บทที่ 3

วิธีการวิจัย

จากงานวิจัยที่เกี่ยวข้องจะเห็นว่าการพัฒนาโปรแกรมบนหน้าเว็บจะต้องใช้ภาษาจาวาสคริปต์ร่วมกับภาษาเอชทีเอ็มแอล เพื่อให้สามารถสร้างโปรแกรมในการตกแต่งภาพบนหน้าเว็บได้ แต่อย่างไรก็ตามการใช้งานจาวาสคริปต์ในการพัฒนาโปรแกรมปกติจะสามารถสั่งงานซีพียู ได้เพียงแกนเดียว ทำให้การทำงานของโปรแกรมไม่รวดเร็ว และไม่สามารถดึงเอาทรัพยากรในเครื่องของผู้ใช้มาได้อย่างมีประสิทธิภาพจึงมีการนำเอาเว็บซีแอลเข้ามาใช้ในการพัฒนาโปรแกรมให้มีการคำนวณแบบขนาน

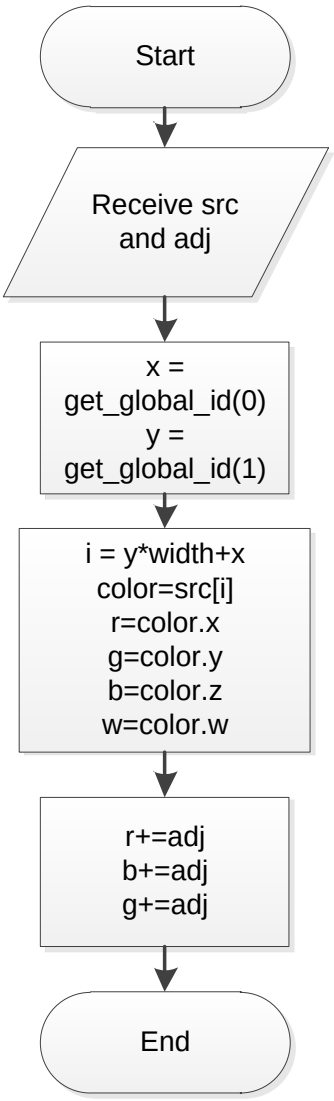
3.1 การออกแบบ

ในการพัฒนาโปรแกรมตกแต่งภาพบนเว็บนั้น จะใช้เว็บบราวเซอร์ Mozilla Firefox รุ่น 33 ในการออกแบบหน้าเว็บ โดยโปรแกรมตกแต่งภาพนั้นจะให้ผู้ใช้เลือกภาพขึ้นมาตกแต่งและแก้ไขภาพได้ ซึ่งภายในโปรแกรมจะสามารถตกแต่งภาพได้หลากหลายรูปแบบ ในการออกแบบจะนำเอาอัลกอริธึมของการประมวลผลภาพที่ทำงานบนจาวาสคริปต์ โดยนำฟังก์ชันในการประมวลผลภาพมาทดสอบ เช่น Brightness Contrast Saturation Inversion Posterize และ Threshold ซึ่งฟังก์ชันที่นำมาทดสอบในส่วนของจาวาสคริปต์ที่นำมาใช้โดยจะนำมาวัดประสิทธิภาพทางด้านเวลา

3.2 อัลกอริธึมที่ใช้ในการทดสอบ

3.2.1 อัลกอริธึมของฟังก์ชันที่ใช้ในการทดสอบด้วยจาวาสคริปต์ และเว็บซีแอล

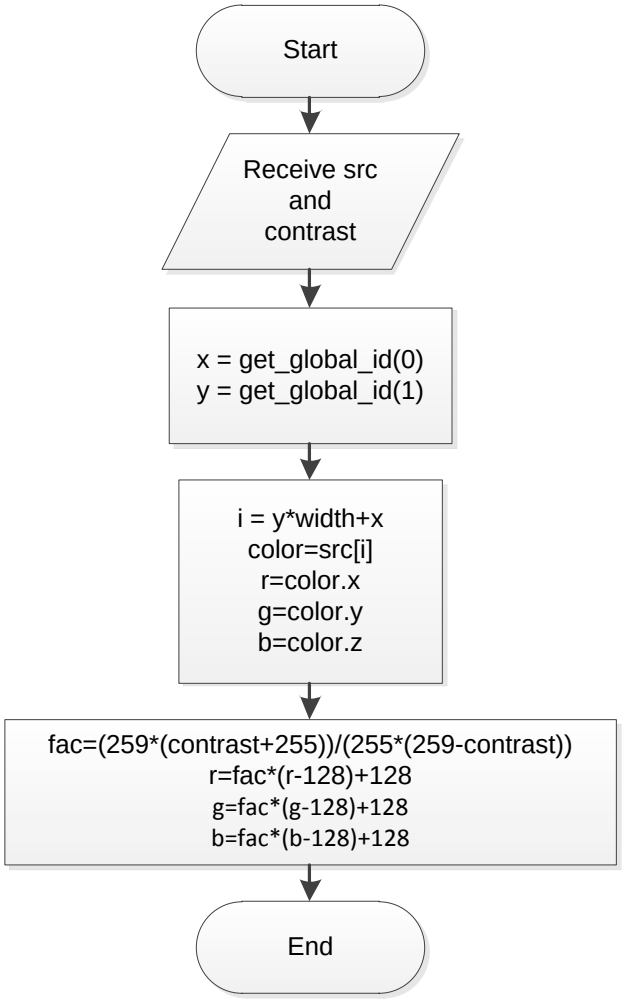
3.2.1.1 ปรับค่าความมืด – ความสว่างให้กับภาพ (Brightness)



ภาพที่ 3.1 ผังงานของอัลกอริธึม Brightness เมื่อมีการทำงานแบบขนาน

จากภาพ 3.1 เมื่อส่งงานไปให้กับจีพียูในที่นี้คือต้นฉบับรูปภาพ (src) และค่าที่ปรับ (adj) จากนั้นจะมีการรับค่าแกนของภาพ โดยให้แกน x เป็นแกนนอน และ แกน y เป็นแกนตั้ง และหาพิกัดอาเรย์ของภาพ ถัดจากนั้นดึงค่าสีมาจากแต่ละ pixel และนำเอาค่าสีแต่ละค่ามาบวกกับค่าที่ปรับ

3.2.1.2 ปรับค่าคอนทราสต์ให้กับภาพ (Contrast)



ภาพที่ 3.2 ผังงานของอัลกอริธึม Contrast เมื่อมีการทำงานแบบขนาน

จากภาพ 3.2 เมื่อส่งงานไปให้กับจีพียูในที่นี้คือต้นฉบับรูปภาพ (src) และค่าที่ปรับ (contrast) จากนั้นจะมีการรับค่าแกนของภาพ โดยให้แกน x เป็นแกนนอน และ แกน y เป็นแกนตั้ง และหาพิกัดอาเรย์ของภาพ ถัดจากนั้นดึงค่าสีมาจากแต่ละ pixel หลังจากนั้นคำนวณค่า factor ที่เป็นค่าใหม่ของ contrast ดังสมการ

$$fac = \frac{(259(contrast + 255))}{255(259 - contrast)}$$

จากนั้นนำเอาค่า factor ไปคูณกับสมการของแต่ละสี

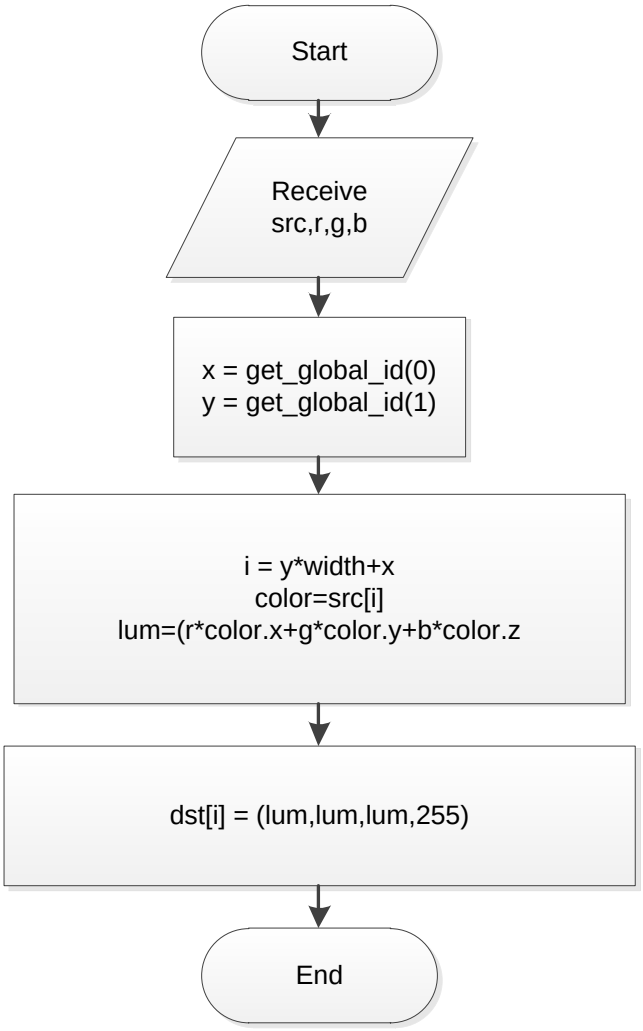
$$r = fac(r - 128) + 128$$

$$g = fac(g - 128) + 128$$

$$b = fac(b - 128) + 128$$

ถัดจากนั้นส่งค่ากลับไปยังฝั่งซีพียู เพื่อนำไปประมวลผลต่อไป

3.2.1.3 ปรับลดความอิ่มสี (Saturation)



ภาพที่ 3.3 ผังงานของอัลกอริธึม Desaturation เมื่อมีการทำงานแบบขนาน

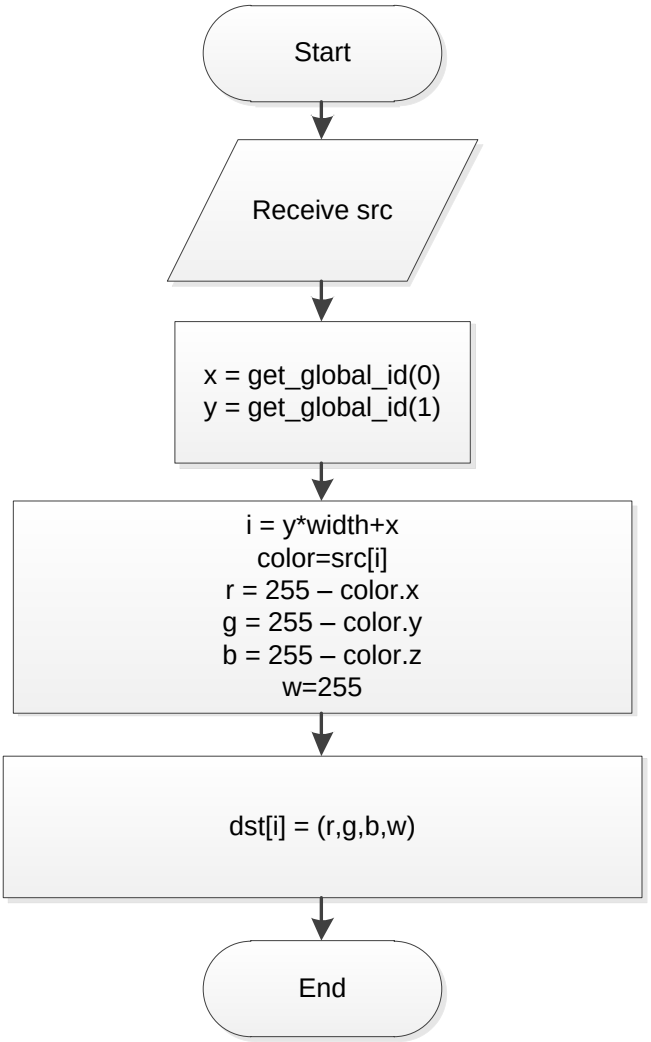
จากภาพที่ 3.3 เมื่อส่งงานไปให้กับจีพียูในที่นี้คือต้นฉบับรูปภาพ (src) และ factor ของสีแดง เขียว และน้ำเงิน (r,g,b) ที่ต้องการเพิ่มลดความอิ่มสี จากนั้นจะมีการรับค่าแกนของภาพ

โดยให้แกน x เป็นแกนนอน และ แกน y เป็นแกนตั้ง และหาพิกัดอาเรย์ของภาพ ถัดจากนั้นดึงค่าสีมาจากแต่ละ pixel หลังจากนั้นคำนวณค่า lum ที่เป็นค่าใหม่ของ แต่ละสีดังสมการ

$$lum = (r(color.x)) + (g(color.y)) + (b(color.z))$$

จากนั้นนำเอาค่า lum ไปใส่ในสีทุกสี

3.2.1.4 กลับสี (Inversion)



ภาพที่ 3.4 ผังงานของอัลกอริธึม Inversion เมื่อมีการทำงานแบบขนาน

จากภาพที่ 3.4 เมื่อส่งงานไปให้กับจีพียูในที่นี้คือต้นฉบับรูปภาพ จากนั้นจะมีการรับค่าแกนของภาพ โดยให้แกน x เป็นแกนนอน และ แกน y เป็นแกนตั้ง และหาพิกัดอาเรย์ของภาพถัดจากนั้นดึงค่าสีมาจากแต่ละ pixel หลังจากนั้นคำนวณค่าสีแดง เขียว และน้ำเงินที่เป็นค่าใหม่ของแต่ละสีดังสมการ

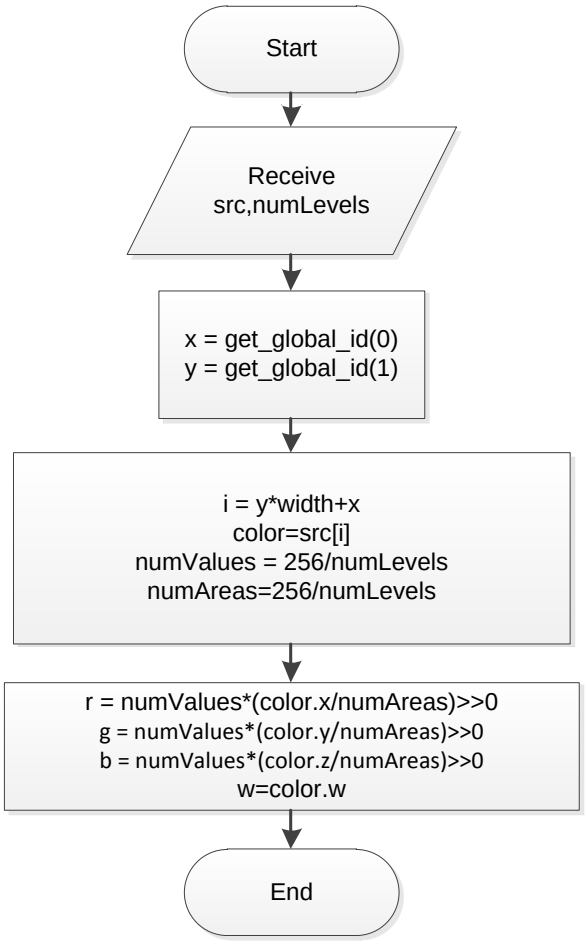
$$r = 255 - color.x$$

$$g = 255 - color.y$$

$$b = 255 - color.z$$

จากนั้นนำค่า r g และ b ใส่กลับเข้าไปที่สีเดิม

3.2.1.5 เร่งแสงสีให้จัดจ้าน (Posterize)



ภาพที่ 3.5 ผังงานของอัลกอริธึม Posterize เมื่อมีการทำงานแบบขนาน

จากภาพที่ 3.5 เมื่อส่งงานไปให้กับจีพียูในที่นี้คือต้นฉบับรูปภาพ และค่าระดับของ Posterize จากนั้นจะมีการรับค่าแกนของภาพ โดยให้แกน x เป็นแกนนอน และ แกน y เป็นแกนตั้ง และหาพิกัดอาเรย์ของภาพ ถัดจากนั้นนำค่าระดับไปคำนวณโดยการหารจาก 256 และหาพื้นที่โดยการนำไปหารจาก 256 เช่นกัน นำค่าที่ได้ไปหาค่าสีใหม่ดังสมการ

$$numValues = 256 / numLevels$$

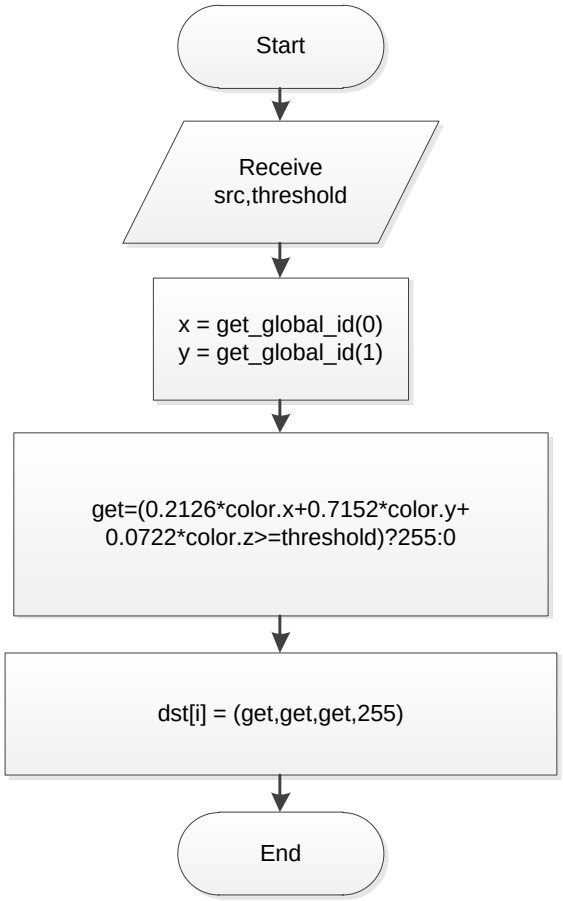
$$numArea = 256 / numLevels$$

$$r = numValues * (color.x / numAreas)$$

$$g = numValues * (color.y / numAreas)$$

$$b = numValues * (color.z / numAreas)$$

3.2.1.6 เทรชโฮลด์ (Threshold)



ภาพที่ 3.6 ผังงานของอัลกอริธึม Threshold เมื่อมีการทำงานแบบขนาน

จากภาพที่ 3.6 เมื่อส่งงานไปให้กับจีพียูในที่นี้คือต้นฉบับรูปภาพ และค่าระดับของ Threshold จากนั้นจะมีการรับค่าแกนของภาพ โดยให้แกน x เป็นแกนนอน และ แกน y เป็นแกนตั้ง และหาพิกัดอาเรย์ของภาพ ถัดจากนั้นนำค่า Threshold ไปเปรียบเทียบกับสีแต่ละสี หากมีค่ามากกว่าให้ตั้งค่าให้เป็นสีขาว หากน้อยกว่าให้เป็นสีดำ โดยอัลกอริธึมเป็นไปดังสมการ

$$get = 0.2126(color.x)) + 0.7152(color.y) + 0.0722(color.z)$$

3.3 การทดสอบ

ในการทดสอบจะมีการเลือกรูปภาพที่มีขนาดต่างกัน 3 ขนาด โดยจะใช้ภาพที่ได้จากกล้องสะท้อนภาพเลนส์เดี่ยวระบบดิจิตอล (DSLR : Digital Single Lens Reflex) ซึ่งปกติแล้วหากภาพมีขนาดใหญ่มากยิ่งขึ้นจะทำให้จำนวนพิกเซลในภาพมากขึ้น และทำให้ปริมาณข้อมูลมากขึ้นตามไปด้วย

สิ่งนี้ส่งผลให้เห็นประสิทธิภาพทางการประมวลผลด้านเวลามากยิ่งขึ้น โดยจะนำภาพ 3 ขนาด โดยขนาดใหญ่ที่สุดจะใช้ภาพที่ได้จากกล้องดิจิตอล DSLR โดยจะนำไปทดสอบกับทุกฟังก์ชัน ดังนี้

3.3.1 ปรับความมืดความสว่างกับภาพ

จะมีการเพิ่ม และลดความมืดความสว่างเข้าไปในภาพ หากเพิ่มความสว่างจะมีการเพิ่มค่าลงไปใน R, G, B ทำให้ภาพสว่างมากยิ่งขึ้น แต่หากลดความสว่าง ค่า R, G, B จะมีค่าลดลง

3.3.2 ปรับคอนทราสต์ของภาพ

จะมีการปรับค่าคอนทราสต์ ให้มีค่าที่แตกต่างกัน ทำให้ค่าคอนทราสต์มีการเปลี่ยนแปลง

3.3.3 ปรับความสดของสี

จะมีการปรับลดความสดของสีจากอัลกอริธึม Desaturation โดยจะมีการปรับค่า Factor ของสีให้ภาพมีค่าความสดของสีต่ำลง

3.3.4 ปรับให้มีการกลับสี

มีการกลับสีโดยการนำเอาค่าของสีขาว (255) ไปลบกับค่าสีทุกสี เพื่อให้เป็นค่าสีตรงข้ามกับสีเดิม

3.3.5 การเร่งแสงและสี

มีการปรับและเร่งแสงและสี โดยการกำหนดระดับที่แตกต่างกัน เพื่อให้ภาพมีความเปลี่ยนแปลง

3.3.6 การปรับค่าเทรสโฮลด์

ในการทดสอบนี้จะมีการปรับภาพให้เป็นสองสี คือขาว (255) และดำ (0) โดยหาค่ากลาง และเปรียบเทียบกับเทรสโฮลด์ที่ต้องการ

ในการทดสอบจะใช้อัลกอริธึมในการจับเวลาของภาษาจาวาสคริปต์ทุกฟังก์ชัน โดยการทดสอบทุกอัลกอริธึมบน ซีพียู 1 แกน ซีพียูหลายแกน จีพียู จากนั้นเปรียบเทียบประสิทธิภาพทางด้านเวลาในแง่ของเวลาหน่วยวินาที และความเร็วที่เพิ่มมากขึ้นเป็นสัดส่วน เพื่อให้เห็นความเปลี่ยนแปลงเมื่อมีการพัฒนาโปรแกรมด้วยเว็บซีแอล