หนังสือรับรองความเป็นเอกลักษณ์

ผู้จัดทำรายงานฉบับนี้ขอรับรองว่ารายงานฉบับนี้เป็นรายงานที่มีความเป็นเอกลักษณ์ โดยที่ผู้จัดทำมิได้คัดลอกมาจากที่ใดที่หนึ่ง เนื้อหาทั้งหมดถูกรวบรวมตามขั้นตอนการดำเนินงานในการจัดทำโครงงาน และหากมีการคัดลอกข้อความหรือส่วนใดส่วนหนึ่งมาจากแหล่งที่มาหรือเอกสารอื่น ผู้จัดทำก็ได้มีการอ้างอิงถึงเอกสารต้นฉบับเหล่านั้นไว้อย่างเหมาะสมแล้วในส่วนของบรรณานุกรม และขอรับรองว่ารายงานฉบับนี้เป็นรายงานที่มิได้เสนอต่อสถาบันใดมาก่อน

.....

(นาย ฐานิศร์ เฉลิมบุญ)

ผู้จัดทำ

17 ตุลาคม พ.ศ. 2551

กิตติกรรมประกาศ

โครงงาน Motion Detection by Optical Flow สามารถดำเนินงานให้บรรลุวัตถุประสงค์ที่ตั้งไว้ เนื่องจากได้รับความสนับสนุนและช่วยเหลือในด้านต่างๆจากบุคคลต่อไปนี้

ดร.นิคม สุวรรณวร ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ได้ให้ความรู้ ข้อเสนอแนะ และช่วยเหลือ ตลอดจนแนวคิดในการจัดทำโครงงาน

ผศ.ดร.ธเนศ เคารพาพงศ์ รศ.ดร.มนตรี กาญจนะเดชะ ซึ่งเป็นอาจารย์ที่ปรึกษาร่วมโครงงาน ที่ได้ให้ข้อแนะนำและตรวจสอบโครงงาน

และขอขอบคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่าน ที่ได้ประสิทธิ์ประสาทวิชาความรู้

ขอขอบคุณเจ้าหน้าที่ของวิศวกรรมคอมพิวเตอร์ทุกท่าน ที่ได้อำนวยความสะดวกในด้านต่างๆ

ขอขอบคุณเพื่อน ๆ และรุ่นพี่ทุกคน ที่ให้ความร่วมมือในการทดสอบโครงงาน และเป็นกำลังใจที่ดีในการจัดทำโครงงานนี้มาโดยตลอด

สุดท้ายนี้ ขอระลึกถึงพระคุณบิดามารดาที่ได้เลี้ยงดู อบรมสั่งสอนจนเติบใหญ่และยังคอยห่วงใยเรื่อยมา

ฐานิศร์ เฉลิมบุญ

ผู้จัดทำ

บทคัดย่อ

โครงงานนี้มีจุดประสงค์เพื่อพัฒนาระบบประมวลผลภาพเพื่อใช้ในระบบตรวจสอบผู้บุกรุกหรือระบบเฝ้าระวัง ซึ่ง motion detection เป็นหนึ่งในวิธีสำคัญเพื่อตรวจจับวัตถุที่กำลังเคลื่อนที่ โดยได้เน้นไปที่กรณีศึกษาในกรณีที่ตัวกล้องสามารถเคลื่อนที่ไปด้วยได้ โดยที่การใช้เทคนิค background subtraction ไม่สามารถที่จะนำไปใช้ได้ การจับความเคลื่อนไหวโดย optical flow จึงได้ถูกนำมาใช้ โดยสามารถใช้ในการหาความเร็ว ทิศทางของวัตถุ และยังสามารถแยกแยะและระบุวัตถุที่กำลังเคลื่อนที่ได้ สำหรับอัลกอริทึมที่ได้พัฒนาขึ้นในโครงงานนี้ถูกพัฒนาขึ้นโดยคำนึงถึงการทำงานในแบบเรียลไทม์ ซึ่งผู้จัดทำหวังเป็นอย่างยิ่งว่าโครงงานนี้จะมีส่วนสำคัญต่อการพัฒนาระบบรักษาความปลอดภัย

Abstract

The aim of this project is emphasized on the development of image processing technique applied for the surveillance system. Motion detection is one of the most important methods in order to detect the moving object in a scene. We focalize on the case study that the camera is also moving which the classical background subtraction techniques can not be applied. Motion detection from optical flow is then considered. Optical flow field image represents the projective motion field of moving objects in the real 3D space. It can be expressed in term of the speed and direction of each local image pixel. By applying our classification method on these characteristics, the moving objects in the considering image could be segmented and identified. Our algorithm developed in this project is also taken in to account the computational time that must be executed in real time. Hopefully, the encouraged results from this project will significantly contribute to the development of an intelligent security system.

บทที่ 1 *บทนำ*

1.1 ความเป็นมาและความสำคัญ

เนื่องจากในสถานการณ์บ้านเมืองปัจจุบัน เกิดความไม่สงบขึ้นภายในประเทศ ทำให้เกิดความวุ่นวาย ความไม่ปลอดภัยต่อทรัพย์สิน และร่างกาย ดังนั้นจึงต้องมีระบบรักษาความปลอดภัย(Security)

ระบบความปลอดภัย (Security) คือ ระบบที่มีไว้ป้องกันทรัพยากรหรือทำการเก็บข้อมูลซึ่งถือว่าเป็นระบบที่มีความจำเป็นต่อสถานที่หรือบริเวณที่ไม่มีผู้บุคคลเข้าออก ซึ่งผู้ทำได้เล็งเห็นถึงความสำคัญในการดูแลความปลอดภัยในบริเวณดังกล่าว จึงมีความพยายามที่จะสร้างระบบตรวจสอบผู้บุกรุกหรือระบบเฝ้าดู (Surveillance System) ภายในบ้านหรือสำนักงานผ่านระบบวิดีโอวงจรปิดขึ้นมา

ระบบตรวจสอบผู้บุกรุกหรือระบบเฝ้าดู (Surveillance System) ภายในบ้านหรือสำนักงานผ่านระบบวิดีโอวงจรปิด เป็นระบบรักษาความปลอดภัยอย่างหนึ่งที่มีการทำงานระหว่างซอฟต์แวร์จับภาพและกล้อง โดยมีการบันทึกภาพผ่านกล้องวงจรปิดเพื่อเก็บเป็นหลักฐาน ซึ่งเหมาะสมกับการจับภาพในบริเวณห่วงห้ามเมื่อมีการเปลี่ยนแปลงเข้ามาในกล้อง

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อวิเคราะห์การเคลื่อนไหวของวัตถุด้วยการประมวลผลภาพ ด้วยวิธีการหา Optical Flow
2. เพื่อแยกวัตถุที่เคลื่อนที่ในทิศทางที่แตกต่างกันได้
3. เพื่อใช้ในระบบความปลอดภัย ซึ่งรับภาพจากกล้อง มาวิเคราะห์
4. สามารถทำงานแบบ Real Time ได้

1.3 ขอบเขตการทำงาน

1. ศึกษาหลักการวิธีการหา Motion Detection ด้วย Optical flow
2. ออกแบบแนวคิด และ อัลกอริทึมที่เหมาะสม
3. แยกแยะวัตถุได้อย่างน้อย 2 วัตถุ
    - วัตถุ
    - ฉากหลัง

1.4 ขั้นตอนการปฏิบัติงาน

โครงงานที่ 1

1. ศึกษาวิธีการเบื้องต้นสำหรับการหาการเคลื่อนไหว (Motion Detection)
2. ศึกษาหลักการของ Optical flow
3. ศึกษาและพัฒนาโปรแกรมภาษาซีร่วมกับ OpenCV เพื่อทดสอบอัลกอริทึมที่ได้ทำการศึกษา
4. ออกแบบเพิ่มเติมหรือปรับเปลี่ยน วิธีการหรืออัลกอริทึมให้เหมาะสมสำหรับการทำ matching ของภาพ ในงานทางด้าน Surveillance System
5. พัฒนาโปรแกรมภาษาซีสำหรับอัลกอริทึมที่ได้ออกแบบไว้แล้วนั้น และทำการทดสอบเปรียบเทียบ

โครงงานที่ 2

1. ปรับปรุง และพัฒนาโปรแกรมในโครงงานที่ 1 ให้ถูกต้องสมบูรณ์
2. ศึกษาวิธีการหรืออัลกอริทึมเบื้องต้นในการแยกการเคลื่อนไหวต่างๆ จาก Optical flow (Motion classification)
3. พัฒนาโปรแกรมภาษาซีสำหรับอัลกอริทึมของการแยกการเคลื่อนไหว ตามที่ได้ศึกษามาแล้ว
4. ทดสอบ พัฒนา และปรับปรุงโปรแกรมให้ใช้งานได้อย่างสมบูรณ์
5. นำโปรแกรมที่ได้ไปใช้งานร่วมกับระบบ Surveillance System (Streaming Real Time Video)

ตารางการปฏิบัติงานโครงงานที่ 1

เดือน	มิ.ย.			ก.ค.				ส.ค.				ก.ย.			
สัปดาห์ที่	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ขั้นตอนที่ 1															
ขั้นตอนที่ 2															
ขั้นตอนที่ 3															
ขั้นตอนที่ 4															
ขั้นตอนที่ 5															

ตาราง 1-1 แสดงแผนการปฏิบัติงานภาคเรียนที่ 1

ตารางการปฏิบัติงานโครงงานที่ 2

เดือน	พ.ย.			ธ.ค.				ม.ค.				ก.พ.			
สัปดาห์ที่	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ขั้นตอนที่ 1															
ขั้นตอนที่ 2															
ขั้นตอนที่ 3															
ขั้นตอนที่ 4															
ขั้นตอนที่ 5															

ตาราง 1-2 แสดงแผนการปฏิบัติงานภาคเรียนที่ 2

ช่วงการทำงานที่ได้วางแผนเอาไว้

สอบกลางภาค

1.5 เครื่องมือที่ใช้ในการพัฒนา

1.5.1 เครื่องมือด้านฮาร์ดแวร์ (Hardware)

- เครื่องคอมพิวเตอร์ส่วนบุคคล (PC)
- เว็บแคมความละเอียด 320 * 240 พิกเซล

1.5.2 เครื่องมือด้านซอฟแวร์ (Software)

- ระบบปฏิบัติการ Microsoft Windows XP
- โปรแกรม Microsoft Visual C++ 6.0
- ไลบรารีภาษาซี OpenCV

บทที่ 2 *ความรู้พื้นฐาน*

ในบทนี้จะกล่าวถึงความรู้พื้นฐานที่จะต้องนำไปใช้ เพื่อทำความเข้าใจเนื้อหาทางเทคนิคในบทอื่นต่อไปโดยจะกล่าวถึงความรู้พื้นฐานของเทคโนโลยีและเครื่องมือที่ใช้ในการพัฒนา

2.1 Surveillance System

ระบบตรวจสอบผู้บุกรุกหรือระบบเฝ้าดู (Surveillance System) ที่ศึกษาในโครงงานประกอบด้วย IP camera หรือ network camera, Network Video Decoder, เครื่องคอมพิวเตอร์ที่ทำหน้าที่ server และเครื่องคอมพิวเตอร์สำหรับการแพร่ภาพ โดยทุกส่วนจะติดต่อกันผ่านระบบ network หรือ ระบบ internet

รูปภาพ 2-1 ตัวอย่างลักษณะของระบบตรวจสอบผู้บุกรุกหรือระบบเฝ้าดู (Surveillance System)

2.2 *Motion Detection*

ระบบตรวจสอบผู้บุกรุกหรือระบบเฝ้าดู (Surveillance System) ที่ได้ศึกษา คือ การหาวัตถุเคลื่อนไหลจากภาพวีดีโอ (Motion Detection) ซึ่งการหาวัตถุเคลื่อนไหลจากภาพวีดีโอสามารถหาได้ด้วยวิธี Background Subtraction และ คำนวณหา Optical Flow ของภาพ โดยในส่วนที่ศึกษาและทำในโครงงานนี้คือ คำนวณหา Optical Flow ของภาพ

2.2.1 *Background Subtraction*

Background Subtraction คือ การแยกวัตถุที่มีการเคลื่อนที่ออกจากฉากหลัง โดยการลบฉากหลัง ด้วยการนำภาพสองเฟรม ณ บริเวรเดียวในเวลาที่ต่างกัน มาทำการลบกัน หากมีการเคลื่อนที่เข้ามาของวัตถุในภาพเฟรมที่สอง เมื่อนำภาพทั้งสองเฟรมมาลบกัน ฉากหลังจะลบหักล้างกันระหว่างภาพสองเฟรม และมีส่วนผลต่างระภาพทั้งสองเฟรม ซึ่งผลต่างนี่เองคือการเคลื่อนไหลของวัตถุที่เข้ามายังภาพเฟรมที่สอง เป็นไปดังสมการต่อไปนี้

$$O = I - B$$

เมื่อ O คือ Object

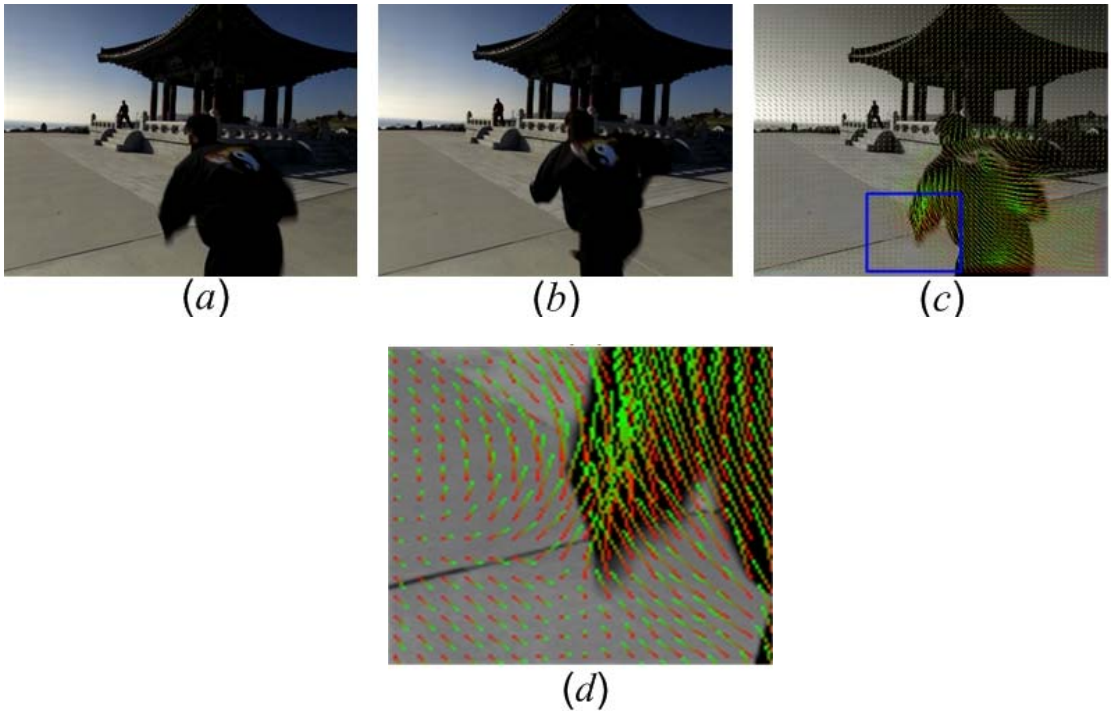
I คือ ภาพของเฟรมหนึ่งๆ

B คือ ภาพพื้นหลัง

ข้อจำกัดของวิธีนี้สามารถใช้ได้กับกล้องแบบติดตั้งอยู่กับที่เท่านั้น เนื่องจากต้องใช้ภาพฉากหลังที่เหมือนกันในการแยกหาวัตถุเคลื่อนที่ในภาพ

2.2.2 *Optical Flow*

optical flow คือ โมเดลการคำนวณหาเว็กเตอร์ลัพธ์ของภาพเคลื่อนไหว เพื่อติดตามการ



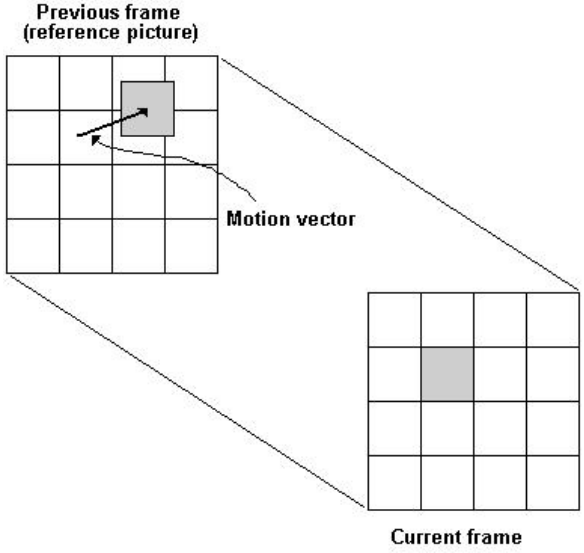
รูปภาพ 2-2 ตัวอย่าง Optical flow

จากรูปที่2-2 (a) คือเฟรมอินพุตแรก (b) คือเฟรมอินพุตที่สอง (c) คือการใช้หลัก estimated optical flow ในการคาดคะเนการไหลเวียนเกี่ยวกับสายตาในการเปรียบเทียบระหว่าง(a)กับ(b) (d) คือภาพขยายจากกรอบสี่เหลี่ยมในรูป(c) จะเห็นได้ว่า(d) เป็นการแสดงเส้นเวกเตอร์ที่แสดงการเปลี่ยนแปลงของการเคลื่อนที่และทิศทางของภาพในหน่วยเวลาที่เปลี่ยนไป โดยที่เปลี่ยนแปลงจากรูป (a) ไปยังรูป (b) จะได้เป็นดังสมการดังนี้

$$\frac{dx}{dt} = u \qquad \frac{dy}{dt} = v$$

2.2.3 Block matching

Block matching คือ เทคนิคมาตรฐานสำหรับการเข้ารหัสภาพเคลื่อนไหวอย่างเป็นลำดับ โดยมีเป้าหมายตรวจสอบการเคลื่อนไหวระหว่างสองบล็อกรูปภาพที่เราสนใจ โดยแบ่งเฟรมรูปภาพเป็นบล็อกจัตุรัสที่ไม่ซ้อนกัน บล็อกแต่ละอันจากเฟรมปัจจุบันจะถูกจับคู่เข้ากับบล็อกในเฟรมถัดไป โดยบล็อกในเฟรมปัจจุบันจะถูกจับคู่กับบล็อกที่มีค่าพิกเซลใกล้เคียงกันในเฟรมถัดไป ณ ที่มีการเปลี่ยนแปลงแต่ละพิกเซล จะมีการคำนวณผลรวมของระยะระหว่างค่าสีเทาของระหว่างสองบล็อก ถ้าผลการคำนวณของผลรวมระยะค่าสีเทามีค่าน้อยที่สุดก็จะคือจุดที่ดีที่สุดที่เราสนใจ



รูปภาพ 2-3 Block matching

การหาการMatchingของภาพสามารถหาได้โดยใช้วิธี mean absolute distance (MAD), mean squared distance (MSD) และ normalized cross-correlation (NCC) โดยมีสมการแต่ละวิธีดังนี้

Mean absolute distance (MAD)

$$\mathrm{MAD}(i,j) = \frac{1}{mn} \sum_{k} \sum_{l} |S_f(k,l) - S_{f-1}(k+i, l+j)|.$$

โดยที่ m * n เป็นขนาดพิกเซลของบล็อก และ $S_f(k,l)$ คือจุดที่เราสนใจในบล็อกของเฟรมถัดไป ที่จะนำมาพิจารณาหา Optical flow

Mean squared distance (MSD)

$$\mathrm{MSD}(i,j) = \frac{1}{mn} \sum_{k} \sum_{l} \left[S_f(k,l) - S_{f-1}(k+i, l+j) \right]^2.$$

โดยที่ m * n เป็นขนาดพิกเซลของบล็อก และ $S_f(k,l)$ คือจุดที่เราสนใจในบล็อกของเฟรมถัดไป ที่จะนำมาพิจารณาหา Optical flow

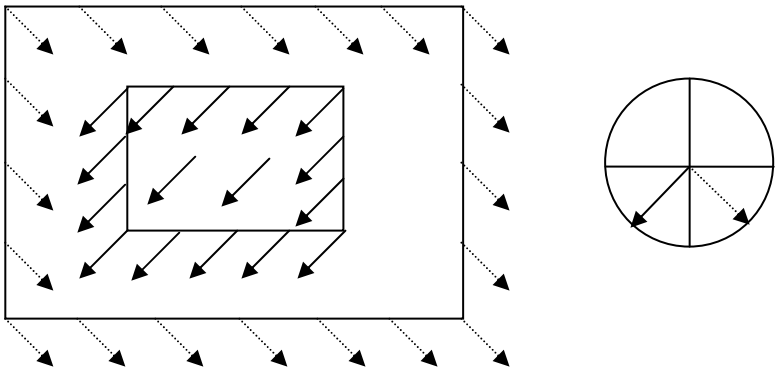
Normalized cross-correlation (NCC)

Cross-correlation ในทาง signal processing คือ ตัวชี้วัดของความคล้ายคลึงกันของสัญญาณ 2 สัญญาณ โดยทั่วไปถูกใช้เพื่อหาลักษณะเฉพาะของสัญญาณที่ไม่รู้จัก เปรียบเทียบกับสัญญาณที่รู้จัก cross-correlation เป็นฟังก์ชั่นที่แสดงความสัมพันธ์ระหว่างเวลากับสัญญาณที่ใช้ในการจำแนกรูปแบบ และ เทคนิคการถอดรหัส

$$\mathrm{NCC} = \frac{\sum [f(x,y) - f(x,y)]\,[g(x,y) - g(x,y)]}{\sqrt{\sum [f(x,y) - f(x,y)]2 \sum [g(x,y) - g(x,y)]2}}$$

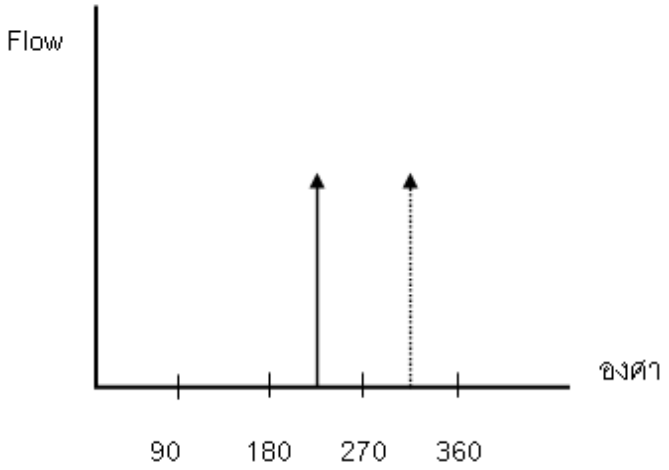
2.3 Motion classification

การแยกการเคลื่อนไหวของวัตถุ(Motion classification) จะใช้มุมของFlowเป็นตัวกำหนดวัตถุที่เคลื่อนที่ว่ามีกี่วัตถุที่เคลื่อนไหว โดยจัดกลุ่มFlowที่มีขนาดมุมใกล้เคียงกันหรือเท่ากัน ว่าเป็นการเคลื่อนไหวของวัตถุเดียวกัน



รูปภาพ 2-4 ตัวอย่างลักษณะของ Flow แสดงการเคลื่อนไหวของวัตถุ 2 วัตถุ

เมื่อสังเกตจากขนาดมุมของFlow จะแบ่งกลุ่ม Flow ได้เป็นสองกลุ่มคือกลุ่มของFlowที่เป็นเส้นประขนาด 315 องศา และ กลุ่มของ Flow ที่เป็นเส้นทึบขนาด 225 องศา ดังรูป 2-5



รูปภาพ 2-5 แผนภูมิการจัดกลุ่มของ Flow ตามขนาดองศา

2.3.1 หลักการของ Motion classification ในโครงงาน

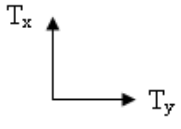
การแยกการเคลื่อนไหวของวัตถุ (Motion classification) โดยใช้ขนาดมุมของ Flow เป็นตัวแยกแยะวัตถุนั้นๆ โดยการหาค่าขนาดมุมของแต่ละ Flow ที่มีค่าเท่ากัน หรือใกล้เคียงกัน แล้วนำขนาดของมุมมาจัดกลุ่มแล้วเฉลี่ย โดยใช้หลักการ K-Mean โดยค่ามุมเฉลี่ยที่ได้แต่ละกลุ่มก็จะเป็นตัวบ่งบอกถึงการเคลื่อนไหวของวัตถุแต่ละวัตถุ ว่ามีวัตถุเคลื่อนไหวอยู่กี่วัตถุ

K – Mean algorithm

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

โดยที่ X_j คือจุดข้อมูลทั้งหมด และ μ_i คือ ค่าเฉลี่ยหรือจุด centroid ของ X_j

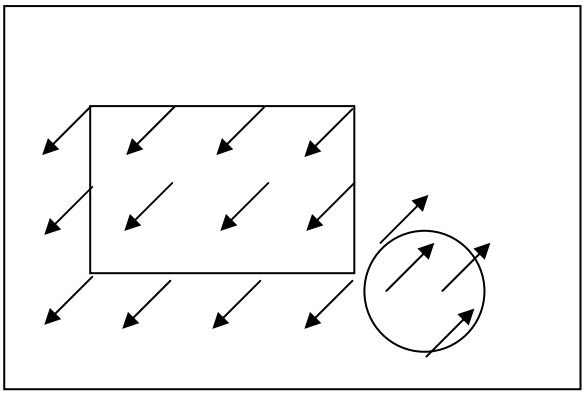
ในโครงงานนี้ใช้หลักในการแยกการเคลื่อนไหวของวัตถุ โดยการพิจารณาการเคลื่อนที่ในสองมิติ T_x และ T_y



โดยสามารถหามุมได้จาก

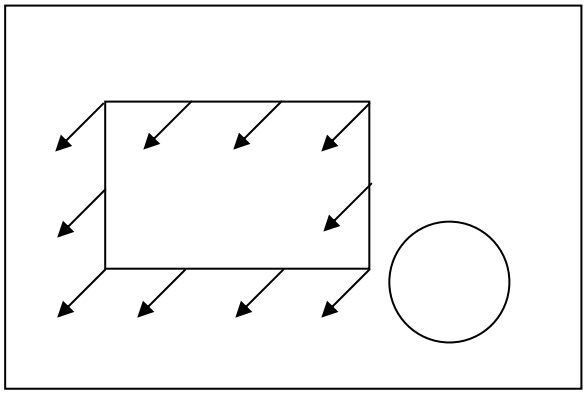
$$\theta = \arctan\left(\frac{T_y}{T_x}\right)$$

Output เริ่มต้น



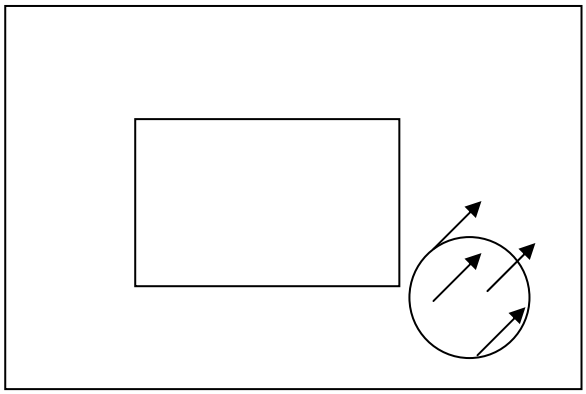
รูปภาพ 2-6 Output **เริ่มต้น**

Output หลังการแยกการเคลื่อนไหวที่ 1



รูปภาพ 2-7 Output **หลังการแยกการเคลื่อนไหวที่** 1

Output หลังการแยกการเคลื่อนไหวที่ 2



รูปภาพ 2-8 Output **หลังการแยกการเคลื่อนไหวที่** 2

2.3.2 ขั้นตอน และ วิธีการหา Motion classification

1. กำหนด class เริ่มต้นของวัตถุ

โดยให้มองวัตถุแต่ละวัตถุเป็น Class และกำหนดค่าเริ่มต้นค่ามุมเฉลี่ยของFlow แต่ละ class ให้เป็นศูนย์

Class

$\mu^{\theta}_{C_1} = 0$	$\mu^{\theta}_{C_2} = 0$	$\mu^{\theta}_{C_3} = 0$	$\mu^{\theta}_{C_n} = 0$

ตาราง 2-1 การให้ค่าเริ่มต้นแต่ละ Class เท่ากับศูนย์

2. Update center of Class

Update ค่า μ ของ class แต่ละ class โดยใช้ค่าเบี่ยงเบนเป็นค่าจำแนกของ Class สมการ Update center of Class

$$\mu^{\theta}_{C_{t+1}} = \alpha F_i + (1-\alpha)\mu^{\theta}_{C_t}$$

โดยที่ α มีค่าเป็นจำนวนจริง ตั้งแต่ 0 -1

F_i คือ ขนาดมุมของ Flow แต่ละ Flow

μ คือ ค่าเฉลี่ยขนาดมุมของ Flow

สมการหาค่าเบี่ยงเบน

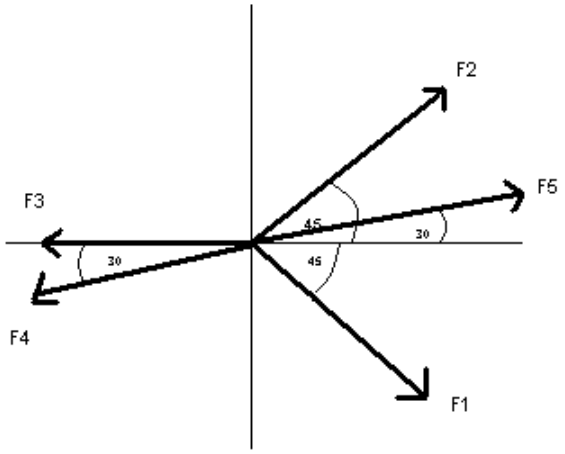
$$\text{ค่าเบี่ยงเบน} = \sqrt{\frac{\sum (X_i - \mu)^2}{N}}$$

โดยที่ X_i คือ ขนาดมุมของFlow แต่ละ Flow

μ คือ ค่าเฉลี่ยขนาดมุมของ Flow

N คือ จำนวน Flow ทั้งหมด

ตัวอย่างการอัพเดต μ ของ Class



รูปภาพ 2-9 ตัวอย่าง Flow

Update ครั้งที่ 1

นำ F1 มาใส่ใน Class1

พิจารณา F2 ว่ามีค่าเบี่ยงเบนจาก F1 เกินค่าเบี่ยงเบนหรือไม่ (แต่ในที่นี้สมมติเป็น 90 องศา) 315 – 45 = 270 ซึ่ง 270 มากกว่า 90 ดังนั้น F2 อยู่กันคนละ Class กับ F1

พิจารณา F3

- เปรียบเทียบกับ F1 315 – 180 = 135 ซึ่งมากกว่า 90 ดังนั้น F3 อยู่กันคนละ Class กับF1

- เปรียบเทียบกับ F2 45 – 180 = 135(ไม่คิดเครื่องหมาย) ซึ่งมากกว่า 90 ดังนั้น F3 อยู่กันคนละ Class กับF2

$\mu_{C_1}^{\theta} = 315$	$\mu_{C_2}^{\theta} = 45$	$\mu_{C_3}^{\theta} = 180$
F1(315)	F2(45)	F3(180)

ตาราง 2-2 Update center of Class ครั้งที่ 1

Update ครั้งที่ 2

พิจารณา F4

- เปรียบเทียบกับ $\mu_{C_1}^{\theta}$ 315 – 210 = 105 ซึ่งมากกว่า 90 ดังนั้น F3 อยู่กันคนละ Class กับF1

- เปรียบเทียบกับ $\mu_{C_2}^{\theta}$ 45 – 210 = 165 (ไม่คิดเครื่องหมาย) ซึ่งมากกว่า 90 ดังนั้น F3 อยู่กันคนละ Class กับF2

- เปรียบเทียบกับ $\mu_{C_3}^{\theta}$ 180 – 210 = 30 (ไม่คิดเครื่องหมาย) ซึ่งน้อยกว่า 90 ดังนั้น F3 อยู่ Class เดียวกับ F3

เมื่อทราบว่า F4 และ F3 อยู่ class เดียวกันต้องคำนวณหาค่า $\mu_{C_3}^{\theta}$ ใหม่จากสมการที่กล่าวมาในตอนต้น จะได้เท่ากับ 195

$\mu_{C_1}^{\theta} = 315$	$\mu_{C_2}^{\theta} = 45$	$\mu_{C_3}^{\theta} = 195$
F1(315)	F2(45)	F3(180)
		F4(210)

ตาราง 2-3 Update center of Class ครั้งที่ 2

Update ครั้งที่ 3

พิจารณา F5

- เปรียบเทียบกับ $\mu_{C_1}^{\theta}$ 315 – 30 = 185 ซึ่งมากกว่า 90 ดังนั้น F5 อยู่กันคนละ Class กับF1

- เปรียบเทียบกับ $\mu_{C_2}^{\theta}$ 45 – 30 = 15 ซึ่งน้อยกว่า 90 ดังนั้น F5 อยู่ Class เดียวกับ F2

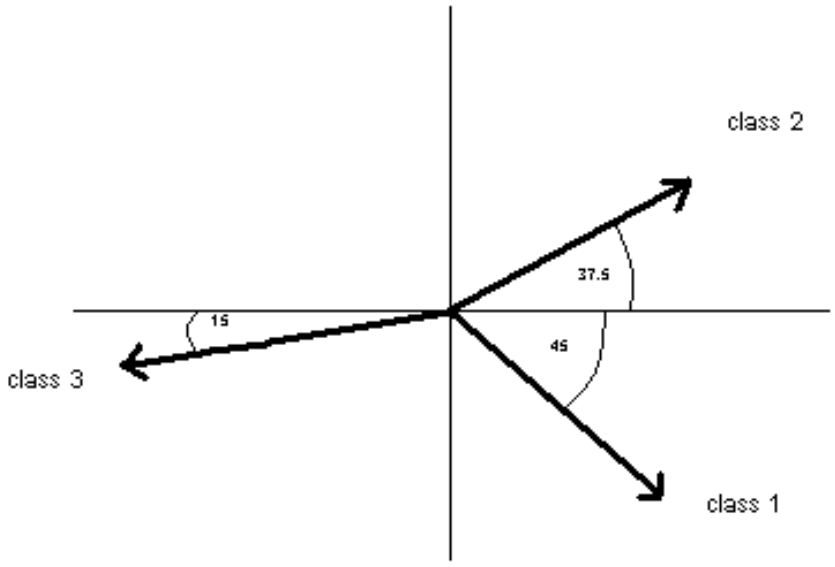
- เปรียบเทียบกับ $\mu_{C_3}^{\theta}$ 195 – 30 = 165 ซึ่งมากกว่า 90 ดังนั้น F5 อยู่กันคนละ Class กับ F3

เมื่อทราบว่า F5 และ F2 อยู่ class เดียวกันต้องคำนวณหาค่า $\mu^{\theta}_{C_2}$ ใหม่จากสมการที่กล่าวมาในตอนต้น จะได้เท่ากับ 37.5

$\mu^{\theta}_{C_1} = 315$	$\mu^{\theta}_{C_2} = 37.5$	$\mu^{\theta}_{C_3} = 195$
F1(315)	F2(45)	F3(180)
	F5(30)	F4(210)

ตาราง 2-4 Update center of Class ครั้งที่ 3

จากตาราง Update center of Class ที่เสร็จแล้ว สามารถบอกได้ว่ามีวัตถุอยู่ 3 วัตถุ คือ วัตถุที่ 1 ที่เคลื่อนที่ทำมุม 315 องศา, วัตถุที่ 2 ที่เคลื่อนที่ทำมุม 37.5 องศา และ วัตถุที่ 3 ที่เคลื่อนที่ทำมุม 195 องศา

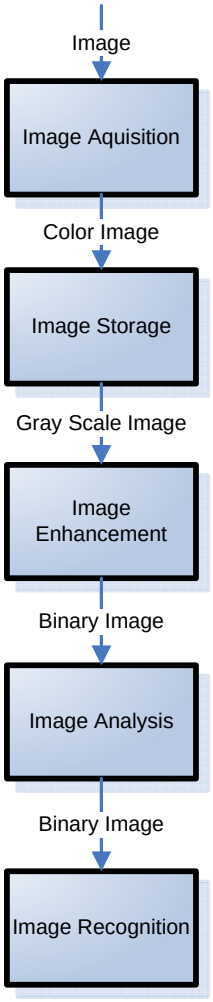


รูปภาพ 2-10 ผลลัพธ์ของการ Classification

2.4 หลักการของ Image processing

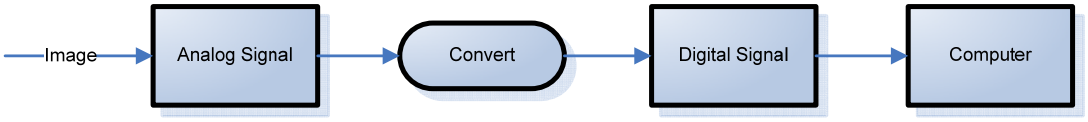
2.4.1 Image processing

Image processing คือ การแปลงข้อมูลภาพให้อยู่ในรูปแบบข้อมูลดิจิตอล ซึ่งสามารถที่จะนำเอาข้อมูลนี้มาวิเคราะห์และจัดการตามกระบวนการต่างๆด้วยคอมพิวเตอร์ได้ ซึ่งลำดับของ Digital Image Processing สามารถแสดงได้ดังรูปที่ 2-11



รูปภาพ 2-11 แสดงลำดับขั้นตอนของ Digital image processing

Image Acquisition เป็นกระบวนการให้ได้มาซึ่งข้อมูลภาพ โดยการแปลงสัญญาณอนาล็อกไปเป็นสัญญาณดิจิตอล ซึ่งสามารถประมวลผลได้ด้วยคอมพิวเตอร์



รูปภาพ 2-12 แสดง Image acquisition process

Image Storage	เป็นกระบวนการที่เก็บข้อมูลภาพ ซึ่งจะเก็บในรูปแบบสัญญาณดิจิตอลไว้ในหน่วยความจำ
Image Enhancement	เป็นกระบวนการที่ประกอบด้วยเทคนิคที่หลากหลายในการจะนำมาซึ่งภาพที่มีความชัดเจนและคมชัดยิ่งขึ้นหรือเป็นการแปลง ภาพให้มีความเหมาะสมสำหรับการวิเคราะห์ภาพโดยมนุษย์หรือคอมพิวเตอร์ต่อไป
Image Analysis	เกี่ยวกับวิธีการอธิบายและการจดจำข้อมูลภาพดิจิตอล ซึ่งอินพุตของระบบข้อมูลภาพดิจิตอลและเอาท์พุตจะเป็นเครื่องหมายที่ใช้แทนข้อมูลภาพดิจิตอลเหล่านั้น ในการวิเคราะห์ภาพมีอยู่หลายวิธีด้วยกันที่ได้นำมาจากการทำงานของภาพนั้น
Image Recognition	เป็นกระบวนการจดจำภาพ

การเก็บข้อมูลภาพลงหน่วยความจำของคอมพิวเตอร์สามารถทำได้โดยการจองหน่วยความจำของเครื่องไว้ในรูปของตัวแปรอะเรย์ (array) โดยค่าในแต่ละช่องของอะเรย์แสดงถึงคุณสมบัติของจุดภาพ (pixel) และตำแหน่งของช่องอะเรย์เป็นตัวกำหนดตำแหน่งของจุดภาพ

2.4.2 รูปภาพแบบสี (Color Images)

การนำภาพแบบมีสีเข้าสู่เครื่องคอมพิวเตอร์จะเปรียบเสมือนการนำภาพเข้าสู่คอมพิวเตอร์ 3 รูปในเวลาเดียวกัน ซึ่งรายละเอียดขึ้นอยู่กับระบบชนิดของสี ในระบบ RGB จะใช้ความเข้มเป็นสีแดง สีเขียว และ สีน้ำเงิน

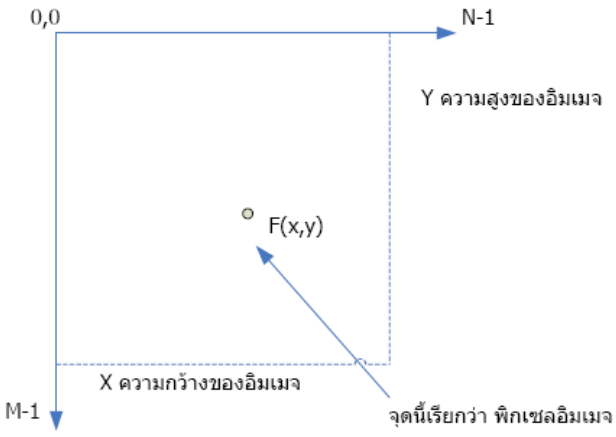
2.4.3 การจับภาพ (Capture)

ผลลัพธ์ที่ได้จากกล้องวิดีโอส่วนมากจะเป็นสัญญาณในระบบ RGB ซึ่งถ้านำอุปกรณ์ดังกล่าวมาต่อกับคอมพิวเตอร์ จำเป็นต้องใช้อุปกรณ์เพิ่มเติมในการแปลงภาพดังกล่าวเข้าสู่คอมพิวเตอร์ โดยที่อุปกรณ์นั้นจะทำหน้าที่รับภาพเป็นเฟรมๆ โดยอาจจะใช้หน่วยความจำของคอมพิวเตอร์ เพื่อที่จะแปลงสัญญาณอนาลอกเป็นดิจิตอล หรืออาจมีหน่วยความจำเป็นของตนเอง ซึ่งหน่วยความจำดังกล่าวเรียกว่า Frame Buffer

2.5 เทคนิคการใช้ Image processing

2.5.1 อิมเมจดิจิตอล

อิมเมจดิจิตอลเป็นผลมาจากการสุ่มค่าในระบบพิกัด Space หรือ (Spatial Coordinate) ดังรูปที่ 1 และการทำ Quantization ของค่าระดับความสว่าง (Brightness Value) หรือความเข้ม (Intensity) ระบบพิกัด Space นี้จะใช้กับการแสดงอิมเมจดิจิตอล ซึ่งจะมีขนาดความกว้างและความสูงของอิมเมจแสดงในแกน Y และ X ตามลำดับ ส่วนจุดใดๆ ที่วางบนระนาบ XY จะเป็นฟังก์ชัน f(x,y) และเรียกว่า พิกเซล (Pixel) ที่แสดงถึงค่าระดับความเข้ม ซึ่งจะเป็นจำนวนที่นับได้จำกัด (Finite Number) แบบไม่ต่อเนื่อง หรือเรียกว่า Discrete Quantity ค่า Discrete Quantity เป็นผลมาจากการทำ Quantization โดยจะใช้การแปลงจากอนาล็อก (Analog) เป็นดิจิตอล (Digital)



รูปภาพ 2-13 ระบบพิกัด Space

2.5.2 วิธีการอ่านข้อมูลพิกเซลของอิมเมจ

จากรูปที่ 2-13 จุดที่วางอยู่ในพิกัด Space นี้ก็คือ พิกเซล (Pixel) หรือ Picture Element ซึ่งก็คือความสว่างหรือค่า Luminance (L) ของอิมเมจ ถ้าอิมเมจนั้นเป็นภาพขาวดำ ขนาด 8 บิต จะมีค่า L เท่ากับ 2^n หรือเท่ากับ 256 ระดับ คือตั้งแต่ 0 (พิกเซลสีดำ) จนถึง 255 (พิกเซลสีขาว) (0<=L<=255) บางครั้งค่าความสว่าง (L) อาจมีความหมายถึงระดับความละเอียดของอิมเมจ (Image Resolution)

ถ้าพิกเซลเป็นภาพขาวดำ จะอ่านค่าอิมเมจดิจิตอลในรูปเมตริก 2 มิติ ขนาด M*N ได้ดังนี้

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0.1) & . & f(0,N-1) \\ f(1,0) & f(1,1) & . & f(1,N-1) \\ . & . & . & . \\ f(M-1,0) & f(M-1,1) & . & f(M-1,N-1) \end{bmatrix}_{M*N}$$

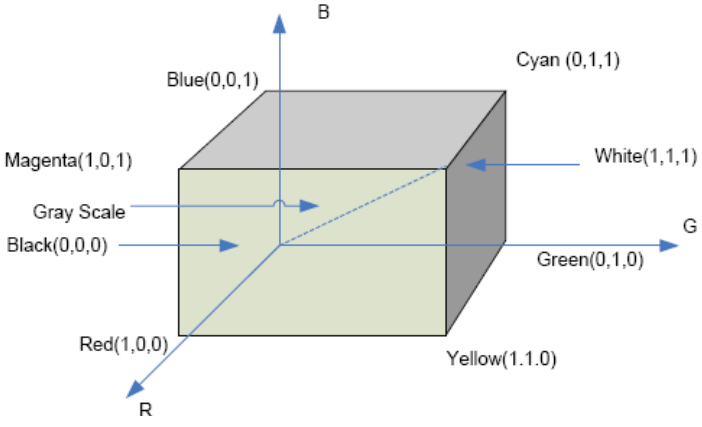
โดยที่ ค่า f(x,y) จะอยู่ในช่วง 0 ถึง 255 (0<=f(x,y)<=L)

สมมุติว่าอ่านค่าพิกเซลจากอิมเมจหนึ่งได้ f(x,y) เท่ากับ 10 แสดงว่า จุดพิกเซลนั้นมีความสว่างน้อยมากหรือค่อนข้างจะดำ ถ้าค่าที่อ่านได้เป็น 250 แสดงว่า จุดพิกเซลนั้นสว่างมาก

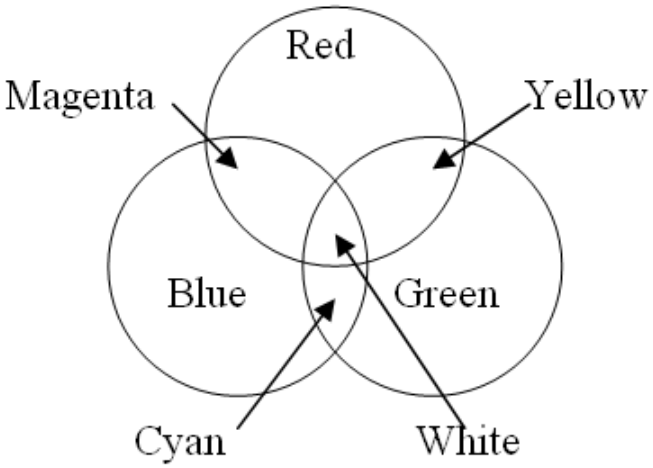
การรับภาพของการจัดเก็บนั้นจะทำการจัดเก็บเป็นโมโนมี 256 ระดับ คือ ได้จากการที่แม่สีแต่ละสีมีขนาด 8 บิต และจะมีการรวมการใช้หลักการของโมเดลสีด้วย

2.5.3 โมเดลสี *(Color Model)*

โมเดลสีหรือ Color Space ประกอบไปด้วย 3 แม่สีหลัก ได้แก่ สีแดง เขียว และน้ำเงิน ถ้านำแต่ละแม่สีมาพล็อตกราฟในระบบพิกัด Color Space โดยแต่ละสีมีค่า 0 ถึง 1 (0 แสดงถึงความมืด และ 1 แสดงถึงความสว่างจะได้จากการผสมสีทางแสงหรือการบวกแม่สีเข้าด้วยกัน (Additive Primary Color) ดังรูปที่ 2-14



รูปภาพ 2-14 โมเดลสีในระบบพิกัด Color Space



รูปภาพ 2-15 การผสมสีทางแสง (Additive Primary Color)

ถ้าแต่ละแม่สีเป็นขนาด 8 บิต รวมทั้งหมด เท่ากับ 24 บิต ซึ่งสามารถสร้างสีใหม่ได้ถึง 256*256*256 เท่ากับ 16, 777,216 สี ในที่นี้จะใช้พิกเซลเป็นภาพสีเท่ากับ 8 บิต หรือเรียกว่า มีความลึกเท่ากับ 24 บิต เป็นหลัก

ถ้าพิกเซลเป็นภาพสีขนาด 24 บิต จะอ่านค่าอิมเมจดิจิตอลในรูปแบบเมตริก 2 มิติขนาด M*N เหมือนกับในสมการ (1.) แต่ค่า f(x,y) จะอยู่ในช่วงที่ประกอบด้วย

R ระดับ 0 จนถึง 255 (0<=R<=255)

G ระดับ 0 จนถึง 255 (0<=G<=255)

และ B ระดับ 0 จนถึง 255 (0<=B<=255)

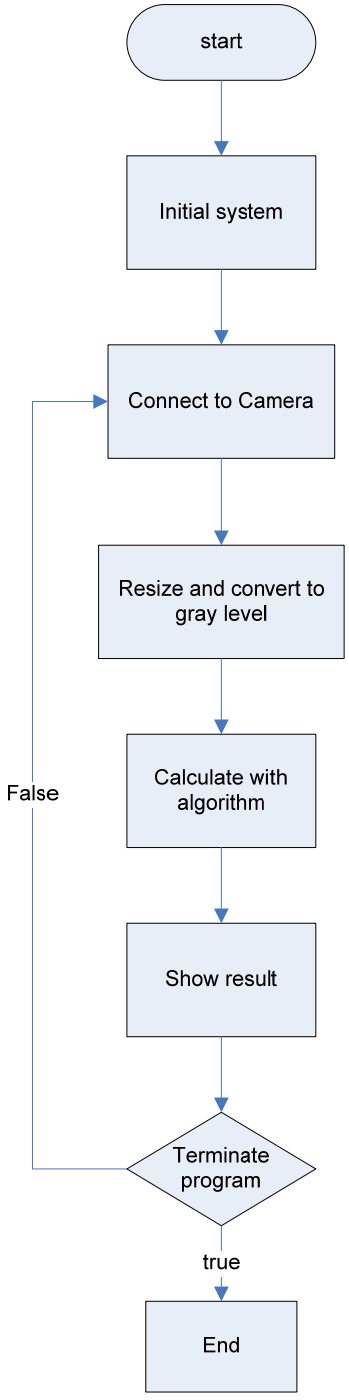
บทที่ 3 *รายละเอียดการทำงาน*

ในส่วนนี้แสดงรายละเอียดและการทำงานของระบบที่ได้ออกแบบไว้ ซึ่งเกี่ยวกับการพัฒนาโปรแกรมภาษาซีร่วมกับไลบรารีการประมวลผลภาพ OpenCV โดยใช้ Visual Studio 6 เป็นเครื่องมือพัฒนาโปรแกรม โดยโปรแกรมที่พัฒนาขึ้นมีการรับภาพจาก Webcam แล้วนำภาพที่ได้มาประมวลผลตามอัลกอริทึมที่ได้ศึกษา

3.1 หลักการทำงานของโปรแกรม

หลักการทำงานของโปรแกรม เมื่อโปรแกรมเริ่มทำงานจะมีการกำหนดค่าเริ่มต้นให้กับตัวแปรที่ได้ประกาศไว้ หลังจากที่ผ่านกระบวนการดังกล่าวแล้ว โปรแกรมจะทำการติดต่อกับ Webcam เพื่อ capture ภาพ จากนั้นลดขนาดของภาพ และแปลงเป็นภาพแบบ 1 แชนแนล เพื่อการคำนวณที่รวดเร็วขึ้น โปรแกรมจะนำภาพที่ได้จากการลดขนาด และแปลงเป็นภาพแบบ 1 แชนแนลแล้ว มาทำการประมวลผลคำนวณหาผลลัพธ์ตามอัลกอริทึมที่ต้องการ ซึ่งโครงงานนี้ใช้การคำนวณค่า Optical Flow แล้วจึงแสดงผลลัพธ์ที่ได้จากการคำนวณมาเป็นเวกเตอร์การเคลื่อนที่ของวัตถุ เมื่อได้เวกเตอร์การเคลื่อนที่ของวัตถุแล้ว จะนำเวกเตอร์การเคลื่อนที่ของวัตถุมาแยกการเคลื่อนไหวในทิศทางต่างๆ กัน หลังจากนั้นโปรแกรมจะทำการตรวจสอบเงื่อนไขว่าต้องการจะหยุดการประมวลผลโปรแกรมต่อไปหรือไม่ ถ้าต้องการ โปรแกรมก็จะหยุดการทำงานของโปรแกรม แต่ถ้าไม่ต้องการ โปรแกรมก็จะทำงานต่อไป

3.1.1 *Flow Chart ลำดับขั้นตอนการทำงานของโปรแกรมโปรแกรม*



รูปภาพ 3-1 Flow chart **แสดงขั้นตอนการทำงานของโปรแกรม**

3.1.2 โปรแกรม

1. Initial System

1.1 ทำการสร้าง Form ขึ้นมา

เพื่อแสดงภาพ Input และ Output โดยมีปุ่มstart ใช้ในการเริ่มการทำงาน และปุ่ม cancel ใช้ในการหยุดการทำงาน

ทำในส่วน Method onInitDialog ของ CDialog

CDialog::OnInitDialog();

1.2 กำหนดตัวแปรของระบบ และกำหนดค่าตัวแปร

สร้างตัวแปร ImageSource เพื่อเก็บภาพ ที่รับได้จากกล้อง

สร้างตัวแปร ImageGray เพื่อเก็บภาพ ที่แปลงเป็นภาพหนึ่งแชนแนล

สร้างตัวแปร ImageOutput เพื่อเก็บภาพ ที่ใช้แสดงเป็นภาพ 0utput

สร้างตัวแปร ImgOpticalFlow เพื่อเก็บภาพ ที่ใช้แสดงภาพผลลัพธ์ของOpticalFlow

สร้างตัวแปร img1 ถึง img8 เพื่อเก็บภาพ ที่ใช้แสดงภาพผลลัพธ์หลังจากการแยกการเคลื่อนไหวทั้ง 8 ทิศทาง

พร้อมทั้งกำหนดขนาดความกว้าง 320 ความสูง 240 พิกเซล และกำหนดขนาดจำนวน 8 บิต และ ขนาด 3 แชนแนล

```
CvSize imgSize = cvSize(320,240);
IplImage *ImageSource = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage *ImageGray   = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage *ImageOutput = cvCreateImage( imgSize1 , IPL_DEPTH_8U, 3);

IplImage* img1 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img2 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img3 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img4 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
```

```
IplImage* img5 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img6 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img7 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img8 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage *ImgOpticalFlow = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
```

สร้างตัวแปร frame1, frame1_1C, *frame2_1C, eig_image, temp_image, pyramid1, pyramid2 เพื่อจะนำไปใช้ในการคำนวณหา Optical Flow

```
static IplImage *frame1  = NULL, *frame1_1C = NULL, *frame2_1C = NULL,
*eig_image  = NULL, *temp_image = NULL, *pyramid1 = NULL, *pyramid2 =
NULL;
```

สร้างตัวแปร frameLast และ frame พร้อมทั้งกำหนดขนาดเท่ากับ imgSize (ความกว้าง 320 ความสูง 240 พิกเซล)และกำหนดขนาดจำนวน 8 บิต และ ขนาด 3 แชนแนล

```
IplImage *frameLast     = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage *frame         = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
```

2. Connect to Camera

โปรแกรมทำการติดต่อรับภาพจากกล้อง ผ่านทางฟังก์ชัน Callback ของ OpenCV โดยทำการกำหนดค่าเริ่มต้นของฟังก์ชันดังนี้

```
cvcamSetProperty(out[0], CVCAM_PROP_ENABLE, CVCAMTRUE);
cvcamSetProperty(out[0], CVCAM_PROP_RENDER, CVCAMTRUE);
cvcamSetProperty(out[0],  CVCAM_PROP_CALLBACK, callback);
      cvcamInit();
      cvcamStart();
```

3. Resize

ใน Application ที่ต้องการการประมวลผลแบบ real time การใช้ภาพที่มีขนาดใหญ่ จะใช้เวลาในการประมวลผลมาก ดังนั้นจึงควรลดขนาดภาพลงเพื่อลดเวลาในการประมวลผลให้เหมาะสมกับการทำงานแบบ real time

```
cvResize(image,ImageOutput, CV_INTER_LINEAR );
```

4. Calculating with Algorithm

4.1 กำหนดตัวแปร และกำหนดค่าตัวแปร

สร้างตัวแปรเก็บภาพที่จะนำไปใช้ในการคำนวณหา Optical Flow พร้อมกำหนดค่าขนาดภาพ

สร้างตัวแปร number_of_features เพื่อใช้ในการเก็บจุดบนภาพจำนวน 400 จุด

```
CvPoint2D32f frame1_features[400];
      int number_of_features;
      number_of_features = 400;
```

หาจุดลักษณะเฉพาะที่สำคัญบนภาพ frame1_1C จำนวน 400 จุด ด้วยฟังก์ชัน cvGoodFeaturesToTrack

```
cvGoodFeaturesToTrack(frame1_1C, eig_image, temp_image,
frame1_features, &number_of_features, .01, .01, NULL);
```

สร้างอะเรย์ frame2_features, optical_flow_found_feature ,optical_flow_feature_error และสร้างอะเรย์ 2 มิติ frame2_features ให้มีขนาด 400 elements ที่จะนำไปใช้ในการคำนวณหา Optical Flow ต่อไป

พร้อมกำหนดค่าขนาดให้กับตัวแปร optical_flow_window ให้มีขนาด 3*3

```
      CvPoint2D32f frame2_features[400];
      char optical_flow_found_feature[400];
      float optical_flow_feature_error[400];
      CvSize optical_flow_window = cvSize(3,3);
      CvTermCriteria optical_flow_termination_criteria
      = cvTermCriteria( CV_TERMCRIT_ITERCV_TERMCRIT_EPS, 20, .3 );
```

4.2 คำนวณหา Optical Flow

โดยเลือกใช้ฟังก์ชั่น cvCalcOpticalFlowPyrLK ซึ่งเป็นที่ยอมรับกันอย่างแพร่หลาย ซึ่งการทำงานของฟังก์ชั่น นำภาพเฟรมแรก ณ เวลา t มาเปรียบเทียบกับภาพเฟรมที่สอง เวลา t+dt โดยใช้หลักฟังก์ชัน CalcOpticalFlowPyrLK นี้คำนวณหา optical flow ระหว่างกลุ่มของจุดบนภาพสองภาพ ฟังก์ชันจะค้นหา flow ด้วย sub pixel อย่างแม่นยำ โดยใช้พารามิเตอร์ pyrA และ pyrB ซึ่งพารามิเตอร์สองตัวนี้ คือค่าพีระมิด ซึ่งจะคำนวณหาได้ตามทฤษฎีของ Lucas Kanade Feature Tracke [10]

```
cvCalcOpticalFlowPyrLK(frame1_1C, frame2_1C,
pyramid1,pyramid2,frame1_features, frame2_features,
number_of_features, optical_flow_window, 5,
optical_flow_found_feature, optical_flow_feature_error,
optical_flow_termination_criteria, 0 );
```

พารามิเตอร์ในฟังก์ชั่น

frame1_1C คือ ภาพเฟรมแรก ณ เวลา t

frame2_1C คือ ภาพเฟรมที่สอง ณ เวลา t + dt

pyramid1 คือ ตัวแปรที่ใช้บัพเฟอร์ค่า pyramid[10] ของเฟรมแรก

pyramid2 คือ ตัวแปรที่ใช้บัพเฟอร์ค่า pyramid[10] ของเฟรมที่สอง

frame1_features คือ อะเรย์ของจุดที่พบFlow

frame2_features คือ อะเรย์ 2 มิติที่เก็บตำแหน่งของ input feature ในรูปที่สอง

number_of_features คือ featuresในเฟรมที่สอง

optical_flow_window คือ ขนาดหน้าต่างในการหาแต่ละ pyramid level

5 คือ ขนาดสูงสุดของ pyramid level

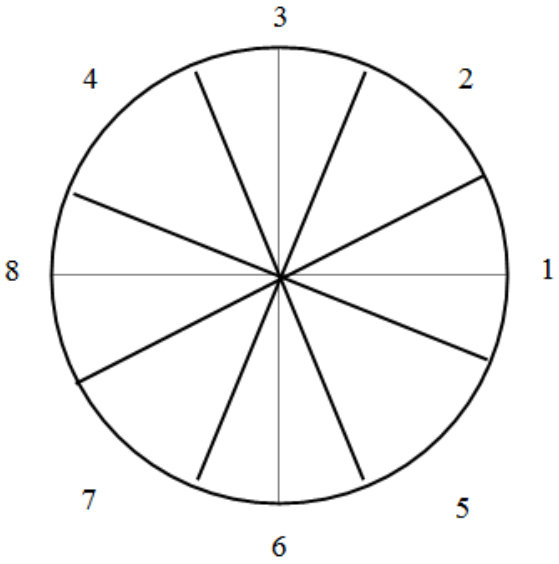
optical_flow_found_feature คือ อะเรย์ที่มีการ set ค่าทุก element เป็น 1 ถ้ามีการพบ flow ที่มีความสอดคล้องกับ feature และ set ค่าทุก element เป็น 0 ถ้าไม่มีการพบ flow

optical_flow_feature_error คือ อะเรย์ที่เก็บความแตกต่างของรอยต่อ ระหว่าง original points กับ moved points

optical_flow_termination_criteria คือ ค่าที่ใช้บอกในการหยุดวนหา flow

5. Classification

ทำการแยกแยะวัตถุเคลื่อนไหวของวัตถุ โดยแบ่งการแยกแยะการเคลื่อนไหวต่างๆ ออกเป็น 8 ทิศทาง คือ ทิศเหนือ ทิศใต้ ทิศตะวันออก ทิศตะวันตก ทิศตะวันออกเฉียงเหนือ ทิศตะวันออกเฉียงใต้ ทิศตะวันตกเฉียงเหนือ และทิศตะวันออกเฉียงใต้



รูปภาพ 3-2 ทิศอ้างอิงในการแยกการเคลื่อนไหว 8 ทิศ

5.1 วนลูปเช็คค่ามุมของ Flow เพื่อแยกการเคลื่อนไหวใน 8 ทิศทาง

```
      int count[8] = {0,0,0,0,0,0,0,0};

Arrow angle1[400],angle2[400],angle3[400],angle4[400],
angle5[400],angle6[400],angle7[400],angle8[400];

for(i = 0; i < number_of_features ; i++)

            {
```

5.2 แยกการเคลื่อนไหวในทิศที่ 1 ในช่วงมุม -22.5 องศา ถึง 22.5 องศา

```
if(angleArr[i].angle >= (-1)*(pi/8) && angleArr[i].angle <= pi/8)
                  {
                        angle1[count[0]] = angleArr[i];
                        count[0]++;
                  }
```

5.3 แยกการเคลื่อนไหวในทิศที่ 2 ในช่วงมุม 22.5 องศา ถึง 67.5 องศา

```
                    else if(angleArr[i].angle >= pi/8 &&
angleArr[i].angle <= (0.375)*pi)
                    {
                          angle2[count[1]] = angleArr[i];
                          count[1]++;
                    }
```

5.4 แยกการเคลื่อนไหวในทิศที่ 3 ในช่วงมุม 67.5 องศา ถึง 112.5 องศา

```
                    else if(angleArr[i].angle >= (0.375)*pi &&
angleArr[i].angle <= (0.625)*pi)
                    {
                          angle3[count[2]] = angleArr[i];
                          count[2]++;
                    }
```

5.5 แยกการเคลื่อนไหวในทิศที่ 4 ในช่วงมุม 112.5 องศา ถึง 157.5 องศา

```
                    else if(angleArr[i].angle >= (0.625)*pi &&
angleArr[i].angle <= (0.875)*pi)
                    {
                          angle4[count[3]] = angleArr[i];
                          count[3]++;
                    }
```

5.6 แยกการเคลื่อนไหวในทิศที่ 5 ช่วงมุม 157.5 องศา ถึง 202.5 องศา

```
                    else if(angleArr[i].angle >= (-1)*(0.375)*pi &&
angleArr[i].angle <= (-1)*(pi/8))
                    {
                          angle5[count[4]] = angleArr[i];
                          count[4]++;
                    }
```

5.7 แยกการเคลื่อนไหวในทิศที่ 6 ช่วงมุม -22.5 องศา ถึง -67.5 องศา

```
                    else if(angleArr[i].angle >= (-1)*(0.625)*pi &&
angleArr[i].angle <= (-1)*(0.375)*pi)
                    {
                          angle6[count[5]] = angleArr[i];
                          count[5]++;
                    }
```

5.8 แยกการเคลื่อนไหวในทิศที่ 7 ช่วงมุม -67.5 องศา ถึง -112.5 องศา

```
                    else if(angleArr[i].angle >= (-1)*(0.875)*pi &&
angleArr[i].angle <= (-1)*(0.625)*pi)
                    {
                          angle7[count[6]] = angleArr[i];
                          count[6]++;
                    }
```

5.9 แยกการเคลื่อนไหวในทิศที่ 8 ช่วงมุม -112.5 องศา ถึง -157.5 องศา

```
                    else if(angleArr[i].angle >= (0.875)*pi &&
angleArr[i].angle <= 1.125*pi)
                    {
                          angle8[count[7]] = angleArr[i];
                          count[7]++;
                    }
```

6. Show Result

หลังจากที่คำนวณหาค่าOptical Flow ได้แล้ว ก็จะวาดเส้นแสดงทิศทางการเคลื่อนที่ของวัตถุ ตามมุมและองศาที่คำนวณได้ โดยอาศัยหลักสมการทางคณิตศาสตร์

6.1 วนลูปเพื่อเช็คว่ามี flow หรือไม่ ถ้ามีให้นำไปวาดเวกเตอร์สีแดง ที่ขนาดเส้นหนา 1 หน่วย

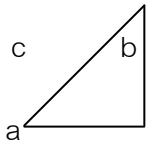
```
for(int i = 0; i < number_of_features; i++)
{
```

```
      if ( optical_flow_found_feature[i] == 0 ) continue;
int line_thickness;
line_thickness = 1;
CvScalar line_color;
line_color = CV_RGB(255,0,0);
```

6.2 คำนวณหามุม และขนาดความยาวเส้น flow ตามสมการทางคณิตศาตร์

$$L = \sqrt{(Py-Qy)^2 + (Px-Qy)^2}$$

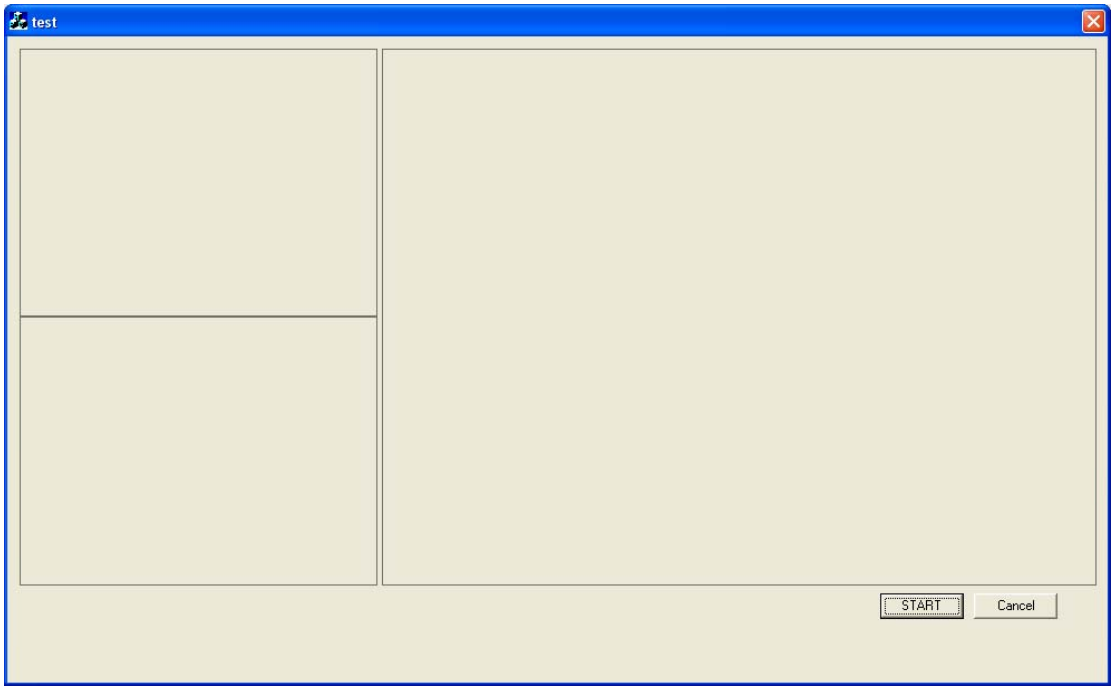
$$\theta = \arctan b/a$$




```
CvPoint p,q;
p.x = (int) frame1_features[i].x;
p.y = (int) frame1_features[i].y;
q.x = (int) frame2_features[i].x;
q.y = (int) frame2_features[i].y;
double angle;
angle = atan2( (double) p.y - q.y, (double) p.x - q.x );
double hypotenuse;
hypotenuse = sqrt( square(p.y - q.y) + square(p.x - q.x) );
q.x = (int) (p.x - 3 * hypotenuse * cos(angle));
      q.y = (int) (p.y - 3 * hypotenuse * sin(angle));
cvLine( frame1, p, q, line_color, line_thickness,CV_AA, 0 );
      p.x = (int) (q.x + 9 * cos(angle + pi / 4));
      p.y = (int) (q.y + 9 * sin(angle + pi / 4));
cvLine( frame1, p, q, line_color, line_thickness, CV_AA, 0 );
      p.x = (int) (q.x + 9 * cos(angle - pi / 4));
      p.y = (int) (q.y + 9 * sin(angle - pi / 4));
cvLine( frame1, p, q, line_color, line_thickness, CV_AA, 0 );
}
cvShowImage("Optical Flow", frame1);
```

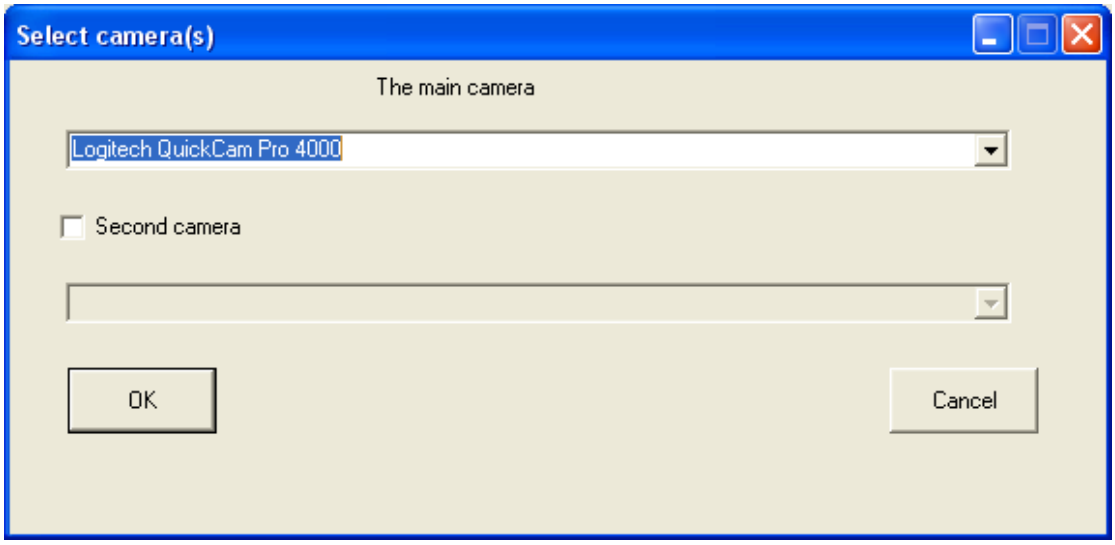
3.2 *Dialog การใช้งานของโปรแกรม*

เมื่อ run โปรแกรม จะมีการสร้างหน้าต่างdialog เพื่อรับค่าจากผู้ใช้งานในการเริ่มต้นรับภาพจากกล้อง โดยการใช้ปุ่ม start และ ใช้ปุ่ม cancel หยุดการทำงานของโปรแกรม



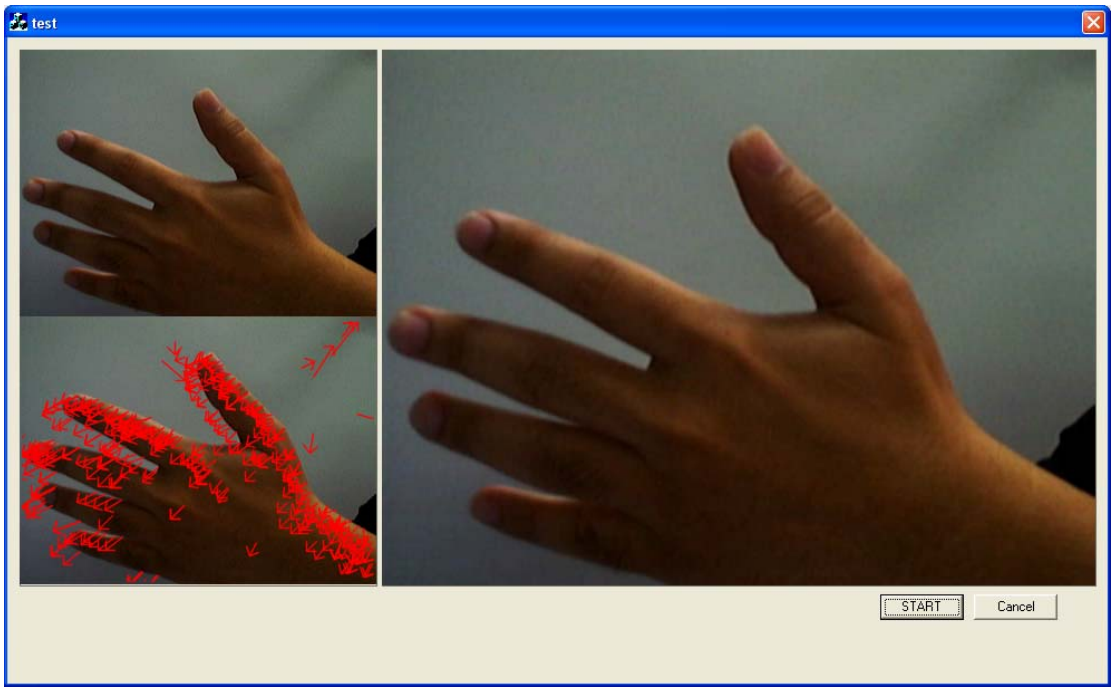
รูปภาพ 3-3 dialog เริ่มต้นการใช้งาน

หลังจาก Start โปรแกรม จะมี dialog ในการเลือกกล้องที่จะใช้งานร่วมกับโปรแกรม



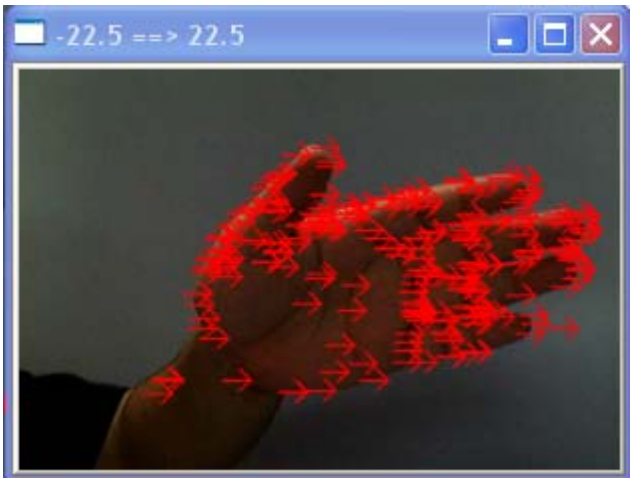
รูปภาพ 3-4 dialog เลือกกล้อง

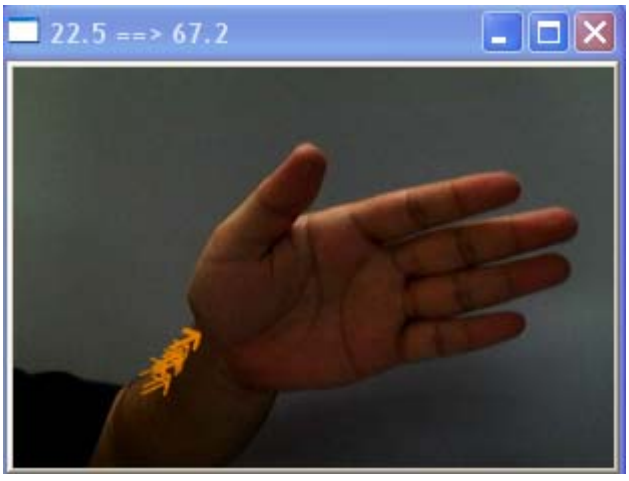
เมื่อเริ่มต้นรับภาพจากกล้องจะมีการแสดงภาพ ที่รับมาแสดงในขนาดความละเอียดของกล้อง 320*240 และแสดงในขนาดขยาย 640*480 พร้อมกับแสดงหน้าต่างการหา Optical Flow ในขนาดภาพ320*240 ดังรูปที่แสดง

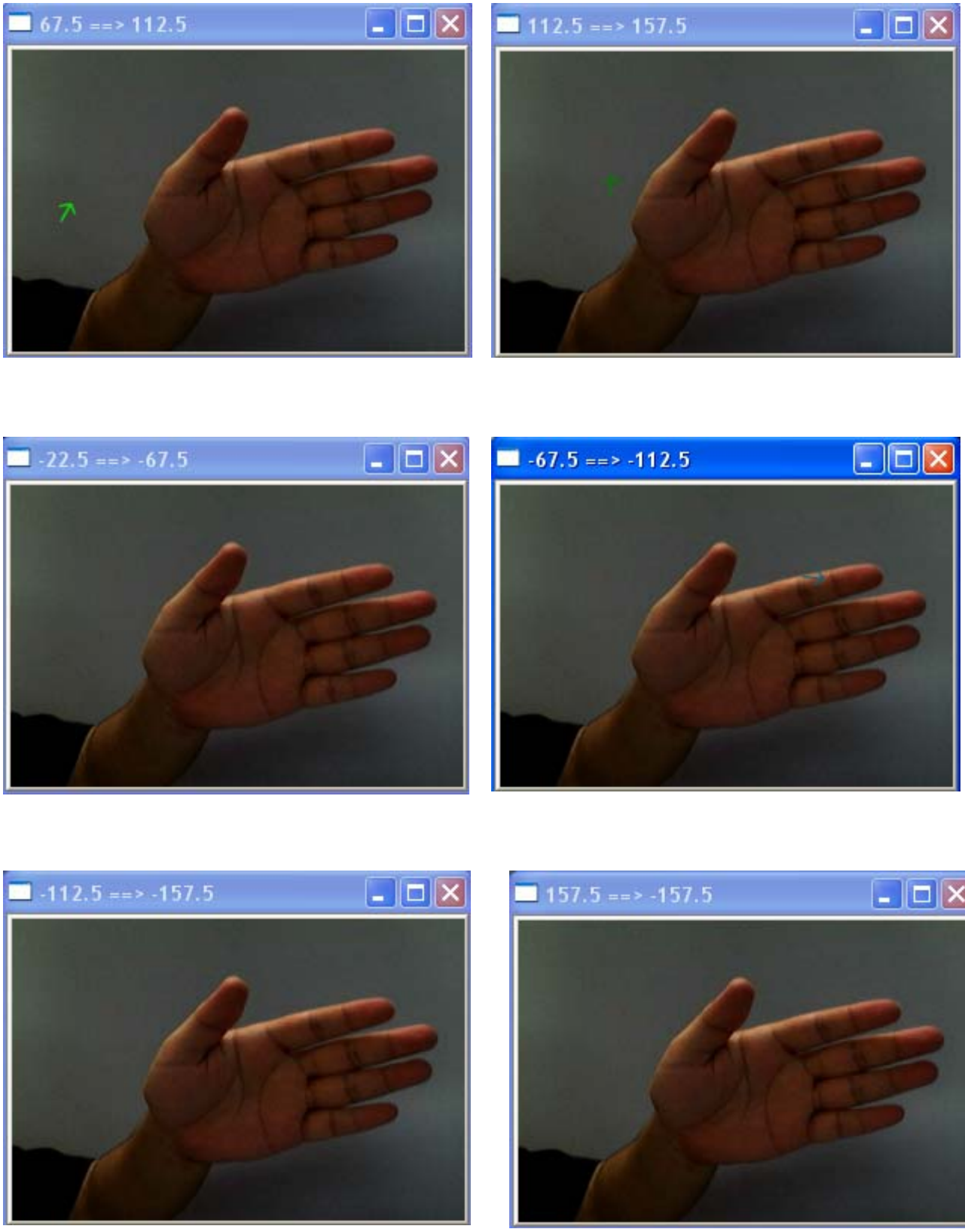


รูปภาพ 3-5 dialog **แสดงการภาพต้นฉบับและหา** Optical Flow

หลังจากโปรแกรมแสดงหน้าการหา Optical Flowแล้ว โปรแกรมจะแสดงหน้าต่างการแยกการเคลื่อนไหว 8ทิศทาง

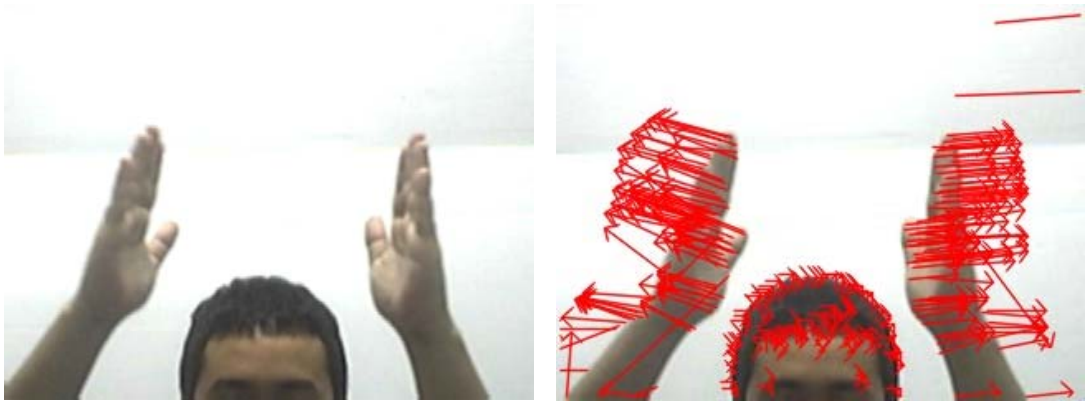




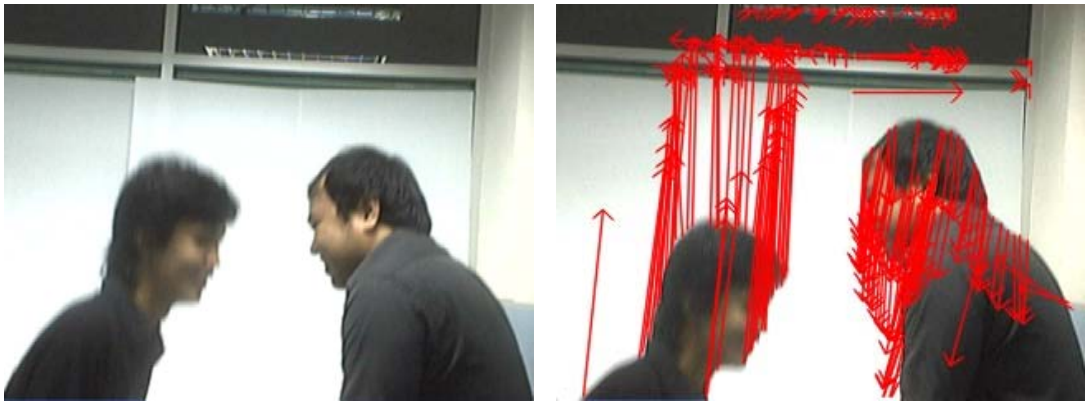


รูปภาพ 3-6 dialog หน้าต่างแยกการเคลื่อนที่ทั้ง 8 ทิศ

3.3 ผลลัพธ์โปรแกรมการคำนวณหา Optical Flow



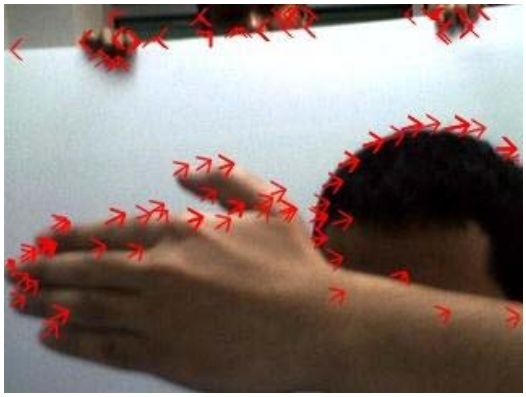
รูปภาพ 3-7 แยกมีอออกจากกัน



รูปภาพ 3-8 เคลื่อนที่ขึ้นลงสลับกัน

รูปภาพ 3-9 หมุนกล้องไปทางขวา

Input



outputเริ่มต้น

ผลลัพธ์หลังการ Classified 8ทิศทาง

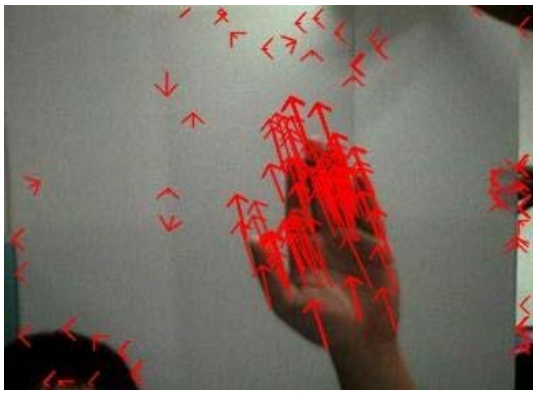
1 2 3

4 5 6

7 8

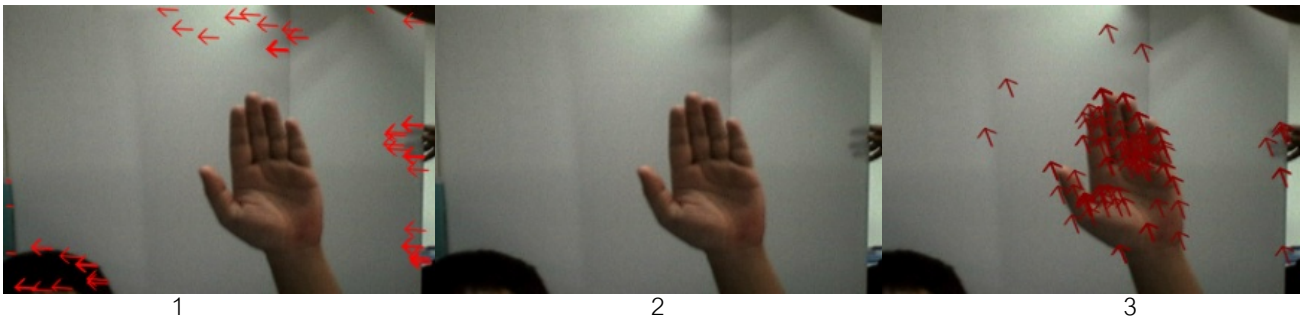
รูปภาพ 3-10 ตัวอย่างการเคลื่อนไหวในแนวทิศที่ 1 ตามทิศอ้างอิง

Input



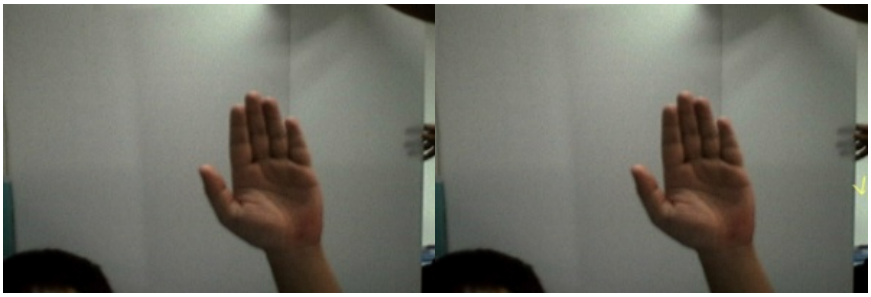
outputเริ่มต้น

ผลลัพธ์หลังการ Classified 8ทิศทาง



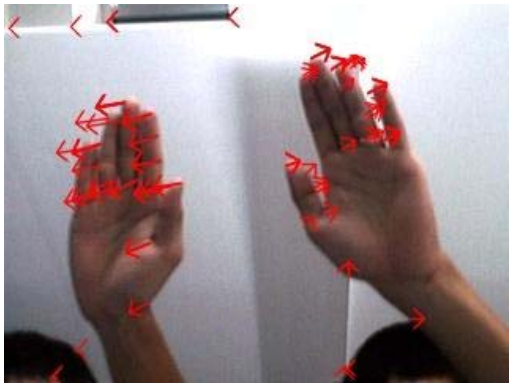
1 2 3

4 5 6



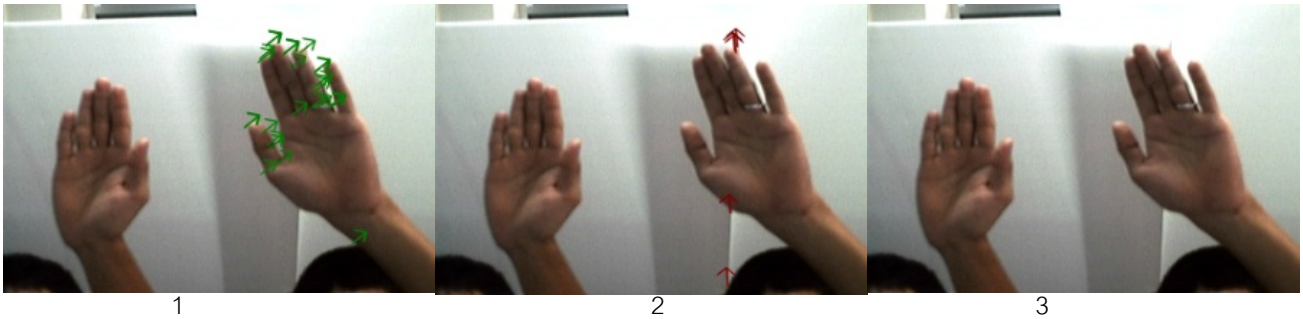
7 8

รูปภาพ 3-11 ตัวอย่างการเคลื่อนไหวในแนวทิศที่ 3 ตามทิศอ้างอิง



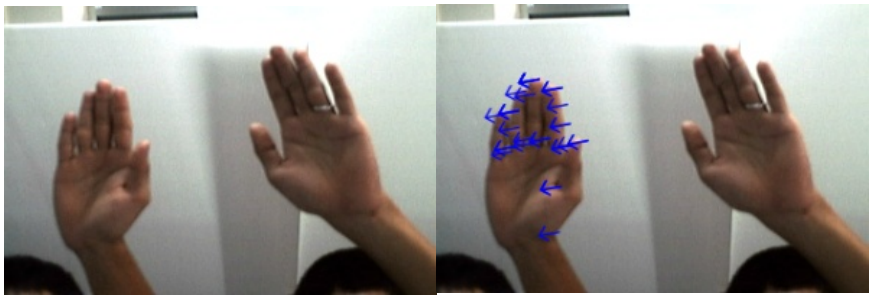
Input output เริ่มต้น

ผลลัพธ์หลังการ Classified 8ทิศทาง



1 2 3

4 5 6



7 8

รูปภาพ 3-12 ตัวอย่างการเคลื่อนไหวในแนวทิศที่ 1 และทิศที่ 8 ตามทิศอ้างอิง

บทที่ 4 *สรุปผลและข้อเสนอแนะ*

4.1 สรุปผลการดำเนินงาน

จากการดำเนินงานในโครงงาน Motion Detection by Optical flow นี้เพื่อที่จะนำไปใช้ในระบบตรวจสอบผู้บุกรุกหรือระบบเฝ้าระวัง ซึ่งการจับความเคลื่อนไหวเป็นหนึ่งในวิธีสำคัญเพื่อตรวจจับวัตถุที่กำลังเคลื่อนที่ โดยได้เน้นไปที่กรณีศึกษาในกรณีที่ตัวกล้องสามารถเคลื่อนที่ไปด้วย ซึ่งเทคนิค background subtraction จึงไม่สามารถใช้ได้ การจับความเคลื่อนไหวโดยเทคนิค optical flow จึงได้ถูกนำมาใช้ โดยสามารถใช้ในการหาความเร็ว ทิศทาง และยังสามารถแยกแยะและระบุวัตถุที่กำลังเคลื่อนที่ได้ หลังจากที่ได้ศึกษาและพัฒนาโครงงานมาเป็นระยะเวลา 2 ภาคการเรียน โดยได้นำเทคนิคการหา opticalflow ที่อ้างอิงจากทฤษฎีของ Lucas – Kanade method มาใช้ในการพัฒนาโครงงาน ซึ่งสามารถทำงานได้ผลลัพธ์ค่อนข้างตรงตามเป้าหมายที่วางไว้ คือสามารถหา optical flow ของวัตถุที่กำลังเคลื่อนที่ได้ และสามารถแยกแยะวัตถุที่กำลังเคลื่อนที่ได้ โดยสามารถแยกแยะวัตถุที่กำลังเคลื่อนที่ต่างกันได้ 8 ทิศทาง

4.2 ปัญหาและอุปสรรค

เนื่องด้วยการเขียนโปรแกรมมีการ include library OpenCV เข้ามา ซึ่งข้าพเจ้าไม่เข้าใจวิธีการใช้งานของฟังก์ชันหลาย ๆ ฟังก์ชัน จึงทำให้เสียเวลาในการศึกษาและพัฒนาโปรแกรมมากกว่าปกติ และเนื่องจากมีประสบการณ์และความชำนาญในด้านการเขียนโปรแกรมค่อนข้างน้อย จึงทำให้โปรแกรมที่พัฒนาขึ้นมามีการจองหน่วยความจำเป็นจำนวนมาก จึงไม่สามารถรันการทำงานของโปรแกรมเป็นเวลานานๆได้ และหลังการรันโปรแกรมทุกครั้ง จะต้องทำการปิดโปรเซสการทำงานของโปรแกรมก่อน เพื่อจะรันโปรแกรมครั้งต่อไปได้ จึงทำให้เกิดความไม่สะดวกในการใช้งาน

บรรณานุกรม

[1] *นิรุธ อำนวยศิลป์* , "คู่มือการเขียนโปรแกรม Microsoft Vision 6.0" บริษัท ส.เอเซีย เพรส (1989) จำกัด

[2] *วรวุฒิ เที่ยงธรรม* , "เรียนลัด Vision C++ 6.0" บริษัท ออฟเซส เพรส จำกัด

[3] J. P. Lewis, "Fast Normalized Cross-Correlation", July,24,2007, http://www.idiom.com/~zilla/Work/nvisionInterface/nip.html

[4] "Optical Flow" http://en.wikipedia.org/wiki/Optical_flow [12/07/2550]

[5] "Lucas–Kanade method" http://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method [17/07/2550]

[6] "Horn–Schunck method" http://groups.google.ws/group/OpenCV/msg/ec125224a1123fc2 [19/07/2550]

[7] Jiangjian Xiao, Hui Cheng, Harpreet Sawhney, Cen Rao, and Michael Isnardi, "Bilateral Filtering-based Optical Flow Estimation with Occlusion Detection", July,30,2007, http://www.cs.ucf.edu/~jxiao/opticalflow.htm

[8] "CalcOpticalFlowBM examples" http://groups.google.ws/group/OpenCV/msg/ec125224a1123fc2 [04/08/2550]

[9] CalcOpticalFlowPyrLK.CV Reference Manual. http://www.cse.iitb.ac.in/~sharat/current/cs663/opencv/ref/opencvref_cv.htm [15/08/2550]

[10] [Bouguet00] Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker.

http://www.cse.iitb.ac.in/~sharat/current/cs663/opencv/ref/opencvref_cv.htm#paper_bouguet00 [15/08/2550]

ภาคผนวก

Source Code ของโปรแกรม

รายละเอียด code ไฟล์ testDlg.cpp

การประกาศค่าเริ่มต้น

```
#include "stdafx.h"
#include "test.h"
#include "testDlg.h"
#include <cxcore.h>
#include <cv.h>
#include "cvcam.h"
#include <highgui.h>
#include <string.h>
#include "calAverageFlow.h"
#include<fstream.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define cvMirror cvFlip
#define ID_USER1    30000

void callback(IplImage *image);

CvSize imgSize = cvSize(320,240);
CvSize imgSize1 = cvSize(640,480);
IplImage *ImageSource = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
```

```
IplImage *ImageGray   = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage *ImageOutput = cvCreateImage( imgSize1 , IPL_DEPTH_8U, 3);


IplImage* img1 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img2 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img3 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img4 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img5 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img6 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img7 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage* img8 = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);


IplImage *ImgOpticalFlow = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);


static IplImage  *frame1 = NULL, *frame1_1C = NULL, *frame2_1C = NULL,
*eig_image = NULL, *temp_image = NULL, *pyramid1 = NULL, *pyramid2 =
NULL;
IplImage    *frameLast = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);
IplImage    *frame = cvCreateImage( imgSize , IPL_DEPTH_8U, 3);


char str[2000];
CvPoint p,q;


CWnd* Source;
CDC* SourceDC;


CWnd* Gray;
CDC* GrayDC;


CWnd* Output;
CDC* OutputDC;


CWnd* OpticalFlow;
```

```
CDC* OpticalFlowDC;
CTestDlg *pMainDialog = NULL;


BITMAPINFO m_BitmapInfo;
BITMAPINFO m_BitmapInfo1;


fstream file_op("c:\\test_file.txt",ios::out);


static const double pi = 3.14159265358979323846;


inline static double square(int a)
{
	return a * a;
}


inline static void allocateOnDemand( IplImage **img, CvSize size, int
depth, int channels )
{
	if ( *img != NULL )	return;


	*img = cvCreateImage( size, depth, channels );
	if ( *img == NULL )
	{
		fprintf(stderr, "Error: Couldn't allocate image.  Out of
memory?\n");
		exit(-1);
	}
}
```

ส่วนการจัดการ Dialog ของโปรแกรม และรับคำสั่งจากผู้ใช้งาน

```
class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	enum { IDD = IDD_ABOUTBOX };
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);  support

protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////
// CTestDlg dialog
CTestDlg::CTestDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CTestDlg::IDD, pParent)
{
```

```
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CTestDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CTestDlg, CDialog)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDSTART, OnStart)

END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////////
// CTestDlg message handlers

BOOL CTestDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
```

```
                  pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
            }
      }
      SetIcon(m_hIcon, TRUE);            // Set big icon
      SetIcon(m_hIcon, FALSE);           // Set small icon

      // TODO: Add extra initialization here
      memset(&m_BitmapInfo,0,sizeof(BITMAPINFO));
      m_BitmapInfo.bmiHeader.biSize = sizeof(BITMAPINFO);
      m_BitmapInfo.bmiHeader.biWidth = 320;
      m_BitmapInfo.bmiHeader.biHeight = 240;
      m_BitmapInfo.bmiHeader.biPlanes = 1;
      m_BitmapInfo.bmiHeader.biBitCount = 24;

      memset(&m_BitmapInfo1,0,sizeof(BITMAPINFO));
      m_BitmapInfo1.bmiHeader.biSize = sizeof(BITMAPINFO);
      m_BitmapInfo1.bmiHeader.biWidth = 640;
      m_BitmapInfo1.bmiHeader.biHeight = 480;
      m_BitmapInfo1.bmiHeader.biPlanes = 1;
      m_BitmapInfo1.bmiHeader.biBitCount = 24;

}

void CTestDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
      if ((nID & 0xFFF0) == IDM_ABOUTBOX)
      {
            CAboutDlg dlgAbout;
            dlgAbout.DoModal();
      }
      else
      {
```

```
            CDialog::OnSysCommand(nID, lParam);
      }
}

void CTestDlg::OnPaint()
{
      if (IsIconic())
      {
            CPaintDC dc(this); // device context for painting

            SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);
            // Center icon in client rectangle
            int cxIcon = GetSystemMetrics(SM_CXICON);
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(&rect);
            int x = (rect.Width() - cxIcon + 1)/2;
            int y = (rect.Height() - cyIcon + 1)/2;
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            CDialog::OnPaint();
      }
}

HCURSOR CTestDlg::OnQueryDragIcon()
{
      return (HCURSOR) m_hIcon;
}

void CTestDlg::OnStart()
{
```

```
int key = -1;

pMainDialog = this;

Source = GetDlgItem(IDC_SOURCE);
SourceDC = Source->GetDC();

OpticalFlow = GetDlgItem(IDC_GRAY);
OpticalFlowDC = OpticalFlow->GetDC();

Output = GetDlgItem(IDC_OUTPUT);
OutputDC = Output->GetDC();

int* out;
int ncams = cvcamGetCamerasCount( );
int nselected = cvcamSelectCamera(&out);

cvcamSetProperty(out[0], CVCAM_PROP_ENABLE, CVCAMTRUE);

cvcamSetProperty(out[0], CVCAM_PROP_RENDER, CVCAMTRUE);

cvcamSetProperty(out[0],  CVCAM_PROP_CALLBACK, callback);

cvcamInit();
cvcamStart();
cvNamedWindow( "-22.5 ==> 22.5");
cvNamedWindow( "22.5 ==> 67.2");
cvNamedWindow( "67.5 ==> 112.5");
cvNamedWindow( "112.5 ==> 157.5");
cvNamedWindow( "-22.5 ==> -67.5");
cvNamedWindow( "-67.5 ==> -112.5");
cvNamedWindow( "-112.5 ==> -157.5");
cvNamedWindow( "157.5 ==> -157.5");
```

```
}


void CTestDlg::OnCancel()
{
	cvReleaseImage(&frame);
	cvReleaseImage(&frame1);
	cvReleaseImage(&frameLast);
	cvReleaseImage(&frame1_1C);
	cvReleaseImage(&frame2_1C);
	cvReleaseImage(&eig_image);
	cvReleaseImage(&temp_image);
	cvReleaseImage(&pyramid1);
	cvReleaseImage(&pyramid2);

	CDialog::OnCancel();

	file_op.close();
}
```

ส่วนการรับภาพจากกล้องเพื่อมาคำนวณ และประมวลตามอัลกอริทึม

```
void callback(IplImage *image)
{
      cvResize(image,ImageOutput, CV_INTER_LINEAR );

            long current_frame = 0;
            cvCopyImage(frameLast, frame);

            allocateOnDemand( &frame1_1C, imgSize, IPL_DEPTH_8U, 1 );
            cvConvertImage(frame, frame1_1C, CV_CVTIMG_FLIP);
            allocateOnDemand( &frame1, imgSize, IPL_DEPTH_8U, 3 );
            cvConvertImage(frame, frame1, CV_CVTIMG_FLIP);

            cvCopyImage(image , frame);

            if (frame == NULL)
            {
                  fprintf(stderr, "Error: Hmm. The end came sooner
than we thought.\n");
            }
            // Query Frame 2
            allocateOnDemand( &frame2_1C, imgSize, IPL_DEPTH_8U, 1 );
            cvConvertImage(frame, frame2_1C, CV_CVTIMG_FLIP);

            allocateOnDemand( &eig_image, imgSize, IPL_DEPTH_32F, 1 );
            allocateOnDemand( &temp_image, imgSize, IPL_DEPTH_32F, 1 );

            CvPoint2D32f frame1_features[400];
            int number_of_features;
            number_of_features = 400;

            cvGoodFeaturesToTrack(frame1_1C, eig_image, temp_image,
frame1_features, &number_of_features, .01, .01, NULL);
```

```
            //percolated point
            cvFindCornerSubPix(     frame1_1C,frame1_features,
number_of_features,

            cvSize(10, 10), cvSize(-1,-1),

            cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.03));
            CvPoint2D32f frame2_features[400];
            char optical_flow_found_feature[400];
            float optical_flow_feature_error[400];

            CvSize optical_flow_window = cvSize(3,3);
            CvTermCriteria optical_flow_termination_criteria
                  = cvTermCriteria( CV_TERMCRIT_ITER |
CV_TERMCRIT_EPS, 20, .3 );

            allocateOnDemand( &pyramid1, imgSize, IPL_DEPTH_8U, 1 );
            allocateOnDemand( &pyramid2, imgSize, IPL_DEPTH_8U, 1 );
            IplImage *velocityX = cvCreateImage(imgSize,
IPL_DEPTH_32F, 1);
          IplImage *velocityY = cvCreateImage(imgSize, IPL_DEPTH_32F,
1);
// calculate Optical Flow
            cvCalcOpticalFlowPyrLK(frame1_1C, frame2_1C, pyramid1,
pyramid2,frame1_features, frame2_features, number_of_features,
optical_flow_window, 5, optical_flow_found_feature,
optical_flow_feature_error, optical_flow_termination_criteria, 0 );

            Arrow *angleArr = new Arrow [number_of_features];
            for(int i = 0; i < number_of_features; i++)
            {
                  if ( optical_flow_found_feature[i] == 0 )  continue;

                  int line_thickness;
                  line_thickness = 1;
```

```
                CvScalar line_color;
                line_color = CV_RGB(255,0,0);

                p.x = (int) frame1_features[i].x;
                p.y = (int) frame1_features[i].y;
                q.x = (int) frame2_features[i].x;
                q.y = (int) frame2_features[i].y;

                angleArr[i].p.x = p.x;
                angleArr[i].p.y = p.y;
                angleArr[i].q.x = q.x;
                angleArr[i].q.y = q.y;

                double angle;
                angle = atan2((double) p.y - q.y, (double) p.x - q.x
);
                angleArr[i].angle = angle;
                double hypotenuse;
                hypotenuse = sqrt( square(p.y - q.y) + square(p.x -
q.x) );

                angleArr[i].hypotenuse = hypotenuse;

            /* cut hypotenuse <3 */
            if (  hypotenuse > 3)
                   {
                q.x = (int) (p.x - 3*hypotenuse * cos(angle));
                q.y = (int) (p.y - 3*hypotenuse * sin(angle));

                cvLine( frame1, p, q, line_color, line_thickness,
CV_AA, 0);

                p.x = (int) (q.x + 9*cos(angle + pi / 4));
```

```
                    p.y = (int) (q.y + 9*sin(angle + pi / 4));
                    cvLine( frame1, p, q, line_color, line_thickness,
CV_AA, 0);
                    p.x = (int) (q.x + 9*cos(angle - pi / 4));
                    p.y = (int) (q.y + 9*sin(angle - pi / 4));
                    cvLine( frame1, p, q, line_color, line_thickness,
CV_AA, 0);
                     }
             }

             cvCopyImage(frame , frameLast);
             cvConvertImage(frame1, frame1, 1);
             cvCopyImage(frame1, ImgOpticalFlow);
             cvSaveImage("source.jpg",frame);
             cvSaveImage("output.jpg",ImgOpticalFlow);
// end calculate Optical Flow

//Classified
             cvCopyImage(image , img1);
             cvCopyImage(image , img2);
             cvCopyImage(image , img3);
             cvCopyImage(image , img4);
             cvCopyImage(image , img5);
             cvCopyImage(image , img6);
             cvCopyImage(image , img7);
             cvCopyImage(image , img8);

             cvConvertImage(img1, img1, CV_CVTIMG_FLIP);
             cvConvertImage(img2, img2, CV_CVTIMG_FLIP);
             cvConvertImage(img3, img3, CV_CVTIMG_FLIP);
             cvConvertImage(img4, img4, CV_CVTIMG_FLIP);
```

```
            cvConvertImage(img5, img5, CV_CVTIMG_FLIP);
            cvConvertImage(img6, img6, CV_CVTIMG_FLIP);
            cvConvertImage(img7, img7, CV_CVTIMG_FLIP);
            cvConvertImage(img8, img8, CV_CVTIMG_FLIP);

            double mue[8] = {0,0,0,0,0,0,0,0};
            double omega[8] = {0,0,0,0,0,0,0,0};
            int count[8] = {0,0,0,0,0,0,0,0};
            Arrow angle1[400],angle2[400],angle3[400],angle4[400],angle5[400],
angle6[400],angle7[400],angle8[400];

            for(i = 0; i <= number_of_features ; i++)
            {
                  //1
                  if( angleArr[i].angle >= (-1)*(pi/8)&&
angleArr[i].angle <= pi/8 )
                  {
                        angle1[count[0]] = angleArr[i];
                        count[0]++;
                  }
                  //2
                  else if(angleArr[i].angle >= pi/8 &&
angleArr[i].angle <= (0.375)*pi )
                  {
                        angle2[count[1]] = angleArr[i];
                        count[1]++;
                  }
                  //3
                  else if(angleArr[i].angle >= (0.375)*pi &&
angleArr[i].angle <= (0.625)*pi)
                  {
                        angle3[count[2]]= angleArr[i];
                        count[2]++;
```

```
                    }
                    //4
                    else if(angleArr[i].angle >= (0.625)*pi &&
angleArr[i].angle <= (0.875)*pi)
                    {
                          angle4[count[3]] = angleArr[i];
                          count[3]++;
                    }
                    //5
                    else if(angleArr[i].angle >= (-1)*(0.375)*pi &&
angleArr[i].angle <=(-1)*(pi/8))
                    {
                          angle5[count[4]] = angleArr[i];
                          count[4]++;
                    }
                    //6
                    else if(angleArr[i].angle >= (-1)*(0.625)*pi &&
angleArr[i].angle <= (-1)*(0.375)*pi)
                    {
                          angle6[count[5]] = angleArr[i];
                          count[5]++;
                    }
                    //7
                    else if(angleArr[i].angle >= (-1)*(0.875)*pi &&
angleArr[i].angle <= (-1)*(0.625)*pi)
                    {
                          angle7[count[6]] = angleArr[i];
                          count[6]++;
                    }
                    //8
                    else if((angleArr[i].angle >= (0.875)*pi) &&
(angleArr[i].angle <= 1.125*pi))
                    {
```

```
                        angle8[count[7]] = angleArr[i];
                        count[7]++;
                  }
            }

//          file_op << count[2]<<count[6]<< "\n";
                  calAverageFlow(angle1 , count[0],mue[0], omega[0]);
                  calAverageFlow(angle2 , count[1],mue[1], omega[1]);
                  calAverageFlow(angle3 , count[2],mue[2], omega[2]);
                  calAverageFlow(angle4 , count[3],mue[3], omega[3]);
                  calAverageFlow(angle5 , count[4],mue[4], omega[4]);
                  calAverageFlow(angle6 , count[5],mue[5], omega[5]);
                  calAverageFlow(angle7 , count[6],mue[6], omega[6]);
                  calAverageFlow(angle8 , count[7],mue[7], omega[7]);

//draw 1
            for( i = 0; i < count[0]; i++)
            {
                  if ( optical_flow_found_feature[i] == 0 )  continue;

                  int line_thickness;
                  line_thickness = 1;

                  CvScalar line_color;
                  line_color = CV_RGB(255,0,0);

                  p = angle1[i].p;
                  q = angle1[i].q;
                  double angle;
                  angle = mue[0]
                  double hypotenuse;      hypotenuse = 5.0;
```

```
                q.x = (int) (p.x - 3*hypotenuse * cos(angle));
                q.y = (int) (p.y - 3*hypotenuse * sin(angle));

                cvLine( img1, p, q, line_color, line_thickness,
CV_AA, 0 );

                p.x = (int) (q.x + 9*cos(angle + pi / 4));
                p.y = (int) (q.y + 9*sin(angle + pi / 4));
                cvLine( img1, p, q, line_color, line_thickness,
CV_AA, 0 );
                p.x = (int) (q.x + 9*cos(angle - pi / 4));
                p.y = (int) (q.y + 9*sin(angle - pi / 4));
                cvLine( img1, p, q, line_color, line_thickness,
CV_AA, 0 );
            }

            //draw2
            for( i = 0; i < count[1]; i++)
            {
                  if ( optical_flow_found_feature[i] == 0 )  continue;
                  int line_thickness;
                  line_thickness = 1;
                  CvScalar line_color;
                  ine_color = CV_RGB(225,125,0);

                  p = angle2[i].p;
                  q = angle2[i].q;
                  double angle;           angle = mue[1];
                  double hypotenuse;      hypotenuse = 5.0;

                  q.x = (int) (p.x - 3*hypotenuse * cos(angle));
                  q.y = (int) (p.y - 3*hypotenuse * sin(angle));
```

```
                    cvLine( img2, p, q, line_color, line_thickness,
CV_AA, 0 );



                    p.x = (int) (q.x + 9*cos(angle + pi / 4));

                    p.y = (int) (q.y + 9*sin(angle + pi / 4));

                    cvLine( img2, p, q, line_color, line_thickness,
CV_AA, 0 );

                    p.x = (int) (q.x + 9*cos(angle - pi / 4));

                    p.y = (int) (q.y + 9*sin(angle - pi / 4));

                    cvLine( img2, p, q, line_color, line_thickness,
CV_AA, 0 );
             }


             //draw3

             for( i = 0; i < count[2]; i++)

             {
                    if ( optical_flow_found_feature[i] == 0 )  continue;

                    int line_thickness;
                    line_thickness = 1;

                    CvScalar line_color;

                    line_color = CV_RGB(0,255,0);




                    p = angle3[i].p;

                    q = angle3[i].q;

                    double angle;           angle = mue[2];

                    double hypotenuse;      hypotenuse = 5.0;


                    q.x = (int) (p.x - 3*hypotenuse * cos(angle));

                    q.y = (int) (p.y - 3*hypotenuse * sin(angle));


                    cvLine( img3, p, q, line_color, line_thickness,
CV_AA, 0 );
```

```
                p.x = (int) (q.x + 9*cos(angle + pi / 4));
                p.y = (int) (q.y + 9*sin(angle + pi / 4));
                cvLine( img3, p, q, line_color, line_thickness,
CV_AA, 0 );
                p.x = (int) (q.x + 9*cos(angle - pi / 4));
                p.y = (int) (q.y + 9*sin(angle - pi / 4));
                cvLine( img3, p, q, line_color, line_thickness,
CV_AA, 0 );
        }

        //draw4
        for( i = 0; i < count[3]; i++)
        {
                if ( optical_flow_found_feature[i] == 0 )  continue;
                int line_thickness;
                line_thickness = 1;
                CvScalar line_color;
                line_color = CV_RGB(0,125,0);

                p = angle4[i].p;
                q = angle4[i].q;
                double angle;           angle = mue[3];
                double hypotenuse;      hypotenuse = 5.0;

                q.x = (int) (p.x - 3*hypotenuse * cos(angle));
                q.y = (int) (p.y - 3*hypotenuse * sin(angle));

                cvLine( img4, p, q, line_color, line_thickness,
CV_AA, 0 );

                p.x = (int) (q.x + 9*cos(angle + pi / 4));
                p.y = (int) (q.y + 9*sin(angle + pi / 4));
```

```
                    cvLine( img4, p, q, line_color, line_thickness,
CV_AA, 0 );

                    p.x = (int) (q.x + 9*cos(angle - pi / 4));

                    p.y = (int) (q.y + 9*sin(angle - pi / 4));

                    cvLine( img4, p, q, line_color, line_thickness,
CV_AA, 0 );

             }


             //draw5

             for( i = 0; i < count[4]; i++)

             {

                    if ( optical_flow_found_feature[i] == 0)   continue;


                    int line_thickness;
                    line_thickness = 1;

                    CvScalar line_color;

                    line_color = CV_RGB(0,0,255);


                    p = angle5[i].p;

                    q = angle5[i].q;

                    double angle;           angle = mue[4];

                    double hypotenuse;      hypotenuse = 5.0;


                    q.x = (int) (p.x - 3*hypotenuse * cos(angle));

                    q.y = (int) (p.y - 3*hypotenuse * sin(angle));


                    cvLine( img5, p, q, line_color, line_thickness,
CV_AA, 0 );


                    p.x = (int) (q.x + 9*cos(angle + pi / 4));

                    p.y = (int) (q.y + 9*sin(angle + pi / 4));

                    cvLine( img5, p, q, line_color, line_thickness,
CV_AA, 0 );
```

```
                p.x = (int) (q.x + 9*cos(angle - pi / 4));

                p.y = (int) (q.y + 9*sin(angle - pi / 4));

                cvLine( img5, p, q, line_color, line_thickness,
CV_AA, 0 );

        }


        //draw6

        for( i = 0; i < count[5]; i++)

        {

                if ( optical_flow_found_feature[i] == 0 )  continue;


                int line_thickness;
                line_thickness = 1;

                CvScalar line_color;

                line_color = CV_RGB(0,100,125);


                //CvPoint p,q;


                p = angle6[i].p;

                q = angle6[i].q;

                double angle;           angle = mue[5];

                double hypotenuse;      hypotenuse = 5.0;


                q.x = (int) (p.x - 3*hypotenuse * cos(angle));

                q.y = (int) (p.y - 3*hypotenuse * sin(angle));


                cvLine( img6, p, q, line_color, line_thickness,
CV_AA, 0 );


                p.x = (int) (q.x + 9*cos(angle + pi / 4));

                p.y = (int) (q.y + 9*sin(angle + pi / 4));

                cvLine( img6, p, q, line_color, line_thickness,
CV_AA, 0 );
```

```
                  p.x = (int) (q.x + 9*cos(angle - pi / 4));

                  p.y = (int) (q.y + 9*sin(angle - pi / 4));

                  cvLine( img6, p, q, line_color, line_thickness,
CV_AA, 0 );
            }


            //draw7

            for( i = 0; i < count[6]; i++)

            {
                  if ( optical_flow_found_feature[i] == 0 )  continue;


                  int line_thickness;
                  line_thickness = 1;

                  CvScalar line_color;

                  line_color = CV_RGB(100,255,255);


                  p = angle7[i].p;

                  q = angle7[i].q;

                  double angle;           angle = mue[6];

                  double hypotenuse;      hypotenuse = 5.0;


                  q.x = (int) (p.x - 3*hypotenuse * cos(angle));

                  q.y = (int) (p.y - 3*hypotenuse * sin(angle));

                  cvLine( img7, p, q, line_color, line_thickness,
CV_AA, 0 );

                  p.x = (int) (q.x + 9*cos(angle + pi / 4));

                  p.y = (int) (q.y + 9*sin(angle + pi / 4));

                  cvLine( img7, p, q, line_color, line_thickness,
CV_AA, 0 );

                  p.x = (int) (q.x + 9*cos(angle - pi / 4));

                  p.y = (int) (q.y + 9*sin(angle - pi / 4));

                  cvLine( img7, p, q, line_color, line_thickness,
CV_AA, 0 );
```

```
            }


            //draw8

            for( i = 0; i < count[7]; i++)

            {
                  if ( optical_flow_found_feature[i] == 0)   continue;


                  int line_thickness;
                  line_thickness = 1;

                  CvScalar line_color;

                  line_color = CV_RGB(255,255,100);


                  p = angle8[i].p;

                  q = angle8[i].q;

                  double angle;           angle = mue[7];

                  double hypotenuse;      hypotenuse = 5.0;


                  q.x = (int) (p.x - 3*hypotenuse * cos(angle));

                  q.y = (int) (p.y - 3*hypotenuse * sin(angle));

                  cvLine( img8, p, q, line_color, line_thickness,
CV_AA, 0 );


                  p.x = (int) (q.x + 9*cos(angle + pi / 4));

                  p.y = (int) (q.y + 9*sin(angle + pi / 4));

                  cvLine( img8, p, q, line_color, line_thickness,
CV_AA, 0 );

                  p.x = (int) (q.x + 9*cos(angle - pi / 4));

                  p.y = (int) (q.y + 9*sin(angle - pi / 4));

                  cvLine( img8, p, q, line_color, line_thickness,
CV_AA, 0 );

            }


            cvFlip( img1,img1 , CV_CVTIMG_FLIP);
```

```
            cvFlip( img2,img2 , CV_CVTIMG_FLIP);
            cvFlip( img3,img3 , CV_CVTIMG_FLIP);
            cvFlip( img4,img4 , CV_CVTIMG_FLIP);
            cvFlip( img5,img5 , CV_CVTIMG_FLIP);
            cvFlip( img6,img6 , CV_CVTIMG_FLIP);
            cvFlip( img7,img7 , CV_CVTIMG_FLIP);
            cvFlip( img8,img8 , CV_CVTIMG_FLIP);

            CvSize showsize = cvSize(300,200);

            IplImage* img1t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            IplImage* img2t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            IplImage* img3t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            IplImage* img4t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            IplImage* img5t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            IplImage* img6t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            IplImage* img7t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            IplImage* img8t = cvCreateImage(showsize,IPL_DEPTH_8U, 3);
            cvResize(img1,img1t, CV_INTER_LINEAR );
            cvResize(img2,img2t, CV_INTER_LINEAR );
            cvResize(img3,img3t, CV_INTER_LINEAR );
            cvResize(img4,img4t, CV_INTER_LINEAR );
            cvResize(img5,img5t, CV_INTER_LINEAR );
            cvResize(img6,img6t, CV_INTER_LINEAR );
            cvResize(img7,img7t, CV_INTER_LINEAR );
            cvResize(img8,img8t, CV_INTER_LINEAR );

            cvShowImage( "-22.5 ==> 22.5",img1t );
            cvShowImage( "22.5 ==> 67.2",img2t );
            cvShowImage( "67.5 ==> 112.5",img3t );
            cvShowImage( "112.5 ==> 157.5",img4t );
```

```
            cvShowImage( "-22.5 ==> -67.5",img5t);

            cvShowImage( "-67.5 ==> -112.5",img6t);

            cvShowImage( "-112.5 ==> -157.5",img7t );

            cvShowImage( "157.5 ==> -157.5",img8t );

            cvSaveImage("1.jpg",img1t);

            cvSaveImage("2.jpg",img2t);

            cvSaveImage("3.jpg",img3t);

            cvSaveImage("4.jpg",img4t);

            cvSaveImage("5.jpg",img5t);

            cvSaveImage("6.jpg",img6t);

            cvSaveImage("7.jpg",img7t);

            cvSaveImage("8.jpg",img8t);

            cvReleaseImage(&img1t);

            cvReleaseImage(&img2t);

            cvReleaseImage(&img3t);

            cvReleaseImage(&img4t);

            cvReleaseImage(&img5t);

            cvReleaseImage(&img6t);

            cvReleaseImage(&img7t);

            cvReleaseImage(&img8t);

//End of Classified

            SetDIBitsToDevice(SourceDC->m_hDC, 0, 0, 320, 240, 0, 0, 0,
320,image-> imageData, &m_BitmapInfo,DIB_RGB_COLORS);

            SetDIBitsToDevice(OpticalFlowDC->m_hDC, 0, 0, 320, 240, 0, 0,
0, 320,ImgOpticalFlow-> imageData, &m_BitmapInfo,DIB_RGB_COLORS);

            SetDIBitsToDevice(OutputDC->m_hDC, 0, 0, 640, 480, 0, 0, 0,
640,ImageOutput-> imageData, &m_BitmapInfo1,DIB_RGB_COLORS);

       //key = cvWaitKey(33);

            _sleep(100);

}
```

รายละเอียด code ไฟล์ calAverageFlow.cpp

```
#include "stdafx.h"
#include "test.h"
#include "testDlg.h"
#include "calAverageFlow.h"
#include <cxcore.h>
#include <cv.h>
#include "cvcam.h"
#include <highgui.h>
#include <string.h>
#include <math.h>
static double alpha = 0.5;
double mue = 0;
double omega = 0;
void calAverageFlow(Arrow *angle , int num_of_feather ,double &mueDst
, double &omegaDst)
{
   double total = 0;
                for(int i = 0 ; i < num_of_feather ; i++)
                {
                 if(angle[i].hypotenuse > 3  )
                 {
                 mue = (alpha * angle[i].angle) + (1 - alpha) *mue ;

                 omega = alpha *pow((angle[i].angle - mue),2) + (1 -
alpha)*omega;
                 }
                }
                mueDst    = mue;
                omegaDst  = omega;
}
```