รหัสโครงการ SUT7-711-48-12-63

รายงานการวิจัย

สภาวะแวดล้อมเชิงวิชวล
สำหรับการพัฒนาระบบการมองเห็นของคอมพิวเตอร์

Visual Development Environment
for Computer Vision System

คณะผู้วิจัย

หัวหน้าโครงการ

รองศาสตราจารย์ ดร.อาทิตย์ ศรีแก้ว

สาขาวิชาวิศวกรรมไฟฟ้า

สำนักวิชาวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ผู้ร่วมวิจัย

นายภาณุพงษ์ เพชรเลิศ

ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปีงบประมาณ พศ.2548

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

สิงหาคม 2553

บทคัดย่อ

งานพัฒนาวิจัยซอฟต์แวร์นี้ได้นำเสนอชุดพัฒนาเรียกว่า VisBuilder สำหรับช่วยพัฒนาสร้างระบบการมองเห็นของคอมพิวเตอร์ได้อย่างสะดวกและมีประสิทธิภาพ ชุดพัฒนานี้ประกอบด้วยซอฟต์แวร์ซึ่งมีองค์ประกอบพื้นฐานในการประมวลผลภาพเพื่อใช้ในการพัฒนาต้นแบบอย่างรวดเร็ว และใช้ต้นรหัสเป็นภาษา C เพื่อให้มีความเร็วในการทำงานเพียงพอสำหรับใช้พัฒนาองค์ประกอบการประมวลผลภาพเพิ่มเติม ชุดซอฟต์แวร์นี้มีการเชื่อมต่อกับผู้ใช้แบบวิชวล (Visual) โดยเน้นการที่ไม่ต้องเขียนโปรแกรมใหม่ ทำให้ผู้ใช้สามารถพัฒนาสร้างระบบได้อย่างง่าย เหมาะสำหรับทั้งผู้พัฒนาระบบการมองเห็นของคอมพิวเตอร์และผู้พัฒนาโปรแกรมอัลกอริทึมการประมวลผลภาพ นอกจากนั้นแล้วชุดซอฟต์แวร์นี้ยังได้พัฒนาการเชื่อมต่อระหว่างแต่ละองค์ประกอบ โดยสามารถเชื่อมต่อ VisBuilder ที่ทำงานบนคอมพิวเตอร์ในเครือข่ายเข้าด้วยกันได้ ลักษณะเด่นดังกล่าวทำให้การพัฒนาสร้างระบบการมองเห็นของคอมพิวเตอร์เป็นไปได้อย่างสะดวกและมีประสิทธิภาพ เหมาะสำหรับการประยุกต์ใช้งานได้ตั้งแต่ระบบพื้นฐานจนถึงระบบที่มีความซับซ้อน ชุดซอฟต์แวร์ VisBuilder นี้ผ่านการทดสอบใช้งานเพื่อตรวจสอบความมีเสถียรภาพและได้ถูกนำเสนอในแบบเปิดต้นรหัสเรียบร้อยแล้ว

Abstract

This work presents a software tool called VisBuilder for conveniently creating and developing computer vision applications. This software is composed of basic image processing components for rapid prototype of computer vision system. C language is used in development to achieve sufficient speed for extending image processing component. The visual GUI with drag-n-drop style is deployed for customizing components in the software environment. Each component's properties can be adjusted without any coding. This software tool is then suitable for both computer vision system developer and image processing programmer. Moreover, this software tool also provides internet connection for each component. This means different VisBuilder agents on different computers can communicate and cooperate via internet. This feature allows user to efficiently build computer vision system together with resource sharing. VisBuilder can be used to build from basic to advanced computer vision system with different skill levels of users. Currently, the stable version of this VisBuilder software has been released as open source.

กิตติกรรมประกาศ

โครงการวิจัยนี้ได้รับความร่วมมือช่วยเหลือในด้านต่างๆ จากหลายๆ ฝ่าย จนสำเร็จไปได้ลุล่วงด้วยดี คณะผู้วิจัยขอขอบพระคุณศูนย์เครื่องมือวิทยาศาสตร์และเทคโนโลยี ที่ให้ความเอื้อเฟื้อทั้งทางด้านสถานที่ เครื่องมือและบุคลากร ขอขอบพระคุณสาขาวิชาวิศวกรรมไฟฟ้าและสำนักวิศวกรรมศาสตร์สำหรับการสนับสนุนในทุกๆ ด้าน การวิจัยครั้งนี้ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปีงบประมาณพ.ศ. 2548

คณะผู้วิจัย

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเทคโนโลยีซอฟท์แวร์ได้มีการปรับปรุงและพัฒนาก้าวหน้าไปอย่างรวดเร็ว ทำให้การเขียนหรือพัฒนาซอฟท์แวร์เป็นไปอย่างมีประสิทธิภาพ และสามารถตอบสนองต่อความต้องการในการใช้งานซอฟท์แวร์ประยุกต์แบบต่างๆ ที่มีอยู่อย่างมากมายหลากหลายได้เป็นอย่างดี โดยเฉพาะในระบบปฏิบัติการไมโครซอฟท์วินโดว์ (Microsoft Windows) ซึ่งได้มีการพัฒนาฐานของเทคโนโลยีที่ช่วยทำให้การพัฒนาซอฟท์แวร์เป็นไปอย่างมีประสิทธิภาพสูงสุด ไม่ว่าจะเป็นในแง่ของเวลาและทรัพยากรในการพัฒนาซอฟท์แวร์

ถึงแม้ว่าฐานของเทคโนโลยีที่ทางไมโครซอฟท์ได้สร้างไว้ให้สำหรับผู้พัฒนาซอฟท์แวร์จะช่วยเพิ่มประสิทธิภาพในการพัฒนาต้นรหัสของซอฟท์แวร์นั้นๆ และทำให้ลดเวลาและทรัพยากรในการพัฒนาซอฟท์แวร์ได้ในระดับหนึ่ง โดยทางไมโครซอฟท์ได้สร้างกรอบงานของการพัฒนาซอฟท์แวร์ในระดับต่ำ (low level programming) เตรียมไว้ให้อย่างเพียงพอและทั่วถึงสำหรับการพัฒนาซอฟท์แวร์ในด้านต่างๆ อย่างไรก็ตามความยุ่งยากซับซ้อนในการเขียนต้นรหัสซอฟท์แวร์ในระดับสูง (high level programming) ยังเป็นส่วนสำคัญที่ทำให้การพัฒนาซอฟท์แวร์โดยรวมเป็นไปอย่างยากลำบาก

พิจารณาการพัฒนาซอฟท์แวร์แบบเปิดต้นรหัส (open source software) ซึ่งส่วนใหญ่ใช้ในระบบปฏิบัติการที่เป็นแบบเปิดต้นรหัสด้วย ยกตัวอย่างเช่นระบบปฏิบัติการลีนุกซ์ (Linux) โดยธรรมชาติของโครงสร้างของระบบปฏิบัติการดังกล่าวมีความซับซ้อน รวมไปถึงการมีโครงสร้างของระบบป้องกันความปลอดภัย (security) ที่ค่อนข้างสูง ทำให้การพัฒนาซอฟท์แวร์ในระดับต่ำ ที่ซึ่งมีความจำเป็นในการเชื่อมต่อระหว่างคอมพิวเตอร์กับฮาร์ดแวร์เป็นไปอย่างยุ่งยากและจำกัด โดยเฉพาะในปัจจุบันที่ซึ่งฮาร์ดแวร์มีการเทคโนโลยีการผลิตที่ทำให้ลดต้นทุนราคาของฮาร์ดแวร์ แต่ว่าฮาร์ดแวร์แต่ละชนิดจะมีโปรแกรมขับ (driver) ที่แตกต่างกันออกไปแล้วแต่ผู้ผลิตจะเตรียมไว้ให้ ปรากฏการณ์น่าสนใจที่เกิดขึ้นคือ บรรดาผู้ผลิตฮาร์ดแวร์ทั้งหลายมุ่งที่จะพัฒนาและใช้เทคโนโลยีการผลิตที่ทำให้ฮาร์ดแวร์สามารถใช้งานได้อย่างง่ายดายบนระบบปฏิบัติการวินโดว์ ซึ่งจะต้องมีการพัฒนาไปตามการเปลี่ยนแปลงไปของระบบปฏิบัติการวินโดว์ ที่ซึ่งตรงกันข้ามกับความรู้สึกที่ว่าการพัฒนาซอฟท์แวร์ควรจะต้องไปในทิศทางตามการพัฒนาของฮาร์ดแวร์ ฮาร์ดแวร์ทั่วๆ ไปจึงได้อยู่ในกรอบการใช้งานบนระบบปฏิบัติการวินโดว์เป็นส่วนใหญ่ การพัฒนาซอฟท์แวร์

ทั้งในระดับต่ำและระดับสูงจึงมีความพร้อมสำหรับระบบปฏิบัติการวินโดว์มากกว่าระบบปฏิบัติการอื่นๆ

นอกจากนั้นแล้ว เมื่อพิจารณาเทคโนโลยีการพัฒนาซอฟท์แวร์ด้วยซอฟท์แวร์ประเภท IDE หรือ Integrated Development Environment ซึ่งช่วยทำให้การเขียนต้นรหัสเป็นไปอย่างง่ายดายและมีประสิทธิภาพ ถึงแม้ว่าซอฟท์แวร์ประเภทดังกล่าวจะได้รับการพัฒนาบนระบบปฏิบัติการแบบเปิดต้นรหัสเพิ่มขึ้นอย่างมากมายภายในช่วงเวลา 4-5 ปีที่ผ่านมา แต่เมื่อเทียบระหว่างระบบปฏิบัติการแล้ว ระบบปฏิบัติการวินโดว์มีซอฟท์แวร์ IDE รองรับการใช้งานได้อย่างพร้อมเพรียงกว่า ไม่ว่าจะเป็น Microsoft Visual Basic หรือ Visual C++ หรือแม้แต่ซอฟท์แวร์จากค่ายอื่นๆ เองเช่น Borland Delphi หรือ C++ Builder เป็นต้น

ในปัจจุบันมีการนำระบบการมองเห็นของคอมพิวเตอร์มาใช้กันอย่างแพร่หลายทั้งในระดับงานวิจัยและการใช้ในชีวิตประจำวัน เช่น การเลือกโฟกัสใบหน้าบุคคลในกล้องถ่ายภาพ การตรวจจับหาผู้บุกรุกในระบบป้องกันภัย เป็นต้น นอกเหนือไปจากการประยุกต์ใช้ในกระบวนการผลิตระดับอุตสาหกรรม โดยใช้ในส่วนการควบคุมคุณภาพของผลิตภัณฑ์การคัดแยกผลิตภัณฑ์ การจัดวางตำแหน่งในการประกอบชิ้นงานและในอีกหลายส่วนของขั้นตอนการผลิต ดังนั้นศาสตร์การพัฒนาและประยุกต์ใช้ในระบบการมองเห็นของคอมพิวเตอร์จึงมีเพิ่มมากขึ้นเรื่อย ๆ แต่การใช้งานดังกล่าวต้องพึ่งพาอุปกรณ์ และชุดพัฒนาซอฟต์แวร์เฉพาะสำหรับอุปกรณ์นั้น ๆ

โดยทั่วไปนั้นลักษณะของงานด้านพัฒนาระบบการมองเห็นของคอมพิวเตอร์ประกอบด้วยสามขั้นตอนหลัก คือ การรับภาพเข้าสู่คอมพิวเตอร์ การประมวลผลภาพ และการแสดงผลภาพในขั้นตอนที่สองซึ่งเป็นส่วนที่เป็นซอฟต์แวร์ และเป็นส่วนที่ต้องมีการปรับเปลี่ยนอยู่ตลอดเวลาในกระบวนการพัฒนาอัลกอริทึมสำหรับการประมวลผลภาพ อีกทั้งยังเป็นส่วนที่ต้องใช้ทั้งทักษะทางด้านการเขียนโปรแกรม และความรู้ทางด้านระบบการมองเห็นของคอมพิวเตอร์ควบคู่กันไปด้วยดังนั้นในการพัฒนาระบบการมองเห็นของคอมพิวเตอร์นั้นจำเป็นต้องใช้ความรู้หลายด้านไม่เพียงต้องมีความรู้ความเข้าใจในศาสตร์ด้านระบบการมองเห็นของคอมพิวเตอร์เป็นอย่างดี แต่ยังต้องอาศัยทักษะทางด้านการเขียนโปรแกรมคอมพิวเตอร์ค่อนข้างสูงโดยเฉพาะการโปรแกรมด้วยภาษาซีที่แม้จะมีการพัฒนาไลบรารีสำเร็จรูปที่มาพร้อมต้นรหัส แต่การใช้งานก็ยังมีความยุ่งยากซับซ้อนมาก ต้องใช้เวลาในการเรียนรู้มากและต้องมีทักษะความรู้ในภาษาโปรแกรมที่ใช้ในระดับสูง ทำให้การปรับเปลี่ยนอัลกอริทึมในการประมวลผลภาพทำได้ยุ่งยาก

จากปัญหาในการพัฒนาระบบการมองเห็นด้วยคอมพิวเตอร์ดังกล่าวข้างต้น การพัฒนาเครื่องมือที่ช่วยลดปัญหาดังกล่าวจึงจัดได้ว่ามีความสำคัญ เพื่อให้ผู้พัฒนาศาสตร์ด้านนี้มุ่งความ

สนใจไปกับการคิดค้นอัลกอริทึมเพียงอย่างเดียว และช่วยให้การนำศาสตร์ทางด้านระบบการมองเห็นของคอมพิวเตอร์ไปประยุกต์ใช้ในงานระดับต่าง ๆ เป็นไปอย่างสะดวกยิ่งขึ้น ทำให้เกิดความจำเป็นในการสร้างซอฟต์แวร์ประยุกต์โดยเฉพาะขึ้น โดยจะดำเนินการออกแบบเพื่อให้รองรับความต้องการ ได้แก่ *การใช้งานที่ง่าย* ช่วยลดเวลาในการพัฒนาระบบการมองเห็น สามารถรับข้อมูลภาพจากภายนอก จัดเตรียมการประมวลผลและการแสดงผลภาพไว้ให้ *มีความยืดหยุ่น* สามารถใช้งานร่วมกับชุดพัฒนาการประมวลผลภาพที่มีอยู่ในปัจจุบันได้ *สามารถขยายระบบได้* สามารถเพิ่มความสามาถในการประมวลผลภาพโดยสร้างชุดการประมวลผลภาพเพิ่มเติมได้โดยง่าย

ในงานวิจัยนี้ได้มุ่งความสนใจกับการพัฒนาเครื่องมือสำหรับช่วยในการสร้างต้นแบบอย่างรวดเร็วของระบบการมองเห็นของคอมพิวเตอร์ โดยคำนึงถึงวัตถุประสงค์ที่ว่าการพัฒนาระบบการมองเห็นของคอมพิวเตอร์ต้องทำได้โดยง่าย สามารถเรียนรู้ได้ในระยะเวลาอันสั้น เครื่องมือที่พัฒนาขึ้นนี้ประกอบด้วยซอฟต์แวร์พร้อมกับองค์ประกอบในการประมวลผลภาพพื้นฐาน และต้นรหัสสำหรับการสร้างองค์ประกอบเพิ่มเติม โดยเรียกซอฟต์แวร์นี้ว่า VisBuilder ซึ่งมีส่วนต่อประสานกราฟิกกับผู้ใช้ที่ทำงานในลักษณะของการลากและวาง (drag and drop) ผู้ใช้สามารถกำหนดคุณสมบัติของระบบได้ตามต้องการ เช่น คุณสมบัติของแต่ละองค์ประกอบในการประมวลผลภาพ การเชื่อมโยงแต่ละองค์ประกอบย่อยเข้าด้วยกัน เป็นต้น

1.2 วัตถุประสงค์การวิจัย

1. เพื่อพัฒนาซอฟท์แวร์ที่ช่วยในการสร้างระบบการมองเห็นของคอมพิวเตอร์ได้อย่างง่ายดายและมีประสิทธิภาพ (หมายเหตุ ประสิทธิภาพในแง่ของการสร้างระบบ ไม่ใช่ประสิทธิภาพของอัลกอริทึมการประมวลผลภาพ)
2. เพื่อพัฒนาระบบการมองเห็นของคอมพิวเตอร์พื้นฐานต้นแบบ ที่มีองค์ประกอบฮาร์ดแวร์ที่จำเป็นและใช้ซอฟท์แวร์ในการสร้างงานระบบการมองเห็นของคอมพิวเตอร์พื้นฐานได้โดยใช้เพียงอุปกรณ์ราคาถูกและผู้ใช้ไม่จำเป็นต้องใช้ความรู้ในการเขียนโปรแกรมแต่อย่างใด
3. เพื่อพัฒนาโครงสร้างการเชื่อมต่อและขยายระบบของซอฟท์แวร์ สำหรับผู้ใช้ที่ต้องการพัฒนาองค์ประกอบของระบบการมองเห็นด้วยตนเอง โดยสามารถเพิ่มการเชื่อมต่อได้ทั้งในด้านซอฟท์แวร์ (เพิ่มชุดการประมวลผล) และฮาร์ดแวร์ (เพิ่มการควบคุมอุปกรณ์ภายนอกระบบ เช่นขับเคลื่อนมอเตอร์ที่ต่ออยู่กับกล้องรับภาพให้เคลื่อนที่เพื่อจับภาพของวัตถุที่ต้องการ)

1.3 ประโยชน์ที่คาดว่าจะได้รับ

ซอฟต์แวร์ที่ได้จากงานวิจัยนี้คาดว่าจะสามารถแก้ปัญหาในการดำเนินงานของหน่วยงานหรือบุคคลใด ๆ ที่ทำการสร้างหรือวิจัยระบบการมองเห็นของคอมพิวเตอร์ได้ เนื่องจากผู้ใช้ไม่จำเป็นต้องเสียเวลาในการศึกษาการเขียนซอฟต์แวร์ แต่สามารถสร้างระบบการมองเห็นได้ทันทีหรือสามารถขยายความสามารถของระบบด้วยการพัฒนาองค์ประกอบการมองเห็นเพิ่มเติมได้อย่างง่ายดาย สามารถนำเอาระบบที่พัฒนาไปประยุกต์ใช้งานได้จริงจากความสามารถในการเชื่อมต่อกับอุปกรณ์ภายนอกระบบได้

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะได้กล่าวถึงการสำรวจซอฟต์แวร์และงานที่เกี่ยวข้องกับงานพัฒนาระบบการมองเห็นของคอมพิวเตอร์ที่ผ่านมามีการวิจัยพัฒนาเครื่องมือหรือซอฟต์แวร์สำหรับงานด้านนี้อย่างมากซึ่งไม่สามารถนำมากล่าวถึงได้ทั้งหมดแต่จะเน้นไปที่เครื่องมือหรือซอฟต์แวร์ที่มีวัตถุประสงค์เพื่องานพัฒนาระบบการมองเห็นของคอมพิวเตอร์โดยทั่วไป ไม่มีความจำเพาะเจาะจงกับงานลักษณะใดลักษณะหนึ่ง และมีความสามารถเป็นไปตามการวิเคราะห์ความต้องการของระบบดังจะได้กล่าวต่อไปในบทที่ 3

2.1 การโปรแกรมเชิงภาพ (Visual Programming)

การโปรแกรมเชิงภาพ เป็นภาษาโปรแกรมที่ใช้รูปภาพหรือสัญลักษณ์แทนการเขียนด้วยตัวอักษรเหมือนภาษาโปรแกรมปกติทั่วไป โปรแกรมย่อย (subroutine) หรือฟังก์ชันต่าง ๆ จะแทนด้วยบล็อก (block) หรือด้วยไอคอน (icon) และใช้เส้นเชื่อมต่อระหว่างบล็อกแทนการไหลของข้อมูลระหว่างโปรแกรมย่อยนั้น ๆ การโปรแกรมมีลักษณะคล้ายกับการเขียนผังงาน (flow chart) รูปแบบคำสั่งที่ยุ่งยากจะถูกแทนที่ด้วยรูปภาพไอคอนทำให้จดจำและทำความเข้าใจได้ง่าย ปัจจุบันมีเครื่องมือที่มีลักษณะการโปรแกรมเชิงภาพหลายชนิด เช่น โปรแกรม LabVIEW (Instruments, 2010) โปรแกรม cAgilent-VEE (Technologies, 2010) โปรแกรม MATLABSimulink (MathWorks, 2010) โปรแกรม VisiQuest (Pegasus, 2010) โปรแกรม OpenDX (IBM, 2010) โปรแกรม ViPEr (Sanner et al., 2002) และ โปรแกรม NeatVision (VisionSystemsGroup, 2010) เป็นต้น

ซอฟต์แวร์การโปรแกรมเชิงภาพที่นิยมใช้กันอย่างแพร่หลาย คือ LabVIEW โดยเฉพาะเมื่อต้องการเชื่อมต่อกับอุปกรณ์ภายนอกต่าง ๆ ภาษาโปรแกรมของ LabVIEW เรียกว่า ภาษา G (Wikipedia, 2010) ซึ่งใช้สำหรับการสร้างแผนผังการทำงาน มีความสามารถเชื่อมต่อกับอุปกรณ์ภายนอกได้โดยตรง

ซอฟต์แวร์ที่ทำงานคล้ายกับ LabVIEW มีลักษณะการใช้งานเช่นเดียวกัน คือ Agilent-VEE สามารถทำงานผ่านบรรทัดคำสั่ง (command-line) ได้ และมีโครงสร้างข้อมูลที่ยืดหยุ่น

Simulink เป็นชุดโปรแกรมเสริมของ MATLAB นิยมใช้ในงานด้านระบบควบคุมสำหรับวิเคราะห์หาผลตอบสนองของระบบ ใช้การแทนฟังก์ชันต่าง ๆ ด้วยบล็อก สามารถปรับเปลี่ยน

หรือวิเคราะห์หาค่าจากบล็อกไดอะแกรมได้โดยตรง ทำให้ลดความยุ่งยากในการเขียนโปรแกรมในลักษณะเป็นคำสั่ง สามารถคำนวณหาค่าสมการต่าง ๆ ที่มีความซับซ้อน

OpenDX ย่อมาจาก Open Data Explorer เป็นซอฟต์แวร์สำหรับการนำเสนอข้อมูลทางด้านวิทยาศาสตร์มีการใช้งานผ่านการโปรแกรมเชิงภาพ ใช้สำหรับการแสดงผลข้อมูลเป็นภาพเพื่อให้สื่อความหมายและเข้าใจได้ง่าย สามารถแสดงข้อมูลในรูปแบบสามมิติ ปรับเปลี่ยนสีได้ตามความต้องการ

ซอฟต์แวร์สำหรับการวิเคราะห์ข้อมูลและประมวลผลภาพ ได้แก่ NeatVision และ VisiQuest มีลักษณะการทำงานคล้ายกัน คือ เป็นลักษณะของการโปรแกรมเชิงภาพ

2.2 กระแสข้อมูล (Dataflow)

การเขียนโปรแกรมโดยอาศัยหลักการไหลของข้อมูล เป็นรูปแบบการส่งผ่านข้อมูลระหว่างองค์ประกอบที่เชื่อมต่อกันเพื่อทำการประมวลผล โดยพิจารณาเหมือนข้อมูลเป็นก้อนเดียวกันที่ไหลผ่านองค์ประกอบการประมวลผลที่เชื่อมต่อกัน ตัวอย่างงานประมวลผลที่ใช้หลักการดังกล่าว เช่น การประมวลผลสัญญาณในเวลาจริง การประมวลผลภาพ เป็นต้น ประโยชน์ของการโปรแกรมโดยอาศัยหลักการไหลของข้อมูลมีมากมายอาทิเช่น

ประโยชน์ที่ช่วยให้การโปรแกรมทำได้ง่ายขึ้น เมื่อพัฒนาองค์ประกอบการประมวลผลมีสิ่งที่ต้องพิจารณามีเพียงข้อมูลที่เข้ามาเท่านั้น ไม่ต้องสนที่มาของข้อมูลว่ามาจากส่วนในของโปรแกรม ส่งผลให้องค์ประกอบการประมวลผลที่พัฒนาขึ้นสามารถปรับเปลี่ยนการทำงานได้ง่ายและยังนำมาใช้ใหม่ได้ง่ายอีกด้วย เช่น เมื่อพัฒนาองค์ประกอบการประมวลผลภาพจะสนใจเฉพาะข้อมูลภาพที่เข้ามา ไม่สนใจที่มาของภาพว่าภาพมาจากที่ใดของโปรแกรม ไม่ว่าจะมาจากกล้องรับภาพ ไฟล์ภาพ หรืออื่น ๆ สิ่งที่องค์ประกอบต้องทำมีเพียงประมวลผลข้อมูลภาพที่เข้ามาเท่านั้น

ประโยชน์ในแง่ของความยืดหยุ่นในการพัฒนา เช่น ในกรณีการประมวลผลภาพจากกล้องรับภาพ เมื่อไม่ต้องการการประมวลผลภาพใด ๆ สามารถผ่านข้อมูลภาพจากกล้องไปที่การแสดงผลโดยตรงได้หรือต้องการเปลี่ยนจากการรับข้อมูลจากกล้องไปรับจากไฟล์ภาพแทนสามารถทำได้ง่ายเหมือนการการนำสายสัญญาณเข้าไปเชื่อมต่อกับแหล่งที่มาอื่น ๆ โดยไม่ต้องปรับเปลี่ยนการทำงานภายในองค์ประกอบเลย

ประโยชน์ในแง่ของการกระจายการประมวลผล สามารถแยกองค์ประกอบไปทำงานที่หน่วยประมวลผลอื่น หรือเครื่องคอมพิวเตอร์เครื่องอื่นได้ง่าย เนื่องจากการส่งข้อมูลสามารถทำได้

และมองเห็นได้ง่าย เช่น หากต้องการส่งข้อมูลภาพไปแสดงผลที่เครื่องคอมพิวเตอร์เครื่องอื่น ๆ สามารถส่งข้อมูลภาพดังกล่าวผ่านระบบเครือข่ายไปที่เครื่องคอมพิวเตอร์โดยตรง

ซอฟต์แวร์ที่ใช้การโปรแกรมเชิงภาพจะใช้หลักการไหลของข้อมูลในการส่งข้อมูลระหว่างบล็อกต่าง ๆ ทิศทางการไหลนั้นจะถูกกำหนดด้วยเส้นเชื่อมระหว่างบล็อกซึ่งแสดงถึงเส้นทางการเดินของข้อมูล โดยกำหนดว่าข้อมูลจะไหลผ่านส่วนการประมวลผลหรือการคำนวณใดบ้าง และจะให้แสดงผลอย่างไร

2.3 โปรแกรมเชิงภาพสำหรับงานพัฒนาระบบการมองเห็นของคอมพิวเตอร์

การวิจัยพัฒนาเครื่องมือที่เหมาะสมกับงานพัฒนาระบบการมองเห็นของคอมพิวเตอร์มีมาอย่างต่อเนื่อง ในที่นี้จะกล่าวถึงเฉพาะงานวิจัยที่พัฒนาซอฟต์แวร์ที่มีการทำงานในลักษณะการโปรแกรมเชิงภาพเท่านั้น และเป็นงานวิจัยที่ให้ซอฟต์แวร์ที่ทำงานได้จริงและซอฟต์แวร์ดังกล่าวยังคงอยู่

Hunt (1992) ได้พัฒนาซอฟต์แวร์ชื่อ IDF (iconic data flow) สำหรับใช้พัฒนาอัลกอริทึมการประมวลผลภาพและระบบการมองเห็นของคอมพิวเตอร์ โดยมุ่งเน้นให้สามารถโปรแกรมได้ง่ายเหมือนการสร้างแผนภูมิภาพ (flow chart)

Koelma et al. (1992) ได้ทำการพัฒนาสภาวะแวดล้อมของการโปรแกรมเชิงภาพที่ชื่อ SCIL-VP โดยใช้หลักการไหลของข้อมูลและสภาวะแวดล้อมแบบ SCIL (standard c interpretive language) ซึ่งมีความสามารถในการโปรแกรมได้ในหลายระดับ SCIL พัฒนาบนพื้นฐานของตัวแปลภาษาซี (c interpreter) ซึ่งมีความยืดหยุ่นในการพัฒนาสูง SCIL-VP รวมความสามารถในการโปรแกรมเชิงภาพและความสามารถของตัวแปลภาษาซีเข้าด้วยกัน

Konstantios and Rasure (1994) ได้พัฒนาซอฟต์แวร์ชื่อ Khoros ภายหลังเปลี่ยนชื่อเป็น VisiQuest เพื่อให้เป็นซอฟต์แวร์เชิงพาณิชย์ มีความสามารถในการประมวลผลภาพและมีความสามารถในการกระจายการมวลผลผ่านระบบเครือข่ายหน่วยประมวลผลย่อยจะแทนด้วยภาพเรียกว่า glyph สำหรับให้ผู้ใช้นำมาเชื่อมต่อกันเพื่อพัฒนาการประมวลผลภาพ

Sim and Park (1996) ได้พัฒนาซอฟต์แวร์ที่ชื่อ VPEDIP ทำงานในลักษณะของการโปรแกรมเชิงภาพสำหรับการประมวลผลสัญญาณภาพ และการพัฒนาระบบการมองเห็นของคอมพิวเตอร์ซอฟต์แวร์นี้ใช้หลักการไหลของข้อมูลในการทำงานสามารถทำการประมวลผลในแบบขนานโดยกระจายการทำงานไปยังเครื่องคอมพิวเตอร์แม่ข่ายโดยอาศัยการเชื่อมต่อด้วยวิธี RPC (remote procedure call) ประสิทธิภาพการทำงานของซอฟต์แวร์นี้จะเพิ่มขึ้นหากมีจำนวนเครื่องคอมพิวเตอร์แม่ข่ายที่มากขึ้น

Whelan and Ghita (2004) พัฒนาซอฟต์แวร์ชื่อ NeatVision สำหรับการวิเคราะห์ข้อมูลภาพขนาดใหญ่ในด้านการแพทย์สร้างขึ้นโดยใช้ภาษาจาวา จึงใช้งานบนระบบปฏิบัติการได้หลายแบบ ทำงานในลักษณะการโปรแกรมเชิงภาพผู้ใช้สามารสร้างโมดูลการประมวลผลเพิ่มเองได้

Lourens (2004) ได้พัฒนาซอฟต์แวร์ชื่อ TiViPE มีลักษณะคล้ายโปรแกรม LabVIEW (Instruments, 2003) และ NeatVision (Whelan and Sadleir, 2004) แต่แตกต่างกันที่ TiViPE สร้างขึ้นเพื่อครอบโมดูลหรือต้นรหัสที่อยู่แล้วซึ่งอาจพัฒนาด้วยภาษาซี ภาษาจาวาหรือภาษาฟอร์แทรน ซึ่งช่วยให้โปรแกรมได้ง่ายขึ้นมากกว่าการเขียนต้นรหัสด้วยตนเอง

2.4 การแบ่งประเภทของซอฟต์แวร์ตามวิธีการใช้งาน

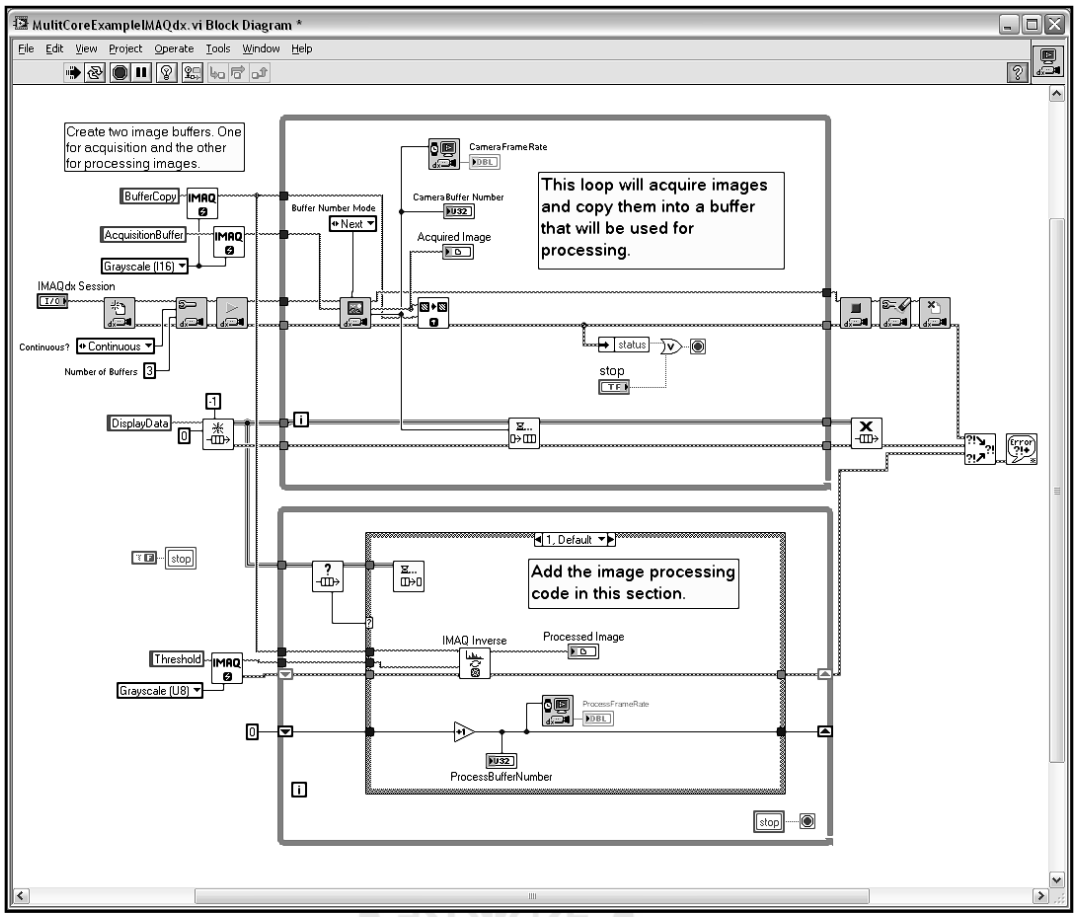
เครื่องมือหรือซอฟต์แวร์สำหรับงานพัฒนาระบบการมองเห็นที่มีอยู่ในปัจจุบันมีมากมายหลายตัว ในที่นี้จะได้จำแนกตามลักษณะวิธีการนำมาใช้งาน ซึ่งสามารถแบ่งได้เป็นสองประเภทคือ ชนิดที่สนับสนุนการโปรแกรมด้วยภาพ และชนิดที่ต้องเขียนรหัสโปรแกรมด้วยตัวเองทั้งหมด ซอฟต์แวร์และชุดพัฒนาที่จะกล่าวถึงทั้งหมดนี้ได้เลือกเฉพาะที่นิยมใช้กันอย่างแพร่หลาย และมีการสนับสนุนที่ดีคือ มีคู่มือ มีแหล่งให้ความช่วยเหลือต่าง ๆ

2.4.1 ซอฟต์แวร์ชนิดโปรแกรมเชิงภาพ

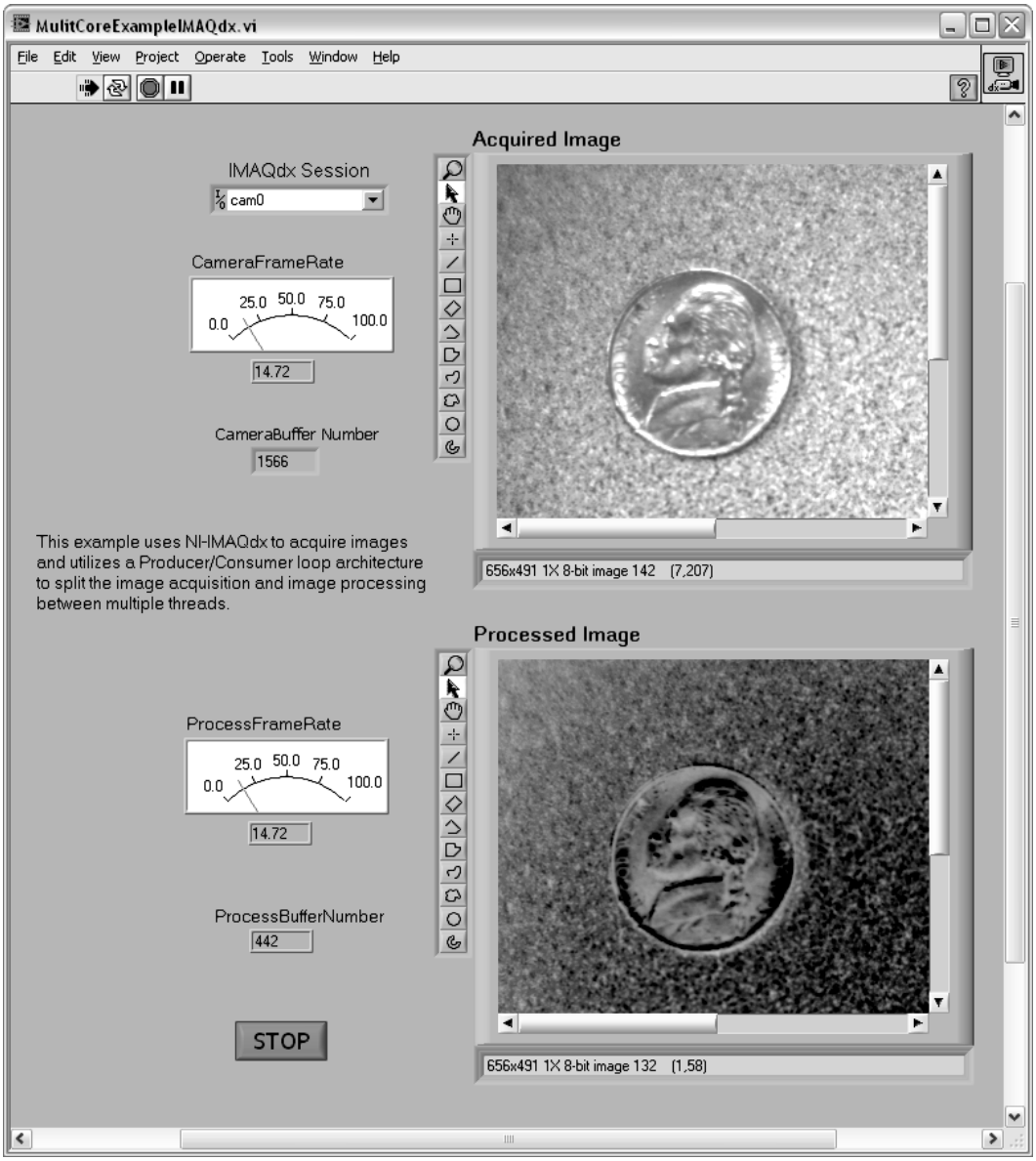
ซอฟต์แวร์ที่จัดอยู่ในกลุ่มนี้มีลักษณะการใช้งานผ่านส่วนต่อประสานกราฟิกกับผู้ใช้มีการแทนฟังก์ชันต่าง ๆ ด้วยภาพหรือไอคอนให้ผู้ใช้ลากและวาง (drag and drop) เพื่อเชื่อมต่อการทำงาน และใช้หลักการไหลของข้อมูลในการสื่อสารระหว่างฟังก์ชัน

2.4.1.1 NI Vision Software

NI Vision Software (Instruments, 2005) เป็นส่วนขยายความสามารถของโปรแกรม LabVIEW พัฒนาโดยบริษัท National Instruments (NI) สำหรับงานด้านการพัฒนาระบบการมองเห็นของคอมพิวเตอร์และการประมวลผลภาพ เป็นซอฟต์แวร์ที่นิยมใช้กันอย่างแพร่หลายและถือเป็นซอฟต์แวร์ชั้นแนวหน้าในด้านนี้ ใช้กันมากในระดับโรงงานอุตสาหกรรมสำหรับงานทางด้านการวัด การทดสอบ และการควบคุม มีฟังก์ชันการทำงานที่เพียบพร้อม เหมาะกับงานที่ต้องการติดต่อกับเครื่องมือวัดหรืออุปกรณ์รับภาพต่าง ๆ ที่เป็นของบริษัท NI โดยเฉพาะส่วนการติดต่อกับอุปกรณ์อื่นที่ไม่ได้ผลิตจากบริษัท NI ก็สามารถทำได้แต่ไม่ง่ายนัก การทำงานของซอฟต์แวร์นี้จะแยกเป็นสองส่วน คือ ส่วนแรกสำหรับการพัฒนาโปรแกรมดังรูปที่ 2.1 และส่วนที่สองสำหรับการแสดงผลและติดต่อกับผู้ใช้ดังรูปที่ 2.2



รูปที่ 2.1 ภาพการทำงานของ NI Vision ส่วน Block Diagram (ที่มา http://zone.ni.com/devzone/cda/epd/p/id/5859)



รูปที่ 2.2 ภาพการทำงานของ NI Vision ส่วน Font Panel (ที่มา:

http://zone.ni.com/devzone/cda/epd/p/id/5859)

2.4.1.2 VisiQuest

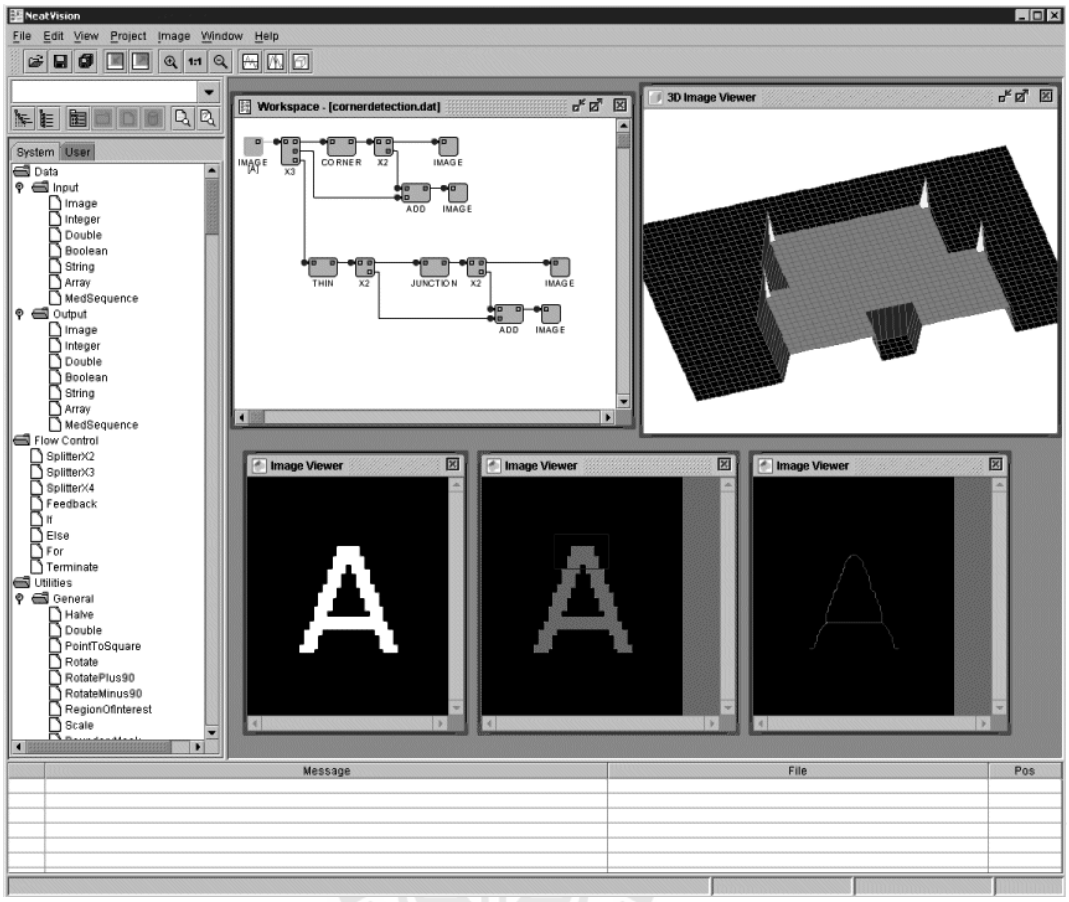
VisiQuest เป็นซอฟต์แวร์ที่เปลี่ยนชื่อมาจาก Khoros ซึ่งเป็นการผันตัวเองจากซอฟต์แวร์ในระดับงานวิจัยสู่ซอฟต์แวร์เชิงพาณิชย์ พัฒนาขึ้นโดยบริษัท AccuSoft (2010) สำหรับการวิเคราะห์ข้อมูลและประมวลผลภาพที่ทำงานในลักษณะของการโปรแกรมเชิงภาพ ทำงานได้บนระบบปฏิบัติการหลายแบบตั้งแต่ วินโดวส์ แมค จนถึงลินุกซ์ รองรับการใช้งานได้หลากหลายโดยผ่านหน่วยประมวลผลย่อยที่แทนด้วยไอคอน เรียกว่า glyph การเพิ่มเติมความสามารถในการประมวลผลหรือเป็นการสร้าง glyph เพิ่มทำได้โดยมีชุดพัฒนาให้สามารถเชื่อมต่อกับ MATLAB ได้

2.4.1.3 WiT

WiT เป็นซอฟต์แวร์เชิงพาณิชย์สำหรับงานด้านการประมวลผลภาพโดยเฉพาะ มีฟังก์ชันที่จำเป็นในการวิเคราะห์ภาพและมีความสามารถในการกระจายการทำงานผ่านระบบเครือข่ายได้ พัฒนาโดยบริษัท DALSA Digital Imaging (2003)

2.4.1.4 NeatVision

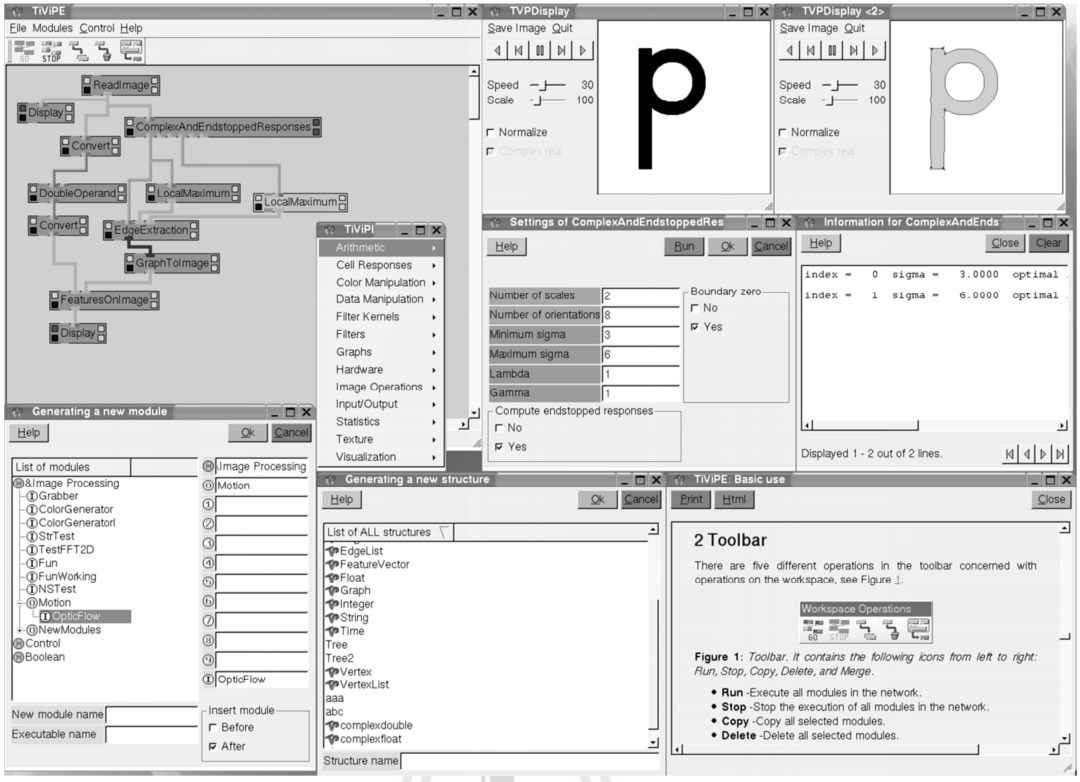
NeatVision เป็นซอฟต์แวร์ที่พัฒนาขึ้นโดย Vision Systems Group: VSG (2010) ซึ่งเป็นกลุ่มวิจัยทางด้านการประมวลผลและวิเคราะห์สัญญาณภาพของมหาวิทยาลัย Dublin City ซอฟต์แวร์นี้ใช้ภาษาจาวาในการพัฒนา มีการแบ่งเป็นสองเวอร์ชัน คือ standard และ developer เมื่อเวอร์ชัน standard มีลักษณะเป็นซอฟต์แวร์เสรี (free software) คือ ไม่มีการเปิดเผยต้นรหัสแต่สามารถนำมาใช้ได้โดยไม่เสียค่าใช้จ่ายใด ๆ ส่วนเวอร์ชัน developer จะต้องเสียค่าใช้จ่ายเหมาะสำหรับผู้ใช้ที่ต้องการพัฒนาการประมวลผลเพิ่มเติมจากที่มีมาให้การทำงานของซอฟต์แวร์แสดงดังรูปที่ 2.3



รูปที่ 2.3 การทำงานของ NeatVision (ที่มา: http://neatvision.eeng.dcu.ie/)

2.4.1.5 TiViPE

TiViPE (Tino's Visual Programming Environment) เป็นซอฟต์แวร์ชนิดเปิดต้นรหัสใช้สัญญาอนุญาตแบบ GPL (GNU general public licence) พัฒนาด้วยภาษา C++ (Lourens, 2006) สามารถใช้กับงานด้านการประมวลผลภาพได้เป็นอย่างดีใช้การโปรแกรมผ่านกราฟิกโดยการสร้างไอคอนแทนโมดูลหรือฟังก์ชันที่เขียนด้วยภาษาซีจาวาหรือฟอร์แทรนก็ได้โดยไม่ต้องแก้ไขโมดูลนั้นใหม่แต่อย่างใด สามารถนำมาใช้ได้โดยตรง TiViPE สามารถสร้างโปรแกรมที่ทำงานด้วยตัวเองโดยไม่ผ่าน TiViPE ได้การทำงานของซอฟต์แวร์แสดงดังรูปที่ 2.4



รูปที่ 2.4 การทำงานของ TiViPE (ที่มา: http://www.dei.brain.riken.jp/)

2.4.1.6 SCIL-Image

SCIL-Image (Balen and Smeulders, 1994) เป็นซอฟต์แวร์เชิงพาณิชย์ที่พัฒนาภายใต้การสนับสนุนจากกลุ่มความร่วมมือ CBP (centre of image processing and pattern recognition) สำหรับการประมวลผลภาพโดยตรง การใช้งานสามารถทำได้ในหลายระดับ เช่น ผู้ใช้สามารถเขียนต้นรหัสลงไปได้โดยตรงซึ่งใช้โครงสร้างภาษาคล้ายภาษาซีหรือใช้การโปรแกรมเชิงภาพก็ได้ เป็นต้น

2.4.2 ซอฟต์แวร์ชนิดที่ต้องเขียนโค้ดด้วยตัวเอง

ซอฟต์แวร์ที่จัดอยู่ในกลุ่มนี้มีลักษณะการใช้งานผ่านการเขียนรหัสโปรแกรมด้วยตนเองทุกขั้นตอนตั้งแต่การรับภาพ การประมวลผล การจัดการข้อมูล และการแสดงผลภาพจะต้องดำเนินการด้วยตัวผู้ใช้เอง รวมทั้งการคอมไพล์ต้นรหัสด้วย

2.4.2.1 OpenCV

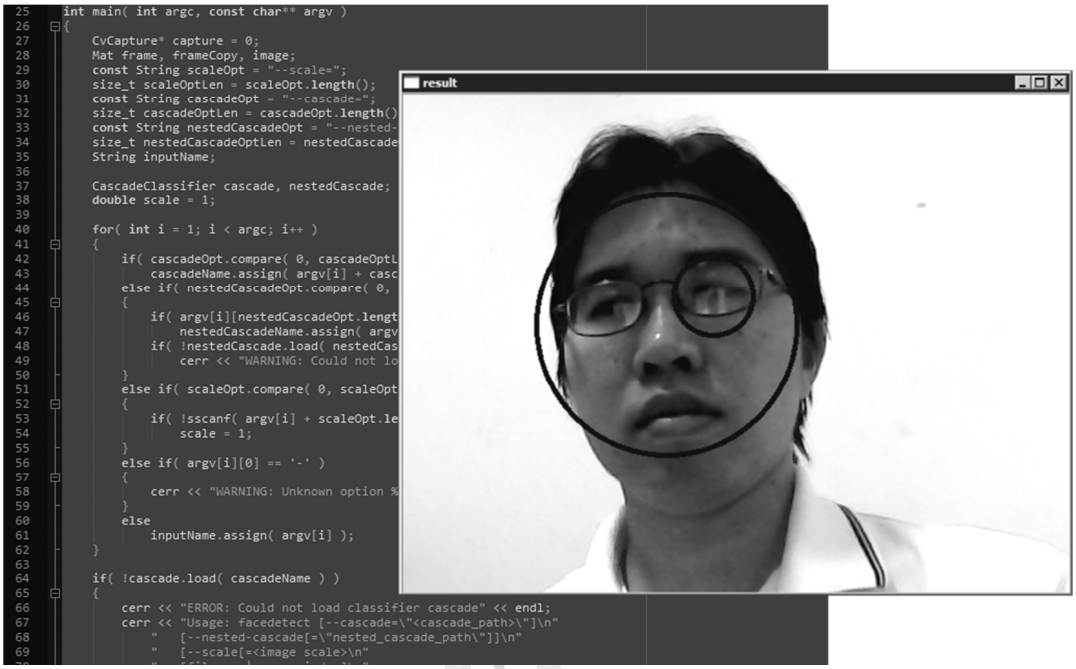
OpenCV พัฒนาขึ้นโดยได้รับการสนับสนุนจากบริษัท Intel Corporation (2010) เป็นซอฟต์แวร์แบบเปิดเผยต้นรหัสสำหรับใช้ในการประมวลผลภาพ สามารถนำไปต่อยอดพัฒนาโปรแกรมต่าง ๆ ได้ง่าย ใช้งานบนระบบปฏิบัติการได้หลายแบบและสามารถพัฒนาโปรแกรมได้หลากหลายภาษา

OpenCV มีฟังก์ชันสำหรับการพัฒนาโปรแกรมทางด้านระบบการมองเห็นของคอมพิวเตอร์ สามารถประมวลผลภาพดิจิตอลได้ทั้งภาพนิ่ง และภาพเคลื่อนไหว มีฟังก์ชันสำเร็จรูปสำหรับจัดการข้อมูลภาพ และการประมวลผลภาพพื้นฐาน เช่น การหาขอบภาพ การกรองข้อมูลภาพ เป็นต้น

การใช้งานฟังก์ชันต่าง ๆ ของ OpenCV จะต้องมีการเขียนโปรแกรมที่เรียกไฟล์ส่วนหัว (header file) เพื่อเข้าถึงไลบรารี่หรือฟังก์ชันต่าง ๆ ที่บรรจุอยู่ในไฟล์ DLL (dynamic link library)

โครงสร้างหลักประกอบด้วย 4 ส่วน คือ (1) CxCore ประกอบด้วยโครงสร้างข้อมูลต่าง ๆ ได้แก่ tree list queue sequence image และฟังก์ชันสำหรับจัดการข้อมูลดังกล่าว (2) CvReference ประกอบด้วยฟังก์ชันสำหรับการวิเคราะห์และประมวลผลภาพ (3) CvAux ประกอบด้วยฟังก์ชันที่อยู่ในระหว่างการพัฒนาหรือทดสอบ รวมถึงฟังก์ชันที่เลิกใช้แล้ว (4) High-GUI สำหรับการแสดงผล การรับภาพและการบันทึกไฟล์ภาพ

OpenCV เป็นชุดพัฒนาที่มีประโยชน์มากสำหรับงานพัฒนาระบบการมองเห็นของคอมพิวเตอร์งานวิจัยนี้ได้นำชุดพัฒนานี้มาใช้เป็นส่วนหลักในการพัฒนาโดยใช้สำหรับจัดการโครงสร้างข้อมูลภายในและฟังก์ชันการประมวลผลภาพต่าง ๆ จะกล่าวถึงในบทที่ 4



รูปที่ 2.5 ตัวอย่างการใช้ทำงาน OpenCV

2.4.2.2 Integrating Vision Toolkit (IVT)

IVT (Integrating Vision Toolkit) เป็นซอฟต์แวร์ที่พัฒนาขึ้นเพื่อการพัฒนาระบบการมองเห็นของคอมพิวเตอร์โดยเฉพาะมีประสิทธิภาพสูงและทำงานได้รวดเร็ว สร้างขึ้นโดยใช้ภาษา C++ ในการพัฒนารูปแบบของการเปิดต้นรหัสสามารถทำงานได้กับระบบปฏิบัติการหลายแบบ (Azad, 2010)

เมื่อเปรียบเทียบกับ OpenCV แล้วการใช้งาน IVT ทำได้ง่ายกว่า เนื่องจากพัฒนาในลักษณะของการโปรแกรมเชิงวัตถุทำให้การทำความเข้าและอ่านรหัสโปรแกรมได้ง่าย การพัฒนาระบบการมองเห็นของคอมพิวเตอร์ทำได้ง่ายและรวดเร็วกว่าการใช้ OpenCV

2.4.2.3 GraphicsMagick

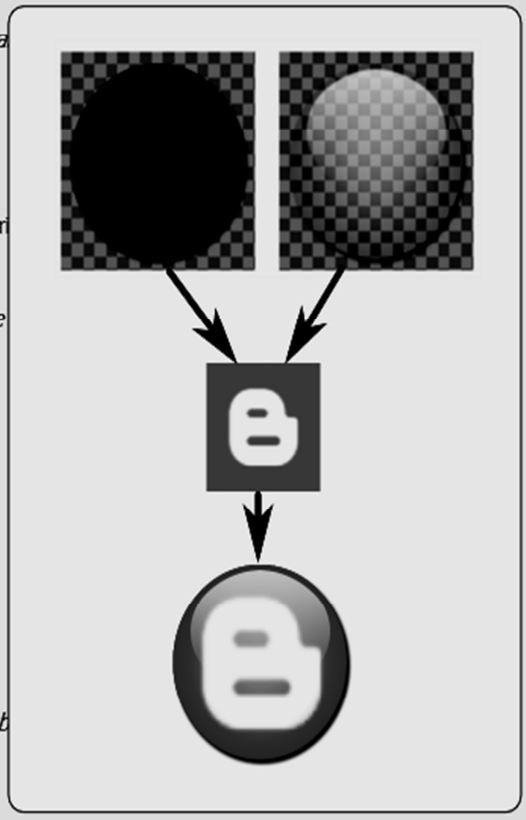
GraphicsMagick (GraphicsMagick, 2010) แยกการพัฒนาออกมาจาก ImageMagick (ImageMagick, 2010) โดยเน้นที่ความง่ายในการใช้งานของ API (application programming interface) และความง่ายในการใช้งานผ่าน command-line

GraphicsMagick สามารถทำงานกับระบบปฏิบัติการได้หลากหลายรองรับการโปรแกรมด้วยภาษาหลายชนิด ได้แก่ C C++ Perl PHP Tcl และ Ruby เหมาะกับการใช้ในการประมวลผลภาพมากกว่าการใช้งานด้านการพัฒนาระบบการมองเห็นของคอมพิวเตอร์

2.4.2.4 Python Image Library (PIL)

PIL (PythonImageLibrary) เป็นส่วนขยายความสามารถของภาษาโปรแกรม Python โดย PythonWare (2010) สำหรับใช้ในงานด้านการประมวลผลภาพ รองรับไฟล์ภาพหลายชนิดไม่มีฟังก์ชันสำหรับการใช้พัฒนาระบบการมองเห็นของคอมพิวเตอร์โดยตรง ตัวอย่างการใช้งานแสดงรูปที่ 2.6 เป็นการสร้างปุ่มจากการรวมภาพหลายภาพเข้าด้วยกัน

```
highlight = Image.open('round.png')
mask = Image.open('round-mask.png')
# Read the icon and convert it into 'RGBA' in case it wa
icon = Image.open('blogger.png').convert('RGBA')
button = Image.new('RGBA', mask.size)

# Resize Icon
icon = ImageOps.fit(
  icon, highlight.size, method=Image.ANTIALIAS, center
)

# Create a helper image that will hold the icon after the
helper = button.copy()
# Cut the icon by the shape of the mask
helper.paste(icon, mask=mask)

# Fill with a solid color by the mask's shape
button.paste((255, 255, 255), mask=mask)
# Get rid of the icon's alpha band
icon = icon.convert('RGB')
# Paste the icon on the solid background
# Note we are using the reshaped icon as a mask
button.paste(icon, mask=helper)

# Get a copy of the highlight image without the alpha b
overlay = highlight.copy().convert('RGB')
button.paste(overlay, mask=highlight)

button.save('button.png')
```



รูปที่ 2.6 ตัวอย่างการใช้งาน PIL (ที่มา: http://nadiana.com/)

2.4.2.5 Adobe Generic Image Library (GIL)

GIL (Bourdev and Jin, 2010) เป็นซอฟต์แวร์ที่ออกแบบเพื่อช่วยพัฒนาอัลกอริทึมการประมวลผลภาพโดยเฉพาะ เน้นที่การพัฒนาอัลกอริทึมให้ใช้ได้กับข้อมูลภาพที่หลากหลายเนื่องจากข้อมูลภาพมีความแตกต่างกันทั้งชนิดของชุดสี (color space) จำนวนบิตที่ใช้

แทนสี (bit depth) หรืออื่น ๆ ดังนั้นจึงเป็นการยากที่จะเขียนโปรแกรมให้ใช้ได้กับข้อมูลภาพที่หลากหลายดังกล่าว GIL จะช่วยให้การเขียนโปรแกรมทำได้ง่าย โดยจัดการเรื่องความแตกต่างของข้อมูลภาพให้ดังนั้นการเขียนต้นรหัสสำหรับการประมวลผลภาพเพียงครั้งเดียวสามารถนำไปใช้ได้กับข้อมูลภาพที่แตกต่างกันได้

GIL ไม่ได้ออกแบบมาเพื่อใช้กับการพัฒนาระบบการมองเห็นของคอมพิวเตอร์จึงไม่มีฟังก์ชันการทำงานด้านนี้ให้แต่สามารถนำไปใช้ร่วมในการพัฒนาได้ GIL สามารถใช้ร่วมกับ OpenCV ได้ช่วยให้การเขียนโปรแกรมทำได้ง่ายขึ้น ซอฟต์แวร์ของงานวิจัยนี้ได้ใช้ GIL ร่วมกับ OpenCV ในการเขียนอัลกอริทึมภายในด้วย

GIL (Josuttis, 1999) มีโครงสร้างคล้ายกลับ STL (standard template library) ของภาษา C++ จึงมีความยืดในการใช้งานสูงและชุดพัฒนานี้ยังได้รับการยอมรับจากกลุ่มผู้พัฒนา Boost C++ (Brinkman, 2006) ให้รวมเข้าไว้ด้วยกันตั้งแต่รุ่น 1.35.0 (C++, 2010) โดยมีลักษณะการใช้งานดังรูปที่ 2.7 เป็นตัวอย่างการคัดลอกข้อมูลพิกเซลของภาพ ทางด้านซ้ายเป็นวิธีการโปรแกรมโดยทั้งไปทางด้านขวาเป็นการใช้ GIL จะเห็นว่ารหัสโปรแกรมสั้นลงมากและสามารถเขียนได้ง่ายขึ้น

General Version	**GIL Version**

void copy_pixels(const rgb_pixel*src,
unsigned char* dst_red,
unsigned char* dst_green,
unsigned char* dst_blue,
int width, int height)
{
constrgb_pixel*src_end=src+width*height;
while (src !=src_end
{
*dst_red++=src->red;
*dst_green++=src->green;
*dst_blue++=src->blue;
}
}
	template<typename SrcView, typename DstView>
	void copy_pixels(const SrcView& src, const DstView&dst)
	{
	std::copy(src.begin(), src.end(), dst.begin());
	}

รูปที่ 2.7 ตัวอย่างต้นรหัสของ GIL

2.4.2.6 ITK

ITK (luis and the insight software consortium, 2005) ย่อมาจาก Insight Segmentation and Registration Toolkit เป็นชุดพัฒนาแบบเปิดต้นรหัสพัฒนาโดย National Library of Medicine ออกแบบมาเพื่องานด้านการวิเคราะภาพทางการแพทย์โดยเฉพาะ ไม่มีฟังก์ชันสำหรับงานด้านการพัฒนาระบบการมองเห็นของคอมพิวเตอร์จึงไม่เหมาะสมที่จะนำมาใช้

2.4.3 เปรียบเทียบคุณสมบัติของซอฟต์แวร์สำหรับการประมวลผลภาพ

ขอนำซอฟต์แวร์ที่กล่าวมาทั้งหมดมาเปรียบเทียบคุณสมบัติต่าง ๆ ไม่ว่าจะเป็นในแง่ของการโปรแกรมเชิงภาพ การเปิดเผยต้นรหัส การทำงานข้ามระบบปฏิบัติการ ความสามารถในการขยายระบบ การรองรับการใช้งานพัฒนาระบบการมองเห็นของคอมพิวเตอร์ดังแสดงในตารางที่ 2.1 จะเห็นได้ว่าซอฟต์แวร์ส่วนใหญ่ที่มีลักษณะของการโปรแกรมเชิงภาพและรองรับการพัฒนาระบบการมองเห็นของคอมพิวเตอร์จะเป็นซอฟต์แวร์ปิดและเป็นซอฟต์แวร์เชิงพาณิชย์ เช่น NI-Vision VisiQuest เป็นต้น ส่วนซอฟต์แวร์ที่เปิดต้นรหัสที่สามารถใช้พัฒนาระบบการมองเห็นของคอมพิวเตอร์ได้ คือ OpenCV และ IVT จะไม่มีลักษณะของการโปรแกรมเชิงภาพซึ่งทำให้ยากต่อการนำมาใช้งาน

2.5 สรุป

ซอฟต์แวร์สำหรับการพัฒนาระบบในลักษณะของการโปรแกรมเชิงภาพมีการพัฒนากันมาอย่างต่อเนื่องทั้งในส่วน ที่เป็นการพัฒนาในระดับงานวิจัย หรือในลักษณะซอฟต์แวร์เชิงพาณิชย์มีทั้งส่วนที่ออกแบบมาเพื่อการประมวล สัญญาณภาพโดยตรง เช่น VisiQuest NeatVision TiViPE และแบบที่เป็นซอฟต์แวร์ที่เพิ่มความสามารถการประมวลผลภาพเข้ามาภายหลัง เช่น NI Vision Software ถึงแม้จะมีการใช้งานที่ง่ายแต่ก็ทำให้ขาดอิสระในการพัฒนาเนื่องจากเป็นซอฟต์แวร์แบบปิดต้นรหัสทำให้การพัฒนาส่วนขยายเพิ่มเติมทำได้ยากหรือต้องเสียค่าใช้จ่ายเพิ่มเติม ส่วนซอฟต์แวร์ที่เปิดต้นรหัส เช่น OpenCV และ IVT ที่ถึงแม้จะมีความสามารถที่หลากหลายแต่การนำมาใช้ยังคงยากมากในแง่ของการโปรแกรม ซึ่งเป็นอุปสรรคสำคัญสำหรับผู้พัฒนาระบบการมองเห็นของคอมพิวเตอร์มีเพียงส่วนน้อยเท่านั้นที่มีความเชี่ยวชาญทั้งสองอย่างควบคู่กันไป ดังนั้นการพัฒนาซอฟต์แวร์ที่ช่วยลดปัญหาและความซับซ้อนต่าง ๆ ของการพัฒนาระบบการมองเห็นของคอมพิวเตอร์จึงควรพิจารณาเป็นอย่างยิ่ง

ตารางที่ 2.1 ตารางเปรียบเทียบคุณสมบัติของซอฟต์แวร์สำหรับการประมวลผลภาพ

Software	VPE	Open Source	Proprietary	Platfroms	Extensibility	Languages	Vision
NI Vision Software	•		•		•	•	•
VisiQuest	•		•	•	•		•
WiT	•		•				
NeatVision	•		•	•	•		•
TiViPE	•	•		•	•	•	•
ScilImage	•		•		•		•
OpenCV		•		•	•	•	•
IVT		•		•	•		•
GraphicsMagick		•		•	•		
PIL		•	•	•	•		
Adobe GIL		•		•	•		
ITK		•		•	•	•	

มหาวิทยาลัยเทคโนโลยีสุรนารี

บทที่ 3
แนวคิดและการออกแบบระบบ

3.1 แนวคิดในการพัฒนา

ในบทนี้จะได้กล่าวถึงความต้องการของระบบในการสร้างซอฟท์แวร์ประยุกต์สำหรับการพัฒนาระบบการมองเห็นของคอมพิวเตอร์การวิเคราะห์ความต้องการของระบบนี้อาศัยการพิจารณาจากการ ใช้งานระบบการมองเห็นในระดับห้องปฏิบัติการ โดยคำนึงถึงผู้ใช้งานและลักษณะการใช้งานเป็นสำคัญ

ระบบการมองเห็นของคอมพิวเตอร์ประกอบด้วยขั้นตอนสำคัญสามขั้นตอน ได้แก่การรับภาพ (capturing) การประมวลผลภาพ (processing) และการแสดงผลภาพ (display image data) ซอฟต์แวร์ที่พัฒนาขึ้นจะต้องรองรับการใช้งานดังกล่าวได้อย่างมีประสิทธิภาพ ความต้องการของระบบในที่นี้ประกอบด้วยความสามารถในการขยายระบบ ความสามารถในการพัฒนาเป็นระบบย่อย การรองรับรูปแบบไฟล์ภาพชนิดต่าง ๆ เป็นต้น ดังจะได้กล่าวรายละเอียดในลำดับต่อไป

ประสิทธิภาพของซอฟต์แวร์ที่กล่าวถึงในที่นี้นั้นไม่ได้หมายถึงความเร็วในการทำงานของซอฟต์แวร์แต่เป็นประสิทธิภาพในเชิงการใช้งาน กล่าวคือ สามารถช่วยให้ผู้ใช้งานสร้างระบบการมองเห็นด้วย คอมพิวเตอร์ได้อย่างรวดเร็ว สามารถสร้างงานวิจัยได้ในปริมาณที่มากขึ้น

3.2 ความต้องการของระบบ

3.2.1 รองรับขั้นตอนพื้นฐานของการประมวลผลภาพ

เมื่อพิจารณาการประมวลภาพด้วยคอมพิวเตอร์นั้นจะดำเนินการด้วยการทำงานของสามส่วนร่วมกัน คือ รับภาพ ประมวลผลภาพ และแสดงผลภาพ การรับภาพโดยทั่วไปจะใช้การเชื่อมต่อกับอุปกรณ์รับภาพต่าง ๆ การประมวลผลภาพเป็นการประยุกต์ใช้หลักการของการประมวลผลภาพแบบต่าง ๆ ดำเนินการกับภาพที่รับเข้ามา การแสดงผลภาพเป็นการนำผลการประมวลผลแสดงออกมาผ่านส่วนต่อประสานกราฟิกกับผู้ใช้ดั้งนั้นซอฟต์แวร์ที่พัฒนาขึ้นนี้จะต้องรองรับการทำงานหลักทั้งสามนี้

3.2.2 รองรับไฟล์รูปภาพชนิดต่าง ๆ

ซอฟต์แวร์ที่พัฒนาขึ้นนี้สามารถรองรับไฟล์ภาพที่มากมายหลายชนิด อาทิ ไฟล์ภาพชนิด JPEG (joint photographic experts Group) ไฟล์ภาพชนิด PNG (portable network graphics) เป็นต้น ซอฟต์แวร์นี้ควรมีต้นรหัสสำหรับการแปลงไฟล์ดังกล่าวข้างต้นให้อยู่ในรูปแบบ

ที่เหมาะสม สามารถนำมาประมวลผลได้โดยตรง รวมถึงมีความสามารถในการบันทึกไฟล์ภาพเป็นชนิดต่าง ๆ การแปลงไฟล์ให้อยู่ในรูปแบบที่เหมาะสมนี้มีความจำเป็นมาก เนื่องจากแต่ละส่วนของซอฟต์แวร์จะต้องทำงานกับข้อมูลภาพในรูปแบบเดียวกันเพื่อให้ทำงานร่วมกันได้อย่างถูกต้อง

3.2.3 ความสามารถในการขยายระบบ

คุณสมบัติที่สำคัญที่จะต้องคำนึงถึงในการออกแบบซอฟต์แวร์สำหรับการประมวลผลภาพด้วยคอมพิวเตอร์นั้นคือความสามารถในการขยายระบบ (extensibility) เพื่อซอฟต์แวร์สามารถรองรับการใช้งานที่มากขึ้นรองรับการพัฒนาระบบการมองเห็นในรูปแบบใหม่การออกแบบซอฟต์แวร์ให้มีคุณสมบัติข้อนี้จะทำโดยการใช้ generic interface การสร้างองค์ประกอบแต่ละส่วน ได้แก่ การรับภาพการประมวลผลและการแสดงผลภาพ จะใช้ออกแบบในลักษณะนี้โดยเป็นการซ่อนฟังก์ชันการทำงานจริงไว้ภายใน

ความสามารถในการขยายระบบนั้นขึ้นอยู่กับคุณสมบัติหลายอย่าง คือ modularity และ portability ซึ่งโดยส่วนใหญ่แล้วความต้องการของระบบนั้นผูกติดกันเป็นอย่างมากความต้องการของระบบแต่ละอย่างจะส่งผลถึงกันเสมอความสามารถในการขยายระบบครอบคลุมถึงความสามารถในการเชื่อมต่อกับชุดพัฒนาอื่น ๆ การรองรับการโปรแกรมด้วยภาษาต่าง ๆ สามารถนำไปพัฒนาด้วยเครื่องมือหรือสภาวะแวดล้อมของการโปรแกรมที่หลากหลาย รวมถึงการนำไปใช้งานบนระบบปฏิบัติการที่สำคัญ เช่น ระบบปฏิบัติการวินโดวส์ระบบปฏิบัติการลินุกซ์ เป็นต้น สำหรับการรองรับระบบปฏิบัติการที่หลากหลายนั้นจะได้กล่าวถึงในส่วนของ portability

การพัฒนาซอฟต์แวร์ให้รองรับการโปรแกรมหลายภาษา หรือทำงานได้กับระบบปฏิบัติการต่าง ๆ ส่วนหนึ่งจะขึ้นอยู่กับสภาวะแวดล้อมของการพัฒนาซอฟต์แวร์หรือเครื่องมือสำหรับการพัฒนาซอฟต์แวร์สภาวะแวดล้อมของการพัฒนาซอฟต์แวร์ที่สำคัญมีสองชนิด คือ ชนิดที่ใช้ Makefile ซึ่งส่วนใหญ่จะใช้สำหรับการพัฒนาบนระบบปฏิบัติการลินุกซ์และชนิดที่ใช้ Project file เช่น ชุดพัฒนาของไมโครซอฟต์ที่ทำงานบนระบบปฏิบัติการวินโดวส์ คือ Visual Studio ซอฟต์แวร์ที่สามารถรองรับการพัฒนาด้วยสภาวะแวดล้อม ทั้งสองชนิดดังกล่าวข้างต้นมีความเป็นไปได้ที่จะนำไปพัฒนาด้วยสภาวะแวดล้อมแบบอื่น ๆ

โดยส่วนใหญ่แล้วผู้พัฒนาซอฟต์แวร์ไม่ได้ชำนาญหรือคุ้นเคยกับการพัฒนาซอฟต์แวร์ด้วยสภาวะแวดล้อมของการพัฒนาซอฟต์แวร์ที่หลากหลายมากนัก ผู้พัฒนาซอฟต์แวร์ที่ไม่เคยใช้ระบบปฏิบัติการลินุกซ์อาจไม่คุ้นเคยกับการใช้ Makefile หรือ Autotools (Gary V. Vaughan and Taylor, 2007) ผู้พัฒนาที่ใช้ระบบปฏิบัติการลินุกซ์จะใช้ Makefile มากกว่าการใช้ Project file ในการจัดการระบบการพัฒนาซอฟต์แวร์ซอฟต์แวร์สำหรับงานพัฒนาระบบการมองเห็นของคอมพิวเตอร์ที่ดีควรจะรองรับความแตกต่างของผู้พัฒนาดังกล่าว

3.2.4 การพัฒนาเป็นองค์ประกอบย่อย

การพัฒนาเป็นองค์ประกอบย่อยเป็นวิธีการออกแบบซอฟต์แวร์โดยแยกออกเป็นส่วนย่อยซึ่งในที่นี้ คือ การแยกการรับภาพ การประมวลผล และการแสดงผลภาพออกเป็นองค์ประกอบย่อยทำงานเป็นอิสระจากกัน การสร้างระบบการมองเห็นจะกระทำโดยการเชื่อมต่อองค์ประกอบดังกล่าวเข้าด้วยกัน การสื่อสารขององค์ประกอบย่อยเหล่านี้มีเพียงการส่งข้อมูลให้กันเท่านั้น การสร้างระบบการมองเห็นที่ซับซ้อนสามารถทำได้โดยการเชื่อมต่อองค์ประกอบย่อยพื้นฐานเข้าด้วยกัน

การสร้างองค์ประกอบย่อยเหล่านี้โดยการซ่อนต้นรหัสการทำงานไว้ภายจะส่งผลโดยตรงต่อความสามารถในการ Portability ต้นรหัสที่ซ่อนอยู่ภายในองค์ประกอบย่อย สามารถเพิ่มให้รองรับระบบปฏิบัติการแบบต่าง ๆ สามารถทำให้รองรับการใช้งานกับอุปกณ์รับภาพที่หลากหลายได้โดยสามารถเลือกที่จะเปิดหรือปิดการใช้งานต้นรหัสในส่วนนั้นได้เองซึ่งขึ้นอยู่กับสภาวะแวดล้อมของการพัฒนาซอฟต์แวร์ในขณะนั้น หรือขึ้นอยู่กับการตั้งค่าของผู้พัฒนาซอฟต์แวร์

การออกแบบเป็นองค์ประกอบย่อยเช่นนี้ช่วยให้ผู้ใช้สามารถสร้างระบบการมองเห็นได้ทันทีช่วยลดเวลาในการเรียนรู้สำหรับการสร้างระบบการมองเห็นขึ้นมาสักระบบและยิ่งกว่านั้นในแง่ของการพัฒนาระบบ การแยกออกเป็นองค์ประกอบย่อยจะช่วยลดเวลาในการพัฒนาและการหาข้อผิดพลาดของซอฟต์แวร์ลงเป็นอย่างมาก

3.2.5 Portability

ความต้องการของระบบข้อนี้เป็นการออกแบบให้ซ่อนฟังก์ชันการทำงานที่แตกต่างและเหมาะสมกับแต่ละระบบปฏิบัติการไว้ภายในองค์ประกอบย่อย ซึ่งมีความเป็นไปได้ที่จะทำให้รองรับทุกระบบปฏิบัติการขึ้นอยู่กับความต้องการใช้งานซอฟต์แวร์นี้การออกแบบซอฟต์แวร์โดยคำนึงถึงความต้องการข้อนี้จะช่วยให้ต้นรหัสเดียวกันสามารถทำงานบนระบบปฏิบัติการอื่น ๆ ได้ด้วยการแก้ไขเพียงเล็กน้อยเท่านั้น ความสามารถในการ portability จะเกิดขึ้นได้ก็ด้วยการใช้การออกแบบโดยแย่งออกเป็นองค์ประกอบย่อยและการออกแบบให้มีสามารถในการขยายระบบได้การออกแบบโดยแย่งออกเป็นองค์ประกอบจะช่วยในกรณีที่องค์ประกอบย่อยนั้นไม่สามารถทำงานได้บนระบบปฏิบัติที่ทำการคอมไพล์องค์ประกอบนั้นก็จะไม่ถูกคอมไพล์ออกมา

ข้อดีของการออกแบบเพื่อความต้องการของระบบดังกล่าวจะส่งผลดีโดยตรงกับผู้พัฒนาซอฟต์แวร์การเขียนต้นรหัสเพียงครั้งเดียวสามารถนำไปใช้กับระบบปฏิบัติการอื่นได้ด้วยเท่าที่ซอฟท์แวร์ที่พัฒนา ขึ้นนี้จะรองรับ และยังส่งผลดีกับผู้ใช้หรือผู้พัฒนาระบบการมองเห็นของ

คอมพิวเตอร์ที่มีความต้องการที่หลากหลาย บางท่านอาจต้องใช้ระบบปฏิบัติการหนึ่งเป็นการเฉพาะ อาจด้วยข้อกำหนดของการทำงานใด ๆ ก็ตาม ในกรณีที่การพัฒนาระบบการมองเห็นจำเป็นต้องใช้อุปกรณ์หรือระบบปฏิบัติการต่าง ๆ กันเพื่อให้ทำงานร่วมกัน ซอฟต์แวร์ที่มีคุณสมบัติข้อนี้จะสร้างความแตกต่างอย่างมากในการพัฒนางานที่พัฒนาขึ้น สามารถทำงานได้ทันทีบนระบบปฏิบัติการต่าง ๆ โดยไม่ต้องมีการแก้ไขใด ๆ

3.2.6 การทำงานแบบระบบงานพร้อมกัน

ความต้องการของระบบข้อนี้มีความจำเป็นในแง่ของการใช้ประโยชน์จากทรัพยากรที่ระบบจัดสรรให้ เช่น หน่วยความจำ ความเร็วของหน่วยประมวลผล เป็นต้น องค์ประกอบย่อยแต่ละตัวสามารถทำได้พร้อมกัน องค์ประกอบการรับภาพทำการรับภาพเข้ามาในขณะที่องค์ประกอบย่อยสำหรับการประมวลผลก็ดำเนินการกับข้อมูลภาพที่ได้รับเข้ามาก่อน และแสดงผลผ่านองค์ประกอบย่อยสำหรับการแสดงผลภาพการทำงานทั้งหมดนี้สามารถเกิดขึ้นได้พร้อมกันโดยการออกแบบซอฟต์แวร์ให้รองรับคุณสมบัติข้อนี้ ยิ่งในปัจจุบันหน่วยประมวลผลหนึ่งอาจมีหน่วยประมวลผลย่อยหลายตัวอยู่ภายใน ซึ่งช่วยเพิ่มความสามารถในการทำงานพร้อมกันของซอฟต์แวร์ให้ดียิ่งขึ้นส่งผลต่อความเร็วในการทำงานโดยรวมของซอฟต์แวร์ในงานด้านระบบการมองเห็นของคอมพิวเตอร์ไม่ได้หยุดอยู่เพียงภาพนิ่งเท่านั้น ยังขยายไปถึงการประมวลผลภาพเคลื่อนไหวหรือในกรณีที่ต้องการผลตอบสนองในทันทีที่รับภาพเข้ามา การออกแบบดังกล่าวจะทำให้ได้ผลตอบสนองที่รวดเร็วเพียงพอที่จะทำงานได้ทันเวลา

3.2.7 Dataflow

การส่งข้อมูลภาพระหว่างองค์ประกอบย่อยเป็นสิ่งจำเป็นในการประมวลผลภาพ ข้อมูลภาพที่รับเข้ามาจะไม่มีประโยชน์อะไรถ้าไม่สามารถส่งต่อให้องค์ประกอบการประมวลผลรวมทั้งองค์ประกอบการแสดงผล ในการสื่อสารหรือการส่งข้อมูลระหว่างองค์ประกอบย่อยจะใช้หลักการของหลักการไหลของข้อมูล (wikipedia, 2010) หลักการส่งข้อมูลดังกล่าวเป็นผลจากการออกแบบให้ซอฟต์แวร์แยกเป็นองค์ประกอบย่อยและให้มีการทำงานแบบระบบงานพร้อมกัน การนำหลักการนี้มาใช้จะกระทำผ่านการออกแบบให้มีรูปแบบการสื่อสารที่มีข้อกำหนดเฉพาะสำหรับการสื่อสารของแต่ละองค์ประกอบย่อย การออกแบบดังกล่าวช่วยลดปัญหาในการพัฒนาซอฟต์แวร์การเขียนต้นรหัสโดยตรงในการส่งข้อมูลจากองค์ประกอบหนึ่งสู่องค์ประกอบหนึ่งจะขัดกับการพัฒนาแบบแยกเป็นองค์ประกอบย่อย ซึ่งจะทำให้องค์ประกอบย่อยขาดการทำงานที่เป็นอิสระจากกัน

การพัฒนางานด้านการมองเห็นของคอมพิวเตอร์นั้นไม่ได้กระทำเฉพาะภาพนิ่งเท่านั้นแต่ยังรวมถึง การประมวลผลภาพเคลื่อนไหวทั้งที่รับจากอุปกรณ์รับภาพและไฟล์ภาพเคลื่อนไหวต่าง ๆ ดังนั้นซอฟต์แวร์ที่ออกแบบสำหรับงานด้านนี้ควรจะรองรับการประมวลผล

ในลักษณะนี้ด้วยเช่นกัน ซึ่งการใช้หลักการของการไหลของข้อมูลจะช่วยให้สามารถดำเนินการดังกล่าวได้โดยง่าย

วิธีการส่งข้อมูลระหว่างองค์ประกอบย่อยตามหลักการของการไหลของข้อมูลมีมากมายหลายวิธี อาทิ การใช้ push การใช้ pull และการใช้ indirect transfers (Manolescu and Nahrstedt, 1998) วิธี simple memory copy operations วิธี memory reference counting (Caluwaerts et al., 1983) เป็นต้น วิธีการเหล่านี้มีคุณสมบัติที่ดีมากมาย แต่สิ่งที่สำคัญที่สุดที่ซอฟต์แวร์นี้ต้องการคือ การส่งข้อมูลภาพระหว่างกัน การออกแบบในส่วนนี้จะได้กล่าวถึงในบทที่ 4

3.3 สรุป

ซอฟต์แวร์ที่พัฒนาขึ้นสำหรับงานด้านการการพัฒนาระบบการมองเห็นของคอมพิวเตอร์จะต้องรองรับความต้องการทั้งหมดดังกล่าวมาข้างต้นกล่าวโดยย่อ คือ รองรับรูปแบบไฟล์ภาพได้หลากหลาย รองรับการใช้ชุดเครื่องมือสำหรับพัฒนาระบบการมองเห็นที่มีอยู่อย่างมากมาย สามารถทำงานร่วมกับอุปกรณ์รับภาพและอุปกรณ์แสดงผลชนิดต่าง ๆ รวมถึงรองรับระบบปฏิบัติการต่าง ๆ ที่สำคัญความต้องการของระบบทั้งหมดนั้นมีความข้องเกี่ยวกันเป็นอย่างมาก ดังนั้นจึงไม่สามารถละเลยข้อใดข้อหนึ่งได้เลย ซอฟต์แวร์ที่รองรับความต้องการทั้งหมดนี้จะส่งผลโดยตรงกับการพัฒนาระบบการมองเห็นของคอมพิวเตอร์ซึ่งจะให้ทุกอย่างที่จำเป็นในการใช้งานของผู้พัฒนาระบบการมองเห็น

มหาวิทยาลัยเทคโนโลยีสุรนารี

บทที่ 4

VisBuilder

4.1 ภาพรวมและสถาปัตยกรรมระบบ

ซอฟต์แวร์ที่ออกแบบและสร้างขึ้นนี้ชื่อว่า VisBuilder ซึ่งได้ออกแบบตามความต้องการของระบบดังที่ได้กล่าวไว้ในบทที่ 3 สามารถรองรับงานพัฒนาระบบ การมองเห็นของคอมพิวเตอร์ได้ดังจะได้แสดงในบทที่ 5 VisBuilder สร้างขึ้นด้วยภาษา C/C++ สามารถใช้งานร่วมกับชุดพัฒนาอื่น ๆ ที่สามารถใช้งานผ่านการโปรแกรมด้วยภาษา C/C++ ได้การติดต่อกับผู้ใช้ในแบบวิชวลโดยเน้นการที่ไม่ต้องเขียนโปรแกรมใหม่

ในขณะนี้ VisBuilder สามารถทำงานได้บนระบบปฏิบัติการวินโดวส์และลินุกซ์สามารถต่อกับอุปกรณ์รับภาพที่สามารถทำงานบนระบบปฏิบัติการทั้งสองนี้ได้การทำงานร่วมกันในระบบเครือข่ายคอมพิวเตอร์สามาถทำได้โดยมีองค์ประกอบสำหรับการรับส่งข้อมูลผ่านระบบเครือข่ายที่ได้จัดเตรียมไว้ให้

ส่วนหลักของส่วนต่อประสานกราฟิกกับผู้ใช้แสดงดังรูปที่ 4.1 จะประกอบไปด้วยสามส่วนหลัก คือ ส่วนทางด้านซ้ายสำหรับแสดงองค์ประกอบที่สามารถใช้ได้รวมถึงองค์ประกอบที่ผู้ใช้สร้างขึ้นเองด้วย ส่วนทางด้านขวาเป็นพื้นที่สำหรับให้ผู้ใช้ลากและวาง องค์ประกอบจากด้านซ้ายและสร้างการเชื่อมต่อเพื่อพัฒนาอัลกอริทึมของระบบการมองเห็นของคอมพิวเตอร์ การกำหนดคุณสมบัติของแต่ละองค์ประกอบทำได้โดยดับเบิ้ลคลิกที่องค์ประกอบนั้น ๆ ซึ่งจะแสดงหน้าต่างคุณสมบัติขึ้นมาดังรูปที่ 4.2 และส่วนสุดท้ายซึ่งอยู่ด้านล่างเป็นที่แสดงข้อความของผลการทำงานหรือแสดงข้อความบ่งบอกความผิดพลาดในการทำงานช่วยให้ผู้ใช้หาข้อผิดพลาดในการโปรแกรมได้ง่ายขึ้น

รูปที่ 4.1 ส่วนต่อประสานกราฟิกกับผู้ใช้ของ VisBuilder

รูปที่ 4.2 ตัวอย่างหน้าต่างคุณสมบัติขององค์ประกอบย่อย

การออกแบบ VisBuilder โดยการพิจารณาระบบเป็นเลเยอร์ดังรูปที่ 4.3 เลเยอร์ที่สูงกว่าจะสร้างจากเลเยอร์ที่ต่ำกว่าเลเยอร์ที่ต่ำกว่าจะไม่สามารถเข้าถึงคลาสหรือ ฟังก์ชันของเลเยอร์ที่สูงกว่าได้ เลเยอร์ทั้งหมดประกอบด้วย แอพพลิเคชันเลเยอร์ (application layer) เฟรมเวิร์คเลเยอร์ (framework layer) เลเยอร์องค์ประกอบย่อย (component layer) เลเยอร์ข้อมูล (data layer) และเลเยอร์การไหลของข้อมูล (dataflow layer)

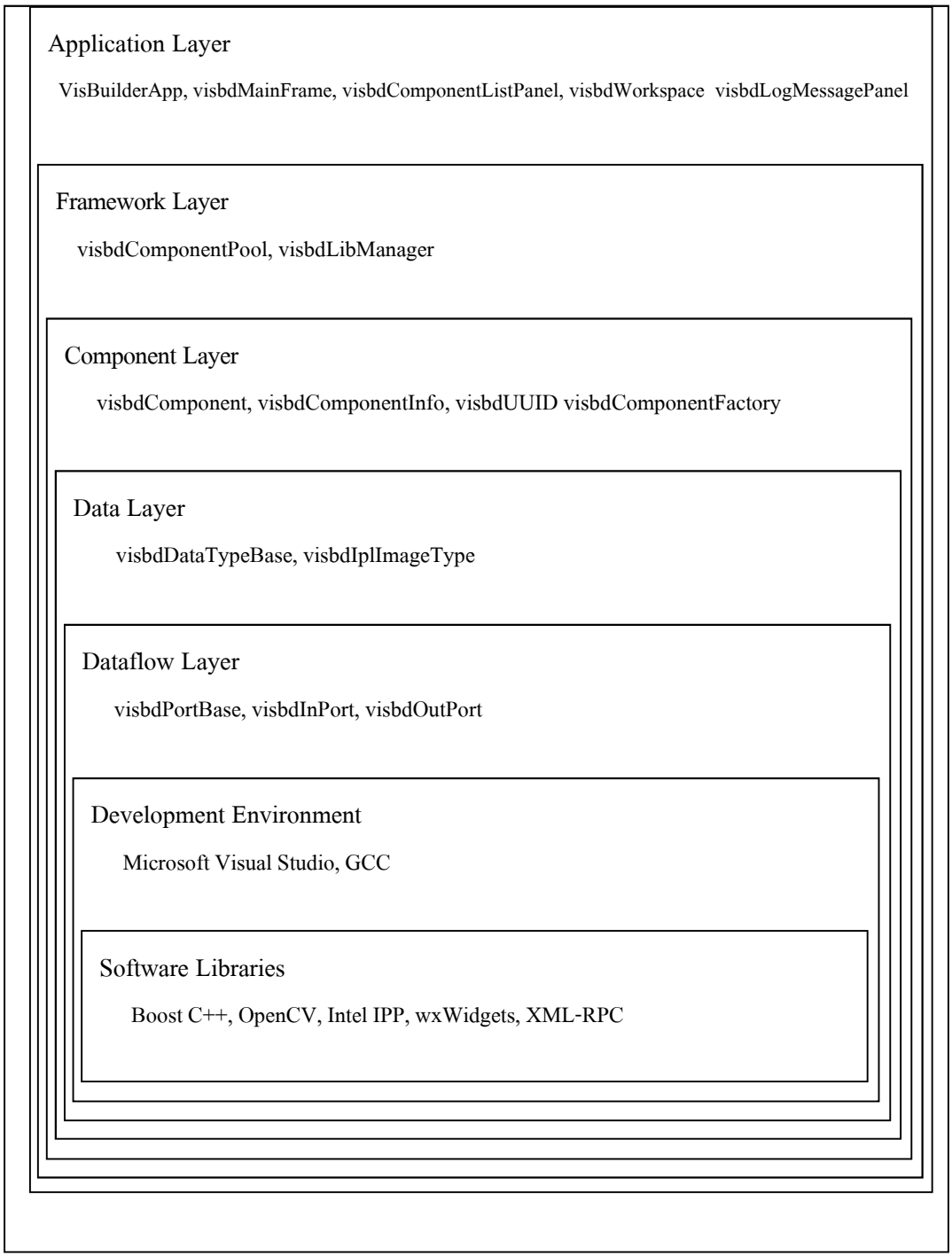
เลเยอร์ที่อยู่ชั้นบนสุด คือ แอพพลิเคชันเลเยอร์ประกอบด้วยคลาสที่รวมคลาสจากเฟรมเวิร์คเข้าด้วยกัน เป็นส่วนที่เชื่อมผู้ใช้กับเฟรมเวิร์คเข้าด้วยกันโดยการใช้การติดต่อกับผู้ใช้ผ่านส่วนต่อประสานกราฟิก คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ (1) VisBuilderApp (2) visbdMainFrame (3) visbdComponentListPanel (4) visbdWorkspace และ (5) visbdLogMessagePanel

เฟรมเวิร์คเลเยอร์ ประกอบด้วยคลาสสำหรับจัดการองค์ประกอบย่อย การเพิ่มหรือลบองค์ประกอบย่อยออกจากระบบการจัดตารางการทำงานขององค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ visbdComponentPool และ visbdLibManager

เลเยอร์องค์ประกอบย่อย ประกอบด้วยคลาสต้นแบบของการสร้างองค์ประกอบย่อย และคลาสที่เป็นคุณสมบัติขององค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ (1) visbdComponent (2) visbdComponentInfo (3) visbdUUID และ visbdComponentFactory

เลเยอร์ข้อมูล เป็นส่วนที่ประกอบด้วยคลาสต้นแบบสำหรับการสร้างชนิดข้อมูลที่ใช้ในซอฟต์แวร์และสำหรับจัดการข้อมูลต่าง ๆ คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่คลาส visbdDataTypeBase และคลาสอนุพันธ์จาก visbdDataTypeBase เช่น visbdIplImageType

เลเยอร์การไหลของข้อมูล เป็นเลเยอร์ที่มีคลาสสำหรับจัดการการส่งข้อมูลภายในระหว่างองค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ คลาส visbdCircularBuffe และvisbdPortBaser โดยที่คลาสที่อนุพันธ์จากคลาส visbdPortBase คือ คลาส visbdInPort และคลาส visbdOutPort

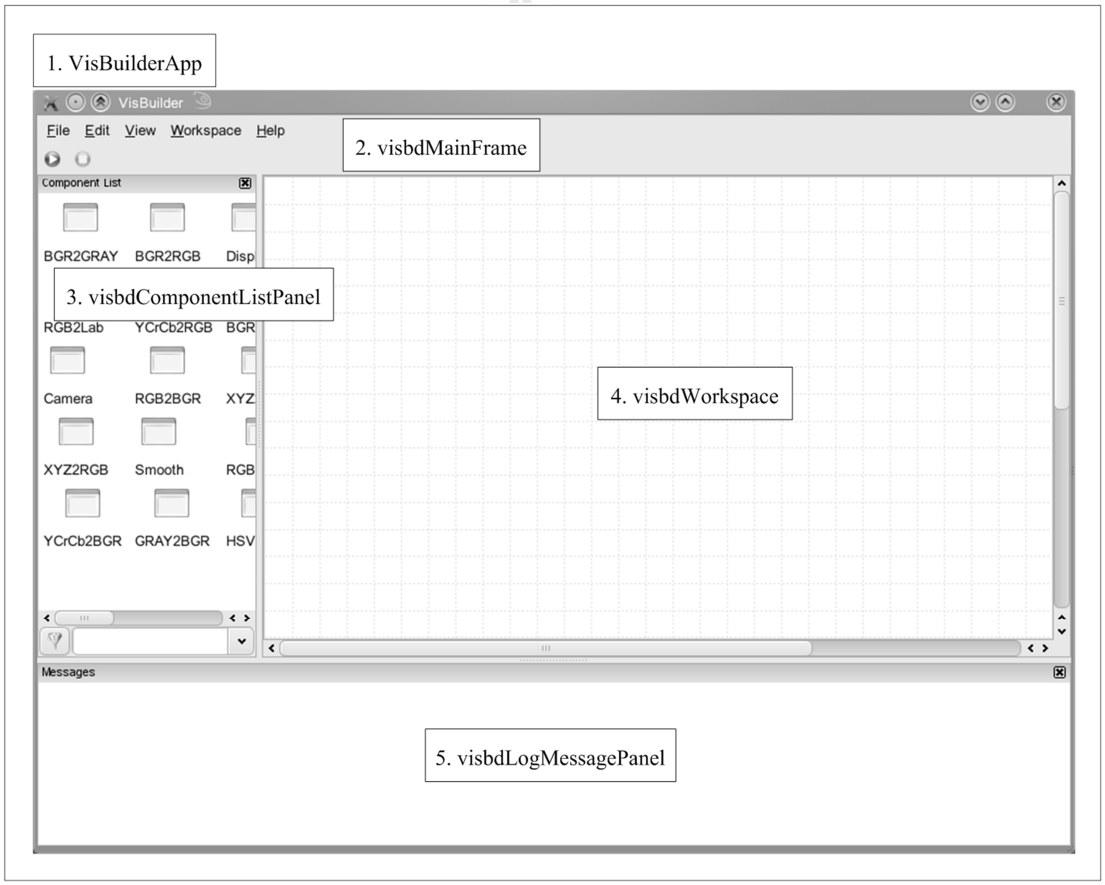


รูปที่ 4.3 การออกแบบสถาปัตยกรรมระบบในลักษณะเลเยอร์

4.2 การออกแบบระบบ

4.2.1 แอพพลิเคชันเลเยอร์ (Application Layer)

แอพพลิเคชันเลเยอร์เป็นเลเยอร์บนสุดของ VisBuilder ทำหน้าที่เชื่อมผู้ใช้กับเฟรมเวิร์คเลเยอร์เข้าด้วยกันโดยการใช้การติดต่อกับผู้ใช้ผ่านส่วนต่อประสานกราฟิก และยังทำหน้าที่ค้นหาองค์ประกอบที่มีอยู่ในระบบประกอบด้วยห้าคลาส คือ (1) VisBuilderApp (2) visbdMainFrame (3) visbdComponentListPanel (4) visbdWorkspace และ (5) visbdLogMessagePanel การจัดวางของคลาสที่สัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้แสดงดังรูปที่ 4.4



รูปที่ 4.4 คลาสที่สัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้ของ VisBuilder

4.2.1.1 คลาส VisBuilderApp

คลาส VisBuilderApp คือ ตัว VisBuilder เป็นจุดเริ่มต้นของระบบมีความสัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้ดังรูปที่ 4.4 ซึ่งคือ ตำแหน่งหมายเลข 1 ใช้สำหรับการกำหนดค่าของซอฟต์แวร์นี้และทำหน้าที่จัดการกับ event loop ของวินโดว์คลาสนี้ อนุพันธ์มาจากคลาส wxApp

4.2.1.2 คลาส visbdMainFrame

คลาส visbdMainFrame ทำหน้าที่สร้างเฟรมวินโดว์ซึ่งเป็นส่วนหลักของส่วนต่อประสานกราฟิก คลาสนี้อนุพันธ์มาจากคลาส wxFrame สำหรับจัดวางองค์ประกอบต่าง ๆ ของส่วนต่อประสานกราฟิก เช่น เมนูบาร์ (menubar) ทูลบาร์ (toolbar) สเตตัสบาร์ (statusbar) เป็นต้น ซึ่งเป็นการกำหนดหน้าตาของ VisBuilder เฟรมวินโดว์ที่สร้างขึ้นนี้สามารถปรับเปลี่ยนขนาดได้และสามารถบรรจุเฟรมวินโดว์ลงไปได้ตำแหน่งของคลาสนี้อยู่ที่หมายเลข 2 ของรูปที่ 4.4

4.2.1.3 คลาส visbdComponentListPanel

คลาส visbdComponentListPanel อนุพันธ์มาจากคลาส wxPanel ทำหน้าที่แสดงองค์ประกอบย่อยที่มีอยู่ในระบบซึ่งสามารถนำไปใช้ได้และยังมีความสามารถค้นหาองค์ประกอบย่อยจากชื่อได้มีความสัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้ดังรูปที่ 4.4 ในตำแหน่งหมายเลข 3

4.2.1.4 คลาส visbdWorkspace

คลาส visbdWorkspace พัฒนาหรืออนุพันธ์มาจากคลาส wxScrolled-Window ตำแหน่งของคลาสนี้อยู่ที่หมายเลข 4 ของรูปที่ 4.4 ทำหน้าที่แสดงองค์ประกอบย่อยและการเชื่อมต่อในรูปแบบกราฟิกเป็นจุดสั่งให้ระบบการมองเห็นที่ออกแบบไว้หยุดหรือทำงาน

คลาสนี้สั่งให้ระบบทำงานโดยเริ่มจากการจัดตารางการทำงานขององค์ประกอบย่อยโดยใช้ข้อมูลจากคลาส ComponentPool ที่อยู่ในเฟรมเวิร์คเลเยอร์เป็นตัวตัดสินใจว่าองค์ประกอบย่อยใดควรทำงานก่อนหลัง

การพัฒนาระบบการมองเห็นของคอมพิวเตอร์จะกระทำที่ตำแหน่งนี้ผู้ใช้สามารถลากองค์ประกอบย่อยจากตำแหน่งที่ 3 ในรูปที่ 4.4 มาวางลงเพื่อเริ่มสร้างระบบการมองเห็นและจัดวางการเชื่อมต่อเข้าด้วยกัน

4.2.1.5 คลาส visbdLogMessagePanel

คลาส visbdLogMessagePanel สำหรับแสดงข้อมูลของความผิดพลาดต่าง ๆ ของการทำงานในรูปแบบข้อความ และยังสามารถใช้แสดงข้อมูลอื่นเพิ่มเติมได้หากผู้พัฒนาต้องการคลาสนี้อนุพันธ์มาจากคลาส wxPanel ซึ่งอยู่ในตำแหน่งที่ 5 ของรูปที่ 4.4

4.2.2 เฟรมเวิร์คเลเยอร์ (Framework Layer)

โครงสร้างของระบบ VisBuilder แสดงในรูปที่ 4.5 โดยในส่วนหลักหรือ VisComponent แบ่งออกเป็นองค์ประกอบย่อย 3 ประเภทหลัก องค์ประกอบทั้งสามแบบถูกจัดให้เป็นองค์ประกอบพื้นฐาน (basic component) ของระบบ และคลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ visbdComponentPool และ visbdLibManager

4.2.2.1 องค์ประกอบแหล่งจ่าย (Source Component)

เป็นองค์ประกอบย่อยในการรับภาพจากอุปกรณ์รับภาพไฟล์ภาพหรือข้อมูลอื่น ๆ จากภายนอกเข้าสู่ตัวซอฟต์แวร์ได้แก่ องค์ประกอบย่อยการรับภาพจากกล้อง (viscomCamera) องค์ประกอบย่อยการอ่านไฟล์ภาพ (viscomImageReader) และองค์ประกอบย่อยการรับข้อมูลจากระบบเครือข่ายคอมพิวเตอร์ (visbdNetInput)

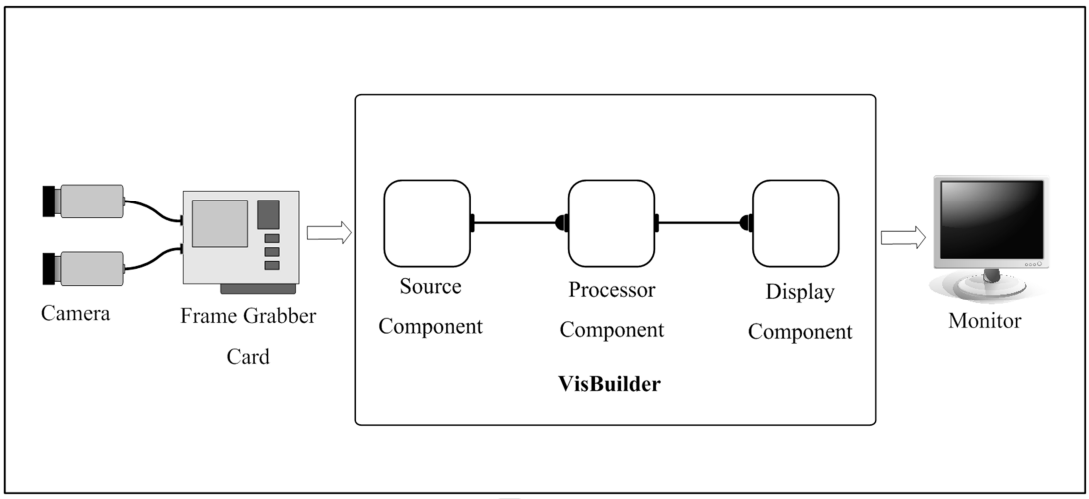
4.2.2.2 องค์ประกอบการประมวลผล (Processing Component)

เป็นองค์ประกอบย่อยในการประมวลผลสัญญาณภาพหรือดำเนินการต่าง ๆ กับข้อมูลภายในและข้อมูลที่รับเข้ามา องค์ประกอบย่อยในส่วนนี้มีมากมายหลายชนิดและสามารถพัฒนาเพิ่มเติมได้ซึ่งในขณะนี้ได้พัฒนาขึ้นบางส่วน ตัวอย่างเช่น

องค์ประกอบย่อยสำหรับการแปลงระบบสีของไฟล์รูปภาพ เช่น (1) viscomRgb2Gray (2) viscomGray2Rgb (3) viscomRgb2Hsv เป็นต้น องค์ประกอบย่อยการหมุนภาพ เช่น (1) viscomFlipVertical (2) viscomFlipHorizontal (3) viscomRotate เป็นต้น และองค์ประกอบย่อยชนิดอื่น ๆ ที่ยังอยู่ในระหว่างการพัฒนา

4.2.2.3 องค์ประกอบการแสดงผล (Display Component)

เป็นองค์ประกอบย่อยในการแสดงผลภาพหรือส่งข้อมูลไปยังอุปกรณ์ภายนอกที่ใช้แสดงผลได้แก่ องค์ประกอบย่อยการแสดงภาพบนหน้าจอ (viscomMonitor) องค์ประกอบย่อยการบันทึกไฟล์ภาพ (viscomImageWriter) และองค์ประกอบย่อยการส่งข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ (visbdNetOutput)



รูปที่ 4.5 โครงสร้างการทำงานของ VisBuilder

4.2.2.4 คลาส visbdComponentPool

คลาส visbdComponentPool เป็นคลาสสำหรับจัดเก็บองค์ประกอบย่อยที่ผู้ใช้เพิ่มเข้าไปในระบบการเพิ่มหรือลบองค์ประกอบย่อยทั้งสามข้างต้นจะดำเนินการภายในคลาสนี้

4.2.2.5 คลาส visbdLibManager

คลาส visbdLibManager เป็นคลาสสำหรับค้นหาไฟล์ที่บรรจุองค์ประกอบย่อย และเพิ่มองค์ประกอบย่อยนั้นเข้าไปในระบบของ VisBuilder เพื่อให้ผู้ใช้ใช้นำมาใช้งานได้ไฟล์ที่บรรจุองค์ประกอบย่อยสำหรับระบบปฏิบัติการวินโดวส์ คือ ไฟล์ DynamicLink Library (DLL) ซึ่งจะมีนามสกุลเป็น *.dll และสำหรับในกรณีของระบบปฏิบัติการลินุกซ์ คือ ไฟล์Shared Object มีนามสกุลเป็น *.so

เมื่อทำการโหลดไฟล์ที่บรรจุองค์ประกอบย่อย ข้อมูลทั้งหมดขององค์ประกอบย่อยจะถูกนำไปเก็บไว้ในคลาส visbdComponentFactoryTable ซึ่งจะจัดเก็บอินสแตน (instance) ของคลาส visbdComponentFactory ไว้ภายใน ดังจะได้กล่าวในหัวข้อ 4.2.3.4

4.2.3 เลเยอร์องค์ประกอบย่อย (Component Layer)

เลเยอร์องค์ประกอบย่อย ประกอบด้วยคลาสต้นแบบของการสร้างองค์ประกอบย่อยและคลาส ที่เป็นคุณสมบัติขององค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ (1) visbdComponent (2) visbdComponentInfo (3) visbdUUID (4) visbdComponentFactory

4.2.3.1 คลาส visbdComponent

คลาส visbdComponent เป็นคลาสฐานของการสร้างองค์ประกอบย่อย ทุกองค์ประกอบย่อยที่พัฒนาขึ้นจะต้องอนุพันธ์จากคลาสนี้ทั้งหมดโดยไม่มีข้อยกเว้น

คลาสนี้ประกอบด้วย4ฟังก์ชันหลักซึ่งเป็นฟังก์ชันที่องค์ประกอบย่อยที่พัฒนาขึ้นจะต้องเขียนรหัสโปรแกรมเพิ่มเติมลงไปได้แก่ ฟังก์ชัน on_create() ฟังก์ชัน on_delete() ฟังก์ชัน process() และฟังก์ชัน on_create_properties_page()

ฟังก์ชัน on_create() จะทำงานในขณะที่มีการเพิ่มองค์ประกอบย่อยเข้าไปในระบบ จึงเป็นส่วนของการกำหนดค่าเริ่มต้นต่าง ๆ ขององค์ประกอบย่อย

ฟังก์ชัน on_delete() ทำงานเมื่อมีการลบองค์ประกอบย่อยออกจากระบบ องค์ประกอบย่อยที่ต้องการให้มีการดำเนินการใด ๆ ก่อนถูกลบจากระบบเช่นบันทึกไฟล์ เป็นต้น สามารถทำได้ในฟังก์ชันนี้

ฟังก์ชัน process() ทำงานเมื่อมีการสั่งระบบให้เริ่มการประมวลผลการดำเนินการคำนวณใด ๆ ขององค์ประกอบย่อยต้องเขียนขึ้นภายในฟังก์ชันนี้

ฟังก์ชัน on_create_properties_page() จะทำงานเมื่อผู้ใช้ต้องการกำหนดค่าคุณสมบัติขององค์ประกอบย่อย ซึ่งจะแสดงหน้าต่างคุณสมบัติขององค์ประกอบนั้น ๆ ขึ้นมาให้ผู้ใช้ทำการปรับแต่ง ฟังก์ชันนี้ช่วยให้มีความยืดหยุ่นในการเพิ่มส่วนต่อประสานกราฟิกกับผู้ใช้สามารถกำหนดหน้าตา ข้อความ หรือปุ่มกดได้ตามความต้องการ

สำหรับฟังก์ชันทั้งสี่นั้นมีเพียงสองฟังก์ชันที่บังคับให้เขียนรหัสโปรแกรมลงไปคือ ฟังก์ชัน on_create() และฟังก์ชัน process() ไม่เช่นนั้นองค์ประกอบย่อยที่พัฒนาขึ้นนี้จะไม่สามารถใช้งานได้

4.2.3.2 คลาส visbdComponentInfo

คลาส visbdComponentInfo เป็นคลาสที่จัดเก็บข้อมูลขององค์ประกอบย่อยอาทิเช่น ชื่อคำอธิบาย รูปไอคอนและตัวระบุที่ไม่ซ้ำกัน (GUID) ข้อมูลทั้งหมดจะถูกนำไปแสดงผลเพื่อให้ผู้ใช้สามารถเลือกใช้องค์ประกอบย่อยได้อย่างถูกต้อง

4.2.3.3 คลาส visbdUUID

คลาส visbdUUID ใช้เก็บค่าตัวระบุที่ไม่ซ้ำกัน (GUID) เป็นค่าสำหรับจำแนกชนิดขององค์ประกอบย่อยต่าง ๆ ซึ่งแต่ละชนิดจะต้องมีค่าไม่ซ้ำกัน เพื่อให้การสร้างองค์ประกอบย่อยเข้าไปในระบบทำได้ง่าย เมื่อผู้ใช้เลือกองค์ประกอบย่อยเพื่อเพิ่มเข้าไปในระบบค่าข้อมูลนี้จะส่งไปให้ระบบเพื่อค้นหาชนิดขององค์ประกอบย่อยที่ระบุนี้แล้วทำการสร้างขึ้น

การสร้างองค์ประกอบย่อยเพิ่มเติมต้องกำหนดค่าตัวระบุที่ไม่ซ้ำกันด้วยตัวเอง ซึ่งสามารถสร้างได้จากคำสั่ง uuidgen ของระบบปฏิบัติการลินุกซ์หรือใช้โปรแกรม GUID Generator ของไมโครซอฟต์ Visual Studio สำหรับระบบปฏิบัติการวินโดวส์ ตัวอย่าง เช่น “21852B6DFD01-4eda-817F-2B8B67C66C19”

4.2.3.4 คลาส visbdComponentFactory

คลาส visbdComponentFactory เป็นคลาสสำหรับการสร้างองค์ประกอบย่อยหรือเป็นโรงงานผลิดองค์ประกอบย่อยนั่นเองทุกองค์ประกอบย่อยจะมีโรงงานผลิดเป็นของตัวเองซึ่งจะถูกโหลดมาด้วยคลาส visbdLibManager ชื่อโรงงานนี้จะกำหนดด้วยค่าตัวระบุที่ไม่ซ้ำกันของแต่ละองค์ประกอบย่อย เมื่อผู้ใช้เลือกองค์ประกอบย่อยที่จะเพิ่มเข้าไปในระบบค่าตัวระบุที่ไม่ซ้ำกันขององค์ประกอบย่อยที่เลือกจะส่งไปให้กับระบบเพื่อค้นหาโรงงานที่ผลิตองค์ประกอบย่อยดังกล่าวแล้วผลิดออกมา การใช้วิธีการเช่นนี้ช่วยให้การพัฒนาเพิ่มเติมองค์ประกอบย่อยใหม่ทำได้โดยไม่ต้องคอมไพล์ VisBuilder ใหม่

4.2.4 เลเยอร์ข้อมูล (Data Layer)

เลเยอร์ข้อมูลเป็นส่วนของการนิยามชนิดข้อมูลที่ใช้ในการประมวลผลของ VisBuilder ทั้งหมดคลาสที่จัดอยู่ในเลเยอร์นี้ ได้แก่ คลาส visbdDataTypeBase เป็นคลาสฐานสำหรับชนิดข้อมูลทั้งหมดและเป็นคลาสแบบนามธรรม (abstract class) การสร้างชนิดข้อมูลใหม่ต้องทำการอนุพันธ์จากคลาสนี้ตัวอย่างคลาสที่อนุพันธ์จากคลาสนี้ คือ คลาส visbdIplImageType เป็นชนิดข้อมูลภาพที่สร้างมาจาก IplImage ซึ่งเป็นชนิดข้อมูลภาพของ OpenCV (OpenCV,2009) เนื่องจาก VisBuilder ใช้ชุดพัฒนานี้เป็นแกนในการประมวล

4.2.5 เลเยอร์การไหลของข้อมูล (Dataflow Layer)

เลเยอร์การไหลของข้อมูลเป็นส่วนที่นิยามคลาสสำหรับการส่งผ่านข้อมูลระหว่างองค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ ได้แก่ คลาส visbdPortBase คลาส visbdCircularBuffer และคลาสที่อนุพันธ์จากคลาส visbdPortBase คือ คลาส visbdInPort และคลาส visbdOutPort

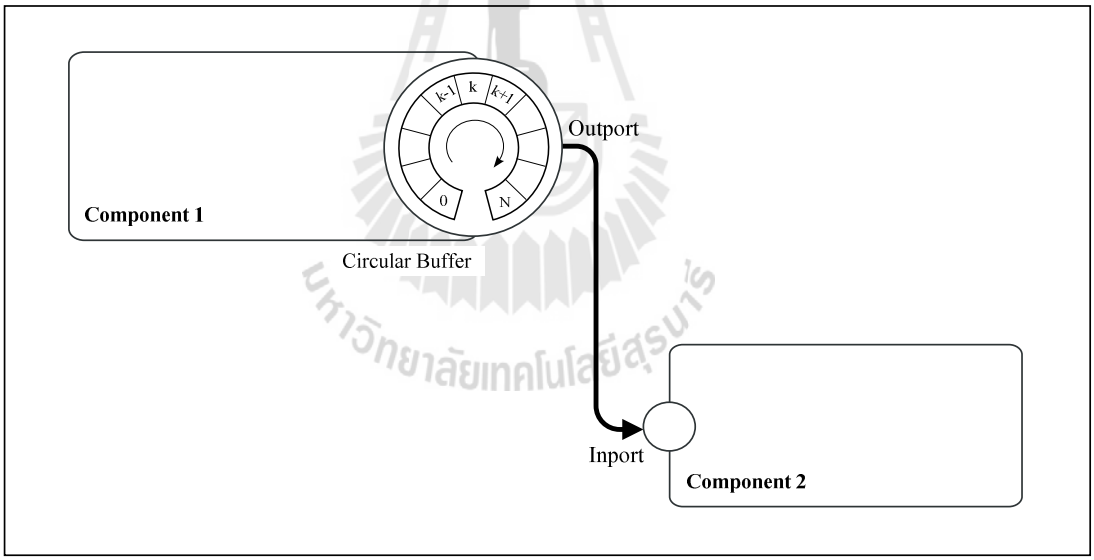
4.2.5.1 คลาส visbdPortBase

การเชื่อมต่อระหว่างองค์ประกอบย่อยจะใช้การสร้างพอร์ทสำหรับการเชื่อมต่อขึ้นคลาสสำหรับการสร้างพอร์ทดังกล่าว คือ คลาส visbdPortBase ซึ่งเป็นคลาสฐานของการเชื่อมต่อทั้งหมดในการเชื่อมต่อระหว่างองค์ประกอบย่อยจะแบ่งเป็นสองประเภทได้แก่ การเชื่อมต่อขาเข้าใช้คลาส visbdInPort และการเชื่อมต่อขาออกใช้คลาส visbdOutPort

คลาส visbdInPort สำหรับการเชื่อมต่อข้อมูลขาเข้า ทำหน้าที่บันทึกการเชื่อมต่อขาเข้าว่ามาจากองค์ประกอบย่อยใด ซึ่งก็คือเป็นข้อมูลขาออกขององค์ประกอบย่อยอื่น เมื่อสั่งให้ VisBuilder ทำการประมวลผลคลาสนี้จะร้องขอข้อมูลจากปลายอีกด้านที่เป็นการเชื่อมต่อขาออกขององค์ประกอบย่อยที่เชื่อมต่ออยู่

คลาส visbdOutPort สำหรับการเชื่อมต่อข้อมูลขาออกมีการเก็บข้อมูลจากการประมวลผลไว้ภายในโดยใช้คลาส visbdCircularBuffer ซึ่งเป็นบัฟเฟอร์รูปแบบหนึ่งที่ใช้ในการประมวลผลโดยอาศัยหลักการของการไหลของข้อมูล

การทำงานระหว่างคลาส visbdInPort และคลาส visbdOutPort แสดงดังรูปที่ 4.6 จะเห็นว่าคลาส visbdOutPort เท่านั้นที่มีการเก็บข้อมูลด้วยบัฟเฟอร์แบบหมุน ข้อมูลที่คลาส visbdInPort ต้องการใช้งานจะเข้าถึงด้วยการอ้างอิงไม่มีการคัดลอกไปไว้ที่ตัวคลาส visbdInPort จึงไม่จำเป็นต้องมีการใช้บัฟเฟอร์แบบหมุน การเข้าถึงข้อมูลด้วยการอ้างอิงช่วยลดเวลาการประมวลผลที่ต้องสูญเสียไปกับการคัดลอกข้อมูลได้มาก



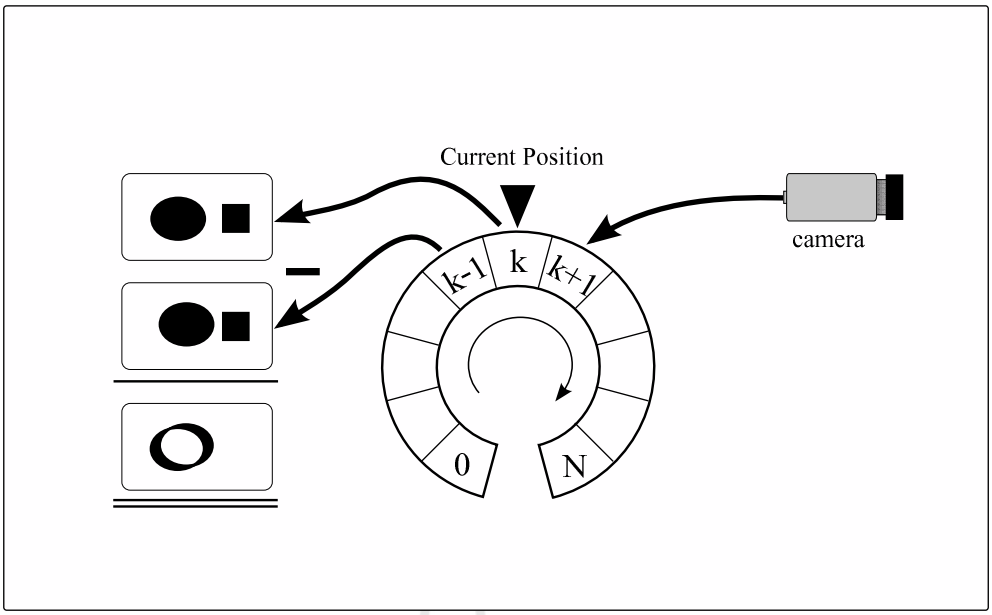
รูปที่ 4.6 ความสัมพันธ์ระหว่างคลาส visbdInPort และคลาส visbdOutPort

4.2.5.2 คลาส visbdCircularBuffer

ในการประมวลผลสัญญาณแบบทันเวลา (real-time processing) จำเป็นที่จะต้องเข้าถึงข้อมูลในขณะปัจจุบัน รวมถึงความสามารถที่จะเข้าถึงข้อมูลก่อนหน้าจำนวนหนึ่งขึ้นกับลักษณะการประมวลผล และการทำงานดังกล่าวจำเป็นต้องมีการปรับเปลี่ยนข้อมูลใหม่อยู่ตลอดเวลา ข้อมูลใหม่เข้ามาแทนที่ข้อมูลชุดเก่าแต่ยังคงต้องรักษาข้อมูลจำนวนหนึ่งที่ผ่านมาไว้ การทำงานลักษณะนี้จะเกิดขึ้นในเวลาเดียวกัน ตัวอย่างเช่น การตรวจจับความเคลื่อนไหวจากไฟล์วีดีโอหรือกล้องรับภาพ วิธีการที่ง่ายที่สุด คือ การหาความแตกต่างระหว่างภาพ (Shuigen et al.,2009) ยิ่งมีความแตกต่างมากก็ยิ่งมีความเคลื่อนไหวมาก การดำเนินการดังกล่าวต้องอาศัยข้อมูลภาพปัจจุบันกับภาพก่อนหน้าซึ่งมีการปรับเปลี่ยนอยู่ตลอดเวลา ภาพใหม่ถูกนำมาแทนที่ภาพเก่า ความจำเป็นดังกล่าวต้องการจำนวนพื้นที่หน่วยความจำจำนวนหนึ่ง หากกำหนดหน่วยความจำมากไปก็จะสิ้นเปลืองน้อยไปก็จะไม่เพียงพอต่อการประมวลผล รวมถึงการจองหน่วยความจำต้องเสียเวลา วิธีการหนึ่งในการจัดการปัญหาดังกล่าวคือการใช้บัฟเฟอร์แบบหมุน (Smith, 1997)

รูปที่ 4.7 แสดงการทำงานของบัฟเฟอร์แบบหมุน เมื่อพิจารณาการตรวจจับความเคลื่อนไหวจากภาพที่รับเข้ามาด้วยกล้องรับภาพ ค่า k เป็นตำแหน่งหน่วยความจำปัจจุบันหรือเป็นภาพปัจจุบัน เมื่อภาพใหม่จากกล้องเข้ามาจะจัดเก็บไว้ที่ตำแหน่ง k+1 ภาพก่อนหน้าอยู่ที่ตำแหน่ง k-1 จะเห็นว่าการประมวลผลสามารถเข้าถึงภาพที่อยู่ก่อนหน้าได้และยังสามารถรับภาพใหม่เข้ามาได้ในขณะเดียวกัน

การจองพื้นที่หน่วยความจำของบัฟเฟอร์แบบหมุนนี้จะเริ่มจองเมื่อสร้างบัฟเฟอร์ชนิดนี้ขึ้นมาในระบบตามขนาดที่ระบุและจะคืนหน่วยความจำเมื่อลบบัฟเฟอร์นี้ออกจากระบบเป็นการทำงานเพียงครั้งเดียว จึงช่วยลดเวลาการทำงานไปได้มาก



รูปที่ 4.7 การทำงานของบัฟเฟอร์แบบหมุน (circular buffer)

4.2.6 สภาวะแวดล้อมการพัฒนา

สภาวะแวดล้อมของการพัฒนาที่ใช้กับ VisBuilder นี้มีสองชนิด คือ Microsoft Visual Studio ในการพัฒนาบนระบบปฏิบัติการวินโดวส์ และ GNU Makefile สำหรับการพัฒนาบนระบบปฏิบัติการลินุกซ์

4.2.7 ชุดเครื่องมือพัฒนาซอฟต์แวร์

ชุดเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ VisBuilder มีดังต่อไปนี้

4.2.7.1 Boost C++

Boost C++ (Karlsson, 2005) เป็นชุดพัฒนาที่ช่วยการเขียนโปรแกรมยืดหยุ่นขึ้น การพัฒนาซอฟต์แวร์จะทำได้รวดเร็วขึ้น ชุดพัฒนานี้มีลักษณะการโปรแกรมเช่นเดียวกับ STL ของภาษา C++ (standard template library) สำหรับซอฟต์แวร์ VisBuilder นี้จะใช้ Boost C++ ในการจัดการหน่วยความจำและจัดการข้อมูลชนิดอาร์เรย์ การนำชุดพัฒนานี้มาใช้ช่วยลดปัญหาในเรื่องการทำงานระหว่างเธรด เนื่องจาก Boost C++ ได้ผ่านการทดสอบการใช้งานในลักษณะนี้มาแล้ว

4.2.7.2 wxWidgets

การออกแบบส่วนต่อประสานกราฟิกกับผู้ใช้ถือเป็นส่วนที่ต้องพิจารณาให้ความสำคัญ เพื่อให้การใช้งาน ซอฟต์แวร์เป็นไปอย่างสะดวกตามวัตถุประสงค์ที่ตั้งไว้ นอกเหนือไปจากเพื่อให้มีความยืดหยุ่น ในการนำไปใช้บนระบบปฏิบัติการต่าง ๆ ได้ด้วย การพัฒนาซอฟต์แวร์ VisBuilder นี้จึงเลือกใช้ชุดพัฒนาส่วนต่อประสานกราฟิกกับผู้ใช้ที่ชื่อว่า wxWidgets (Smart et al., 2005) เป็นชุดพัฒนาที่ใช้กันอย่างแพร่หลายมีเสถียรภาพสูง เปิดต้นรหัส และทำงานได้หลายแพลทฟอร์ม ทั้งวินโดวส์ แมค และลินุกซ์

4.2.7.3 OpenCV

OpenCV เป็นชุดพัฒนาที่มีประโยชน์มากสำหรับงานพัฒนาระบบการมองเห็นของคอมพิวเตอร์งานวิจัยนี้ได้นำชุดพัฒนานี้มาใช้เป็นส่วนหลักในการพัฒนาโดยใช้สำหรับจัดการโครงสร้างข้อมูลภายในและฟังก์ชันการประมวลผลภาพต่าง ๆ ซึ่งในจะนำมาใช้เป็นแกนหลักในการทำงานของ VisBuilder ทั้งในส่วนที่เป็นโครงสร้างข้อมูล เช่น IplImageType และส่วนการประมวลผลอื่น ๆ องค์ประกอบที่พัฒนาในขณะนี้ใช้การฟังก์ชันของ OpenCV ในการประมวลผล เช่น การแปลงสีภาพ การหมุนภาพ การรับภาพจากกล้อง เป็นต้น

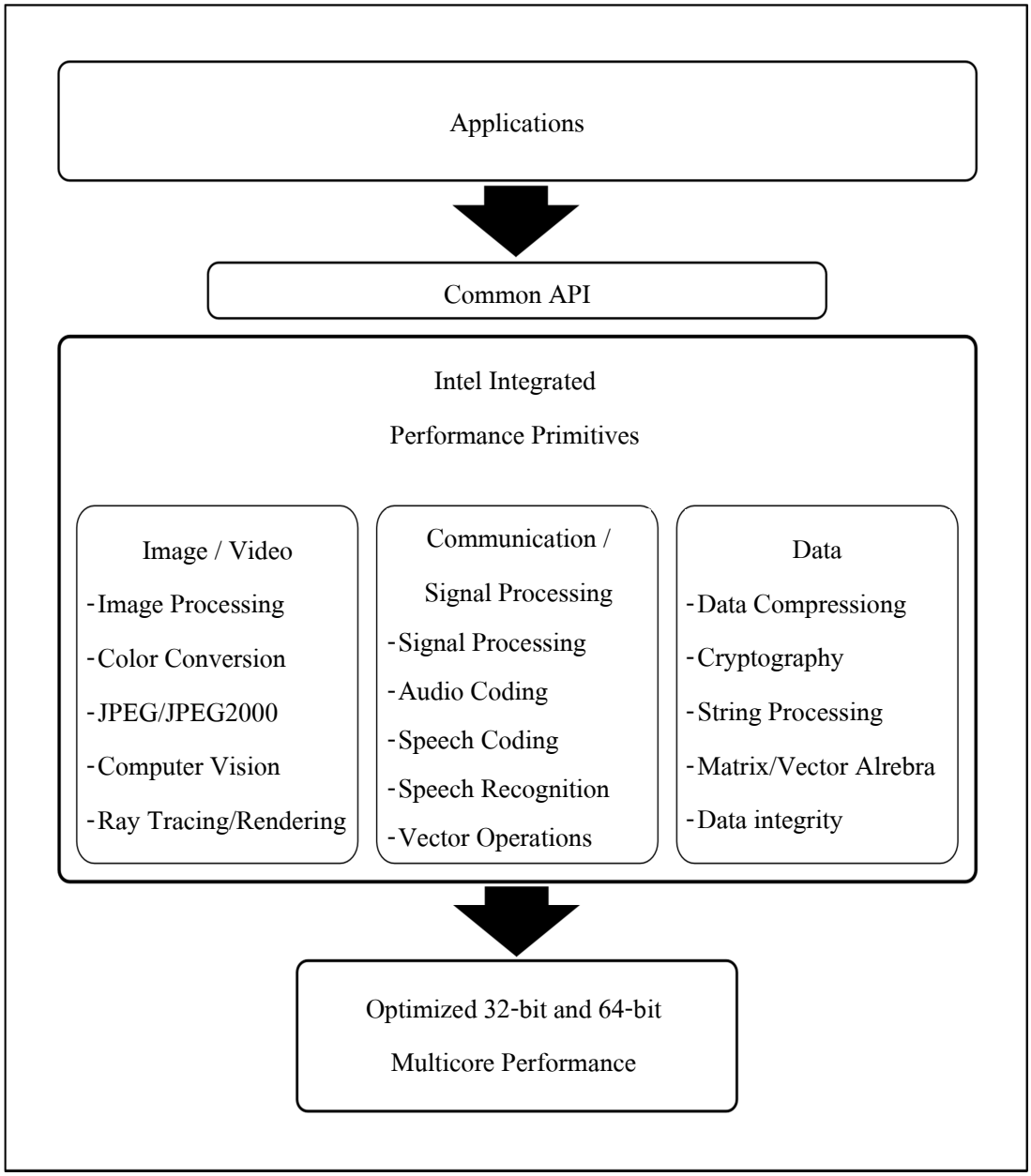
4.2.7.4 Intel IPP

ชุดพัฒนาซอฟต์แวร์สำหรับการประมวลผลที่ใช้ซีพียูของอินเทล (intel integrated performance primitives หรือ IPP) ดังแสดงในรูปที่ 4.8 เป็นชุดพัฒนาซอฟต์แวร์ทางบริษัทอินเทลได้พัฒนาเพื่อใช้งานการประมวลผลที่ต้องการประสิทธิภาพการทำงานสูงสุดบนซีพียูของอินเทลเอง (Intel, 2003) โดยทำเป็นชุดพัฒนาซอฟต์แวร์ในระดับต่ำ (low level software layer) สามารถใช้งานได้ทั้งบนระบบปฏิบัติการ Windows Linux และ Mac มีความยืดหยุ่นสูง สามารถนำต้นรหัสที่เขียนขึ้นไปคอมไพล์ได้ทั้งสามระบบปฏิบัติการโดยไม่ต้องแก้ไขต้นรหัสแต่อย่างใด นอกเหนือไปจากการประมวลผลภาพและสัญญาณ 2 มิติและการมองเห็นของคอมพิวเตอร์แล้ว ในเวอร์ชันล่าสุดของ IPP นั้นมีฟังก์ชันการใช้งานที่หลากหลายครอบคลุมการพัฒนาซอฟต์แวร์ด้านต่าง ๆ ได้ทั่วถึง เช่น การประมวลผลสัญญาณเสียง การคำนวณทางคณิตศาสตร์การบีบอัดข้อมูล เป็นต้น

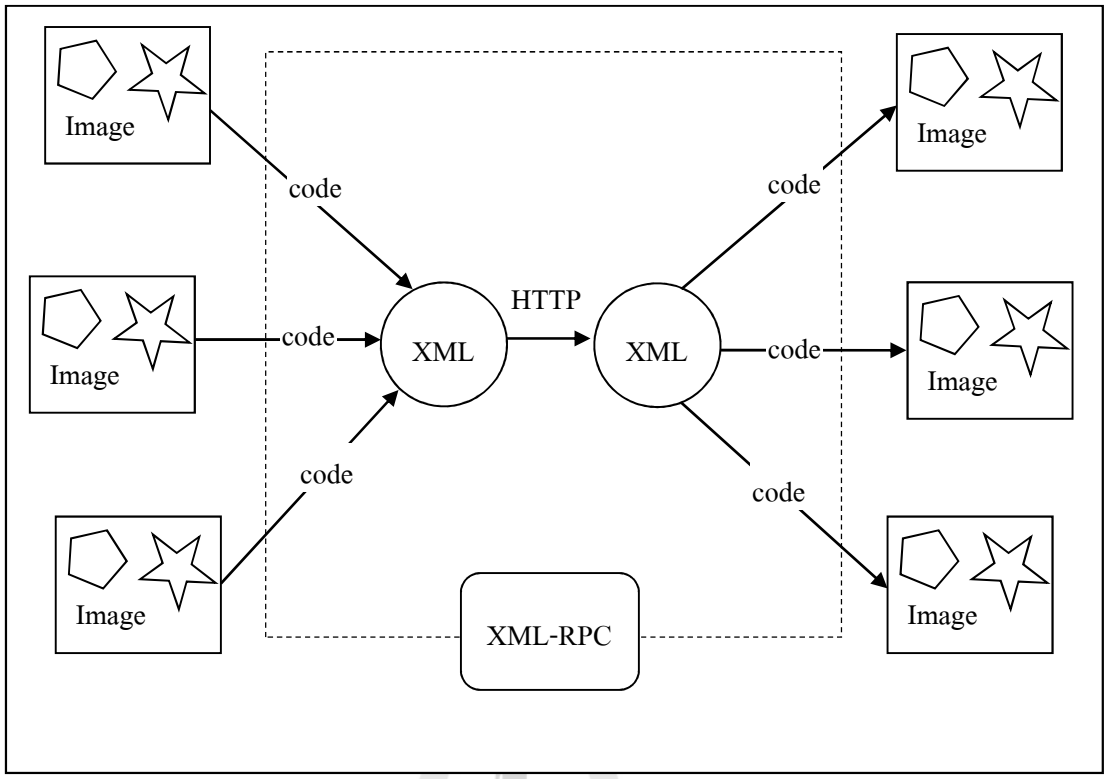
4.2.7.5 XML-RPC

การเรียกกระบวนงานระยะไกล (remote procedure call หรือ RPC) คือ เทคโนโลยีการเรียกกระบวนงานระยะไกล เป็นการอนุญาตให้โปรแกรมบนคอมพิวเตอร์เครื่องหนึ่งสามารถเรียกใช้โปรแกรมที่อยู่บนอีกเครื่องหนึ่งผ่านระบบเครือข่ายคอมพิวเตอร์ได้ (Allman, 2003) เสมือนหนึ่งว่าเป็นการเรียกใช้โปรแกรมบนเครื่องเดียวกัน กล่าวคือ มีการซ่อนการสื่อสารผ่าน ระบบเครือข่ายคอมพิวเตอร์ไว้ภายในในระบบแม่ข่ายลูกข่ายมักจะติดต่อกับแม่ข่ายโดยผ่านทางการเรียกกระบวนงานระยะไกลซึ่งช่วยให้การติดต่อกันระหว่างระบบย่อยเป็นไปได้ด้วยดีและทันเวลา ขั้นตอนการทำงานจะเริ่มที่ลูกข่ายเรียกใช้โปรแกรมบนเครื่องแม่ข่ายผ่านทางการเรียกกระบวนงานระยะไกล แม่ข่ายจะดำเนินการตามโปรแกรมที่ลูกข่ายร้องขอจากนั้นผลลัพธ์ที่ได้จึงถูกส่งกลับไปให้ลูกข่าย การเลือกใช้เทคโนโลยี RPC ดังกล่าวทำให้การใช้ทรัพยากรร่วมกันในระบบการมองเห็นของคอมพิวเตอร์เป็นไปอย่างสะดวกและมีประสิทธิภาพยกตัวอย่าง เช่น อุปกรณ์จับสัญญาณภาพเข้าสู่คอมพิวเตอร์อาจจะไม่จำเป็นจะต้องอยู่ที่เดียวกับระบบประมวลผลสัญญาณภาพก็ได้

รูปที่ 4.9 แสดงการทำงานของ XML-RPC ในการส่งข้อมูลภาพผ่านโปรโตคอล HTTP โดยจะต้องแปลงภาพให้อยู่ในรูปแบบข้อมูล XML แล้วส่งผ่านเครื่องข่ายคอมพิวเตอร์ด้วยโปรโตคอล HTTP เมื่ออีกด้านได้รับข้อมูลจึงแปลงกลับเป็นข้อมูลภาพตามเดิม

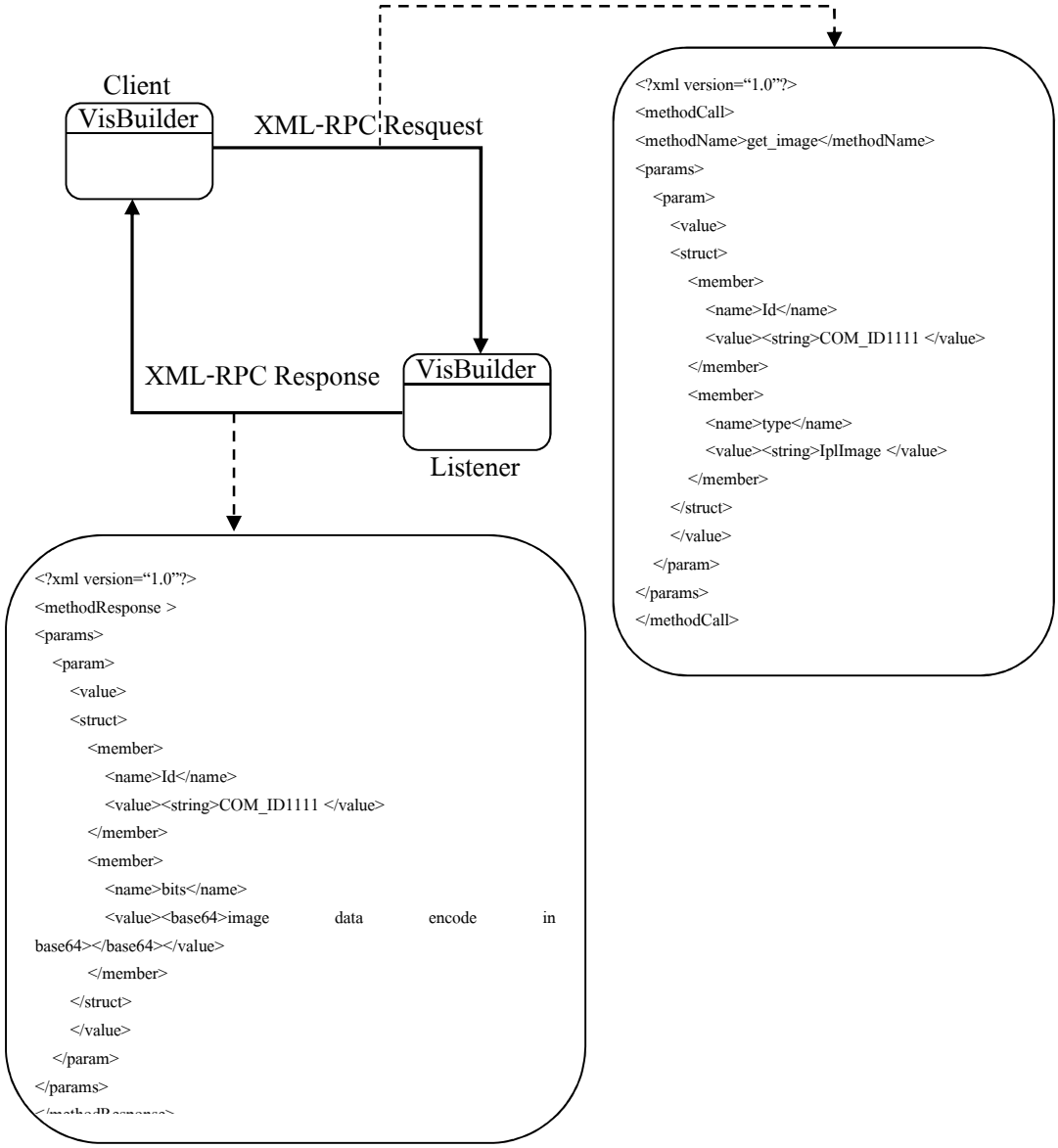


รูปที่ 4.8 โครงสร้างของ Intel IPP



รูปที่ 4.9 การทำงานของ XML-RPC

รูปที่ 4.10 แสดงการทำงานของ VisBuilder ในการส่งข้อมูลภาพด้วย XMLRPC ด้าน VisBuilder ที่เป็น client คือ ผู้ร้องขอข้อมูลจะส่งรูปแบบข้อมูลในรูปแบบของ XML ด้วยข้อกำหนดการใช้งานของ XML-RPC โดยใช้คำสั่งที่ชื่อ methodCall โดยผ่านชื่อฟังก์ชันที่จะเรียกใช้ที่เครื่องปลายทาง ซึ่งในรูป คือ get_image ฝ่าย VisBuilder ที่เป็น listener จะส่งข้อมูลกลับมาในรูปแบบ XML เช่นเดียวกัน โดยจะส่งมาด้วยคำสั่ง methodResponse เพื่อระบุการส่งกลับข้อมูลซึ่งจะมีข้อมูลภาพกลับมาด้วยข้อมูลภาพจะต้องแปลงด้วยวิธี BASE64 เป็นการแปลงข้อมูลไบนารีเป็นข้อมูลที่เป็นข้อความเพื่อให้ใช้กับรูปแบบ XML ได้



รูปที่ 4.10 การส่งข้อมูลของ XML-RPC

ในการทำงานร่วมกันของซอฟต์แวร์นี้ผ่านทางระบบเครือข่ายคอมพิวเตอร์จะใช้การเชื่อมต่อผ่านทางการเรียกกระบวนงานระยะไกล (Sim and Park, 1996) โดยในงานวิจัยนี้ได้เลือกใช้ไลบรารีที่ชื่อว่า Xmlrpc c (Henderson, 2004) สำหรับการโปรแกรมภาษาซีซึ่งจะส่งข้อมูลในรูปแบบ XML (Bray and Maler, 2000) ผ่านทางโปรโตคอล HTTP การพัฒนาตรงส่วนนี้จึงต้องทำการแปลงข้อมูลขององค์ประกอบเป็นข้อมูลในรูปแบบ XML ซึ่งรองรับชนิดข้อมูลตามที่ได้กล่าวไว้ในหัวข้อที่ 4.2.4

4.3 วิธีการออกแบบที่ใช้กับ VisBuilder

กระบวนการออกแบบ VisBuilder ใช้แนวคิดการออกแบบหลายวิธี เช่น การออกแบบเชิงวัตถุ (object oriented design) เพื่อให้ฟังก์ชันอินเตอร์เฟซต่าง ๆ มีความง่ายและชัดเจนในการใช้งานและการซ่อนข้อมูลไว้ภายใน ฉะนั้นจึงทำให้ปลอดภัยจากการเปลี่ยนแปลงอย่างบังเอิญ อาจกล่าวได้ว่าข้อมูลและฟังก์ชันของข้อมูลถูก encapsulate ไปเป็นก้อนเดียวกันรวมถึงความสามารถในการนำกับมาใช้ใหม่ได้ (reusability) กล่าวคือ เมื่อได้เขียนสร้างและแก้ไขข้อบกพร่องของคลาสแล้ว สามารถนำคลาสนี้ไปแทนในโปรแกรมเดิมได้โดยไม่ส่ง ผลกระทบกับการทำงานร่วมกันของแต่ละคลาส

การตรวจสอบการส่งกลับข้อมูลความผิดพลาดของซอฟต์แวร์ที่พัฒนาด้วยภาษา C/C++ เป็นสิ่งที่สำคัญมาก ภาษา C/C++ มีวิธีจัดการกับการส่งกลับข้อมูลความผิดพลาด คือ exception แต่วิธีการนี้ไม่ได้นำมาใช้ในการสร้าง VisBuilder เนื่องจากจะส่งผลต่อประสิทธิภาพในการทำงานของซอฟต์แวร์

VisBuilder ไม่ใช้ตัวแปลส่วนกลาง (globalvariable) เนื่องจากปัญหาหลายอย่างในการพัฒนาซอฟต์แวร์เกิดจากการใช้ตัวแปลชนิดนี้

4.3.1 การโปรแกรมเชิงวัตถุ (Object-Oriented Programming)

การสร้าง VisBuilder ด้วยการพิจารณาแบบเลเยอร์และการใช้การออกแบบเชิงวัตถุช่วยให้ระบบง่ายต่อการใช้และการพัฒนา การโปรแกรมได้ใช้ภาษา C++ ซึ่งมีความสามารถที่จำเป็นของการโปรแกรมเชิงวัตถุ เช่น การ encapsulate การสร้างคลาสแบบ abstract การอนุพันธ์คลาส (inheritance) เป็นต้น

คลาสฐานของทุกองค์ประกอบย่อยจะเป็นแบบ abstract ซึ่งจะทำให้ได้ฟังก์ชันการทำงานพื้นฐานที่ไม่เปลี่ยนแปลงสามารถนำไปใช้ได้กับทุกระบบปฏิบัติการ ความสามารถในการอนุพันธ์คลาสสามารถดูได้จากการสร้างองค์ประกอบใหม่ซึ่งสามารถทำได้โดยการอนุพันธ์จากคลาสฐานนี้จึงเป็นไปได้ที่จะเพื่อความสามารถการประมวลผลที่ยังขาดเข้าไปในระบบหรือสามารถสร้างองค์ประกอบย่อยที่ครอบชุดพัฒนาอื่น ๆ ไว้ภายในได้การเพิ่มองค์ประกอบย่อยเข้าไปสามารถทำได้อย่างไม่จำกัดยิ่งกว่านั้น ความสามารถในการ encapsulate ช่วยให้เลเยอร์ที่อยู่สูงกว่าไม่ได้รับผลกระทบในกรณีที่มีการเปลี่ยนแปลงการทำงานภายในทำให้การปรับปรุงประสิทธิภาพของระบบสามารถทำได้ง่าย

4.3.2 การจัดการ error code

สิ่งหนึ่งที่สำคัญมากในการออกแบบซอฟต์แวร์คือ การตรวจสอบสถานะของค่าที่ส่งกลับ (return code) จากฟังก์ชันหรือสถานะการทำงานของส่วนใด ๆ ในต้นรหัส VisBuilder ไม่ใช้ exception ในการการจัดการสถานะของค่าที่ส่งกลับ เพราะการใช้ exception จะส่งผลต่อความเร็วในการประมวลผล การจัดการสถานะของค่าที่ส่งกลับด้วยตัวเองจะให้การทำงานที่รวดเร็วกว่า

ค่าที่ส่งกลับจากฟังก์ชันจะมีเพียงสองสถานะเท่านั้นเพื่อให้สามารถจัดการได้ง่าย และมีประสิทธิภาพสถานะที่ใช้ในที่นี้ คือ VISBD_SUCCESS สำหรับการทำงานของฟังก์ชันที่เสร็จสมบูรณ์ซึ่งมีค่าเท่ากับศูนย์และ VISBD_FAIL สำหรับการทำงานที่เกิดข้อผิดพลาดใด ๆ ซึ่งมีค่าเท่ากับลบหนึ่งทุกฟังก์ชันที่มีการส่งกลับค่าสถานะการทำงานจะต้องส่งค่าใดค่าหนึ่งจากสองค่านี้พร้อมด้วยข้อมูลเพิ่มเติมของสถานะการทำงาน

ข้อมูลเพิ่มเติมสำหรับอธิบายสถานะการทำงานจะใช้การจัดการด้วยแมโครซึ่งจะส่งสถานะข้อมูลเพิ่มเติมหรือคำอธิบายและค่าประจำตัวขององค์ประกอบกลับไปให้ระบบ ข้อมูลทั้งหมดจะถูกนำมาแสดงที่ส่วนสำหรับแสดงข้อผิดพลาดของส่วนต่อประสานกราฟิกกับผู้ใช้ที่อยู่ด้านล่างของหน้าต่างหลัก สิ่งเหล่านี้จะช่วยให้ผู้พัฒนาซอฟต์แวร์สามารถตรวจสอบหาความผิดพลาดที่เกิดขึ้นได้สามารถรู้ส่วนของต้นรหัสที่ทำให้เกิดความผิดพลาดนี้ทำให้การแก้ไขปรับปรุงทำได้ง่าย VisBuilder มีความสามารถในการจัดการกับความผิดพลาดได้โดยไม่ต้องปิดโปรแกรม การจัดการความผิดพลาดด้วยวิธีนี้ช่วยให้คาดคะเนความเร็วได้ต่างจากการใช้ exception ซึ่งจะขึ้นอยู่กับชุดพัฒนาที่ใช้คอมไพล์ทำให้ไม่สามารถคาดคะเนได้โดยตรง

4.3.3 การทำงานแบบระบบงานพร้อมกัน

หลักการที่สำคัญในการออกแบบ VisBuilder คือ การทำให้องค์ประกอบย่อยทำงานไปพร้อมกัน โดยการใช้ความสามารถของเธรด (thread) ซึ่งเป็นการแบ่งงานออกเป็นส่วนย่อยและให้ทำงานได้พร้อมกัน การใช้เธรดไม่ได้เป็นข้อกำหนดตายตัวของการทำงานของ VisBuilder อาจจะใช้การแบ่งงานออกด้วยการใช้โปรเซส (process) แทนได้โปรเซสนั้นจะแตกต่างจากเธรดตรงที่ว่าเธรดเป็นระบบงานย่อยที่ทำงานอยู่ในโปรเซส โปรเซสให้ประสิทธิภาพการทำงานที่ดีกว่า แต่วิธีการใช้ทรัพยากรร่วมกันระหว่างโปรเซสจะแตกต่างจากเธรด ซึ่งจะมีความซับซ้อนมากกว่าการเลือกใช้วิธีการใดไม่ว่าจะเป็นเธรดหรือโปรเซสหรือวิธีการอื่น ๆ ไม่ใช่สิ่งที่ต้องนำมาพิจารณามากนักเพราะสามารถแทนกันได้แต่สิ่งที่สำคัญในที่นี้คือ VisBuilder มีความสามารถแบบระบบงานพร้อมกัน

สำหรับ VisBuilder ในขณะนี้ได้เลือกใช้เธรดด้วยเหตุผลที่ว่า การใช้วิธีการแยกงานเป็นโปรเซส นั้นเกินความจำเป็น เนื่องจากสามารถสร้างองค์ประกอบย่อยที่ส่งรับข้อมูลระหว่าง

โปรเซสแทนได้การใช้งานโดยการเปิด VisBuilder หลาย ๆ ตัวให้สื่อสารกันผ่านองค์ประกอบย่อยดังกล่าว ซึ่งก็ไม่ต่างจากการแบ่งงานเป็นโปรเซส

การพัฒนาซอฟต์แวร์สำหรับระบบการมองเห็นโดยการใช้ประโยชน์จากคุณสมบัติของการทำงานแบบระบบงานพร้อมกันจะช่วยให้ได้ประสิทธิภาพการทำงานของระบบที่ดีอีกทั้งยังช่วยให้การออกแบบระบบซอฟต์แวร์ทำได้ง่าย แต่ละเธรดจะทำงานอย่างอิสระภายใต้เงื่อนไขการทำงานของระบบการมองเห็นที่ได้รับมอบหมายให้ทำ ตัวอย่างเช่น องค์ประกอบย่อยสำหรับการรับภาพจากอุปกรณ์รับภาพสามารถรับภาพและรอรับภาพถัดไปเมื่อพิจารณาจะเห็นว่าการทำงานมีเพียงการรับภาพและรอภาพถัดไปเท่านั้น การพิจารณาในลักษณะนี้สามารถใช้กับการออกแบบองค์ประกอบย่อยชนิดอื่น ๆ ได้เช่นเดียวกัน เมื่อมองในแง่ของความเร็ว ในการทำงานความเร็วของเธรดจะขึ้นอยู่กับกรรมวิธีทางระบบการมองเห็นที่เธรดได้รับทำให้สามารถประเมินความเร็วของแต่ละองค์ประกอบย่อยได้ซึ่งช่วยในการปรับปรุงประสิทธิภาพการทำงาน

4.3.4 การไม่ใช้ตัวแปลส่วนกลาง (global variable)

VisBuilder ได้รับการออกแบบให้ไม่มีการใช้ตัวแปลส่วนกลางในทุกส่วนของโปรแกรม ปัญหาหลายอย่างในการพัฒนาซอฟต์แวร์เกิดจากการใช้ตัวแปลชนิดนี้ คำว่าส่วนกลาง (global) หมายถึงตัวแปรที่บรรจุข้อมูลไว้นั้นจะถูกประกาศ (declare) เอาไว้ภายนอกทุก ๆ ฟังก์ชันซึ่งทำให้ฟังก์ชันทุก ๆ ฟังก์ชันสามารถที่จะเข้าถึงตัวแปรเหล่านี้ได้ฟังก์ชันเหล่านี้จะดำเนินการต่อข้อมูลด้วยวิธีการต่าง ๆ ไม่ว่าจะเป็นการอ่านข้อมูล การวิเคราะห์ข้อมูล การปรับข้อมูล การจัดข้อมูลใหม่การแสดงข้อมูล การเขียนข้อมูลกลับไปยังดิสก์เป็นต้น

อย่างไรก็ตามอาจมีการใช้ตัวแปลส่วนกลางในชุดพัฒนาซอฟต์แวร์ที่นำมาใช้เช่นจากการใช้ไฟล์ส่วนหัว (header file) ของระบบที่ใช้พัฒนาซอฟต์แวร์หรืออาจมาจากการนำชุดพัฒนาอื่น ๆ มาใช้การไม่มีตัวแปลส่วนกลางช่วยให้ไม่ต้องกังวลปัญหาเนื่องการทำงานขององค์ประกอบย่อยแต่ละตัวมีการซ้อนทับกันในการเข้าถึงตัวแปล องค์ประกอบย่อยแต่ละตัวจะมีตัวแปลเป็นของตัวเองสำหรับ การบรรจุข้อมูล การเก็บสถานะการทำงาน ซึ่งตัวแปลดังกล่าวจะถูกซ่อนไว้ภายใน ป้องการการเข้าถึงได้โดยตรงจากองค์ประกอบย่อยอื่น ๆ

4.3.5 การจัดเก็บไฟล์

การจัดเก็บอัลกอริทึมของระบบการมองเห็นของคอมพิวเตอร์ที่ผู้ใช้ออกแบบจะใช้การบันทึกในรูปแบบ XML การจัดการกับไฟล์ลักษณะนี้จะใช้หลักการ XML Parser เป็น API (application programming interface) ที่ช่วยในการอ่านหรือเขียนข้อมูลโดย API ที่นิยมกันมากคือ DOM (Marini, 2002) และ SAX (Brownell, 2004) ทั้งสองวิธีจะใช้การเข้าถึงข้อมูลที่แตกต่างกัน

การเข้าถึงข้อมูลแบบ DOM จะมองเอกสาร XML ในลักษณะของโครงสร้างต้นไม้ โดยมีหลักการในการอ่านเอกสาร XML มาวางในหน่วยความจำในลักษณะต้นไม้ซึ่งประกอบด้วย element และ attribute ต่าง ๆ การเข้าถึงข้อมูลจะใช้การเดินไปตามกิ่งก้านของต้นไม้แบบต่อเนื่องไปเรื่อย ๆ หรือจะใช้การอ้างอิงกิ่งก้านแบบเฉพาะเจาะจงลงไป หรือจะเข้าถึงแบบสุ่ม (random access) ก็ได้ข้อจำกัดของ DOM คือต้องการหน่วยความจำที่เพียงพอกับปริมาณข้อมูลที่ใช้ในการประมวลผลเนื่องจาก DOM จะอ่านข้อมูลทั้งหมดมาเก็บไว้ในหน่วยความจำในครั้งเดียว แต่ก็มีข้อดี คือ การเขียนโปรแกรมสามารถทำได้ง่ายกว่า

การเข้าถึงเอกสาร XML แบบ SAX จะจัดการกับข้อมูลด้วยแนวทาง event driven การทำงานของ SAX จะไม่อ่านข้อมูลทั้งหมดมาไว้ในหน่วยความจำแต่จะอ่านข้อมูลจากไฟล์ตั้งแต่เริ่มต้นไล่ลงไปเรื่อย ๆ แล้วสร้าง event ต่าง ๆ ออกมา เช่น การเปิด element การปิด element การพบ attribute เป็นต้น ดังนั้นผู้เขียนโปรแกรมจึงมีหน้าที่ดักจับ event เหล่านั้นเพื่อจัดการกับข้อมูล

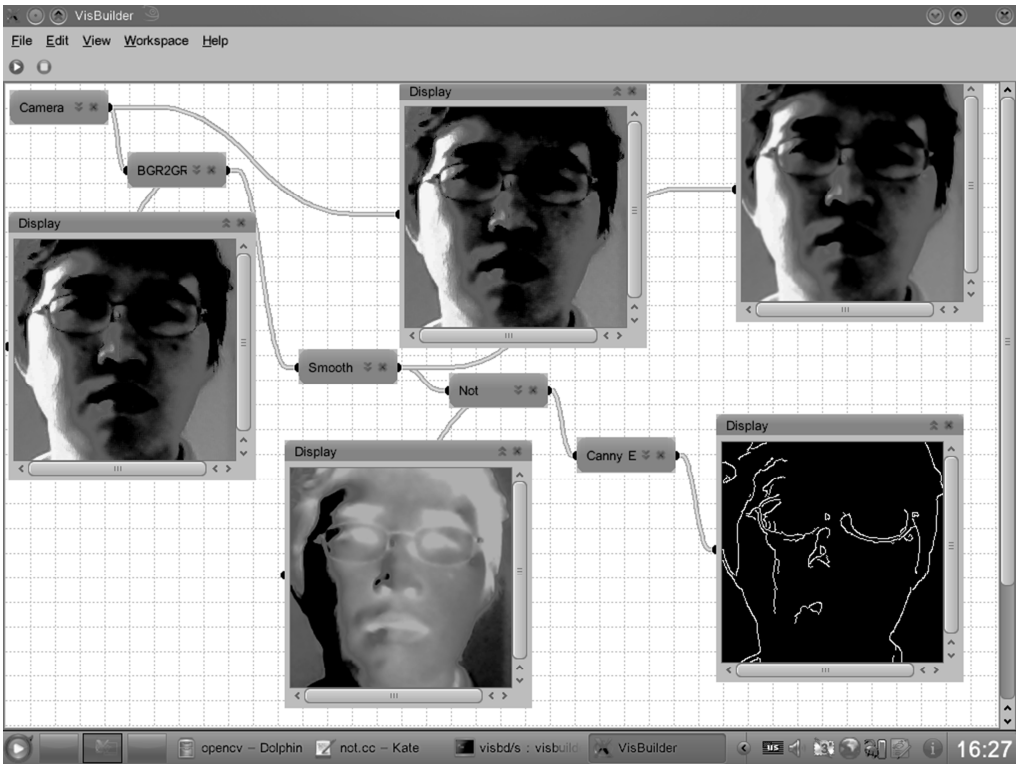
การจัดเก็บไฟล์ของ VisBuilder จะใช้ทั้งสองวิธีตามความเหมาะสมในการใช้งาน กล่าวคือ ในการบันทึกไฟล์จะใช้ API แบบ DOM เพราะบันทึกไฟล์สามารถทำให้เสร็จในครั้งเดียว ได้ส่วนการอ่านไฟล์จะใช้ SAX เนื่องจากการสร้างระบบการมองเห็นจากไฟล์ขึ้นใหม่จะทำได้ง่ายในแง่ของการโปรแกรม เช่น เมื่อเกิด event จากการเปิด element ที่ชื่อว่า COMPONENT ก็จะทำการสร้างองค์ประกอบย่อยขึ้นและเพิ่มเข้าไปในระบบ เมื่ออ่านถึง element ชื่อ LINK ก็จะสร้างการเชื่อมต่อระหว่างองค์ประกอบย่อยตามที่ระบุไว้เป็นต้น

4.3.6 การจองหน่วยความจำ

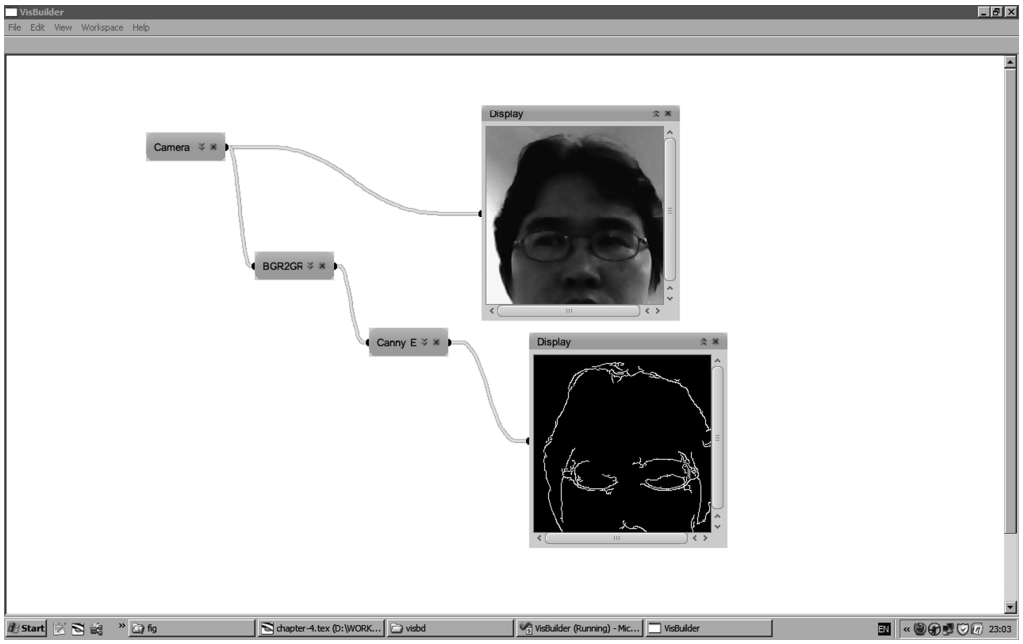
VisBuilder ได้รับการออกแบบให้ทำงานกับข้อมูลภาพอย่างต่อเนื่อง ในหลายส่วนของซอฟต์แวร์นี้มีการจองหน่วยความจำไว้ล่วงหน้าหรือการสร้างบัฟเฟอร์เพื่อบรรจุข้อมูลสำหรับการประมวลผล หน่วยความจำหรือบัฟเฟอร์ดังกล่าวจะถูกใช้งานซ้ำแล้วซ้ำอีกตลอดเวลาที่มีการเรียกใช้ฟังก์ชัน ที่มีความเกี่ยวข้องกับบัฟเฟอร์นี้การจองหน่วยความจำขนาดใหญ่สำหรับการใช้งานในช่วงเวลาสั้น ๆ ในระหว่างการทำงานแล้วคืนกลับให้ระบบ การทำเช่นนี้หลาย ๆ ครั้งจะส่งผลกระทบต่อความเร็ว นการทำงานโดยรวมของซอฟต์แวร์ดังนั้นการจองหน่วยความจำไว้ล่วงหน้าเพื่อการประมวลผล แล้วคืนให้ระบบเมื่อองค์ประกอบนั้น ๆ ถูกลบออกจากระบบจะช่วยเพิ่มประสิทธิภาพการทำงานให้ดียิ่งขึ้น

4.4 สรุป

ซอฟต์แวร์ VisBuilder ที่สร้างขึ้นเป็นไปตามความต้องการของระบบตามที่ได้กล่าวไว้ในบทที่ 3 ซึ่งรองรับการทำงานแบบระบบงานพร้อมกันโดยการใช้เธรดพัฒนาในลักษณะขององค์ประกอบย่อย สามารถขยายความสามารถของระบบโดยการเพิ่มองค์ประกอบย่อยเข้าไป มีความยืดหยุ่นในการใช้งานรองรับการทำงานบนระบบปฏิบัติการต่าง ๆ ดังแสดงในรูปที่ 4.11 และรูปที่ 4.12 คลาสในเฟรมเวิร์คสำหรับการรับภาพ การประมวลผลภาพ และการแสดงผลภาพสร้างจากการอนุพันธ์จากคลาสฐานที่ชื่อว่า Component ซึ่งมีความสามารถในการทำงานแบบระบบงานพร้อมกันทำให้การรับภาพ การประมวลผลภาพต่าง ๆ และการแสดงผลทำงานได้พร้อมกัน ผู้พัฒนาระบบการมองเห็นของคอมพิวเตอร์สามารถเพิ่มคการประมวลผลหรือการเชื่อมต่อกับอุปกรณ์ต่าง ๆ ได้โดยใช้ต้นแบบที่ได้เตรียมไว้ให้



รูปที่ 4.11 การทำงานของ VisBuilder บนระบบปฏิบัติการลินุกซ์



รูปที่ 4.12 การทำงานของ VisBuilder บนระบบปฏิบัติการวินโดวส์

ความสามารถในการทำงานในลักษณะระบบงานพร้อมกันช่วยลดความล่าช้าในการประมวลผลภาพและลดความซับซ้อนของระบบความล่าช้าเป็นผลมาจากความจำเป็นในการรอผลการประมวลผลภาพและการแสดงภาพ ก่อนที่จะรับภาพใหม่เข้ามา องค์ประกอบย่อยการประมวลผลบางชนิดอาจทำงานได้เร็วกว่าองค์ประกอบย่อยอื่น ๆ ดั้งนั้นการทำงานได้พร้อมกันขององค์ประกอบย่อยจะช่วยลดความล่าช้าในการทำงานโดยรวมลงไปได้มาก

VisBuilder พัฒนาในลักษณะแยกการทำงานออกเป็นส่วนย่อยหรือองค์ประกอบย่อย แต่ละส่วนมีความอิสระจากกัน ซึ่งช่วยให้การจัดการองค์ประกอบย่อยและการพัฒนาร่วมกับชุดพัฒนาอื่น ๆ โดยการเรียกใช้ภายในองค์ประกอบย่อยจะทำได้ง่าย การเปลี่ยนแปลงแก้ไขต้นรหัสใด ๆ ที่เกิดขึ้นกับองค์ประกอบย่อยจะไม่ส่งผลกระทบกับส่วนอื่น ๆ ของซอฟท์แวร์ องค์ประกอบย่อยดังกล่าวสามารถนำไปใช้ร่วมกับ VisBuilder เดิมโดยไม่ต้องคอมไพล์ใหม่ในกรณีขององค์ประกอบย่อยที่สร้างขึ้นใหม่สามารถนำมาใช้ได้ทันทีซึ่งมีการทำงานคล้ายกับการต่อปลั๊กของอุปกรณ์ใหม่เข้ากับระบบเดิมที่มีอยู่

การออกแบบ VisBuilder ใช้การแบ่งออกเป็นเลเยอร์ซึ่งมี 5 เลเยอร์โดยเลเยอร์ที่อยู่ด้านบนสร้างจากเลเยอร์ที่อยู่ด้านล่างและไม่มีการใช้ตัวแปลส่วนกลางเพื่อให้เหมาะสมกับการพัฒนาแบบแยกเป็นส่วนย่อยองค์ประกอบย่อยต่าง ๆ สำหรับรับภาพประมวลผลภาพ และแสดงผลมีการทำงานที่อิสระจากกันและมีต้นรหัสที่ง่ายต่อการทำความเข้าใจ

การคอมไพล์ต้นรหัสขององค์ประกอบย่อยต่าง ๆ สามารถเลือกคอมไพล์เฉพาะส่วนที่เหมาะสมกับระบบปฏิบัติการนั้น ๆ หรือให้สอดคล้องกับชุดพัฒนาที่เกี่ยวข้องได้โดยทั้วไปการคอมไพล์สามารถทำได้สองวิธี คือ การใช้ Makefile หรือการใช้ Project file ซึ่ง VisBuilder รองรับทั้งสองวิธี

VisBuilde มีความสามารถในการขยายระบบรองรับการพัฒนาบนสภาวะแวดล้อมของการพัฒนาได้หลายแบบทั้งการพัฒนาบนระบบปฏิบัติการวินโดวส์และระบบปฏิบัติการลินุกซ์และรองรับการสร้างองค์ประกอบย่อยการรับภาพ การประมวลผลภาพ และการแสดงผลภาพเพิ่มเติมเข้าไปในระบบ

การสื่อสารระหว่างองค์ประกอบย่อยใช้หลักการไหลของข้อมูลซึ่งได้สร้างไว้ภายในผู้ใช้ไม่มีความจำเป็นต้องเขียนโปรแกรมสำหรับการส่งข้อมูลระหว่างองค์ประกอบด้วยตนเอง การสื่อสารดังกล่าวจะเกิดขึ้น ภายในโดยไหลไปตามจุดเชื่อมต่อระหว่างองค์ประกอบย่อย การเพิ่มชนิดข้อมูลใหม่สามารถทำได้ผ่านทาง การอนุพันธ์จากคลาสที่เตรียมไว้ให้

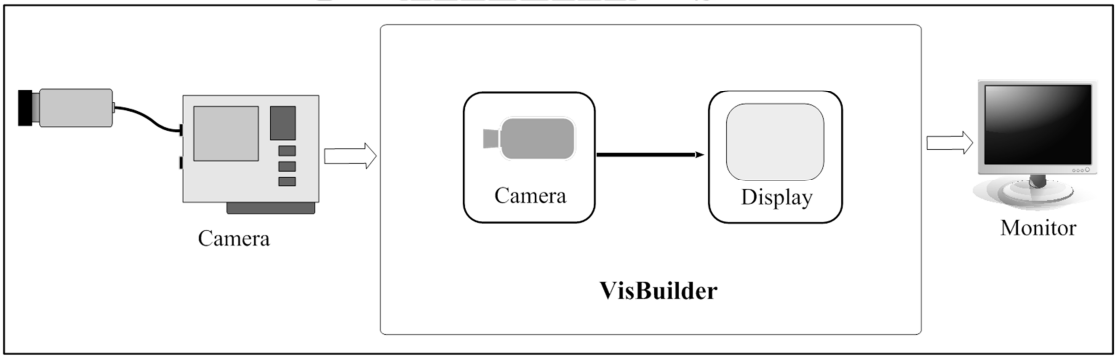
มหาวิทยาลัยเทคโนโลยีสุรนารี

บทที่ 5
การทดสอบ VisBuilder

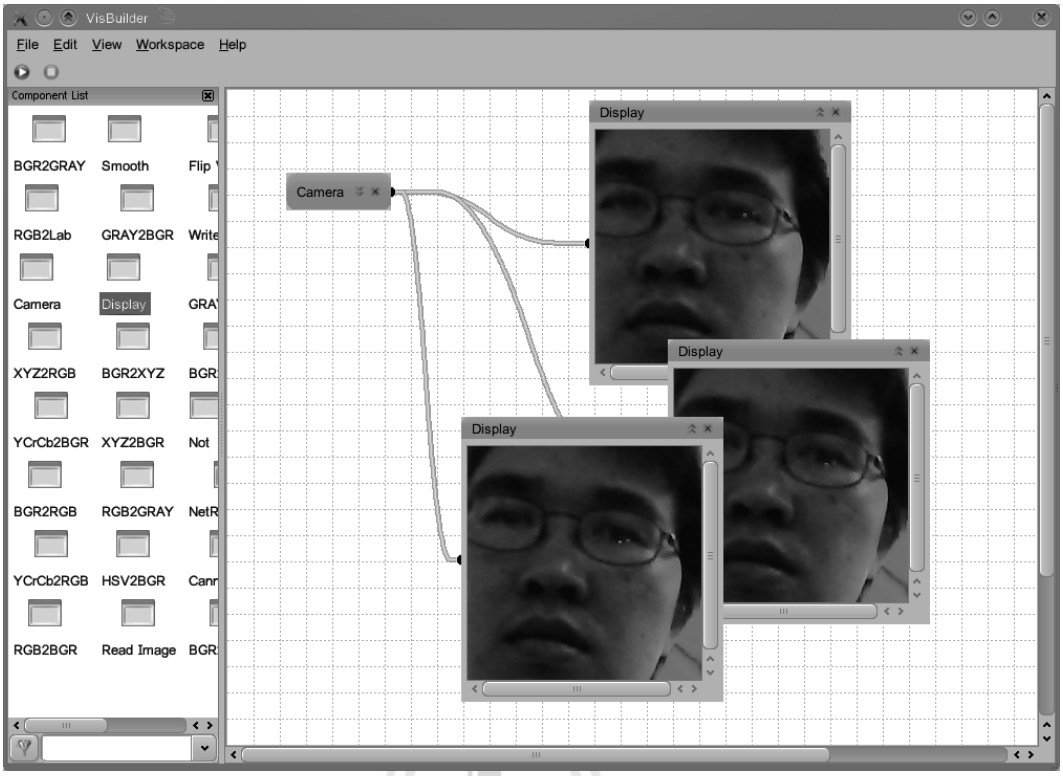
ในบทนี้จะได้ทำการทดสอบซอฟต์แวร์ VisBuilder ที่ได้พัฒนาขึ้น โดยจะทดสอบการใช้งานเบื้องต้นที่เป็นการใช้งานพื้นฐาน เช่น การเชื่อมต่อกับอุปกรณ์รับภาพ การตรวจจับวัตถุ การเชื่อมต่อผ่านระบบเครือข่ายคอมพิวเตอร์เป็นต้น และรวมถึงการวัดประสิทธิภาพการทำงานของซอฟต์แวร์นี้

5.1 การเชื่อมต่อกับอุปกรณ์รับภาพ

การเชื่อมต่อกับอุปกรณ์รับภาพแสดงการทำงานดังรูปที่ 5.1 การทำงานจะเริ่มจากการรับภาพจากองค์ประกอบย่อยการรับภาพที่เรียกว่า Camera แล้วส่งต่อให้กับองค์ประกอบย่อยการแสดงผล คือ Display ซึ่งสามารถเชื่อมต่อกับองค์ประกอบการแสดงผลได้หลายตัวพร้อมกัน รูปที่ 5.2 แสดงการทำงานรับภาพจากกล้องโดยใช้ VisBuilder จะเห็นว่าสามารถแสดงผลได้หลายตัวพร้อมกัน จำนวนองค์ประกอบการแสดงผลสามารถเพิ่มได้อย่างไม่จำกัดเพียงแต่จะถูกกำหนดด้วยขนาดของหน่วยความจำของเครื่องคอมพิวเตอร์ที่ใช้ และยังขึ้นอยู่กับประสิทธิภาพในการประมวลผลของซีพียูด้วย



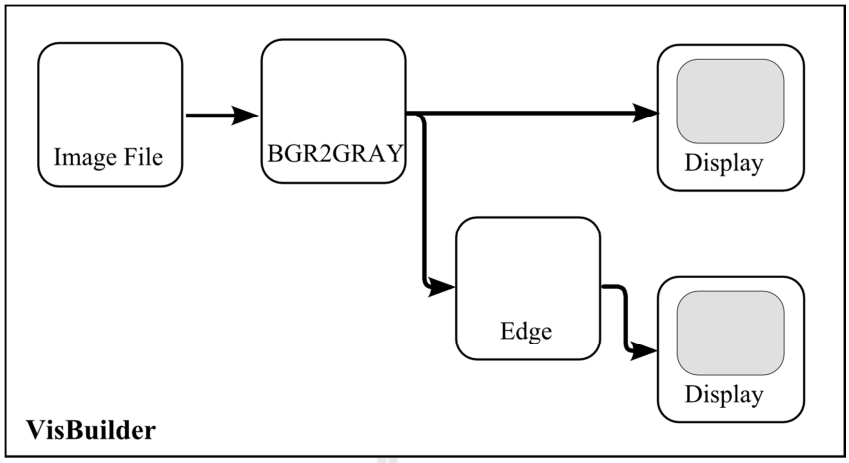
รูปที่ 5.1 ขั้นตอนการใช้งานร่วมกับอุปกรณ์รับภาพ



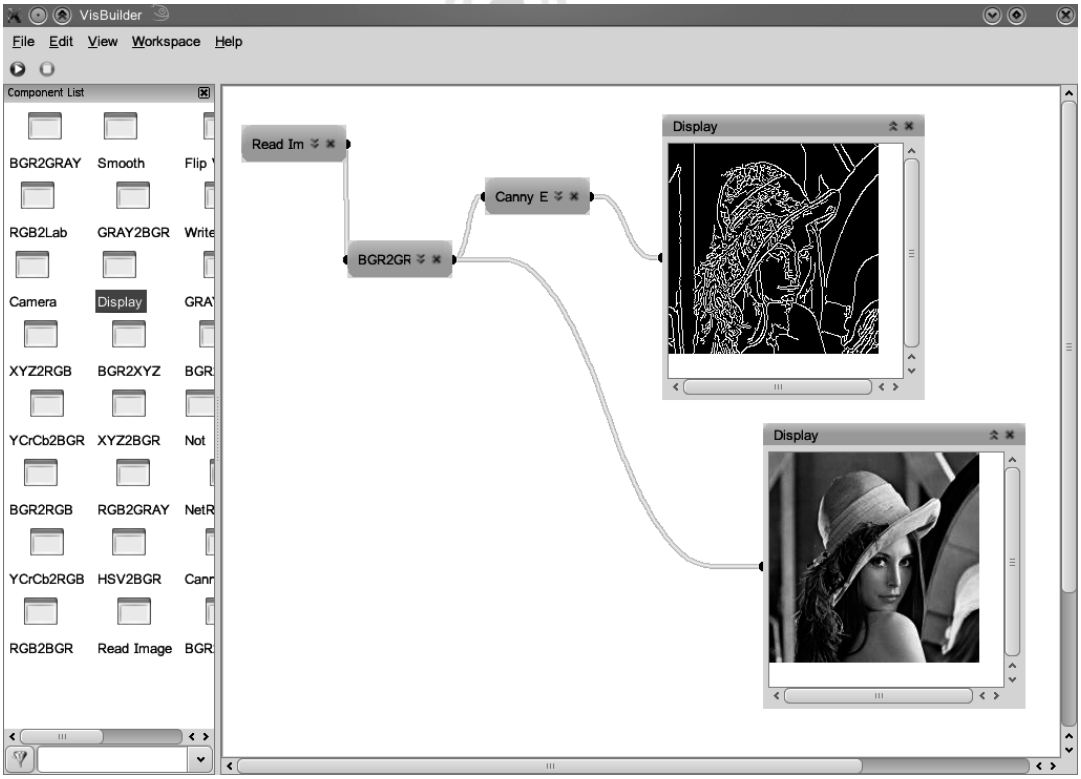
รูปที่ 5.2 การเชื่อมต่อกับอุปกรณ์รับภาพด้วย VisBuilder

5.2 การหาขอบภาพ

การหาของภาพด้วยซอฟต์แวร์ VisBuilder แสดงขั้นตอนดังรูปที่ 5.3 ขึ้นตอนเริ่มด้วยการอ่านภาพจากไฟล์ภาพแล้วแปลงเป็นภาพสีเทาด้วยองค์ประกอบย่อย BGR2GRAY การใช้ BGR เนื่องจาก OpenCV ใช้การลำดับชั้นของสีไม่เหมือนโปรแกรมโดยทั่วไปที่เริ่มด้วย RGB หรือสีแดงมาก่อน เมื่อผ่านการแปลงภาพแล้วส่งไปให้องค์ประกอบย่อย CannyEdge สำหรับการหาขอบภาพแล้วส่งต่อไปให้องค์ประกอบการแสดงผล การเชื่อมต่อด้วย VisBuilder แสดงดังรูปที่ 5.4



รูปที่ 5.3 ขั้นตอนการหาขอบภาพ



รูปที่ 5.4 การหาขอบภาพด้วย VisBuilder

5.3 การเชื่อมต่อผ่านเครือข่ายคอมพิวเตอร์

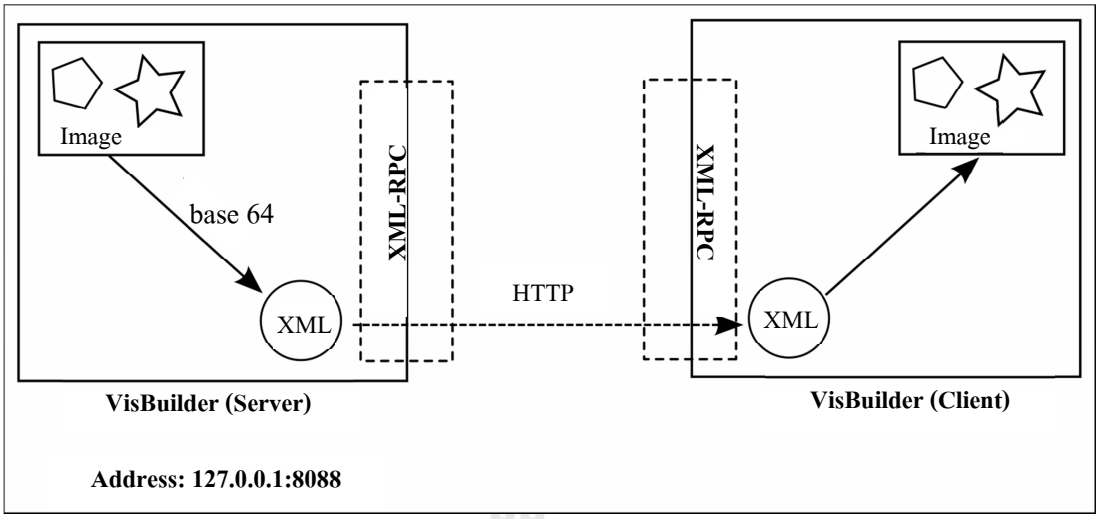
ในการทำงานร่วมกันของ VisBuilder ผ่านระบบเครือข่ายคอมพิวเตอร์จะใช้วิธีการเชื่อมต่อผ่านโปรโตคอล HTTP โดยใช้ XMLRPC ช่วยในการสื่อสารโดยข้อมูลที่รับส่งจะแปลงเป็นข้อมูลในรูปแบบ XML ข้อมูลรูปภาพที่เป็นบิตจะถูกแปลงเป็นตัวอักษรโดยใช้วิธีแปลงแบบ base 64 การทำงานแสดงดังรูปที่ 5.5

ในการทดสอบจะกำหนดให้ทำงานภายในเครื่องเดียวกันเพื่อให้สะดวกในการทดสอบการทำงาน ซึ่งผลที่ได้จะไม่แตกต่างกันในแง่ของของการทำงาน แต่เมื่อพิจารณาถึงความเร็วในการส่งข้อมูล การจำลองการทำงานภายในเครื่องเดียวกันจะให้ผลรวดเร็วกว่า ซึ่งในที่นี้ไม่ได้สนใจในแง่ของประสิทธิภาพในการทำงาน เป็นการทำการทดลองเพื่อยืนยันความสามารถในการเชื่อมต่อผ่านระบบเครือข่ายคอมพิวเตอร์เท่านั้น

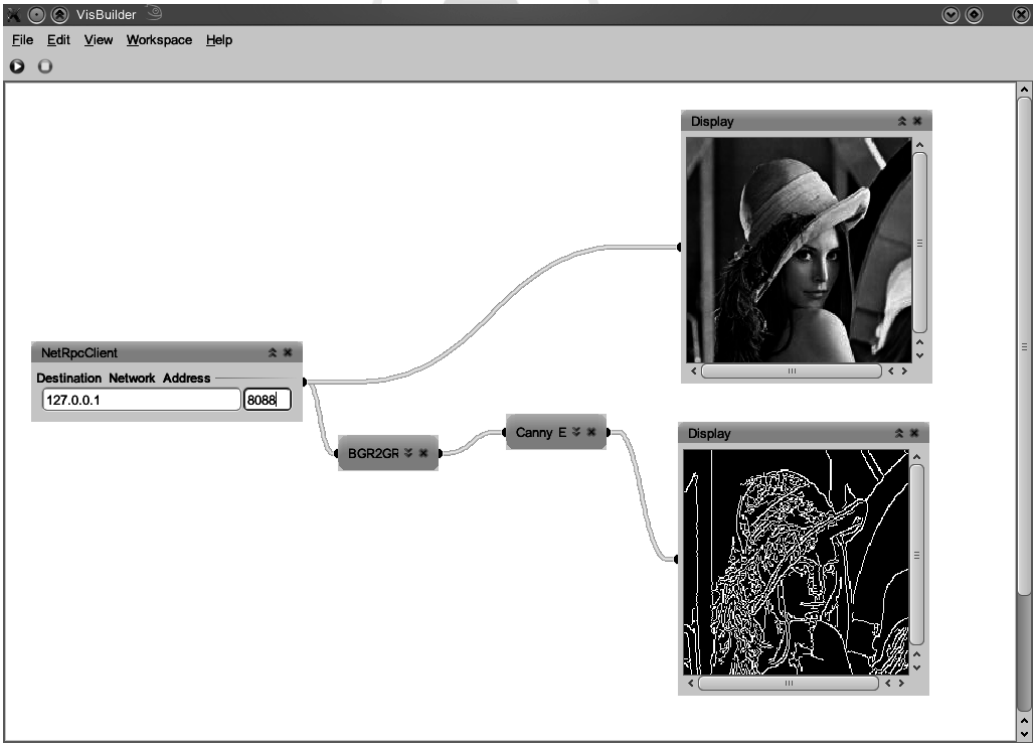
จากรูปที่ 5.6 เป็นการสร้างระบบการเชื่อมต่อผ่านระบบเครือข่ายคอมพิวเตอร์ด้วย VisBuilder โดยกำหนดให้ระบบนี้ทำหน้าที่เป็นเครื่องแม่ข่าย โดยการทำงานจะเริ่มจากการอ่านข้อมูลภาพด้วยองค์ประกอบย่อย ReadImgae นำมาแสดงผล พร้อมกับส่งต่อไปยังองค์ประกอบย่อย NetRpcServer เพื่อส่งข้อมูลภาพผ่านระบบเครือข่าย ซึ่งข้อมูลภาพดังกล่าวจะยังไม่ถูกส่งไปจนกระทั่งมีการร้องขอข้อมูลจากทางฝั่งลูกข่าย องค์ประกอบย่อย NetRpcServer จึงจะส่งผ่านข้อมูลภาพไปให้ องค์ประกอบย่อย NetRpcServer นี้จะต้องมีการกำหนดหมายเลขพอร์ทสำหรับการเชื่อมต่อผ่านเครือข่ายซึ่งจะต้องใช้หมายเลขเดียวกันกับเครื่องลูกข่าย ในที่นี้กำหนดให้เป็นหมายเลข 8088

ในรูปที่ 5.7 เป็นการสร้างระบบด้วย VisBuilder ให้ทำหน้าที่ดึงข้อมูลจากระบบเครือข่ายคอมพิวเตอร์หรือทำหน้าที่เป็นเครื่องลูกข่ายนั่นเอง โดยขั้นตอนจะเริ่มจากการที่องค์ประกอบย่อย NetRpcClient เชื่อมต่อผ่านระบบเครือข่ายคอมพิวเตอร์ไปที่หมายเลขเครือข่ายตามที่กำหนด คือ 127.0.0.1 ซึ่งเป็นหมายเลขเครือข่ายของเครื่องแม่ข่าย ซึ่งในที่นี้เป็นหมายเลขเครือข่ายที่อ้างอิงถึงเครื่องที่ใช้งาน VisBuilder ซึ่งเป็นการอ้างอิงถึงเครื่องตนเอง และต้องใช้หมายเลขพอร์ตเดียวกับเครื่องแม่ข่ายที่จะเชื่อมต่อไป คือ 8088 เมื่อได้รับข้อมูลภาพมาแล้วจะนำไปแสดงผล พร้อมกับประมวลภาพเบื้องต้น คือ ทำการแปลงข้อมูลภาพเป็นสีเทาด้วยองค์ประกอบ BGR2GRAY และส่งไปที่องค์ประกอบ CannyEdge สำหรับหาขอบของภาพแล้วจึงนำไปแสดงผล

การทดสอบที่ทำขึ้นนี้ใช้การประมวลภาพแบบง่ายเพื่อแสดงถึงความสามาถในการทำงานร่วมกันของซอฟต์แวร์ VisBuilder โดยผ่านระบบเครือข่ายคอมพิวเตอร์ การทดสอบกับงานที่เป็นแบบทันเวลา (real-time) จะได้ทำการพัฒนาและทดสอบต่อไปในอนาคตของการวิจัยพัฒนาซอฟต์แวร์นี้



รูปที่ 5.5 ขั้นตอนการเชื่อมต่อผ่านระบบเครือข่าย

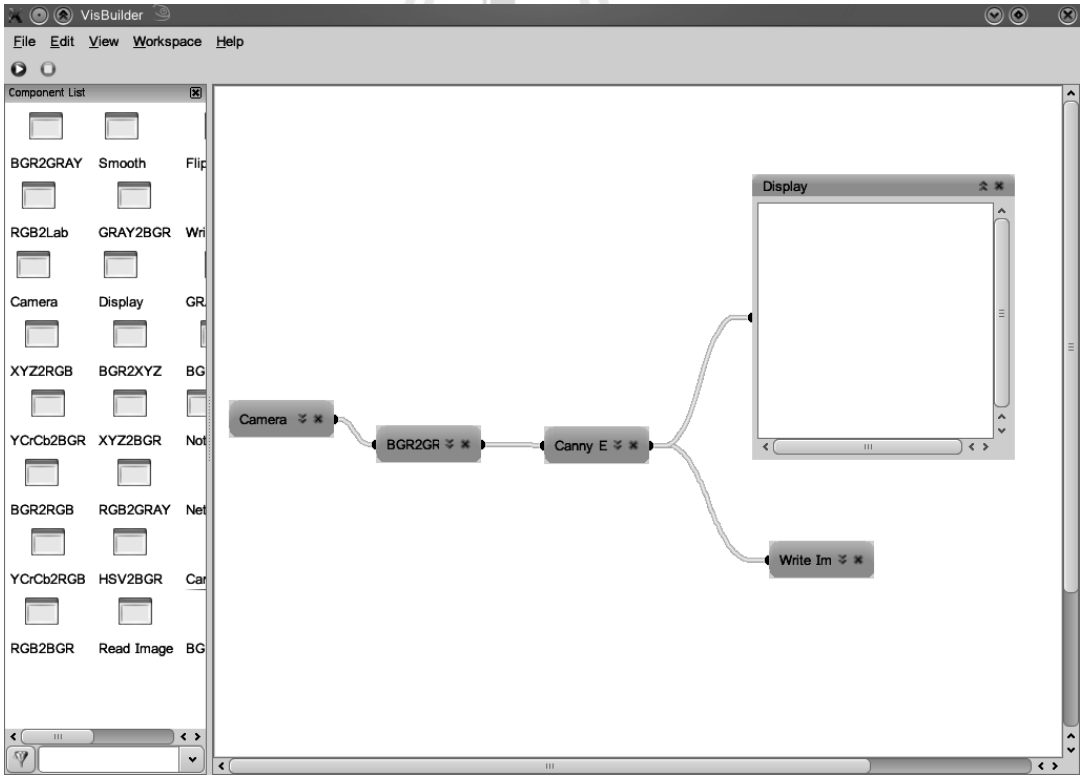


รูปที่ 5.6 การกำหนดให้ VisBuilder ทำหน้าที่เป็นแม่ข่าย

5.4 การวัดประสิทธิภาพการทำงาน

การวัดประสิทธิภาพการทำงานของซอฟต์แวร์ VisBuilder จะใช้การประเมินความเร็วในการรับภาพจากอุปกรณ์รับภาพ โดยจะทำการนับจำนวนเฟรมของภาพที่สามารถประมวลผลได้ในเวลาที่กำหนด

ในการทดสอบหาความเร็วในการทำงานของ VisBuilder จะใช้การนับจำนวณเฟรมของภาพที่ได้ในระยะเวลาการประมวลผลที่กำหนด การทำงานแสดงดังรูปที่ 5.8 โดยจะเริ่มจากการที่ VisBuilder รับภาพจากอุปกรณ์รับภาพเข้ามาผ่านการใช้องค์ประกอบย่อย Camera ส่งต่อไปให้กับองค์ประกอบย่อย BGR2GRAY สำหรับแปลงภาพเป็นภาพสีเทา แล้วผ่านไปให้กับองค์ประกอบย่อย CannyEdge เพื่อหาของภาพแล้วนำไปบันทึกเป็นไฟล์ภาพพร้อมทั้งแสดงผลภาพไปด้วย และนับจำนวนไฟล์ภาพที่ได้ ผลการทดสอบแสดงดังตารางที่ 5.1 ซึ่งได้ทำการทดสอบเป็นจำนวน 10 ครั้ง แล้วคำนวณหาจำนวนเฟรมที่ได้ต่อวินาที จากนั้นนำมาหาค่าเฉลี่ยซึ่งได้ค่าเท่ากับ 2 เฟรมต่อวินาที



รูปที่ 5.7 การวัดประสิทธิภาพในการประมวลผลของ VisBuilder

ตารางที่ 5.1 การหาความเร็วในการทำงานของ VisBuilder

ครั้งที่	เวลาในการทดสอบ (วินาที)	จำนวนเฟรมภาพที่ประมวลได้	เฟรมต่อวินาที
1	10	21	2.1
2	20	33	1.65
3	30	63	2.1
4	40	78	1.95
5	50	91	1.82
6	60	123	2.05
7	70	147	2.1
8	80	167	2.08
9	90	183	2.03
10	100	211	2.11

5.5 สรุป

ซอฟต์แวร์ VisBuilder สามารถทำงานในการประมวลผลภาพเบื้องต้นได้ในระดับหนึ่ง ซึ่งความสามารถดังกล่าวยังคงมีจำกัดอยู่เนื่องจากองค์ประกอบการประมวลผลยังมีน้อย จึงจำเป็นต้องพัฒนาเพิ่มเติมซึ่งต้องใช้เวลาในการทำความเข้าใจฟังก์ชันการประมวลผลภาพของ OpenCV ที่ซอฟต์แวร์นำมาใช้ต้องศึกษาทดสอบก่อนการนำมาใช้ใน VisBuilder

ซอฟต์แวร์นี้สามารถเชื่อมต่ออุปกรณ์รับภาพต่าง ๆ ที่สามารถเชื่อมต่อผ่านเข้ากับระบบปฏิบัติการลินุกส์ได้ซึ่งเป็นไปตามข้อตกลงดังได้กล่าวไว้แล้วในบทที่ 1

VisBuilder ยังมีความสามารถในการทำงานร่วมกันโดยเชื่อมต่อผ่านระบบเครือข่ายคอมพิวเตอร์ ซึ่งได้แสดงการทดสอบการทำงานไปแล้วข้างต้น
ประสิทธิภาพของ VisBuilder ยังไม่มีความล่าช้าในการประมวลผลอยู่ค่อนข้างมาก
ซึ่งในขณะนี้สามารถประมวลผลในลักษณะการทำงานที่มีรอบการทำงานหลายครั้งได้
แต่ยังไม่เพียงพอต่อสำหรับการประมวลผลในแบบทันเวลาซึ่งต้องใช้ความรวดเร็วในการประมวลผลสูงมาก

มหาวิทยาลัยเทคโนโลยีสุรนารี

บทที่ 6
สรุปผลงานวิจัยและข้อเสนอแนะ

6.1 สรุปผลงานวิจัย

งานวิจัยนี้เป็นการสร้างซอฟต์แวร์ให้เป็นทางเลือกที่เพิ่มขึ้นสำหรับผู้พัฒนาระบบการมองเห็นของ คอมพิวเตอร์ซึ่งสอดคล้องกับการวิเคราะห์ความต้องการของระบบต่าง ๆ ที่สำคัญและมีความจำเป็นสำหรับงานด้านนี้ได้แก่การรองรับขั้นตอนการทำงานพื้นฐาน สำหรับงานด้านนี้ความสามารถในการขยายระบบให้รองรับความต้องการใช้งานที่มากขึ้น การแยกการพัฒนาเป็นส่วนย่อยความสามารถในการทำงานได้บนระบบปฏิบัติการที่หลากหลาย และความสามารถในการทำงานได้พร้อมกันขององค์ประกอบย่อย

ผลของการวิจัยนี้ทำให้ได้ซอฟต์แวร์ที่ชื่อ VisBuilder ซึ่งรองรับความต้องการของระบบดังกล่าวข้างต้น รองรับการทำงานจริงดังที่ได้แสดงไว้ในบทที่ 4 มีการจัดเตรียมองค์ประกอบพื้นฐานที่จำเป็นไว้ให้การโปรแกรมเชิงภาพของ VisBuilder ช่วยให้การพัฒนาระบบการมองเห็นของคอมพิวเตอร์ทำได้ง่ายและรวดเร็ว การใช้งานองค์ประกอบสำหรับการประมวลผลภาพผ่านทางการแทนด้วยภาพ ช่วยให้ผู้ใช้มองเห็นกระบวนการทำงานอย่างชัดเจนจึงสามารถมุ่งความสนใจไปในส่วนการพัฒนาอัลกอริทึมของการประมวลผลภาพเพียงอย่างเดียวไม่ต้องกังวลถึงการเขียนโปรแกรม และในกรณีที่ต้องการใช้งานที่มากกว่าที่จัดเตรียมไว้ผู้ใช้งานสามารถเพิ่มเติมความสามารถเข้าไปได้ โดยได้จัดเตรียมโครงร่างสำหรับการสร้างองค์ประกอบย่อยไว้ให้

ซอฟต์แวร์นี้ยังอยู่ในขั้นงานวิจัยเท่านั้น ซึ่งยังมีความไม่สมบูรณ์อยู่รวมถึงคุณสมบัติต่าง ๆ ที่ยังขาด การพัฒนาเพิ่มเติมอย่างต่อเนื่องจึงมีความจำเป็น สำหรับแนวทางการพัฒนาได้กล่าวไว้แล้วในบทที่ 6.2 โดยมีเป้าหมายหลักที่กำลังอยู่ในการพัฒนาชุดซอฟต์แวร์นี้คือความสามารถในการสร้างองค์ประกอบใหม่จากองค์ประกอบที่มีอยู่โดยผู้ใช้ไม่จำเป็นต้องเขียนรหัสเอง ความสามารถนี้จะทำให้การพัฒนาสร้าง ระบบการมองเห็นด้วยคอมพิวเตอร์เป็นไปอย่างสะดวกและมีประสิทธิภาพมากยิ่งขึ้น

การพัฒนา VisBuilder เพิ่มเติมยังคงต้องดำเนินการต่อไปซึ่งมีสิ่งที่ต้องทำหลายอย่างมากกว่าที่ได้กล่าวไว้ในบทนี้ทิศทางของการพัฒนาดังจะได้กล่าวในหัวข้อถัดไปเป็นเป้าหมายหลักเพื่อให้ VisBuilder เป็นซอฟต์แวร์ที่ดีมีความสมบูรณ์เหมาะสมกับงานด้านการพัฒนาระบบการมองเห็นของคอมพิวเตอร์ซึ่งต้องใช้เวลาอีกนานพอสมควรกว่าจะพัฒนาได้สมบูรณ์ทั้งหมด โดยจะพัฒนาให้ครอบคลุมการทำงานสำหรับ ระบบการมองเห็นของคอมพิวเตอร์ให้มากที่สุดเท่าที่จะเป็นไปได้

ถึงแม้ว่า VisBuilder จะมีเป้าหมายอยู่ที่งานพัฒนาระบบการมองเห็นของคอมพิวเตอร์ แต่งานด้านอื่นที่มีความใกล้เคียงกันก็อาจนำ VisBuilder ไปใช้ได้ด้วยเช่นกัน เช่น การประมวลผลสัญญาณเสียง เป็นต้น

6.2 ทิศทางของการพัฒนา

มีหลายสิ่งหลายอย่างซึ่งยังไม่ได้รับการพัฒนาสำหรับ VisBuilder ในบทนี้จะได้กล่าวถึงทิศทางและแนวทางในการพัฒนาเพิ่มเติม เพื่อขยายความสามารถและเพื่อให้รองรับงานวิจัยระบบการมองเห็นของคอมพิวเตอร์ได้ดียิ่งขึ้น ยังมีอีกหลายส่วนที่ยังไม่ได้ศึกษาและพัฒนา เช่น ชนิดข้อมูลอื่น ๆ ที่นอกเหนือจากข้อมูลภาพหรือวิดีโอ เป็นต้น ในขณะนี้ VisBuilder ยังไม่ได้มีประสิทธิภาพการทำงานที่ดีมากนัก การปรับปรุงประสิทธิภาพดังกล่าวมีความสำคัญเป็นอย่างมาก

6.2.1 การใช้ Design Pattern

การนำ design pattern มาใช้กับ VisBuilder จะช่วยยืนยันและตรวจสอบความเหมาะสมในการออกแบบที่เป็นอยู่ในขณะนี้และยังสามารถนำมาช่วยออกแบบคลาสที่มีอยู่อีกครั้งเพื่อให้มีความถูกต้องเหมาะสม การใช้ design pattern จะช่วยปรับปรุงประสิทธิภาพของ VisBuilder ซึ่งจะช่วยให้ซอฟต์แวร์นี้มีความเหมาะสมกับงานด้านการพัฒนาระบบการมองเห็นของคอมพิวเตอร์มากขึ้น

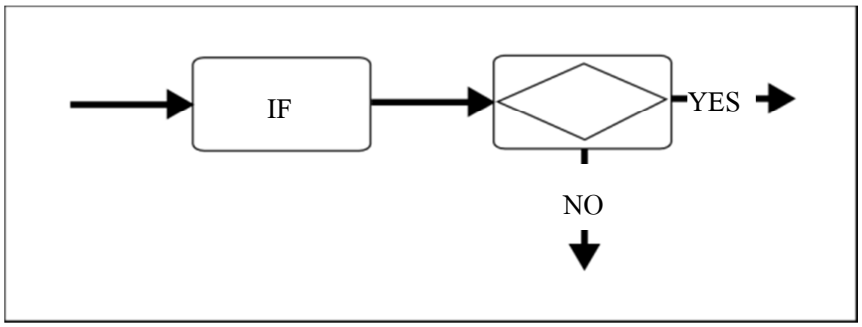
6.2.2 การนำ VisBuilder ไปใช้กับงานด้านอื่น

ยังมีงานหลายประเภทที่อาจจะนำ VisBuilder ไปใช้ได้ เช่น การประมวลผลสัญญาณเสียงเป็นต้น การพัฒนา VisBuilder ให้รองรับงานด้านอื่นที่มีลักษณะงานคล้ายคลึงกับงานพัฒนาระบบการมองเห็น ของคอมพิวเตอร์มีความเป็นไปได้อาจทำโดยการเพิ่มชนิดข้อมูลของงานนั้น ๆ เข้าไป และเพิ่มองค์ประกอบย่อยสำหรับงานนั้นตามความเหมาะสม

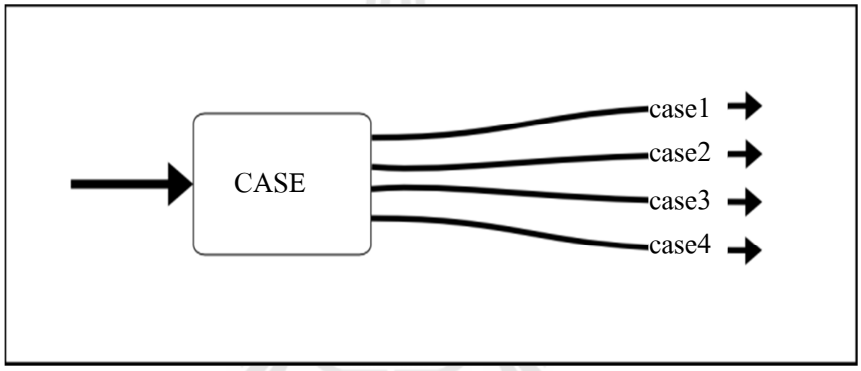
6.2.3 การเพิ่มองค์ประกอบการควบคุม

การทำงานที่ขาดไม่ได้สำหรับการโปรแกรมก็คือ การดำเนินตามเงื่อนไข ความสามารถนี้มีความจำเป็นมาก เพราะไม่ใช่ทุกกรณีที่องค์ประกอบย่อยชุดหนึ่ง ๆ จะดำเนินการประมวลผล ซึ่งขึ้นอยู่กับเงื่อนไขหรือยังกำหนดในการทำงานขณะนั้น เช่น การตรวจจับวัตถุในภาพ อาจมีการดำเนินการที่แตกต่างกันในกรณีที่พบวัตถุกับไม่พอวัตถุเป็นต้น

องค์ประกอบย่อยที่ควรสร้างเพิ่มในที่นี้มี 2 ชนิด คือ องค์ประกอบย่อยสำหรับตรวจสอบเงื่อนไขเดียว แสดงในรูปที่ 6.1 และองค์ประกอบย่อยสำหรับการเลือก แสดงในรูปที่ 6.2 การเพิ่มองค์ประกอบย่อยดังกล่าวจะช่วยให้ VisBuilder มีความยืดหยุ่นในการใช้งานมากขึ้น



รูปที่ 6.1 องค์ประกอบย่อยการตรวจสอบเงื่อนไขเดียว



รูปที่ 6.2 องค์ประกอบย่อยการเลือก

6.2.4 การเพิ่มเติมส่วนต่อประสานกราฟิกกับผู้ใช้

ส่วนต่อประสานกราฟิกกับผู้ใช้ของ VisBuilder ยังไม่สมบูรณ์มากนักยังคงต้องออกแบบเพิ่มเติม และเพิ่มความสามารถที่ยังขาดเข้าไป เช่น การค้นหา การเรียงลำดับและการจัดกลุ่มองค์ประกอบย่อย รวมถึงการปรับปรุงความสวยงามต่าง ๆ ของ VisBuilder เช่น ไอคอน เป็นต้น

ในขณะนี้ส่วนต่อประสานกราฟิกกับผู้ใช้ของ VisBuilder ยังคงผูกติดหรือรวมอยู่กับส่วนการทำงาน หรือส่วนประมวลผลในลักษณะเป็นก้อนเดียวกัน ซึ่งจะสร้างข้อจำกัดในการนำ VisBuilder ไปใช้งาน เช่น หากต้องการสร้างระบบการมองเห็นของคอมพิวเตอร์ที่ประมวลผลเพียงอย่างเดียวโดยไม่ต้องการ การแสดงผลผ่านส่วนต่อประสานกราฟิกกับผู้ใช้ในกรณีนี้ VisBuilder ยังไม่รองรับ

การแยกส่วนต่อประสานกราฟิกกับผู้ใช้ออกจากส่วนการทำงานหรือการประมวลผลจะส่งผลดีหลายอย่าง ตัวอย่างเช่น การปรับเปลี่ยนส่วนต่อประสานกราฟิกกับผู้ใช้ทำได้ง่ายไม่ส่งผลกระทบกับระบบการทำงานเดิม การนำระบบการมองเห็นที่ออกแบบไปใช้กับงานจริงโดยให้ทำงานอยู่เบื้องหลังเป็นต้น

รายการอ้างอิง

AccuSoft (2010). **Accusoft visiquest 4.1 matlab integration**.

AccuSoft (2010). **Visiquest : A problem solving environment for scientific computing and visualization**.

Agilent, T. I. (2 0 1 0) . **Agilent vee**. http://www.home.agilent.com/agilent/product.jspx?nid=-34095.0.00 &cc=US&lc=eng. [Online; accessed 14-April-2010].

Allman, M. (2003). **An evaluation of xml-rpc**. SIGMETRICS Perform. Eval. Rev., 30(4):2–11.

Azad, P. (2010). **Integrating vision toolkit (ivt)**. http://ivt.sourceforge.net/index.html. [Online; accessed 14-April-2010].

Bourdev, L., and Jin, H. (2010). **Adobe generic image library (gil)**. http://opensource.adobe.com /wiki/display/gil/Generic+Image+Library. [Online; accessed 14-April-2010].

Brinkman, T. (2006). **Boost mailing page : Gil accepted**. http://lists.boost.org/Archives/boost/2006/11/112896.php. [Online; accessed 15-May-2010].

Brownell, D. (2004). **Sax: Simple api for xml**. http://www.saxproject.org/. [Online; accessed 14-May-2010].

C++, B. (2010). **Boost library documentation**. http://www.boost.org/doc/libs/. [Online; accessed 12-May-2010].

Caluwaerts, L. J., Debacker, J., and Peperstraete, J. A. (1983). Implementing streams on a data flow computer system with paged memory. **In ISCA '83: Proceedings of the 10th annual international symposium on Computer architecture.**: 76–83.

Vaughan, G. V., Ben, E, T. T., and Taylor, I. L. (2007). **Gnu autoconf, automake and libtool**. http://sourceware.org/autobook/.

GraphicsMagick (2010). **Graphicsmagick image processing system**. http://www.graphicsmagick .org/index.html. [Online; accessed 14-April-2010].

Henderson, B. (2004). **Xml-rpc for c and c++**. http://xmlrpc-c.sourceforge.net. [Online; accessed 14-April-2010].

Hunt, N. (1990). IDF: A graphical data flow programming language for image processing and computer vision. **In Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on**.: 351–360.

IBM (2010). Opendx - **the open source software project based on ibm's visualization data explorer**. http://www.opendx.org/. [Online; accessed 14-April-2010].

ImageMagick (2010). **Imagemagick**. http://www.imagemagick.org/script/index.php. [Online; accessed 14-April-2010].

Imaging, C. (2003). **WiT Getting Started**. Coreco Imaging Inc., Quebec, Canada, 8.0 edition.

Instruments, N. (2003). LabVIEW User Manual. National Instruments Corporate.

Instruments, N. (2005). **NI Vision for LabVIEWTM User Manual**. National Instruments Corporate.

Instruments, N. (2010). **Ni labview**. http://www.ni.com/labview/. [Online; accessed 14-April-2010].

Intel (2001-2003). **Integrated performance primitives for Intel architecture - reference manual**. INTEL Corporation.

Josuttis, N. M. (1999). **The C++ standard library: a tutorial and handbook**. Addison-Wesley, illustrated edition.

Karlsson, B. (2005). **Beyond the C++ Standard Library: An Introduction to Boost**. Addison Wesley Professional.

Koelma, D., Van, B. R., and Smeulders, A. (1992). SCIL-VP : a multi-purpose visual programming environment. **Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's.**: 1188–1198.

Konstantinides, K., and Rasure, J. (1994). The khoros software development environment for image and signal processing. **Image Processing, IEEE Transactions on**, 3:243 – 252.

Lourens, T. (2004). TiViPe - tino's visual programming environment. **In Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC2004).**: 10–15.

Lourens, T. (2006). **Installation of TiViPe**. http://www.dei.brain.riken.jp/~emilia/Collaboration/Tino/TiViPE/download/readme.html. [Online; accessed 14-April-2010].

Luis Ib´a˜nez, Will, S. L. N. J. C., and the Insight Software Consortium. (2005). **The ITK Software Guide**, second edition.

Manolescu, D. A., and Nahrstedt, K. (1998). A scalable approach to continuous-media processing. **In Research Issues In Data Engineering, 1998.'Continuous-Media Databases and Applications'. Proceedings., Eighth Internatinal Workshop on**,: 84–91.

Marini, J. (2002). **Document Object Model: Processing Structured Documents**. McGraw-Hill/Osborne, illustrated edition.

MathWorks (2010). **Matlab-simulink**. http://www.mathworks.com/products/simulink/. [Online; accessed 14-April-2010].

OpenCV (2009). **OpenCV 2.0 - The Reference Manual**. OpenCV (2010). Open source computer vision library (opencv). http://sourceforge.net/projects/opencvlibrary/. [Online; accessed 14-April-2010].

Paul, F., Whelan, R. J. S., and Ghita, O. (2004). **NeatVision - visual programming for computer aideddiagnostic applications**. Radiographics, 24.: 1779–1789.

Pegasus, A. (2010). **Visiquest**. http://www.accusoft.com/. [Online; accessed 14-April-2010].

PythonWare (2010). **Python imaging library (pil)**. http://www.pythonware.com/products/pil/. [Online; accessed 14-April-2010].

Richard, V. B., Dennis, K. T. K. t. K. B. M., and Smeulders, A. W. (1994). Scilimage: A multi-layered environment for use and development of image processing software. **Experimental Environments for Computer Vision and Image Processing**.:107–126.

Sanner, M., Stoffler, D., and Olson, A. (2002). Viper, a visual programming environment for python. **In Proceedings of the 10th International Python conference**.: 103–115.

Shuigen, W., Zhen, C., and Hua, D. (2009). Motion detection based on temporal difference method and optical flow field. **In Electronic Commerce and Security, 2009. ISECS '09. Second International Symposium on**, volume 2.: 85 –88.

Smart, J., Hock, K., and Csomor, S. (2000). **Cross Platform GUI Programming with wxWidgets (Bruce Perens Open Source)**. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Smith, S. W. (1997). **The Scientist and Engineer's Guide to Digital Signal Processing**. California Technical Publishing, San Diego, California.

T. Bray, J. Paoli, C. M. S.-M. and Maler, E. (2000). **Extensible markup language (xml) 1.0 (second edition)**. Technical report, WorldWideWeb Consortium.

VisionSystemsGroup (2010). **Neatvision: Image analysis and software development environment**. http://ww.neatvision.com.

Whelan, P. F. and Sadleir, R. J. (2004). **A visual programming environment for machine vision engineers**. Sensor Review, 24(3).: 265–270.

Wikipedia (2010a). **Dataflow**. http://en.wikipedia.org/wiki/Dataflow. [Online; accessed 15-May 2010].

Wikipedia (2010b). **Labview : wikipedia, the free encyclopedia**. http://en.wikipedia.org/w/index.php?title=LabVIEW&oldid=361991658. [Online; accessed 14-April-2010].

Young-Seok Sim, Chae-Seong Lim, Y.-S. M. and Park, S.-H. (1996). Design and implementation of the visual processing environment for the distributed image processing. **In IEEE International Conference on Image Processing**, volume 1.: 149–152.

ภาคผนวก ก

บทความวิชาการที่ได้รับการตีพิมพ์เผยแพร่

รายชื่อบทความวิชาการที่ได้รับการตีพิมพ์เผยแพร่

ภาณุพงษ์ เพชรเลิศ และอาทิตย์ ศรีแก้ว, **VisBuilder: สภาวะแวดล้อมเชิงวิชวลสำหรับการพัฒนาระบบการมองเห็นของคอมพิวเตอร์**, การประชุมวิชาการนานาชาติร่วมสาขาวิทยาการคอมพิวเตอร์และวิศวกรรมซอฟต์แวร์ครั้งที่ 7 ประจำปี 2553 (JCSSE 2010). มหาวิทยาลัยรามคำแหง, กรุงเทพ , 12-14 พฤษภาคม 2553.

ภาคผนวก ข

ประวัติผู้วิจัย

ประวัติผู้วิจัย

ดร.อาทิตย์ ศรีแก้ว เกิดเมื่อวันที่ 19 พฤศจิกายน พ.ศ. 2515 สำเร็จการศึกษาระดับปริญญาตรีในสาขาวิชาวิศวกรรมอิเล็กทรอนิกส์จาก สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เมื่อ พ.ศ. 2537 สำเร็จการศึกษาระดับ ปริญญาโทและปริญญาเอกสาขาวิศวกรรมไฟฟ้าจาก Vanderbilt University ประเทศสหรัฐอเมริกา เมื่อ พ.ศ. 2540 และ 2543 ตามลำดับ ปัจจุบันดำรงตำแหน่งรองศาสตราจารย์ สาขาวิชาวิศวกรรมไฟฟ้า สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี มีความสนใจงานวิจัยทางด้านการมองเห็นของคอมพิวเตอร์และหุ่นยนต์ การประมวลผลภาพ และระบบทางปัญญาประดิษฐ์

นายภาณุพงษ์ เพชรเลิศ เกิดเมื่อวันที่ 28 มิถุนายน พ.ศ.2522 ณ จังหวัดชลบุรี สำเร็จการศึกษาระดับชั้นมัธยมศึกษา จากโรงเรียนพนัสพิทยาคาร อ.พนัสนิคม จ.ชลบุรี ในปีการศึกษา 2540 และสำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตร์บัณฑิต (วิศวกรรมไฟฟ้า) สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารีในปีการศึกษา 2545 หลังจากนั้นได้เข้าศึกษาต่อในระดับปริญญาโท สาขาวิชาวิศวกรรมไฟฟ้า สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารีในปีการศึกษา 2547 มีความสนใจในการเขียนโปรแกรมตั้งแต่เริ่มเข้าศึกษาในระดับปริญญาตรี โดยได้มีโอกาสฝึกฝนทักษะทางด้านนี้จากรุ่นพี่และตำราต่างๆ และเมื่อขึ้นชั้นปีที่ ได้สมัครเป็นผู้ช่วยวิจัยให้กับอาจารย์ ดร.กัณทิมา ศิริจิระชัย ซึ่งในขณะนั้นกำลังศึกษาอยู่ในระดับปริญญาเอกสาขาเคมีที่มหาวิทยาลัย Dalhousie ประเทศแคนาดา โดยได้รับผิดชอบในการพัฒนาโปรแกรมจำลองระบบเคมีโดยใช้ PBE (Population Balance Equation) ทำให้ได้รับความรู้ในหลายด้าน อาทิ การโปรแกรมด้วยภาษา Fortran C และ MATLAB ร่วมไปถึงระเบียบวิธีเชิงตัวเลข ทำให้มีความสนใจในการศึกษาต่อในระดับสูงขึ้นไป