เพื่อการคำนวณที่รวดเร็วขึ้น โปรแกรมจะนำภาพที่ได้จากการลดขนาดและแปลงเป็นภาพแบบ Grayscale แล้วมาทำการประมวลผลคำนวณหาผลลัพธ์ตามอัลกอริทึมที่ต้องการซึ่งโครงงานนี้ที่นี้คือ การทำBackground Subtraction ด้วย Running Average อัลกอริทึม หลังจากที่คำนวณหาผลลัพธ์ของพื้นหลังด้วยอัลกอริทึมแล้วก็นำมาหาวัตถุพร้อมทั้งจำลองรูปแบบการเคลื่อนไหวของภาพวัตถุโครงร่างด้วยวิธี Simulate image with MHI เพื่อแสดงผลภาพเค้าโครงของวัตถุได้



รูปภาพ 3-2 Flow Diagram *การทำงานของโปรแกรม*

บทที่ 4 รายละเอียดการทำงาน

4.1 การทำงานของโปรแกรมตัดพื้นหลังเชิงเทคนิค

1. Initial System

1.1 ทำการสร้าง Form ขึ้นมา

ทำในส่วน method onInitDialog ของ CDialog

```
CDialog::OnInitDialog();
```

1.2 กำหนดตัวแปรของระบบ

```
// define constant string
#define WCV_NAME                "LOADIMAGE"
#define WCV_BKG                 "BACKGROUND"
#define WCV_OBJ                 "OBJECT"
#define WCV_FILE_NAME           "file1.bmp"
#define WCV_IP          "192.168.17.131"

// Initial system variable
IplImage *gray_img, *bkg_img, *acc_img, *obj_img, *result_img;
IplImage *rsize_img,*rsize_acc ,*rsize_bkg;

// Set system variable value
rsize_img = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 3 );
rsize_acc = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 3 );
rsize_bkg = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 3 );
```

```
gray_img = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 1 );

acc_img = cvCreateImage( cvSize(320, 240), IPL_DEPTH_32F, 1 );

obj_img = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 1 );

bkg_img = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 1 );

result_img= cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 3 );
```

ทำการสร้างตัวแปรของภาพ พื้นหลัง วัตถุ และ ภาพแสดงผล ในขนาดความกว้าง 320 ความสูง 240 พิกเซล และกำหนดขนาดจำนวน bit และ จำนวน channel

1.3 สร้างหน้าต่างแสดงผล

```
// Create background and object windows

    cvNamedWindow(WCV_BKG);

    cvNamedWindow(WCV_OBJ);
```

สร้างส่วนแสดงผลของพื้นหลังและวัตถุ

2. Connect to Server Program

```
    m_Xmpeg.Connect(WCV_IP);
```

เริ่มการเชื่อมต่อไปยัง Server program โดยใช้ค่า WCV_IP คือ 192.168.17.131

3. Capturing Image Stream

```
    IplImage *img;

    m_Xmpeg.SaveImage(WCV_FILE_NAME);

    img = cvLoadImage(WCV_FILE_NAME);
```

เนื่องจาก input ของโปรแกรมเป็น stream ดังนั้นเราต้องจับภาพเป็น realtime เพื่อทำหารโหลดข้อมูลภาพมาใช้ในแต่ละเฟรมใช้หลักการเดียวกับบัฟเฟอร์

4. Resize and Convert to Gray

```
cvResize(img, rsize_img);

cvReleaseImage(&img);



cvCvtColor( rsize_img, gray_img, CV_BGR2GRAY );
```

เพื่อการคำนวณภาพที่รวดเร็วยิ่งขึ้นจำเป็นต้องใช้ภาพที่ resize แบบ Gray Scale

5. Calculating with Algorithm

นำค่าของภาพที่ resize แล้วมาคำนวนโดยใช้วิธี Running Average เพื่อหาค่าของพื้นหลังเพื่อนำไปใช้ในการจำลองระบบ

```
cvRunningAvg(*img, *acc, alpha, NULL);

cvConvertScale(*acc, *bkg, 1.0, 0.0);
```

โดยที่

alpha = 0.005

acc คือ ค่าที่ทดเอาไว้เพื่อนำมาใส่ค่าใน bkg พร้อมทั้งเปลี่ยนขนาด 1 เท่า

6. Silmulate motion with MHI

เป็นการจำลองระบบด้วยวิธี MHI (Motion History Image) ใน function

```
update_mhi( rsize_img,bkg_img, result_img, 30 );
```

โดยที่

rsize_img คือ ค่า รูปภาพ capture ที่ทำการ resize แล้ว

bkg_imgคือ ค่าพื้นหลังของรูปที่ทำการคำนวณจาก algorithm แล้ว

result_img คือ ค่าของภาพที่ต้องการให้แสดงผลในหน้าต่าง object

100 คือ ค่า threshold ที่กำหนดไว้

1) Initial Variable

```
// numble of cyclic frame
```

```
const int N = 4;


// ring image buffer
IplImage **buf = 0;


IplImage* silh = 0;  //ค่าของรูปจำลอง output object
 CvSeq* seq;   //ค่า segmentation mask
IplImage *mhi = 0; // MHI    ค่า MHI
IplImage *orient = 0; // orientation   ค่ามุมองศา
IplImage *mask = 0; // valid orientation mask  ค่า mask ของมุมองศา
IplImage *segmask = 0; // motion segmentation map    ค่า segmentation mask
CvMemStorage* storage = 0; // temporary storage  ค่าหน่วยเก็บข้อมูลชั่วคราว
```

2) Allocating buffer image

```
buf = (IplImage**)malloc(N*sizeof(buf[0]));
```

สร้าง array ของตัวแปร IplImage ตามจำนวน N x buf ตัวแรก

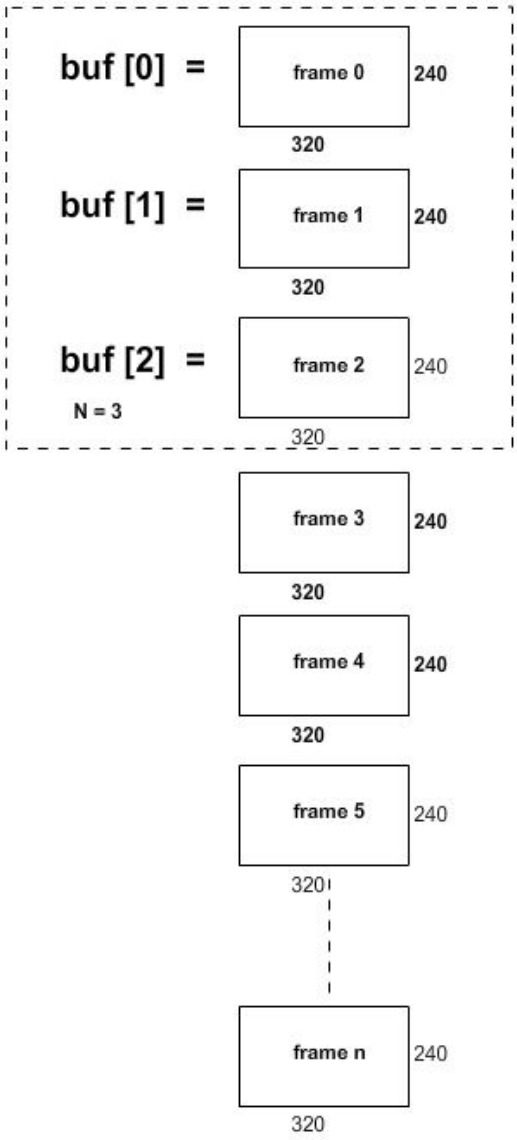
```
memset( buf, 0, N*sizeof(buf[0]));
```

จองหน่วยความจำให้แก่ buf จำนวน N x buf ตัวแรก

3) Create buffering cyclic

```
for( i = 0; i < N; i++ ) {
    cvReleaseImage( &buf[i] );
    buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
    cvZero( buf[i] );
  }
```

สร้าง buffer image N (N = 4) ตัวที่มีการ update ตลอดเวลา พร้อมทั้งให้ค่าเริ่มต้นเป็น 0 จากรูป



รูปภาพ 4-1 ภาพประกอบการทำ frame buffer

```
Create MHI and Segmentation variable

// create MHI image with 32 bit color 1 channel

mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );

cvZero( mhi ); // clear MHI at the beginning

orient = cvCreateImage( size, IPL_DEPTH_32F, 1 );

segmask = cvCreateImage( size, IPL_DEPTH_32F, 1 );
```

```
mask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
```

4) Change image to gray scale buffer

```
cvCvtColor( img, buf[last], CV_BGR2GRAY ); // convert frame to grayscale
```

5) Find object and Update MHI

```
1  cvAbsDiff( buf[idx1], bkg, silh ); // get difference between frames

2  cvThreshold( silh, silh, diff_threshold, 1, CV_THRESH_BINARY ); // and threshold
it

3 cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); // update MHI
```

// Note ทำการหาผลต่างของ frame โดยใช้ function cvAbsDiff() จากค่า

$$buf[idx1] - bkg = silh$$

// Note ทำการปรับค่า threshold ให้ object เพื่อการแสดงผลที่ดีขึ้น โดยใช้ thresholde = 100

// Note ทำการ update MHI จาก object silh ไปสู่ mhi ในช่วงเวลา 1 ลูกคลื่นของนาฬิกา

6) Convert scale MHI and Calculate Motion Gradian

```
1  cvCvtScale( mhi, mask, 255./MHI_DURATION,
        (MHI_DURATION - timestamp)*255./MHI_DURATION );
   cvZero( dst );
2  cvCvtPlaneToPix( mask, 0, 0, 0, dst );

   // calculate motion gradient orientation and valid orientation mask
3  cvCalcMotionGradient( mhi, mask, orient, MAX_TIME_DELTA,
   MIN_TIME_DELTA, 3 )
```

1) ทำการ convert ภาพ MHI ไปเป็นภาพ mask 8 bit
2) นำค่า mask ไปใส่ใน dst เพื่อรอการวาด

3) คำนวณองศา motion gradian จาก MHI ไปใส่ใน mask จาก MIN_TIME_DELTA ไปสู่ MAX_TIME_DELTA

7) Create Storage and Segmentation

```
if( !storage )
    storage = cvCreateMemStorage(0);
  else
    cvClearMemStorage(storage);
  // ขณะที่ยังไม่มีการสร้าง storage ก็ให้สร้างมิฉะนั้น จะเคลียร์ storage
  // segment motion: get sequence of motion components
  // สร้าง segment เพื่อเก็บการเคลื่อนไหวโดยรวมของภาพ
  seq = cvSegmentMotion( mhi, segmask, storage, timestamp, MAX_TIME_DELTA );

  // iterate through the motion components,
  for( i = -1; i < seq->total; i++ ) {
      // ถ้าเป็นการทำงานรอปแรกให้กำหนดเส้นสีขาว มีรัศมี 100 พิกเซลและกล่องขนาด
320 x 240
    if( i < 0 ) { // case of the whole image
      comp_rect = cvRect( 0, 0, size.width, size.height );
      color = CV_RGB(255,255,255); //color = white
      magnitude = 100;
    }
    else { // i-th motion
      // หา element ตัวที่ i ของอนุกรมภาพ ( sequence image)
      comp_rect = ((CvConnectedComp*)cvGetSeqElem( seq, i ))->rect;
```

```
            //ถ้า ขนาดของ element ตัวที่ i เล็กมากให้วนรอปต่อไป
              if( comp_rect.width + comp_rect.height < 50 ) // skip very small components
               continue;
            // กำหนดสีของเส้นเป็นสีแดง และมีรัศมี 100 พิกเซล
              color = CV_RGB(255,0,0);  // color = red
              magnitude = 30;                          // fixed magnitude
            }
```

8) Select Component ROI

```
            // select component ROI
            cvSetImageROI( silh, comp_rect );
            cvSetImageROI( mhi, comp_rect );
            cvSetImageROI( orient, comp_rect );
            cvSetImageROI( mask, comp_rect );
```

นำภาพ silh , mhi , orient , mask ไปใส่ไว้ในกล่องสี่เหลี่ยม

9) Draw empty circles

```
            // หาจุดศูนย์กลางของภาพ
            center = cvPoint( (comp_rect.x + comp_rect.width/2),
                              (comp_rect.y + comp_rect.height/2) );
          // วาดวงกลมมีเฉพาะขอบให้มีสีที่ระบุ ความหนา 3 pixel รัศมี magnitude x 1.2
            cvCircle( dst, center, cvRound(magnitude*1.2), color, 3, CV_AA, 0 );
```

7. Draw Background and Result

```
            cvShowImage(WCV_BKG, bkg_img);
            cvShowImage(WCV_OBJ, result_img);
          //สั่งให้โปรแกรมแสดงภาพพื้นหลัง และ ภาพเค้าร่างวัตถุ
```

รูปภาพ 4-2 ภาพต้นฉบับ

รูปภาพ 4-3 ภาพวัตถุ

รูปภาพ 4-4 ภาพพื้นหลัง

4.2 ขั้นตอนวิธีการทำ Running Gaussian Average

1) Initial Parameter

- กำหนดค่า k , α

static double alpha = 0.5;

static int k = 8;

- ทำการสร้างรูปที่เก็บค่า gray scale ของ σ และ μ

IplImagemue = cvCreateImage(cvSize(320, 240), IPL_DEPTH_8U, 1);

IplImageomega= cvCreateImage(cvSize(320, 240), IPL_DEPTH_8U, 1);

Process realtime image

1) Receive output

- ทำการรับค่า output จากเฟรมก่อนหน้า ได้แก่ค่า μ , σ โดยรับค่าจาก function calRGA ที่เขียนขึ้นมา

VSAAPI(void) calRGA(IplImage *input ,IplImage *mue, IplImage *omega , IplImage *output)

{

.

.

.

}

2) Compute output

- หาค่า $I_{out} = | I_{in} - I_{\mu} | > kI_{\sigma}$

```
if( abs( ((int)(unsigned char)input->imageData[i] -(int)(unsigned char)mue-
>imageData[i]))  >

        (k * (int)(unsigned char)omega->imageData[i]) )



        output->imageData[i] = (char)-1;  //ค่า -1 เป็นสีดำ

else

        output->imageData[i] = (char)0;   // ค่า 0 เป็นสีขาว
```

3)Update μ

- หาค่า $I_{\mu t+1} = \alpha (I_{in}) + (1- \alpha) I_{\mu t}$

```
temp      = (alpha * (int)(unsigned char)input->imageData[i]) +

((1 - alpha) * (int)(unsigned char)mue->imageData[i]);

mue->imageData[i] = (char)(int)temp;
```

4) Update σ

- $I_{(\sigma^2)t+1} = \alpha (I_{in} - I_{\mu t+1})2 + (1- \alpha) I_{\sigma t}$

```
temp      = alpha * pow(((int)(unsigned char)input->imageData[i] - (int)(unsigned
char)mue->imageData[i]) , 2);

temp      += (1 - alpha) * pow(((int)(unsigned char)omega->imageData[i]), 2);
```

5) Show input / μ / σ / output

```
#define WCV_INPUT              "INPUT"

#define WCV_OUTPUT             "OUTPUT"

#define WCV_MUE                "MUE"

#define WCV_OMEGA              "OMEGA"

cvShowImage(WCV_INPUT, gray_img);

cvShowImage(WCV_MUE, mue_img);

cvShowImage(WCV_OMEGA, omega_img);

cvShowImage(WCV_OUTPUT, output_img);
```

5) Release Parameter

- ลบ pointer ของ I_{in} I_{out} I_{σ} I_{μ} ออกจากโปรแกรม

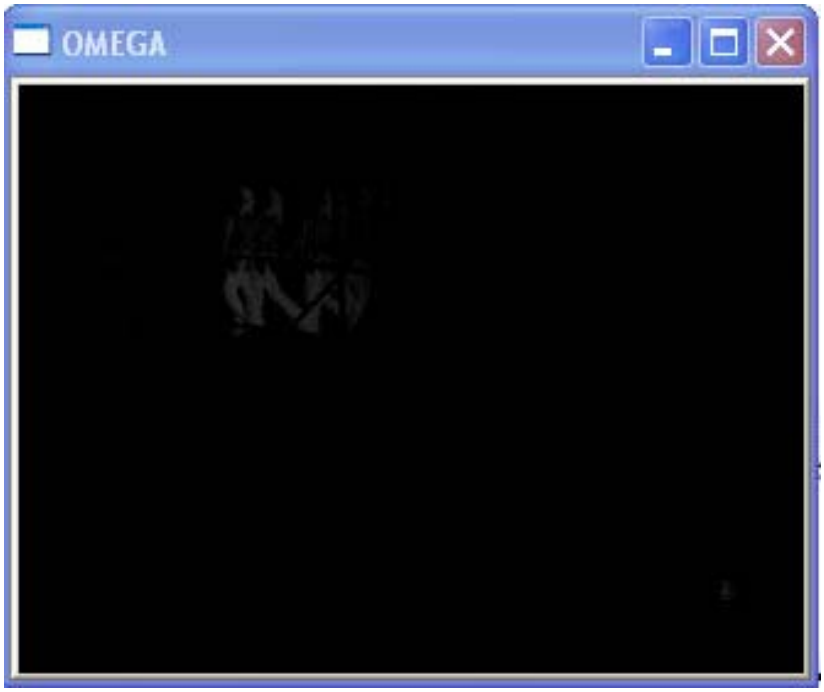
```
cvReleaseImage(&mue);

cvReleaseImage(&omega)
```

ผลลัพธ์ของโปรแกรม

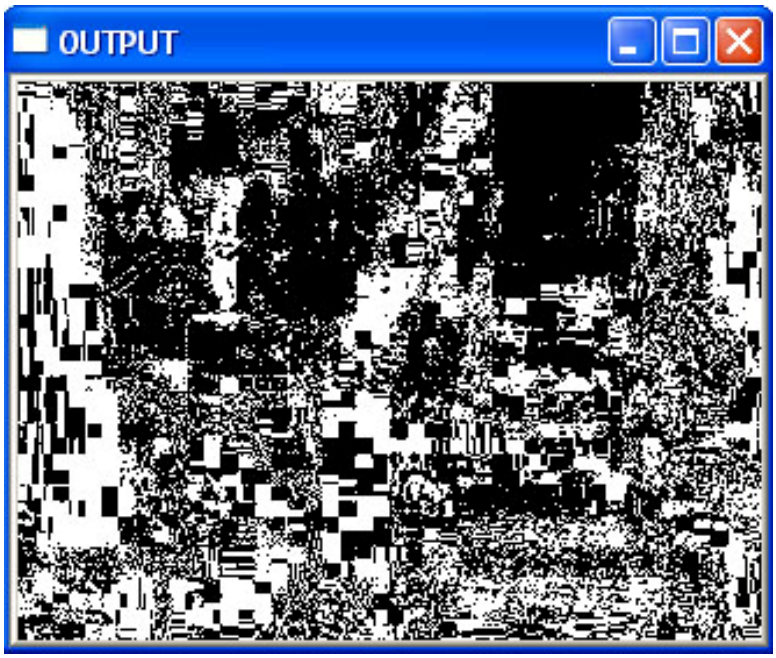
กำหนดค่า $\alpha = 0.5$, $k = 8$

รูปภาพ 4-5 ภาพต้นฉบับ

รูปภาพ 4-6 ภาพค่าเฉลี่ย



รูปภาพ 4-7 ภาพค่าความแปรปรวน



รูปภาพ 4-8 ภาพผลลัพธ์

4.3 ขั้นตอนการทำ Erosion / Dilation

1) Initialization

```
IplConvKernel* element = 0;

int element_shape = CV_SHAPE_ELLIPSE;

int max_iters = 8;

int erode_dilate_pos = 0;
```

```
erode_dilate_pos = max_iters;

source = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 1 );

dest = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 1 );

cvNamedWindow("Erode/Dilate");
```

```
cvCreateTrackbar("iterations","Erode/Dilate",&erode_dilate_pos,max_iters*2+1,ErodeDilat
e);
```

- ทำการกำหนด element shape template โดยให้ค่ารูปร่าง template เป็นวงรี CV_SHAPE_ELLIPE พร้อมกับสร้างภาพเพื่อทำสำเนาในการคำนวณ และภาพผลลัพธ์ขนาด 320 x 240 pixel
- กำหนดค่าการวนรอบเริ่มต้นเป็น 8 และสร้าง Track Bar เพื่อการแสดงผลของการทำ Erosion / Dilation ในหน้าต่างของ "Erode/Dilate"

2) Processing

```
int n = erode_dilate_pos - max_iters;

int an = n > 0 ? n : -n;

element = cvCreateStructuringElementEx( an*2+1, an*2+1, an, an, element_shape, 0 );

if( n < 0 )

{
```

```
    cvErode(source,dest,element,1);
  }
  else
  {
    cvDilate(source,dest,element,1);
  }

      cvShowImage("Erode/Dilate",dest);
```

- ทำการสร้าง template ขนาด (n x 2)+ 1 รูปวงรี พร้อมกับเปรียบเทียบค่าของ n ที่ได้จาก Track bar โดยที่

  ถ้า n < 0 ไปทำ function Erode

  ถ้า n > 0 ไปทำ function Dilate

- แสดงผลในหน้าต่าง "Erode/Dilate" โดยใช้ภาพ dest

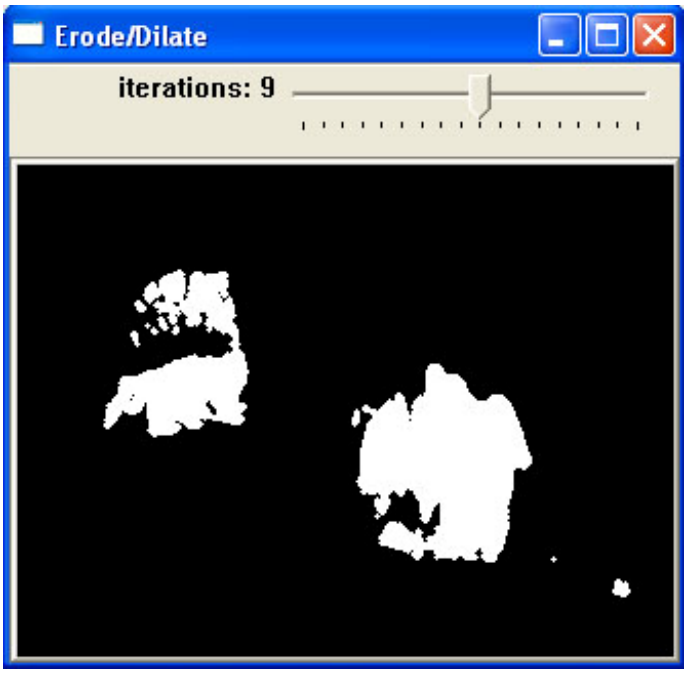
3) Release Parameter

```
cvReleaseImage(&sourc);
cvReleaseImage(&dest);
cvDestroyWindow("Erode/Dilate");
cvReleaseStructuringElement(&element);
```

- ทำการลบพารามิเตอร์ของขั้นตอนทั้งหมดไป

ผลลัพธ์จากการทำ Erosion / Dilation

รูปภาพ 4-9 ภาพผลลัพธ์



รูปภาพ 4-10 ภาพหลังการทำ Erode/Dilate

4.4 Running Gaussian Average By Selectivity

เข้าไปแก้การทำงานของ Running Gaussian Average ดังนี้

1) เพิ่มเติมส่วน Initialization

```
static int k1       = 10;

static int k2       = 15;

static int k3       = 5;

IplImage *output2   = 0;

IplImage *output3   = 0;
```

```
output2             = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 3 );

output3             = cvCreateImage( cvSize(320, 240), IPL_DEPTH_8U, 3 );

cvNamedWindow("OUTPUT2");

cvNamedWindow("OUTPUT3");
```

- เป็นการกำหนดค่า K1 , K2 , K3 และส่วนแสดงผล output ที่ 2 และ 3

2) เพิ่มเติมส่วน Processing

```
//output 2


if(     (unsigned char)omega->imageData[i] > k2)

{

        output2->imageData[i*3] = (char)255;

        output2->imageData[i*3+1] = (char)0;

        output2->imageData[i*3+2] = (char)0;

}
```

```
else
{
        output2->imageData[i*3] = (char)0;
        output2->imageData[i*3+1] = (char)0;
        output2->imageData[i*3+2] = (char)0;

}

//output 3
if(     (unsigned char)omega->imageData[i] > k3)
{
        output3->imageData[i*3] = (char)0;
        output3->imageData[i*3+1] = (char)255;
        output3->imageData[i*3+2] = (char)0;
}
else
{
        output3->imageData[i*3] = (char)0;
        output3->imageData[i*3+1] = (char)0;
        output3->imageData[i*3+2] = (char)0;

}
```

```
}

double percentMotion =  (cvCountNonZero(output) / (320 * 240) ) * 100;

//Running Gaussian By Selectiviy
if(percentMotion <= 1)    // case motion < 1% of total area
{
        //find average without object
        cvAbsDiff(mue, obj , acc);
        cvCopyImage(acc , mue);
}
else if(percentMotion >= 90)  // case motion > 90% of total area
{
        mue = mue;
}
cvShowImage("OUTPUT2",output2);
cvShowImage("OUTPUT3",output3);
```

- ทำการหาค่าเปอร์เซ็นต์ของวัตถุเทียบจากพื้นที่ทั้งหมด แล้วนำมาตรวจสอบว่าอยู่ในกรณีใดของ Running Gaussian Average By Selectivity พร้อมทั้งแจกแจงผลลัพธ์ตามค่า k ซึ่งแสดงผลในแต่ละหน้าต่าง

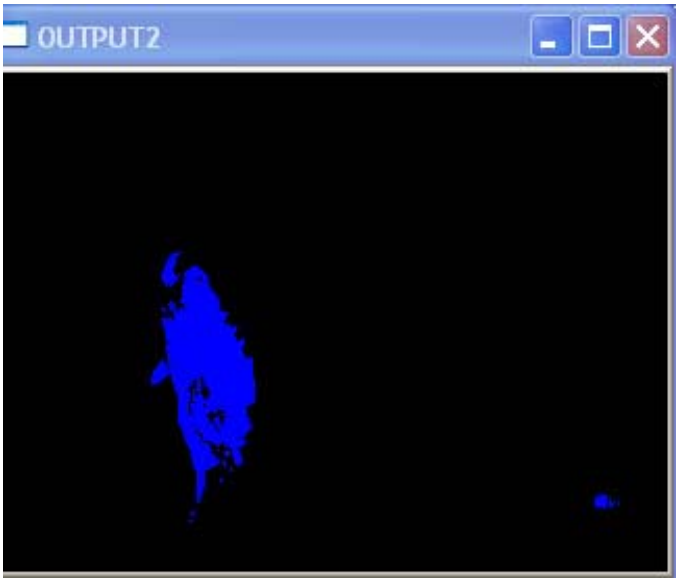
3) เพิ่มเติมส่วน Release Parameter

```
cvReleaseImage(&output2);

cvReleaseImage(&output3);



cvDestroyWindow("OUTPUT2");

cvDestroyWindow("OUTPUT3");
```

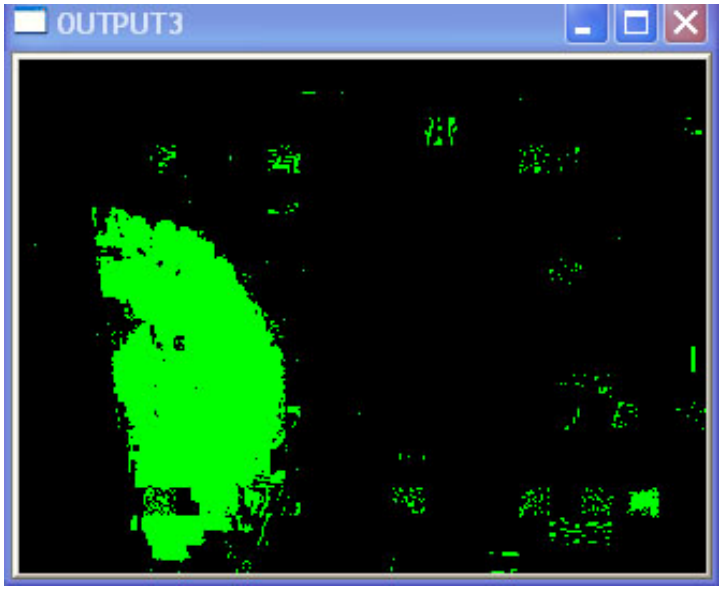
ผลลัพธ์จากการทำ Running Gaussian Average By Selectivity

รูปภาพ 4-11 ภาพต้นฉบับ

รูปภาพ 4-12 K1=10



รูปภาพ 4-13 K2 = 20



รูปภาพ 4-14 K3 = 1

บทที่ 5 *สรุปผลและข้อเสนอแนะ*

สรุปผล

1. ศึกษาอัลกอริธึมหาพื้นหลังสำหรับวิธี Background Substraction พร้อมทั้งเปรียบเทียบข้อดีและข้อด้อยของแต่ละวิธี

2. ทดสอบระบบกล้อง IP Camera จากโปรแกรม capture วิดีโอสตรีมของทางฝั่ง server พร้อมทั้งเรียนรู้ขั้นตอนการทำงานของโปรแกรม

3. เรียนรู้การใช้ Library openCV ที่ในหัวข้อ Background Substraction และ Motion Template

4. ศึกษาวิธีการทำ Motion Template เพื่อทำการหาภาพโครงร่างของวัตถุที่เคลื่อนไหว

5. ทำความเข้าใจ source code ของโปรแกรมฝั่ง client ที่มีอยู่ให้

6. implement โปรแกรมในข้อ 5 เพื่อหาวัตถุโดยใช้วิธี Background Substraction ของอัลกอริธึม Running Average จาก input ของโปรแกรมฝั่ง server

7. นำโปรแกรมในข้อ 6 เพื่อหาภาพโครงร่างของวัตถุที่เคลื่อนไหวโดยวิธี Motion Template

8. implement โปรแกรมในข้อ 5 เพื่อหารูผลลัพธ์โดยใช้วิธี Background Substraction ของอัลกอริธึม Running Guassian Average จาก input ของโปรแกรมฝั่ง server

9. ปรับปรุงผลลัพธ์ด้วยวิธีของ Morphology Image Processing จาก อัลกอริธึม Running Guassian Average

10. เพิ่มเติมอัลกอริธึมแบบ Running Guassian Average By Selectivity พร้อมทั้งแจกแจงผลลัพธ์ตามค่า k

ยิ่งค่า k มาก วัตถุที่ได้จะคมชัด

ยิ่งค่า k น้อย วัตถุที่ได้จะมี noise มาก

ปัญหาและแนวทางการแก้ไข

1. Overflow pixel value เป็นการแสดงค่าของ pixel เกินจากการใช้งานจริงเช่น

ค่าของ pixel ณ จุดพิกัด x,y ของเฟรมแรกและเฟรมถัดไปเป็น 200 และ 150 เมื่อมีการนำค่า 2 ค่านี้มาคูณกันตามสมการ Running Guassain Average จะมีการ overflow เกิดขึ้นถ้ามีการ

2. ค่าที่คำนวณเป็นผลไม่ตรงกับทฤษฎี เนื่องจากการ casting character ใน image เพื่อนำมาคำนวณ เช่น

```
temp      = (alpha * input->imageData[i]) +
((1 - alpha) * mue->imageData[i]);

mue->imageData[i] = temp;
```

หลังการ casting ที่ถูกต้องจะเป็น

```
temp      = (alpha * (int)(unsigned char)input->imageData[i]) +
((1 - alpha) * (int)(unsigned char)mue->imageData[i]);

mue->imageData[i] = (char)(int)temp;
```

ทั้งนี้เนื่องจากค่าของ pixel ในตัวแปร sign char มีต่าตั้งแต่ -127 ถึง 127 แต่ค่าที่เราใช้งานจริงมีค่าตั้งแต่ 0 ถึง 255

3. การหาค่า k ที่เหมาะสมกับสมการ เนื่องจากค่า k เป็นค่าที่เหมาะสมของกราฟรูประฆังคว่ำ ยิ่งค่า k มากๆ noise ที่ได้จะลดลง ในขณะที่ภาพที่ได้จะห่างไกลจากค่าเฉลี่ยเรื่อยๆ

บรรณานุกรม

1. A robust and computationally ef_cient motion detection algorithm based on Σ-Δ background estimation. By A . Manzanera J. C. Richefeu

F-75739 PARIS CEDEX 15

http://www.ensta.fr/_ manzaner

2. M. Piccardi. Background subtraction techniques: a review.

http://www-staff.it.uts.edu.au/¬ massimo/

3. S.-C. Cheung and C. Kamath. Robust techniques for background

subtraction in urban traf_c video. In *Proc. SPIE*

Video Communications and Image Processing, San Jose -

CA, 2004.

4. A. Elgammal, D. Harwood, and L. Davis. Non-parametric

Model for Background Subtraction. In *Proc. IEEE European*

Conference on Computer Vision, Dublin - Ireland,

2000.

5. Open Source Computer Vision Library Reference Manual

Copyright © 1999-2001 Intel Corporation

All Rights Reserved

Issued in U.S.A.

Order Number: TBD

World Wide Web: http://developer.intel.com

ดัชนีฟังก์ชัน

CVAPI(IplImage*) cvCreateImage(CvSize size, int depth, int channels)

size คือ ค่าของขนาดเฟรมของภาพทั้งความยาวและความสูง

depth คือ ค่าของจำนวนบิตต่อชาเนลของจุดสี

channel คือ จำนวนชาเนล

รายละเอียด

เป็นฟังก์ชันการสร้างภาพขึ้นมากำหนดขนาดเฟรม ค่าของจำนวนบิตต่อชาเนลของจุดสี และจำนวนชาเนล

CV_INLINE CvSize cvSize(int width, int height)

width คือ ค่าของความกว้างของเฟรม

height คือ ค่าของความกว้างของเฟรม

รายละเอียด

เป็นฟังก์ชันการสร้างขนาดเฟรมของภาพขึ้นมา

CVAPI(int) cvNamedWindow(const char* name)

name คือ ชื่อของหน้าต่าง

รายละเอียด

เป็นฟังก์ชันการหน้าต่างสำหรับแสดงภาพ

CVAPI(void) cvShowImage(const char* name, const CvArr* image)

name คือ ชื่อของหน้าต่าง

image คือ ค่าของภาพ

รายละเอียด

เป็นฟังก์ชันการแสดงผลหน้าต่างที่ใช้สำหรับแสดงภาพ

CVAPI(IplImage*) cvLoadImage(const char* filename)

filename คือ ชื่อของไฟล์ภาพที่จะโหลดมาใช้

รายละเอียด

เป็นฟังก์ชันที่จะโหลดภาพมาใช้

CVAPI(void) cvCvtColor(const CvArr* src, CvArr* dst, int code)

src คือ ภาพ input

dst คือ ภาพ output

code คือ รหัสสำหรับการแปลงสี

รายละเอียด

เป็นฟังก์ชั่นที่ใช้สำหรับแปลงสีจากภาพสีเป็นขาวดำ

CVAPI(void) cvRunningAvg(const CvArr* image, CvArr* acc, double alpha,

const CvArr* mask CV_DEFAULT(NULL))

image คือ ภาพ input

acc คือ ภาพ output

alpha คือ ค่าของอัลฟ่าที่ใช้สำหรับคำนวนสูตร

mask คือ ค่า mask ปกติเป็น 0

รายละเอียด

เป็นฟังก์ชั่นที่ใช้สำหรับหาพื้นหลังของภาพด้วยวิธี Running Average

CVAPI(void) cvAbsDiff(const CvArr* src1, const CvArr* src2, CvArr* dst)

src1 คือ ภาพ input1

src2 คือ ภาพ input2

dst คือ ภาพ output

รายละเอียด

เป็นฟังก์ชั่นที่ใช้สำหรับหาหาผลต่างของภาพ input 1 และ input 2

```
CVAPI(void)  cvUpdateMotionHistory( const CvArr* silhouette, CvArr* mhi,
                                    double timestamp, double duration );
```

silhouette คือ ภาพโครงร่าง

mhi คือ ภาพ MHI

timestamp คือ เวลาที่ผ่านไปตั้งแต่เริ่มโปรแกรม

duration คือ ช่วงเวลาที่ต้องการ

รายละเอียด

เป็นฟังก์ชั่นที่ใช้การทำ Update MHI

```
CVAPI(void)  cvCalcMotionGradient( const CvArr* mhi, CvArr* mask, CvArr* orientation,
                                   double delta1, double delta2,
                                   int aperture_size CV_DEFAULT(3)))
```

mhi คือ ภาพ MHI

mask คือ ค่า mask

orientation คือ ค่าองศา

delta1 คือ ค่าเวลาที่เปลี่ยนแปลงสูงสุด

delta2 คือ ค่าเวลาที่เปลี่ยนแปลงสูงสุด

aperture_size คือ ขนาดของอนุพันธ์ ปกติจะเป็น 3

รายละเอียด

เป็นฟังก์ชั่นที่ใช้การคำนวณ Motion Gradient