โครงการวิจัยย่อยลำดับที่ 19

เรื่อง การแบ่งย่อยภาพ Magnetic Resonance ปีที่ 2

(Magnetic Resonance Image Segmentation)

1. ผู้รับผิดชอบโครงการ อาจารย์ ดร. ชาญชัย ปลื้มปิติวิริยะเวช

บทนำ

เทคโนโลยีการถ่ายภาพสะท้อนแม่เหล็ก (Magnetic Resonance Imaging)หรือ MRI ได้ถูกนำมาใช้ในวงการแพทย์อย่างแพร่หลายมากขึ้นเรื่อย ๆ ในปัจจุบัน ทั้งนี้เพราะ มันมีศักยภาพสูงในการนำมาวิเคราะห์เพื่อวินิจฉัยโรคต่าง ๆ ในขั้นเบื้องต้นได้ดีโดยไม่จำเป็นต้องมีการผ่าตัด ในช่วงแรก MRI ถูกนำมาใช้ในการถ่ายภาพสมอง เพื่อศึกษาปฏิกิริยาทางเคมีและปฏิกิริยาทางกายภาพในส่วนต่าง ๆ ของระบบสมอง ซึ่งเป็นผลเนื่องมาจากยาบางชนิดหรือสิ่งเร้าทางกายภาพจากภายนอก ภาพ magnetic resonance ของสมองยังถูกนำมาใช้ในการศึกษาการทำงานของสมองขณะนึกคิดหรือปฏิบัติงานบางอย่าง และตรวจหาความผิดปกติของสมองด้วย ต่อมาเทคโนโลยีการถ่ายภาพสะท้อนแม่เหล็ก MRI นี้ได้ถูกพัฒนาให้ดีขึ้นเรื่อย ๆ จนสามารถนำมาใช้ถ่ายภาพอวัยวะนุ่ม (soft tissue) อื่น ๆ ได้อีก เช่น หัวใจ และ ตับ เป็นต้น

ไม่นานมานี้MRI ได้ถูกนำมาใช้ในการถ่ายภาพหัวใจเพื่อศึกษาการทำงานของหัวใจ รวมถึงการตรวจหาความผิดปกติของหัวใจในผู้ป่วยโรคหัวใจ นอกจากนี้ปัจจุบันยังมีงานวิจัยใหม่ ๆ ที่นำเอา MRI มาใช้ในการดูแล(monitor) การทำงานของหัวใจในผู้ป่วยที่เพิ่งผ่านการผ่าตัดเปลี่ยนหัวใจโดยไม่ต้องใช้วิธีสุ่มเนื้อเยื่อหัวใจออกมาตรวจ (Biopsy) อีกด้วย งานวิจัยเหล่านี้จำเป็นต้องมีการศึกษาและวิเคราะห์ผลจากภาพถ่าย magnetic resonance เป็นจำนวนมาก เนื่องจากข้อมูลภาพหัวใจเหล่านี้ถูกบันทึกตลอดระยะเวลาการเต้นของหัวใจ และขั้นตอนเบื้องต้นที่สำคัญที่สุดในการวิเคราะห์ภาพเหล่านี้คือการแบ่งย่อยภาพ (image segmentation) เพื่อให้ได้มาซึ่งส่วนต่าง ๆ ของหัวใจเพื่อนำไปวิเคราะห์ข้อมูลการทำงานที่ถูกต้องของหัวใจต่อไป ทว่าวิธีการแบ่งย่อยภาพหัวใจที่ใช้กันอยู่ในปัจจุบันคือให้ผู้เชี่ยวชาญตรวจหาส่วนต่าง ๆ ของหัวใจในภาพ magnetic resonance แต่ละภาพแล้วทำการแบ่งย่อยภาพด้วยการวาดเส้นแบ่งเองด้วยมือ (manual segmentation) แม้วิธีนี้จะให้ผลที่มีความถูกต้องมาก การแบ่งย่อยภาพด้วยมือนี้ ล้าช้า ใช้แรงงานอย่างมาก และขาดประสิทธิภาพในการคำนวณ จึงมีความจำเป็นอย่างยิ่งที่ควรนำเทคนิคการปฏิบัติการทางภาพ (image processing) โดยคอมพิวเตอร์เข้ามาช่วย เพื่อให้ผู้เชี่ยวชาญสามารถวิเคราะห์ ข้อมูลชุดภาพ magnetic resonance ของหัวใจได้เร็วขึ้น และมีประสิทธิภาพมากขึ้น ที่สำคัญ เครื่องมือแบ่งย่อยภาพ magnetic resonance ของหัวใจนี้ควรจะสามารถแบ่งย่อย ส่วนต่าง ๆ ของหัวใจได้เองให้ได้มากที่สุด และพึ่งการทำงานกับผู้เชี่ยวชาญ (expert interaction) เฉพาะในส่วนที่จะเป็นจริง ๆ เท่านั้น

2.วัตถุประสงค์ของโครงการวิจัย

2.1 ศึกษา และ วิเคราะห์การปรับคุณภาพของภาพ magnetic resonance ของหัวใจให้ดีขึ้น (image enhancement)

2.2 ศึกษา และ วิเคราะห์วิธีการแบ่งย่อยภาพ magnetic resonance ของหัวใจ (cardiac MR image segmentation)

2.3 พัฒนาและเพิ่มเติมเทคนิค active contour เพื่อใช้ในการแบ่งย่อยภาพ magnetic resonance ให้มีประสิทธิภาพมากขึ้น เพื่อให้ได้มาซึ่งอวัยวะหรือสิ่งต่างๆของหัวใจที่สนใจ เพื่อนำไปวิเคราะห์ข้อมูลต่อไป เช่นการไหลเวียนของโลหิต เพื่อตรวจหาโรคหัวใจ เป็นต้น

2.4 พัฒนาเครื่องมือการแบ่งย่อยภาพ magnetic resonance ที่ง่ายต่อการใช้งาน

2.5 พัฒนาวิธีทดสอบ และวัดประสิทธิภาพ การใช้งานของเครื่องมือแบ่งย่อยภาพ

3.ขอบเขตของโครงการวิจัย

ในโครงงานวิจัยนี้ เราจะเน้นที่การวิเคราะห์หา กระบวนการแบ่งย่อยภาพที่มีประสิทธิภาพเมื่อใช้กับภาพ magnetic resonance, ซึ่งมีลักษณะเฉพาะที่แตกต่างจากภาพอื่นๆ โดยทั่วไป ในขั้นเบื้องต้น เราจะเน้นการค้นหาอวัยวะหัวใจในภาพ magnetic resonanceเป็นหลัก หลังจากนั้น เราจะออกแบบสร้างเครื่องมือการแบ่งย่อยภาพ magnetic resonanceที่ง่ายต่อการใช้งาน เพื่อเป็นประโยชน์ให้แพทย์ผู้เชี่ยวชาญนำไปวิเคราะห์ตรวจหาโรคหัวใจต่อไป

4.ผลการดำเนินการ

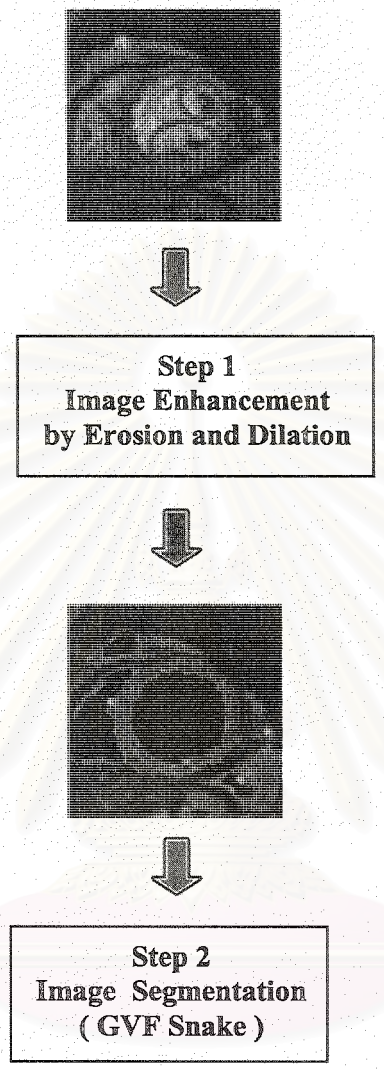
4.1 ศึกษาและวิเคราะห์การปรับคุณภาพของภาพ magnetic resonance ของหัวใจให้ดีขึ้น (image enhancement)

การปรับคุณภาพของภาพให้ดีขึ้นนั้น มีความสำคัญอย่างยิ่งซึ่งจะส่งผลถึงผลลัพธ์ถึงการแบ่งย่อยภาพ (image segmentation) ในขั้นตอนถัดไปด้วย ดังแสดงในรูป 4.1

ใน step ที่ 1 นี้ จะเป็นการทำ Image Enhancement โดยมีจุดประสงค์หลักเพื่อต้องการที่จะกำจัดจุดด่างดำ ซึ่งเป็นส่วนของ papillary muscles ดังปรากฏภายในวงกลมสีแดงของภาพบนสุด ในรูปที่ 4.1 ในเพราะจุดด่างดำพวกนี้จะผลกระทบอย่างมากในการทำ Image Segmentation ด้วยวิธี GVF snake ใน Step 2

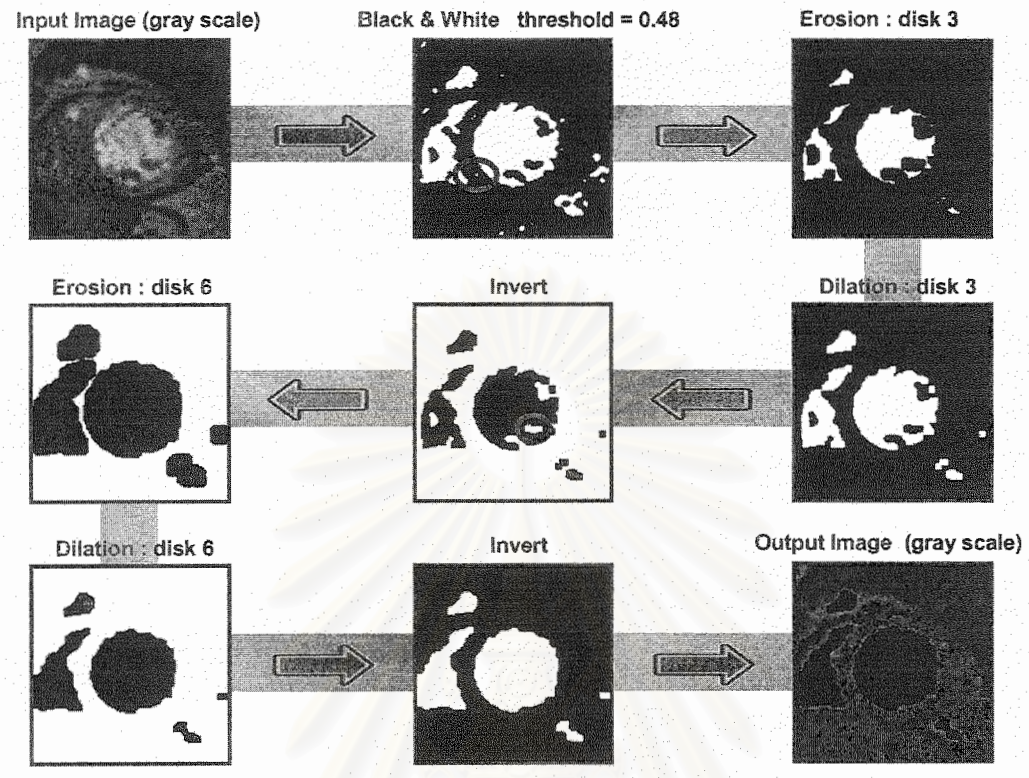
ในการทำ image enhancement หรือ pre-processing นี้ของโครงงานนี้ เราได้เลือกใช้ Morphological Operations 2 ตัว ด้วยกัน ก็คือ Erosion และ Dilation โดยมีกระบวนการคราวๆดังนี้ คือ เราจะเริ่มต้นโดยการทำภาพ gray scale ให้เป็นภาพขาวดำ จากนั้นจะทำ Erosion เพื่อกร่อนจุดด่างดำพวกนั้นให้หายไป และจากนั้นเราจะทำ Dilation เพื่อขยายส่วนอื่นๆให้กลับมาสู่สภาพเดิม ซึ่งจุดด่างดำที่ถูกกร่อน

ไปจนหมดแล้วก็จะไม่ถูกขยายกลับคืนมา ทำให้เราได้ภาพที่ไม่มีจุดด่างดำพวกนั้นแล้ว ดังแสดงในกลางถัดจาก บล็อก step 1 ในรูปที่ 4.1 จากนั้นเราก็จะส่งต่อไปยัง Step ที่ 2



รูปที่ 4.1 แสดงบล็อกไดอะแกรมแสดงผลการทำงานของ step 1 : Image Enhancement

ในการทำ Erosion และ Dilation นี้ จะเลือกใช้ Structuring element แบบ disk เพราะภาพที่ได้ออกมาจะมีลักษณะโดยรวมมนๆ ซึ่งจะดูดีกว่าแบบ Square เพราะภาพที่จะได้จะเป็นเหลี่ยม ๆ ต่อไปจะอธิบายวิธีการทำ Step1 และผลการทดลองโดยละเอียด



รูปที่ 4.2 แสดงผลการทดลองของ step 1 : Image Enhancement

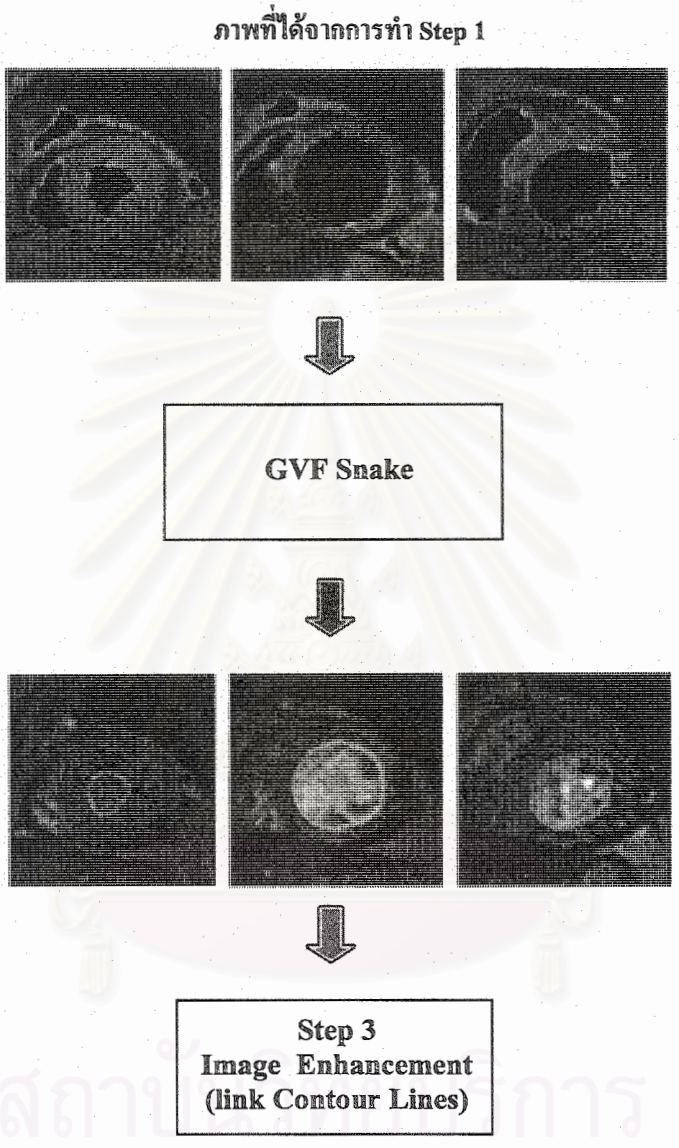
จากรูปที่ 4.2 จะเป็นผลการทดลองที่ได้จาก Step 1 ซึ่งในรูปแรกจะเป็นภาพ Input ที่เป็นภาพ gray scale เราจะต้องทำการแปลงภาพให้เป็นภาพขาวดำ โดยใช้ ค่า thresholds เป็นตัวแบ่งระหว่างสีขาวกับสีดำ ซึ่งในการเลือกค่า thresholds นี้ จะมีความสำคัญอย่างมากในการทำ Step นี้ เพราะถ้าเลือกค่า thresholds ที่ไม่เหมาะสมจะทำให้ได้ภาพ output ที่ไม่ดี เช่น ทำให้ลบจุดด่างดำไม่หมด หรืออาจจะทำให้หัวใจห้องล่างขาวกับหัวใจห้องล่างซ้ายติดกันได้ หรืออาจจะทำให้ในเลือกค่า Disk ของการทำ Erosion และ Dilation นั้นเลือกได้ยากขึ้น ซึ่งในที่นี้ได้เลือกใช้ค่า thresholds = 0.48 ซึ่งทำให้ได้ผลดีมาก ไม่เกิดการติดกันของห้องหัวใจและยังสามารถใช้ค่าเดียวกันนี้ได้กับภาพหัวใจ ทั้ง 3 slice

เมื่อเราได้ภาพขาวดำมาแล้วจากนั้น เราจะเริ่มกำจัดจุดที่เราไม่ต้องการ โดยจากภาพที่สอง ในวงกลมสีแดงจะเห็นได้ว่าหลังจากทำการแปลงภาพ gray scale ให้เป็นภาพขาวดำนั้น จะเกิดจุดสีขาวขึ้นระหว่างหัวใจห้องล่างขาวกับห้องล่างซ้ายซึ่งจุดเหล่านี้ในการทำ Project ในตอนแรกได้มองข้ามไป โดยที่ไปเริ่มกำจัดจุดที่อยู่ในหัวใจห้องล่างซ้ายเลย ซึ่งได้เกิดปัญหาขึ้นในภายหลัง ปัญหาก็คือ ในการที่ทำ Erosion และ Dilation นั้น จุดสีขาวดังกล่าวได้ถูกผลกระทบจากการทำ Erosion และ Dilation ทำให้ห้องหัวใจล่างขาวและล่างซ้ายเกิดการติดกันขึ้นมา ดังนั้นจึงได้มีการทำ Erosion และ Dilation สองครั้งด้วยกัน โดยในครั้งแรกจะเป็นการกำจัดจุดที่อยู่ระหว่างห้องหัวใจ เพื่อไม่ให้เกิดปัญหาขึ้นอีกในภายหลัง โดยจะทำ Erosion และ Dilation ซึ่งได้เลือกใช้ Disk ที่มีขนาดรัศมีเท่ากับ 3 เพราะจุดที่อยู่ระหว่างห้องหัวใจนั้นมี

ขนาดไม่ใหญ่มาก จากนั้นเราก็จะกำจัดจุดที่อยู่ในหัวใจห้องล่างซ้ายโดยจะต้องมีการแปลงภาพที่ได้จากการกำจัดจุดที่อยู่ระหว่างห้องหัวใจ กลับขาวเป็นดำ กลับดำเป็นขาว เพราะในการเขียนโปรแกรมนั้นจุดที่เราต้องการที่จะกำจัดต้องเป็นสีขาว ในการกำจัดจุดในห้องล่างซ้ายนี้ เราได้เลือกใช้ Disk ที่มีขนาดรัศมีเท่ากับ 6 เพราะจุดดังกล่าวมีขนาดค่อนข้างใหญ่ ซึ่งขนาดของ Disk ก็เป็นอีกพารามิเตอร์หนึ่งที่ต้องมีการเลือกใช้อย่างเหมาะสม ในแต่ละภาพของ Input เพราะอาจจะมีผลกระทบทำให้การกำจัดจุดที่เราไม่ต้องการออกไป ไม่หมด หรือทำให้ห้องหัวใจทั้งสองห้องติดกัน ในที่นี้ ค่าที่เลือกใช้คือ 3 กับ 6 เป็นค่าที่ทำให้ได้ภาพ output อย่างที่เราต้องการ และสามารถใช้ค่าเดียวกันนี้ได้กับทั้ง 3 slice

จากนั้นเราก็ทำการ กลับขาวเป็นดำ กลับดำเป็นขาวอีกครั้งหนึ่ง เพราะในขั้นตอนสุดท้ายของ Step 1 นี้ การเขียนโปรแกรมได้มีการกำหนดให้ส่วนที่เป็นสีขาวของภาพขาวดำ จะเป็นตัวที่ทำให้ที่ pixel ที่ตำแหน่งเดียวกันของภาพ Input จะเป็นสีดำ เราก็จะได้ภาพ Output ที่เป็นภาพ gray scale เหมือนกับภาพ input แต่จะต่างกันตรงที่ บริเวณ ที่เป็นหัวใจห้องล่างซ้ายและห้องล่างขาวจะเป็นสีดำ สาเหตุที่เลือกให้ห้องหัวใจเป็นสีดำเพราะ จะทำให้ง่ายต่อการทำงานใน step ที่ 2 ซึ่งจะได้กล่าวต่อไป

4.2 **ศึกษา และ วิเคราะห์วิธีการแบ่งย่อยภาพ magnetic resonance ของหัวใจ (cardiac MR image segmentation)**

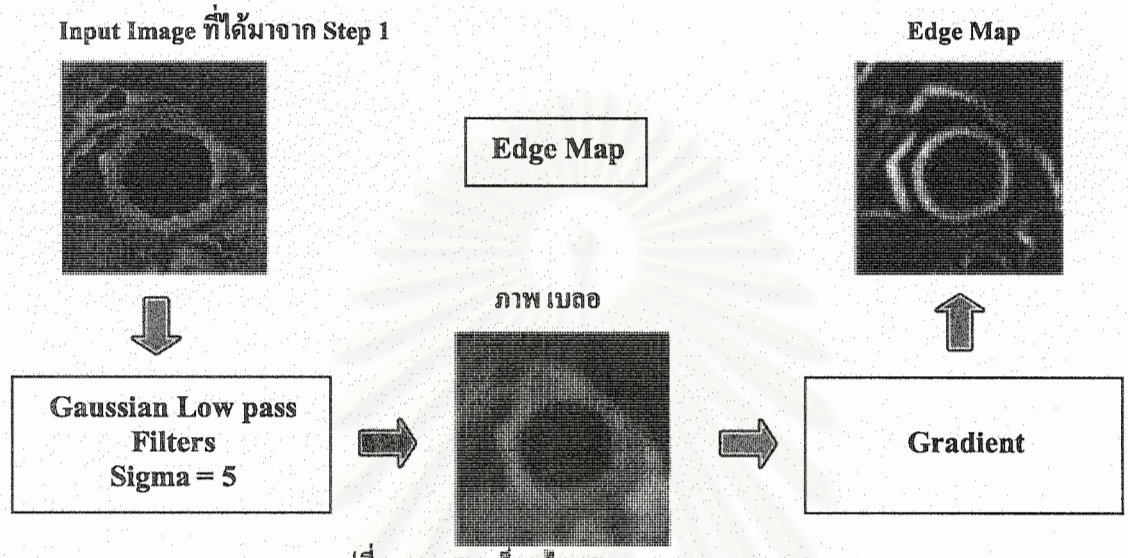


รูปที่ 4.2 แสดงบล็อกไดอะแกรมการทำงานของ **step 2 : Image Segmentation**

ใน Step ที่ 2 นี้จะเป็นการทำ Image Segmentation คือเป็นการแยกส่วนที่เราต้องการและสนใจออกมาจากภาพ Input และส่วนที่เราสนใจใน Project นี้ ก็คือ หัวใจห้องล่างซ้าย โดยเลือกใช้วิธี GVF snake [5] ในการทำ Segmentation ซึ่งภาพ input ของ Step ที่ 2 นี้ ก็คือภาพที่ได้มากจาก Step ที่ 1 นั่นเอง โดยวิธี GVF Snake นี้ สามารถเป็นออกเป็นขั้นตอนใหญ่ๆได้ 3 ขั้นตอน คือ 1.Edge Map 2. GVF field และ 3. Snake Deform และหลังจากที่เราผ่านขั้นตอนต่างๆ เหล่านั้นมาแล้วเราจะได้ Contour line ออกมา 3 Contour lines ด้วยกัน (1 slice ได้ 1 contour) จากนั้นเราก็จะส่ง Contour lines

เหล่านี้ไปยัง Step ที่ 3 เพื่อเข้ากระบวนการในการสร้างภาพ 3 มิติ ต่อไปจะอธิบายวิธีการทำ Step2 และผลการทดลองโดยละเอียด

ดังที่กล่าวไว้แล้วว่า วิธี GVF Snake นี้ สามารถเป็นออกเป็นขั้นตอนใหญ่ๆได้ 3 ขั้นตอน คือ 1.Edge Map 2. GVF field และ 3. Snake Deform โดยจะเริ่มอธิบายในขั้นตอนแรกก็คือ Edge Map โดยภาพ Input คือภาพที่ได้จาก Step ที่ 1



รูปที่ 4.3 แสดงบล็อกไดอะแกรมของการหา Edge Map

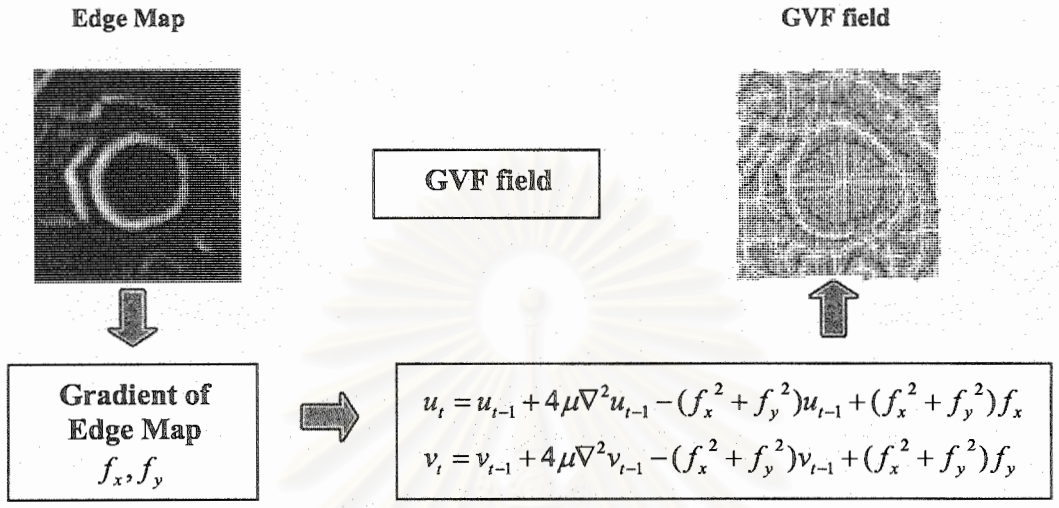
ในขั้นตอนแรกเราจะนำภาพที่ได้จาก Step ที่ 1 มาผ่าน Gaussian Low pass Filters โดยใช้ค่า sigma เท่ากับ 5 ซึ่งค่า sigma ก็คือค่า standard deviation ของ Gaussian Function นั่นเอง ภาพที่ได้จะเบลอ และถ้าค่า sigma ยิ่งมีค่ามาก ภาพที่ได้ก็จะเบลอมากขึ้นตาม สาเหตุที่เราต้องทำแบบนี้ก็เพราะ เราต้องการที่จะให้ขอบของรายละเอียดในภาพใหญ่ขึ้นและมนขึ้นไม่เป็นยักๆบริเวณขอบ หลังจากที่เราได้ภาพ เบลอ มาแล้ว เราก็จะทำการหาขอบของภาพโดยใช้ gradient ซึ่งมีสูตรดังนี้

$$|\nabla f| = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

โดยที่ $f(x,y)$ คือ ภาพเบลอ และ $|\nabla f|$ คือ Edge Map

หลังจากที่เราได้ Edge Map ที่เราต้องการแล้ว เราก็จะส่ง Edge Map ที่ได้ไปยัง ขั้นตอนที่ 2 ก็คือ GVF field ซึ่งจะอธิบายในหน้าถัดไป

จากในขั้นตอนที่แล้วเมื่อเราได้ Edge Map มาแล้วเราก็จะทำการคำนวณค่า GVF ซึ่ง GVF ย่อมาจาก Gradient Vector Flow ซึ่งคำนวณได้จากสูตรข้างล่างนี้ (จาก Paper[5])



รูปที่ 4.4 แสดงบล็อกไดอะแกรมของการหา GVF field

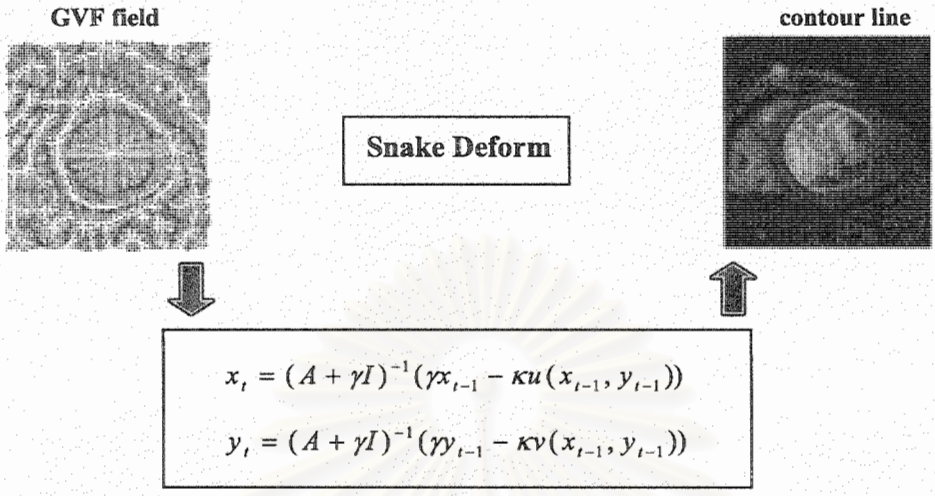
จากสูตรจะเห็นได้ว่ามีค่า พารามิเตอร์อยู่ตัวหนึ่งที่เราต้องกำหนดค่าให้ นั่นก็คือ μ : regularization parameter ซึ่งในการกำหนดค่า μ นั้นก็สำคัญมากเพราะถ้าเลือกค่า μ ที่ไม่เหมาะสมจะทำให้การเคลื่อนที่ ของ Snake ในขั้นตอนต่อไปจะไม่สามารถเคลื่อนที่ไปยังบริเวณที่เราต้องการได้

$\mu = 0.05$ $\mu = 0.3$

รูปที่ 4.5 GVF filed ที่ค่า μ ต่างๆ

จาก GVFfield ทั้งสองรูปข้างบนจะเห็นได้ว่าถ้าเราเลือกค่า μ ที่ไม่เหมาะสม(μ = 0.3) GVFfield ที่ได้ก็จะมีลักษณะที่แตกต่างออกไปจากภาพ Edge Map อย่างมาก คือมองไม่ออกเลยว่าส่วนใดเป็นขอบของหัวใจห้องล่างซ้าย แต่ถ้าเราเลือกใช้ μ ที่เหมาะสมกับภาพ($\mu = 0.05$) เราก็จะสังเกตเห็นได้ว่า GVFfield จะมีลักษณะเหมือนกับ Edge Map โดยส่วนที่เป็นขอบนั้น จะเป็นบริเวณที่ลูกศรหันหัวชนกันนั้นเอง ซึ่งในแต่ละภาพ input ก็อาจจะมีค่า μ ที่เหมาะสมแตกต่างกันไปของใครของมันไม่จำเป็นต้องมีค่าเดียวกัน ถึงจะเหมาะสม แต่ทว่าในการทดลองนี้ได้เลือกใช้ค่า μ = 0.05 ซึ่งใช้ได้ดีกับภาพหัวใจทั้ง 3 slice

เมื่อเราได้ GVF field (u,v) มาแล้ว จากนั้นเราก็จะเข้าสู่ขั้นตอน Snake Deform ซึ่งสามารถคำนวณได้จากสูตรในกรอบสี่เหลี่ยมข้างล่างนี้ (Paper[4])



รูปที่ 4.6 แสดงบล็อกไดอะแกรมของการคำนวณ Snake Deform

โดยที่ A คือ Pentadiagonal matrix สามารถหาได้จาก

$$A = \begin{bmatrix} c & b & a & 0 & \cdots \\ d & c & b & a & \ddots \\ e & d & c & b & \ddots \\ 0 & e & d & c & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \qquad \begin{aligned} a &= \beta \\ b &= -\alpha - 4\beta \\ c &= 2\alpha + 6\beta \\ d &= -\alpha - 4\beta \\ e &= \beta \end{aligned}$$

Alpha : elasticity parameter

Beta : rigidity parameter

Gamma : viscosity parameter

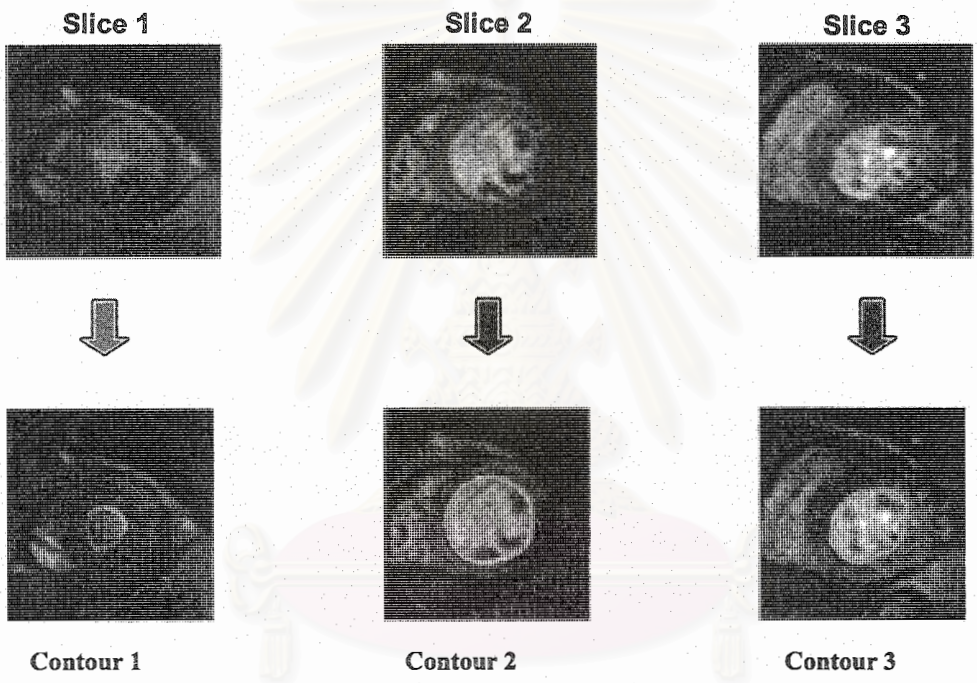
Kappa : external force weight

จากสมการในกรอบสี่เหลี่ยม ก็คือ สมการ Snake จาก paper [4] เมื่อ snake เคลื่อนที่ไปยังตำแหน่งที่เราต้องการแล้ว มันจะเหมือนกับอยู่กับที่ และเมื่อคำนวณครบแล้ว snake ก็จะหยุด ซึ่งจำนวนรอบในการคำนวณเราไม่สามารถรู้ได้ว่าต้องคำนวณกี่รอบถึงจะผลตามที่เราต้องการ

และอีกเรื่องที่สำคัญก็คือ ตำแหน่งเริ่มต้นของ snake เพราะถ้าตำแหน่งเริ่มต้นไม่เหมาะสมกับภาพ input การเคลื่อนที่ของ snake อาจจะไม่เป็นไปตามที่เราต้องการก็ได้ ใน project นี้ เลือก snake เริ่มต้นที่เป็นวงกลม และตำแหน่งที่จุดกึ่งกลางของภาพ เพราะภาพหัวใจห้องล่างซ้ายอยู่กึ่งกลางของภาพ

จากการทำใน Step 1 : Image Enhancement และ Step 2 : Image Segmentation เมื่อทำครบทั้ง 3 slice แล้วเราจะได้ contour lines 3 อัน แสดงในรูปข้างล่าง ซึ่งในแต่ละ contour line นั้นจะมีจำนวนจุด(x,y)

ไม่เท่ากัน เช่น ในการทดลองนี้ contour 1 มีจำนวน 90 จุด contour 2 มีจำนวน 214 จุด และ contour 3 มีจำนวน 175 จุด ซึ่งถ้าเรานำ contour line ที่ได้เหล่านี้ไปทำการเชื่อมต่อ เพื่อที่จะสร้างเป็นภาพ 3 มิติ นั้นจะมีข้อเสียอยู่ด้วยกันหลายประการ เช่น เส้นที่ลากเชื่อมต่อระหว่าง contour จะมีมากเกินไป วิธีการที่จะใช้ในการเชื่อมต่อนั้นอาจจะทำได้ยากเพราะจำนวนจุดไม่เท่ากัน และ ภาพที่ออกมาอาจจะไม่สวยงาม เป็นต้น ด้วยเหตุผลนี้เองเราจึงต้องมีการทำอะไรบ้างอย่างกับ contour line ก่อนที่จะมีการเชื่อมต่อ contour line ซึ่งสิ่งที่เราต้องการเพื่อที่จะให้การเชื่อมต่อ contour ของเราทำได้ง่ายขึ้นและสวยงาม ก็คือ จำนวนจุดของแต่ละ contour ต้องเท่ากัน และมีจำนวนจุดที่ไม่มากเกินไป ซึ่งใน Project นี้ ได้เลือกใช้วิธีที่ไม่ยากมากและยังให้ผลดีอีกด้วย ซึ่งจะอธิบายในหน้าถัดไป

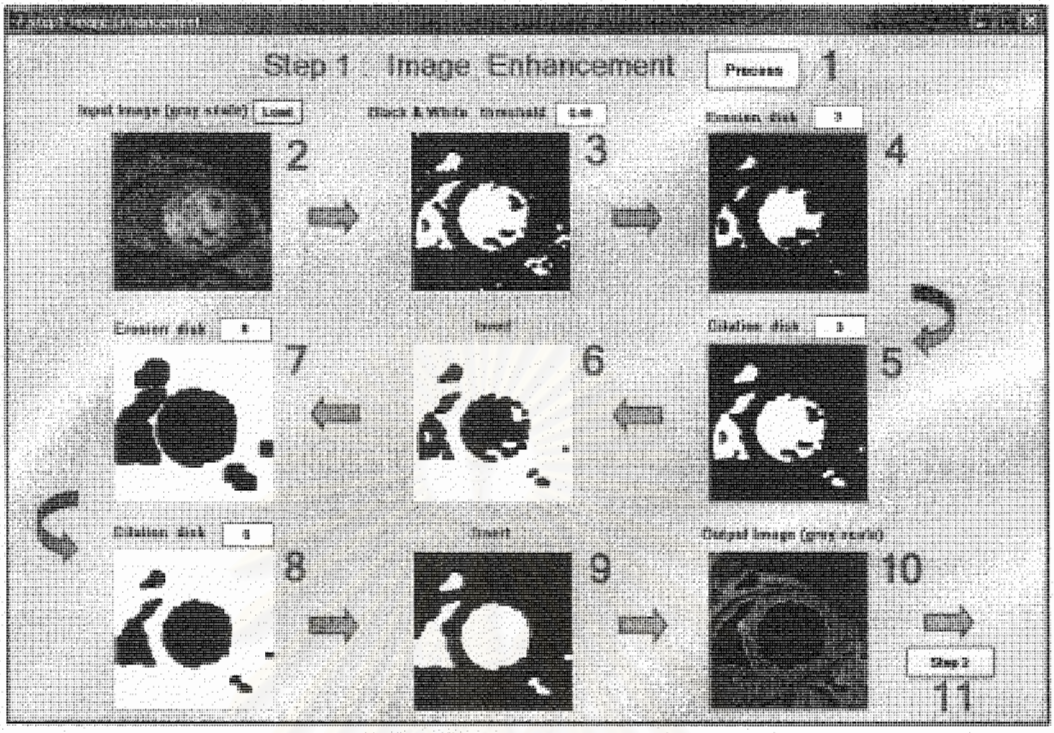


รูปที่ 4.7 แสดงภาพ MRI ทั้ง 3 slices ที่ Segmentation เสร็จแล้ว

4.3 พัฒนา Graphical User Interface (GUI) ด้วย MATLAB

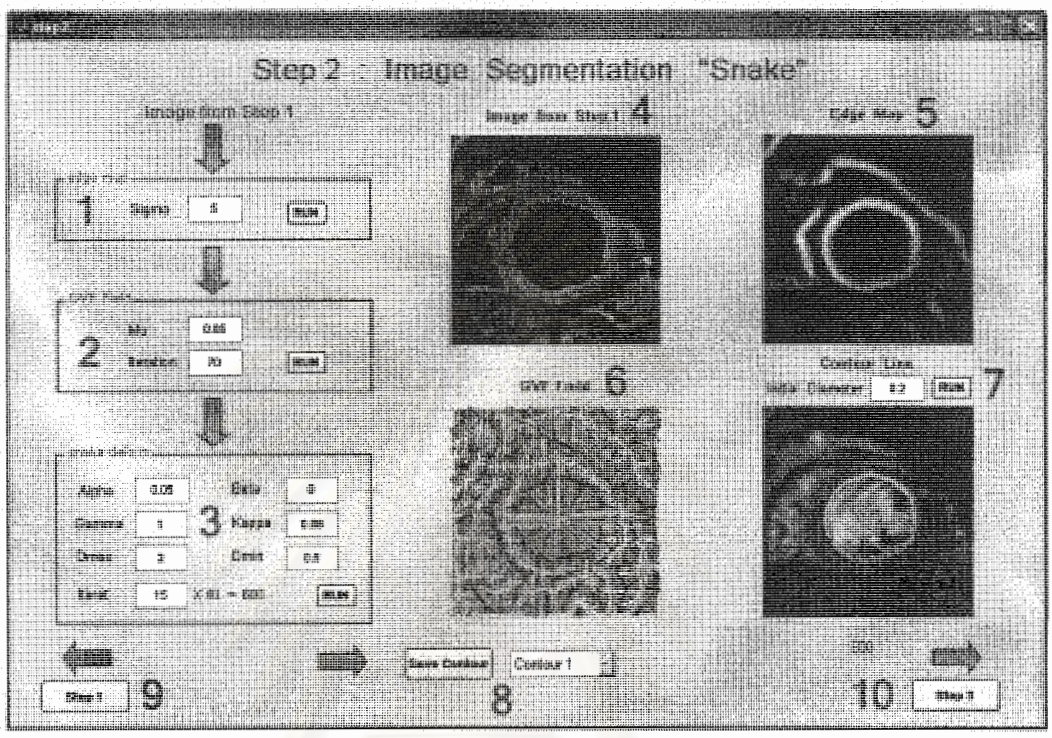
เพื่อให้ง่ายต่อการใช้งาน ในโครงงานนี้ เราได้จัดทำ GUI 2 ส่วนคือ ส่วนของ Step 1 image enhancement และ ส่วนของStep 2 Image Segmentation โดยใช้ GUI ของ MATLAB ซึ่งแต่ละส่วนมีการใช้งาน ดังนี้ สำหรับ ตัว MATLAB โปรแกรม หรือ source code นั้น เราได้รวบรวมไว้ในภาคผนวกแล้ว

GUI Step 1 : Image Enhancement



หมายเลข	หน้าที่
1	ปุ่ม Process เริ่มต้นกระบวนการใหม่ทั้งหมดใน Step1
2	แสดง input Image ซึ่งสามารถเลือก input ได้จากปุ่ม Load
3	แสดงภาพขาวดำ ของภาพ input ซึ่งสามารถปรับค่า threshold ได้ในช่วง [0,1]
4	แสดงภาพที่ถูก erosion โดยสามารถปรับค่าขนาดของ disk ได้
5	แสดงภาพที่ถูก dilation โดยสามารถปรับค่าขนาดของ disk ได้
6	แสดงภาพ Invert
7	แสดงภาพที่ถูก erosion โดยสามารถปรับค่าขนาดของ disk ได้
8	แสดงภาพที่ถูก dilation โดยสามารถปรับค่าขนาดของ disk ได้
9	แสดงภาพ Invert
10	แสดงภาพ output Image ซึ่งเป็นภาพ grayscale
11	ไป Step 2 : Image Segmentation

GUI Step 2 : Image Segmentation



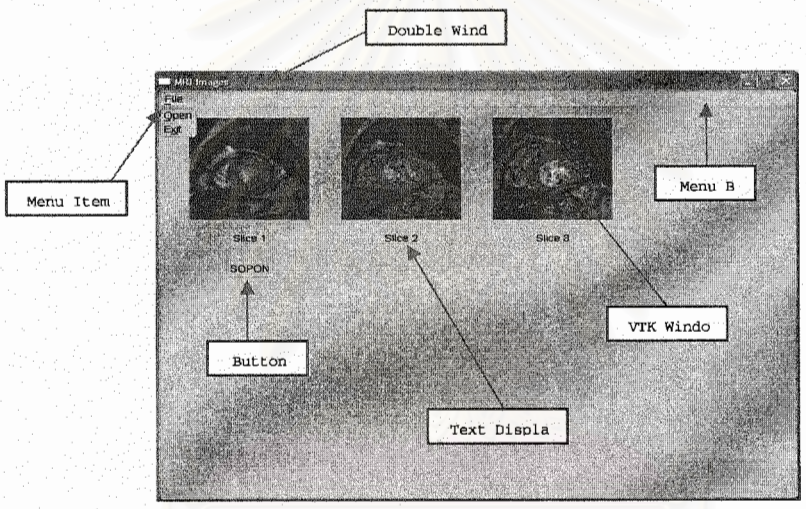
หมายเลข	หน้าที่
1	ส่วนที่กำหนดค่าพารามิเตอร์ต่างๆ ของ function EdgeMap
2	ส่วนที่กำหนดค่าพารามิเตอร์ต่างๆ ของ function GVFfield
3	ส่วนที่กำหนดค่าพารามิเตอร์ต่างๆ ของ function Snake Deform
4	แสดง Input Image ของ Step 2 ซึ่งได้มาจาก Step1
5	แสดง Edge Map
6	แสดง GVF field
7	กำหนดขนาดของวงกลมที่เป็น snake เริ่มต้น
8	Save contour ที่ได้ โดยเลือก save ให้เป็น contour ที่ 1 , 2 หรือ ที่ 3
9	กลับไปยัง Step 1 : Image Enhancement
10	ไป Step 3 : 3D Reconstruction

4.4 พัฒนา Graphical User Interface (GUI) ด้วย C++

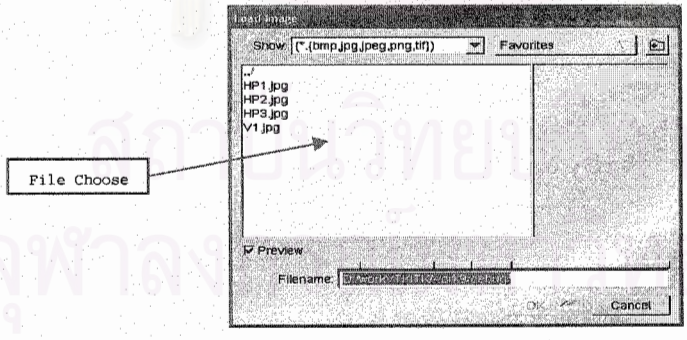
เพื่อให้สะดวกต่อการนำไปใช้งานจริง กับวงการแพทย์ ซึ่งอาจจะไม่มี โปรแกรม MATLAB เราเห็นความจำเป็นในการสร้าง GUI ด้วย โปรแกรม C++ โดยเราได้นำ Open Source Toolkit ต่อไปนี้ ITK VTK, FLTK, และ vtkFLTK มาใช้

Open Source Toolkit		หน้าที่	Version	Web Site
ITK	The Insight Toolkit	Image Processing	2.2.0	http://www.itk.org/
VTK	The Visualization Toolkit	Image Processing , Visualization	4.2	http://www.vtk.org/
FLTK	The Fast Light Toolkit	GUI	1.1.6	http://www.fltk.org/
vtkFLTK	The vtkFLTK bridge library	bridge library	0.6.1	http://vtkfltk.sourceforge.net/

ตัวอย่างการสร้าง GUI ดังรูป ที่ 4.8



(a)



(b)

รูปที่ 4.8 แสดงส่วนต่างๆ ของ GUI

จากรูป ที่ 4.8 มี source code C++ ของ โปรแกรม ดังนี้

```
// ITK
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageToVTKImageFilter.h"
#include "itkFlipImageFilter.h"
```

ในส่วนนี้จะเป็นการประกาศการใช้ class ของ ITK ซึ่งในที่นี้จะเลือกใช้ ด้วยกัน 4 ตัว คือ 1. "itkImage.h" ใช้ในการประกาศ type ใหม่ ให้เป็น ImageType 2. "itkImageFileReader.h" ใช้ในการ read file รูปภาพ 3. "itkImageToVTKImageFilter.h" ใช้ในการ เชื่อมต่อระหว่าง ITK กับ VTK ในที่นี้จะทำหน้าที่เป็นตัวส่งต่อรูปภาพที่อ่านโดย ใช้ ITK เพื่อที่จะแสดงรูปภาพโดยใช้ VTK 4. "itkFlipImageFilter.h" ใช้ในการกลับภาพให้แสดงภาพได้ปกติโดยใช้ VTK เพราะ ภาพที่อ่านมาจาก ITK จะกลับด้าน จึงต้อง กลับด้วยกลับอีกครั้งก่อนที่จะแสดงภาพโดยใช้ VTK

ซึ่ง list ของ class ทั้งหมดของ ITK และ วิธีการใช้งาน class นั้นๆสามารถดูได้จาก เว็บไซต์ http://www.itk.org/Doxygen16/html/files.html

```
// VTK Rendering
#include "vtkImageMapper.h"
#include "vtkActor2D.h"
```

ในส่วนนี้จะเป็นการประกาศการใช้ class ของ VTK ซึ่งในที่นี้จะเลือกใช้ ด้วยกัน 2 ตัว คือ "vtkImageMapper.h" และ "vtkActor2D.h" ซึ่งใช้ในการแสดงภาพ

ซึ่ง list ของ class ทั้งหมดของ ITK และ วิธีการใช้งาน class นั้นๆ สามารถดูได้จาก เว็บไซต์ http://www.vtk.org/doc/release/4.2/html/files.html

```
// FLTK
#include <FL/Fl.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Menu_Bar.H>
#include <FL/Fl_Menu_Item.H>
#include <FL/Fl_File_Chooser.H>
#include <FL/Fl_Text_Display.H>
#include <FL/Fl_Output.H>
```

ในส่วนนี้จะเป็นการประกาศการใช้ class ของ FLTK ซึ่งในที่นี้จะเลือกใช้ ด้วยกัน 8 ตัว คือ "Fl.H" "Fl_Button.H" "Fl_Double_Window.H" "Fl_Menu_Bar.H" "Fl_Menu_Item.H" "Fl_File_Chooser.H" "Fl_Text_Display.H" "Fl_Output.H" ซึ่งถ้าเราต้องการใช้มากกว่านี้ก็สามารถประกาศเพิ่มเติมได้ โดยดู list ของ class ทั้งหมดของ FLTK และ วิธีการใช้งาน class นั้นๆสามารถดูได้จาก เว็บไซต์ http://www.fltk.org/documentation.php/doc-1.1/toc.html

```
// vtkFLTK
#include "Fl_VTK_Window.H"
```

ในส่วนนี้จะเป็นการประกาศการใช้ class ของ vtkFLTK ซึ่งเป็นตัวสำคัญที่ใช้ในการติดต่อระหว่าง VTK กับ FLTK เพื่อที่จะได้แสดงภาพที่ GUI ได้

```
// GUI
Fl_Double_Window*       MainWindow;
Fl_Menu_Bar*            MainMenuBar;
Fl_Text_Display*        TextDisplay1;
Fl_Text_Display*        TextDisplay2;
Fl_Text_Display*        TextDisplay3;
Fl_VTK_Window*          mView1;
Fl_VTK_Window*          mView2;
Fl_VTK_Window*          mView3;
Fl_Button*              OpenButton1;
// vtk
vtkActor2D*             ImageActor1;
vtkActor2D*             ImageActor2;
vtkActor2D*             ImageActor3;
vtkImageMapper*         ImageMapper1;
vtkImageMapper*         ImageMapper2;
vtkImageMapper*         ImageMapper3;
//itk
typedef itk::RGBPixel<unsigned char>                    PixelType;
typedef itk::Image<PixelType,2>                         ImageType;
typedef itk::ImageFileReader<ImageType>                 ReaderType;
typedef itk::ImageToVTKImageFilter<ImageType>           FilterType;
```

ในส่วนนี้จะเป็นการประกาศการตัวแปรต่างๆที่ต้องใช้ ให้เป็นตัวแปรชนิด ต่างๆ โดยจะมีการประกาศ ทั้ง FLTK ITK และ VTK ในส่วนของ FLTK ก็คือส่วนที่เป็น GUI ที่ต้องการใช้ เช่นในที่นี้ตัวที่สำคัญก็คือ mainwindow ที่ต้องใช้ เป็นต้น ส่วน object อื่นก็เช่นพวกปุ่ม เมนูบาร์ ตัวแสดงข้อความ เป็นต้น และส่วนที่ใช้ในการแสดงรูปภาพก็คือ VTKWindow ซึ่งส่วนนี้เองที่ต้อง ใช้ vtkFLTK ในการเชื่อม ต่อ class ของ VTK กับ FLTK ในการแสดงผลภาพ

```
// Load Image
VTK_FLTK_IMPLEMENT(void)
Loadcb(Fl_Widget*,void*)
{
        char* fileName = fl_file_chooser("Load Image","(*.{bmp,jpg,jpeg,png,tif})",NULL);
        if (fileName == NULL) return;
}
// Exit
VTK_FLTK_IMPLEMENT(void)
Exitcb(Fl_Widget*,void*)
{
        exit(0);
}
```

ส่วนนี้เป็นฟังก์ชั่นย่อย ซึ่งประกอบไปด้วย ฟังก์ชั่นที่ใช้ในการเลือก ไฟล์ภาพ ที่ต้องการ load โดย เมื่อเลือกฟังก์ชั่นนี้ จะแสดง หน้าตาในการเลือกไฟล์ดังรูปที่ 2.3.1 (b) ซึ่งสามารถกรองชนิดของไฟล์ได้ เช่น " (*.{bmp,jpg,jpeg,png,tif}) " เป็นต้น

```
Fl_Menu_Item menu_mMenuBar[] = {
        {"&File", 0,  0, 0, 64, 0, 0, 14, 56},
        {"&Open", 0,Loadcb,NULL, 0, 0, 0, 14, 56},
        {"E&xit", 0,Exitcb,NULL, 0, 0, 0, 14, 56},
        {0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0}
        };
```

ส่วนนี้เป็นการกำหนดคุณลักษณะของ Item menu_mMenuBar ว่าต้องการที่จะมีเมนูอะไรบ้างซึ่ง รูปแบบในการกำหนดคุณลักษณะต่าง ของ Item menu_mMenuBar มีดังนี้

```
struct Fl_Menu_Item {
  const char* text; // label()
  ulong               shortcut_;
  Fl_Callback*        callback_;
  void*               user_data_;
  int                 flags;
  uchar               labeltype_;
  uchar               labelfont_;
  uchar               labelsize_;
  uchar               labelcolor_;
};
```

```
int main (int argc, char* argv[])
{
//GUI
        MainWindow = new Fl_Double_Window(100,100,800, 600, "MRI Images");
        MainWindow->box(FL_PLASTIC_THIN_DOWN_BOX);
        MainWindow->color((Fl_Color)30);
```

ส่วนนี้เป็นการสร้าง MainWindow โดยรูปแบบในการกำหนดคุณลักษณะการสร้าง ของ MainWindow เช่น ขนาด 800x600 เป็นต้น โดยมีรูปแบบดังนี้

Fl_Double_Window::Fl_Double_Window(int x, int y, int w, int h, const char *label = 0)

```
TextDisplay1 = new Fl_Text_Display(85,225, 60, 0);
TextDisplay1->label("Slice 1");
TextDisplay2 = new Fl_Text_Display(275,225, 60, 0);
TextDisplay2->label("Slice 2");
TextDisplay3 = new Fl_Text_Display(465,225, 60, 0);
TextDisplay3->label("Slice 3");
```

ส่วนนี้เป็นการสร้าง TextDisplay 3 อัน โดยรูปแบบในการกำหนดคุณลักษณะการสร้าง ของ TextDisplay มีดังนี้

Fl_Text_Display(int X, int Y, int W, int H, const char *l = 0);

```
MainMenuBar = new Fl_Menu_Bar(0, 0, 800, 25, "mMenuBar");
MainMenuBar->box(FL_PLASTIC_THIN_DOWN_BOX);
MainMenuBar->down_box(FL_PLASTIC_THIN_UP_BOX);
MainMenuBar->color((Fl_Color)61);
MainMenuBar->selection_color((Fl_Color)3);
MainMenuBar->menu(menu_mMenuBar);
```

ส่วนนี้เป็นการสร้าง MainMenuBar 3 อัน โดยรูปแบบในการกำหนดคุณลักษณะการสร้าง ของ MainMenuBar มีดังนี้

Fl_Menu_Bar::Fl_Menu_Bar(int x, int y, int w, int h, const char *label = 0)

```
OpenButton1 = new Fl_Button(85,250,60,25,"SOPON");
OpenButton1->box(FL_PLASTIC_UP_BOX);
OpenButton1->down_box(FL_PLASTIC_DOWN_BOX);
OpenButton1->color(FL_GREEN);
OpenButton1->selection_color(FL_YELLOW);
OpenButton1->callback(Loadcb,NULL);
```

ส่วนนี้เป็นการสร้าง Button โดยรูปแบบในการกำหนดคุณลักษณะการสร้าง ของ Button มีดังนี้

Fl_Button::Fl_Button(int x, int y, int w, int h, const char *label = 0)

```
mView1 = new Fl_VTK_Window(40, 40, 150, 150);
mView2 = new Fl_VTK_Window(230, 40, 150, 150);
mView3 = new Fl_VTK_Window(420, 40, 150, 150);
```

ส่วนนี้เป็นการสร้าง VTK_Window 3 อัน โดยมีขนาดเท่ากันที่ 150x150 ซึ่งเป็นส่วนที่ใช้ในการแสดงภาพ

```
//V1
        ImageActor1 = vtkActor2D::New();
        ImageMapper1 = vtkImageMapper::New();
        ReaderType::Pointer reader1 = ReaderType::New();
        FilterType::Pointer connector1 = FilterType::New();
        reader1->SetFileName("HP1.jpg");
        typedef itk::FlipImageFilter<ImageType> FilterType1;
        FilterType1::Pointer filter1 = FilterType1::New();
        typedef FilterType1::FlipAxesArrayType  FlipAxesArrayType;
        FlipAxesArrayType flipArray;
        flipArray[0] = atoi("0");
        flipArray[1] = atoi("90");
        filter1->SetFlipAxes(flipArray);
        filter1->SetInput(reader1->GetOutput());
        connector1->SetInput(filter1->GetOutput());
        ImageMapper1->SetInput(connector1->GetOutput());
        ImageMapper1->SetColorWindow(255);
        ImageMapper1->SetColorLevel(127);
        ImageActor1->SetMapper(ImageMapper1);
        ImageMapper1->Delete();
        mView1->AddProp(ImageActor1);
        View1->Update();
```

ส่วน V1 นี้ เป็นอ่านไฟล์ภาพ "HP1.jpg" จากนั้นจะทำการกลับภาพให้เหมือนเดิมเพราะการอ่านภาพจากไฟล์ของ ITK จะทำให้ภาพกลับด้าน จึงต้องมีการกลับภาพให้เหมือนเดิมก่อนที่จะนำไปแสดงภาพที่ mView1 โดยใช้ VTK

```
//V2
        ImageActor2 = vtkActor2D::New();
        ImageMapper2 = vtkImageMapper::New();
        ReaderType::Pointer reader2 = ReaderType::New();
        FilterType::Pointer connector2 = FilterType::New();
        reader2->SetFileName("HP2.jpg");
        typedef itk::FlipImageFilter<ImageType> FilterType2;
        FilterType2::Pointer filter2 = FilterType2::New();
        filter2->SetFlipAxes(flipArray);
        filter2->SetInput(reader2->GetOutput());
        connector2->SetInput(filter2->GetOutput());
        ImageMapper2->SetInput(connector2->GetOutput());
        ImageMapper2->SetColorWindow(255);
        ImageMapper2->SetColorLevel(127);
        ImageActor2->SetMapper(ImageMapper2);
        ImageMapper2->Delete();
        mView2->AddProp(ImageActor2);
        mView2->Update();
```

```
//V3
        ImageActor3 = vtkActor2D::New();
        ImageMapper3 = vtkImageMapper::New();
        ReaderType::Pointer reader3 = ReaderType::New();
        FilterType::Pointer connector3 = FilterType::New();
        reader3->SetFileName("HP3.jpg");
        typedef itk::FlipImageFilter<ImageType> FilterType3;
        FilterType3::Pointer filter3 = FilterType3::New();
        filter3->SetFlipAxes(flipArray);
        filter3->SetInput(reader3->GetOutput());
        connector3->SetInput(filter3->GetOutput());
        ImageMapper3->SetInput(connector3->GetOutput());
        ImageMapper3->SetColorWindow(255);
        ImageMapper3->SetColorLevel(127);
        ImageActor3->SetMapper(ImageMapper3);
        ImageMapper3->Delete();
        mView3->AddProp(ImageActor3);
        mView3->Update();
```

```
        MainWindow->callback(Exitcb,NULL);
        MainWindow->show(argc,argv);
        int fl_ret = Fl::run();
        delete MainWindow;
        delete mView1;
        delete mView2;
        delete mView3;
        return fl_ret;
}
```

ส่วน V2 และ V3 ก็จะเหมือนกับ V1 จะต่างกันตรงที่ไฟล์ภาพที่อ่านเข้ามาเพื่อแสดงภาพ

5. สรุปผลการทดลองและข้อเสนอแนะ

ในโครงงานนี้ เราได้นำเสนอวิธีการแบ่งย่อยภาพ MR ของหัวใจ ที่มีประสิทธิภาพ โดยใช้วิธีแอกตีฟคอนทัวร์ โดยเราได้เพิ่มสมรรถภาพของวิธีการแบ่งย่อยภาพนี้ให้มีความแม่นยำมากขึ้นด้วยการเพิ่งส่วนที่เป็น pre-processing ในขั้นตอนแรก ซึ่งเราได้นำวิธีการ morphological processing เช่น dilation and erosion เข้ามาใช้ ซึ่งเป็นวิธีการที่ไม่ซับซ้อน และใช้เวลาน้อย

อีกทั้งเรายังสร้างโปรงแกรม GUI ทั้งในรูปของ MATLAB และ C++ เพื่อความสะดวกต่อการนำไปใช้งาน

<u>ส่วนที่จะดำเนินการต่อไป</u>

- ศึกษาและพัฒนาวิธีการสร้างภาพ 3 มิติ ของจากส่วน ที่เราสร้างขึ้นให้สามารถแบ่งย่อยส่วนอื่น ๆ ของหัวใจในภาพได้พร้อม ๆ กัน

- ปรับปรุงและพัฒนาโปรแกรม MATLAB มีประสิทธิภาพมากขึ้น และสามารถคำนวณการแบ่งย่อยภาพได้เร็วขึ้น

- เพื่อยืนยันความน่าเชื่อถือของวิธีการนี้ ทดสอบโปรแกรมที่มีอยู่กับ ภาพชุด magnetic resonance อื่น ๆ และเปรียบเทียบผลการแบ่งย่อยโดยโปรแกรมของเรากับผลที่ได้จากการแบ่งย่อยโดยผู้เชี่ยวชาญ

8. เอกสารอ้างอิง

1. Kass, M., Witkin, A., & Terzopolous, D. Snakes: Active Contour Models. International Journal of Computer Vision, 1(4), 321--331., 1988.
2. Cohen L. D., Cohen I., "A finite element method applied to new active contour models and 3D reconstruction from cross sections", Proceedings of Third International Conference on Computer Vision, Osaka, Japan, 1990.
3. C. Xu and J. L. Prince, "Snakes, Shapes, and Gradient Vector Flow," IEEE Transactions on Image Processing, 7(3), pp. 359-369, March 1998.
4. Chan T.F. and Vese, L.A., "Active contours without edges, " IEEE Transactions on Image Processing, Volume 10, Issue 2, pp. 266 – 277, Feb. 2001.

ภาคผนวก

โปรแกรม MATLAB Source Code สำหรับ GUI ที่ เรา เขียนขึ้น ในโครงงานนี้

GUI Step1

```
% GUI step1
% Sopon Phumeechanya ID : 4870541521
function varargout = step1(varargin)
% STEP1 M-file for step1.fig
% STEP1, by itself, creates a new STEP1 or raises the existing
% singleton*.
%
% H = STEP1 returns the handle to a new STEP1 or the handle to
% the existing singleton*.
%
% STEP1('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in STEP1.M with the given input arguments.
%
% STEP1('Property','Value',...) creates a new STEP1 or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before step1_OpeningFunction gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to step1_OpeningFcn via varargin.
%
% See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Copyright 2002-2003 The MathWorks, Inc.
% Edit the above text to modify the response to help step1
% Last Modified by GUIDE v2.5 31-Jan-2006 19:03:58
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
```

```
'gui_OpeningFcn', @step1_OpeningFcn, ...
'gui_OutputFcn', @step1_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before step1 is made visible.
function step1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to step1 (see VARARGIN)
% Choose default command line output for step1
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
26
% UIWAIT makes step1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = step1_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbuttonLoad.
function pushbuttonLoad_Callback(hObject, eventdata, handles)
% hObject handle to pushbuttonLoad (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
global IMGI IMGO;
[filename, pathname] = ...
uigetfile({'*.jpg';'*.bmp';'*.*'},'File Selector');
IMGI = imread([pathname,filename]);
threshold = str2num(get(handles.editthreshold,'String'));
E1 = str2num(get(handles.editE1,'String'));
D1 = str2num(get(handles.editD1,'String'));
E2 = str2num(get(handles.editE2,'String'));
D2 = str2num(get(handles.editD2,'String'));
IAA = imread('arrow3.jpg');
IAB = imread('arrow2.jpg');
IAC = imread('arrow4.jpg');
IAD = imread('arrow5.jpg');
axes(handles.axesA1);imshow(IAA);
axes(handles.axesA2);imshow(IAA);
axes(handles.axesA5);imshow(IAD);
axes(handles.axesA6);imshow(IAD);
axes(handles.axesA9);imshow(IAA);
axes(handles.axesA10);imshow(IAA);
axes(handles.axesA11);imshow(IAA);
axes(handles.axesA3);imshow(IAB);
axes(handles.axesA8);imshow(IAC);
IMGI = im2double(IMGI);
IMGI = mat2gray(IMGI);
%convert grayscale to black&white
```

```
BW1 = im2bw(IMGI,threshold);
BW2 = ErosionDisk(BW1,E1);
BW3 = DilationDisk(BW2,D1);
BW4 = not(BW3);
BW5 = ErosionDisk(BW4,E2);
BW6 = DilationDisk(BW5,D2);
BW6 = im2double(BW6);
BW7 = not(BW6);
IMGO = IMGI;
[i,j] = find(BW7);
27
for k = 1:length(i)
IMGO(i(k),j(k)) = 0;
End
axes(handles.axes1);imshow(IMGI);
axes(handles.axes2);imshow(BW1);
axes(handles.axes3);imshow(BW2);
axes(handles.axes4);imshow(BW3);
axes(handles.axes5);imshow(BW4);
axes(handles.axes6);imshow(BW5);
axes(handles.axes7);imshow(BW6);
axes(handles.axes8);imshow(BW7);
axes(handles.axes9);imshow(IMGO);
set(handles.pushbuttonprocess,'Enable','on');
set(handles.pushbuttonS1ST2,'Visible','on');
%***********************************************
% --- Executes on button press in pushbuttonprocess.
function pushbuttonprocess_Callback(hObject, eventdata, handles)
% hObject handle to pushbuttonprocess (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%***********************************************
global IMGI IMGO;
```

```
threshold = str2num(get(handles.editthreshold,'String'));
E1 = str2num(get(handles.editE1,'String'));
D1 = str2num(get(handles.editD1,'String'));
E2 = str2num(get(handles.editE2,'String'));
D2 = str2num(get(handles.editD2,'String'));
IMGI = im2double(IMGI);
IMGI = mat2gray(IMGI);
%convert grayscale to black&white
BW1 = im2bw(IMGI,threshold);
BW2 = ErosionDisk(BW1,E1);
BW3 = DilationDisk(BW2,D1);
BW4 = not(BW3);
BW5 = ErosionDisk(BW4,E2);
BW6 = DilationDisk(BW5,D2);
BW6 = im2double(BW6);
BW7 = not(BW6);
IMGO = IMGI;
[i,j] = find(BW7);
for k = 1:length(i)
IMGO(i(k),j(k)) = 0;
end
axes(handles.axes1);imshow(IMGI);
axes(handles.axes2);imshow(BW1);
axes(handles.axes3);imshow(BW2);
axes(handles.axes4);imshow(BW3);
axes(handles.axes5);imshow(BW4);
axes(handles.axes6);imshow(BW5);
axes(handles.axes7);imshow(BW6);
axes(handles.axes8);imshow(BW7);
axes(handles.axes9);imshow(IMGO);
%************************************************
28
function editthreshold_Callback(hObject, eventdata, handles)
```

```
% hObject handle to editthreshold (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editthreshold as text
% str2double(get(hObject,'String')) returns contents of editthreshold as a
double
% --- Executes during object creation, after setting all properties.
function editthreshold_CreateFcn(hObject, eventdata, handles)
% hObject handle to editthreshold (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editE1_Callback(hObject, eventdata, handles)
% hObject handle to editE1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editE1 as text
% str2double(get(hObject,'String')) returns contents of editE1 as a double
% --- Executes during object creation, after setting all properties.
function editE1_CreateFcn(hObject, eventdata, handles)
% hObject handle to editE1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
```

```
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editD1_Callback(hObject, eventdata, handles)
% hObject handle to editD1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editD1 as text
% str2double(get(hObject,'String')) returns contents of editD1 as a double
% --- Executes during object creation, after setting all properties.
function editD1_CreateFcn(hObject, eventdata, handles)
% hObject handle to editD1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
29
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editE2_Callback(hObject, eventdata, handles)
% hObject handle to editE2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editE2 as text
% str2double(get(hObject,'String')) returns contents of editE2 as a double
% --- Executes during object creation, after setting all properties.
function editE2_CreateFcn(hObject, eventdata, handles)
% hObject handle to editE2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editD2_Callback(hObject, eventdata, handles)
% hObject handle to editD2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editD2 as text
% str2double(get(hObject,'String')) returns contents of editD2 as a double
% --- Executes during object creation, after setting all properties.
function editD2_CreateFcn(hObject, eventdata, handles)
% hObject handle to editD2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in pushbuttonS1ST2.
function pushbuttonS1ST2_Callback(hObject, eventdata, handles)
% hObject handle to pushbuttonS1ST2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
step2();
%************************************************
```

GUI Step2

```
% GUI step2
% Sopon Phumeechanya ID : 4870541521
function varargout = step2(varargin)
% STEP2 M-file for step2.fig
% STEP2, by itself, creates a new STEP2 or raises the existing
% singleton*.
%
% H = STEP2 returns the handle to a new STEP2 or the handle to
% the existing singleton*.
%
% STEP2('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in STEP2.M with the given input arguments.
%
% STEP2('Property','Value',...) creates a new STEP2 or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before step2_OpeningFunction gets called. An
% nrecognized property name or invalid value makes property application
% stop. All inputs are passed to step2_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% nstance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Copyright 2002-2003 The MathWorks, Inc.
% Edit the above text to modify the response to help step2
% Last Modified by GUIDE v2.5 30-Jan-2006 23:44:20
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @step2_OpeningFcn, ...
'gui_OutputFcn', @step2_OutputFcn, ...
```

```
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
31
% --- Executes just before step2 is made visible.
function step2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to step2 (see VARARGIN)
% Choose default command line output for step2
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes step2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = step2_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
%************************************************
% initial
global IMGI IMGO I;
axes(handles.axesV1);
cla;
axes(handles.axesV2);
cla;
axes(handles.axesV3);
cla;
axes(handles.axesV4);
cla;
I = IMGO;
axes(handles.axesV1);imshow(I);
A1 = imread('arrow6.jpg');
IAA = imread('arrow3.jpg');
IAB = imread('arrow5.jpg');
axes(handles.axesA1);imshow(A1);
axes(handles.axesA2);imshow(A1);
axes(handles.axesA3);imshow(A1);
axes(handles.axesA4);imshow(IAA);
axes(handles.axesA6);imshow(IAA);
axes(handles.axesA5);imshow(IAB);
set(handles.axesV2,'Visible','off');
set(handles.axesV3,'Visible','off');
set(handles.axesV4,'Visible','off');
set(handles.editsigma,'String','5');
set(handles.editMu,'String','0.05');
set(handles.editIterations,'String','70');
32
set(handles.editAlpha,'String','0.05');
set(handles.editBeta,'String','0');
set(handles.editGamma,'String','1');
set(handles.editKappa,'String','0.08');
```

```
set(handles.editDmin,'String','0.5');
set(handles.editDmax,'String','2');
set(handles.editIterat,'String','15');
set(handles.editDiameter,'String','0.2');
set(handles.textIteratsnake,'String',' ');
set(handles.pushbuttonSC,'Enable','off');
%************************************************
function editsigma_Callback(hObject, eventdata, handles)
% hObject handle to editsigma (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editsigma as text
% str2double(get(hObject,'String')) returns contents of editsigma as a
double
% --- Executes during object creation, after setting all properties.
function editsigma_CreateFcn(hObject, eventdata, handles)
% hObject handle to editsigma (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editMu_Callback(hObject, eventdata, handles)
% hObject handle to editMu (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editMu as text
% str2double(get(hObject,'String')) returns contents of editMu as a double
% --- Executes during object creation, after setting all properties.
```

```
function editMu_CreateFcn(hObject, eventdata, handles)
% hObject handle to editMu (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
33
function editIterations_Callback(hObject, eventdata, handles)
% hObject handle to editIterations (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editIterations as text
% str2double(get(hObject,'String')) returns contents of editIterations as a
double
% --- Executes during object creation, after setting all properties.
function editIterations_CreateFcn(hObject, eventdata, handles)
% hObject handle to editIterations (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editAlpha_Callback(hObject, eventdata, handles)
% hObject handle to editAlpha (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editAlpha as text
% str2double(get(hObject,'String')) returns contents of editAlpha as a
double
% --- Executes during object creation, after setting all properties.
function editAlpha_CreateFcn(hObject, eventdata, handles)
% hObject handle to editAlpha (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editBeta_Callback(hObject, eventdata, handles)
% hObject handle to editBeta (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editBeta as text
% str2double(get(hObject,'String')) returns contents of editBeta as a
double
34
% --- Executes during object creation, after setting all properties.
function editBeta_CreateFcn(hObject, eventdata, handles)
% hObject handle to editBeta (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
```

```
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editGamma_Callback(hObject, eventdata, handles)
% hObject handle to editGamma (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editGamma as text
% str2double(get(hObject,'String')) returns contents of editGamma as a
double
% --- Executes during object creation, after setting all properties.
function editGamma_CreateFcn(hObject, eventdata, handles)
% hObject handle to editGamma (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editKappa_Callback(hObject, eventdata, handles)
% hObject handle to editKappa (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editKappa as text
% str2double(get(hObject,'String')) returns contents of editKappa as a
double
% --- Executes during object creation, after setting all properties.
function editKappa_CreateFcn(hObject, eventdata, handles)
% hObject handle to editKappa (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
35
function editDmin_Callback(hObject, eventdata, handles)
% hObject handle to editDmin (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editDmin as text
% str2double(get(hObject,'String')) returns contents of editDmin as a
double
% --- Executes during object creation, after setting all properties.
function editDmin_CreateFcn(hObject, eventdata, handles)
% hObject handle to editDmin (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editDmax_Callback(hObject, eventdata, handles)
% hObject handle to editDmax (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of editDmax as text
% str2double(get(hObject,'String')) returns contents of editDmax as a
double
% --- Executes during object creation, after setting all properties.
function editDmax_CreateFcn(hObject, eventdata, handles)
% hObject handle to editDmax (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function editIterat_Callback(hObject, eventdata, handles)
% hObject handle to editIterat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editIterat as text
% str2double(get(hObject,'String')) returns contents of editIterat as a
double
% --- Executes during object creation, after setting all properties.
function editIterat_CreateFcn(hObject, eventdata, handles)
% hObject handle to editIterat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
36
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
```

```
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in pushbedgemap.
function pushbedgemap_Callback(hObject, eventdata, handles)
% hObject handle to pushbedgemap (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
global I f;
%load input image
sigma = str2num(get(handles.editsigma,'String'));
I = im2double(I);
%scale imageinput to the range [0,1]
I = mat2gray(I);
f = EdgeMap(I,sigma);
%show edgemap
axes(handles.axesV2);imshow(f);
%************************************************
% --- Executes on button press in pushbGVFField.
function pushbGVFField_Callback(hObject, eventdata, handles)
% hObject handle to pushbGVFField (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
global f IMGI x y u v Diameter;
%load input image
Mu = str2num(get(handles.editMu,'String'));
Iterations = str2num(get(handles.editIterations,'String'));
axes(handles.axesV3);
[u,v] = GVFfield(f,Mu,Iterations);
xSpace=(1:size(f,1)/64:size(f,1));
ySpace=(1:size(f,2)/64:size(f,2));
qx=interp2(u,xSpace, ySpace');
```

```
qy=interp2(v,xSpace, ySpace');
quiver(xSpace,ySpace,qx,qy);
axis off; axis equal; axis('ij');
%************************************************
Dmax = str2num(get(handles.editDmax,'String'));
Dmin = str2num(get(handles.editDmin,'String'));
Diameter = str2num(get(handles.editDiameter,'String'));
axes(handles.axesV4);imshow(IMGI);
t = 0:0.05:6.28;
x = (size(IMGI,1)/2 + Diameter/2*size(IMGI,1)*cos(t))';
y = (size(IMGI,2)/2 + Diameter/2*size(IMGI,2)*sin(t))';
axis('square', 'off');
SnakeDisplay(x,y,'r',1);
%************************************************
37
% --- Executes on button press in pushbsnakedeform.
function pushbsnakedeform_Callback(hObject, eventdata, handles)
% hObject handle to pushbsnakedeform (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
global x y u v IMGI Diameter;
cla;
Alpha = str2num(get(handles.editAlpha,'String'));
Beta = str2num(get(handles.editBeta,'String'));
Gamma = str2num(get(handles.editGamma,'String'));
Kappa = str2num(get(handles.editKappa,'String'));
Iterat = str2num(get(handles.editIterat,'String'));
Dmax = str2num(get(handles.editDmax,'String'));
Dmin = str2num(get(handles.editDmin,'String'));
set(handles.textIterat,'String',num2str(Iterat*40));
axes(handles.axesV4);imshow(IMGI);
t = 0:0.05:6.28;
```

```
x = (size(IMGI,1)/2 + Diameter/2*size(IMGI,1)*cos(t))';
y = (size(IMGI,2)/2 + Diameter/2*size(IMGI,2)*sin(t))';
axis('square', 'off');
SnakeDisplay(x,y,'r',1);
for i=1:Iterat,
[x,y] = SnakeDeform(x,y,Alpha,Beta,Gamma,Kappa,u,v,40);
[x,y] = SnakeInterP(x,y,Dmax,Dmin);
SnakeDisplay(x,y,'r',1);
set(handles.textIteratsnake,'String',num2str(i*40));
pause(0.5);
end
axis('square', 'off');
cla;
imshow(IMGI);
hold on
SnakeDisplay(x,y,'g',3);
set(handles.pushbuttonSC,'Enable','on');
%************************************************
% --- Executes on button press in pushbuttonStep1.
function pushbuttonStep1_Callback(hObject, eventdata, handles)
% hObject handle to pushbuttonStep1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
step1();
%************************************************
% --- Executes on button press in pushbuttonStep3.
function pushbuttonStep3_Callback(hObject, eventdata, handles)
% hObject handle to pushbuttonStep3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
step3();
```

```
%************************************************
38
% --- Executes on button press in pushbuttonSC.
function pushbuttonSC_Callback(hObject, eventdata, handles)
% hObject handle to pushbuttonSC (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
global x y ;
popup_sel = get(handles.popupmenuContour,'Value');
switch popup_sel
case 1
save C1.mat x y ;
case 2
save C2.mat x y ;
case 3
save C3.mat x y ;
end
set(handles.pushbuttonSC,'Enable','off');
%************************************************
function editDiameter_Callback(hObject, eventdata, handles)
% hObject handle to editDiameter (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of editDiameter as text
% str2double(get(hObject,'String')) returns contents of editDiameter as a
double
% --- Executes during object creation, after setting all properties.
function editDiameter_CreateFcn(hObject, eventdata, handles)
% hObject handle to editDiameter (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
```

```
% See ISPC and COMPUTER.
if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in pushbuttonDiameter.
function pushbuttonDiameter_Callback(hObject, eventdata, handles)
% hObject handle to pushbuttonDiameter (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%************************************************
global IMGI x y Diameter;
cla;
Dmax = str2num(get(handles.editDmax,'String'));
Dmin = str2num(get(handles.editDmin,'String'));
Diameter = str2num(get(handles.editDiameter,'String'));
axes(handles.axesV4);imshow(IMGI);
t = 0:0.05:6.28;
x = (size(IMGI,1)/2 + Diameter/2*size(IMGI,1)*cos(t))';
y = (size(IMGI,2)/2 + Diameter/2*size(IMGI,2)*sin(t))';
axis('square', 'off');
SnakeDisplay(x,y,'r',1);
%************************************************
```