

Report Progress

Division of Applied Mathematics, Department of Mathematics

Faculty of Science, Silpakorn University

Date : 23 พฤศจิกายน 2561

Advisor : ผู้ช่วยศาสตราจารย์ ดร.นพดล ชุมชอบ

Student : นายภัคพล พงษ์ทวี รหัส 07580028

Project Title : ขั้นตอนวิธีเชิงตัวเลขชนิดใหม่สำหรับการต่อเติมภาพที่ใช้การแปรผันรวมกับการประยุกต์สำหรับซ่อมแซมภาพจิตรกรรมไทยโบราณและการลบบทบรรยายจากอนิเมะ

(A new numerical algorithm for TV-based image inpainting with its applications for restoring ancient Thai painting images and removing subtitles from animes)

1 ที่มาและความสำคัญ

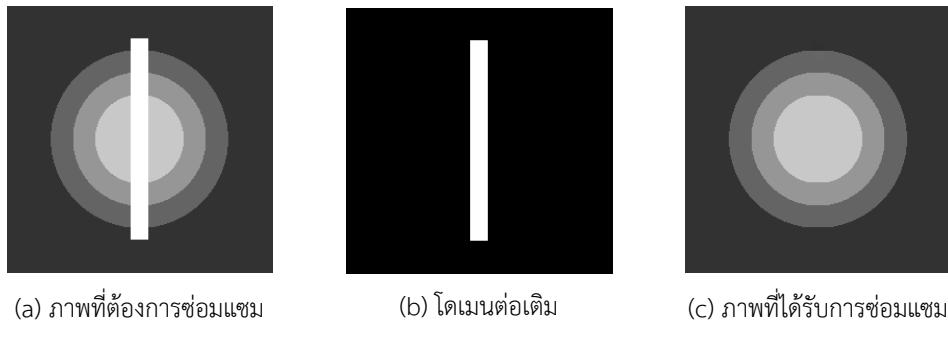
ในปัจจุบันการใช้ภาพดิจิตัล (digital images) ในสังคมเครือข่ายได้รับความนิยมอย่างแพร่หลายเนื่องจากโทรศัพท์เคลื่อนที่มีราคาถูกลงแต่มีความสามารถที่ชาญฉลาด สามารถทำหน้าที่ได้ตั้งแต่การเป็นกล้องดิจิตัลคอมแพค (compact digital camera) คุณภาพดีให้ภาพดิจิตัลที่มีความคมชัดสูงจนไปถึงการทำหน้าที่ดังเช่นเครื่องคอมพิวเตอร์ส่วนบุคคลที่สามารถเชื่อมต่อกับระบบเครือข่ายไร้สายเพื่อรับส่งภาพดิจิตัลในสังคมเครือข่ายด้วยความสะดวกและรวดเร็ว

นอกจากภาพดิจิตัลจะได้รับจากการถ่ายภาพด้วยโทรศัพท์เคลื่อนที่แล้ว ภาพดิจิตัลยังได้รับการถ่ายภาพด้วยกล้องดีเอสแอลอาร์ หรือ กล้องสะท้อนเลนส์เดียวแบบดิจิตัล (digital single lens reflex camera) กล้องโทรศัพท์มือถือ (หรือ กล้องดูดาว) หรือ เครื่องมือสร้างภาพถ่ายทางการแพทย์ (medical imaging device)

โดยทั่วไปภาพดิจิตัลจะได้รับการประมวลผลภาพก่อนนำไปใช้งานเพื่อให้สามารถใช้ข้อมูลที่ปรากฏบนภาพได้ตรงวัตถุประสงค์ของการใช้งานมากที่สุด ตัวอย่างเช่น ภาพบุคคล (portrait) อาจจำเป็นต้องได้รับการกำจัดสัญญาณรบกวนออกจากภาพและ/หรือปรับเพิ่มความละเอียดข้อมูลของความเข้มของสีและความสว่างของสีบนบริเวณใบหน้าก่อนนำภาพไปใช้งานเพื่อจัดทำต้นฉบับวารสารหรือหนังสือของสำนักพิมพ์ เป็นต้น

การต่อเติมภาพ (image inpainting) เป็นวิธีการประมวลผลภาพชนิดหนึ่งที่มีเป้าหมายเพื่อซ่อมแซมภาพด้วยการต่อเติมข้อมูลของความเข้มของสีบนบริเวณที่กำหนด (ต่อไปจะเรียกบริเวณนี้ว่าโดเมนต่อเติม (inpainting domain)) โดยอาศัยข้อมูลของความเข้มของสีที่ปรากฏในภาพ ตัวอย่างเช่น กำหนดให้รูปที่ 1.1 (a) แสดงภาพที่ต้องการซ่อมแซมระดับความเข้มของสีบนบริเวณแห่งวัตถุรูปร่างสีเหลืองสีขาว การต่อเติมภาพ

ตั้งกล่าวจะเริ่มด้วยการกำหนดให้บริเวณแห่งวัตถุรูปร่างสีเหลี่ยมสีขาวเป็นโหมดเมนการต่อเติมดังรูปที่ 1.1 (b) จากนั้นภาพที่ได้รับการซ่อมแซมหรือภาพที่ได้รับการต่อเติม (restored or inpainted image) ซึ่งแสดงในรูปที่ 1.1 (c) ได้มาจากขั้นตอนวิธีการต่อเติมภาพ (inpainting algorithm) ซึ่งได้รับการออกแบบเพื่อนำข้อมูลที่ปรากฏบนภาพในบริเวณใกล้เคียงกับขอบของโmodeนต่อเติมมาซ่อมแซมภาพ



รูปที่ 1.1: ตัวอย่างการซ่อมแซมภาพ; (a) ภาพที่ต้องการซ่อมแซม; (b) โmodeนต่อเติม; (c) ภาพที่ได้รับการซ่อมแซม

เท่าที่ผู้วิจัยศึกษาและค้นคว้ามาจนถึงขณะนี้ ผู้วิจัยพบว่าการต่อเติมภาพมักนิยมนำไปใช้งานสำหรับการปรับแต่งความสวยงามของภาพบุคคลที่ถ่ายจากโทรศัพท์เคลื่อนที่ เช่น การลบร่องรอยของรอยตีนกา การลบร่องรอยแผลเป็นที่เกิดจากสิ่วเสี้ยน การลดร่องรอยของความชรา หรือ การเพิ่มความใสและความเนียนของสีผิวนบนบริเวณใบหน้าผ่านโปรแกรมแอปพลิเคชันแต่งรูปภาพที่มีอยู่ในแอปสโตร์ (App Store) หรือ กูเกิลเพลย์ (Google Play) เป็นต้น

1.1 การซ่อมแซมภาพจิตรกรรมไทยโบราณ

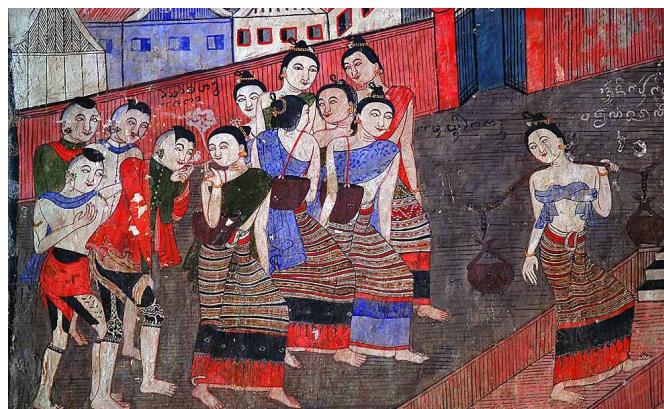
ภาพจิตรกรรมไทย คือ ภาพเขียนที่มีเอกลักษณ์ความเป็นศิลปะไทยซึ่งได้เด่นและแตกต่างจากภาพเขียนของชนชาติอื่น ในอดีต ช่างไทยได้สร้างสรรค์ลวดลายและสีสันบนภาพวาดเพื่อสะท้อนประเพณีและวัฒนธรรมในสังคมไทยที่เกี่ยวกับศาสนา ประวัติศาสตร์ โบราณคดี ชีวิตความเป็นอยู่ วัฒนธรรมการแต่งกายตลอดจนการแสดงการเล่นพื้นเมืองต่าง ๆ ของแต่ละยุคสมัย

อย่างไรก็ตาม ภาพจิตรกรรมไทยจำนวนไม่น้อยที่เสื่อมสภาพตามกาลเวลา และรอคوعการซ่อมแซมจากช่างในสมัยปัจจุบันที่ต้องไม่สร้างความเสียหายให้กับภาพเขียนเพิ่มขึ้นมากกว่าเดิม ที่ผ่านมาภาพที่ผ่านการซ่อมแซมมาแล้วจำนวนไม่น้อยได้รับความเสียหายหลังจากการซ่อมแซม ถึงแม้สภาพโดยรวมของภาพจิตรกรรมเดิมยังคงอยู่ แต่รายละเอียดในตัวภาพเขียนได้เปลี่ยนไป ก่อให้เกิดความเสียหายที่ประเมินค่าไม่ได้

การซ่อมแซมภาพจิตรกรรมไทยโบราณโดยใช้การต่อเติมภาพเป็นขั้นตอนของการซ่อมแซมแบบหนึ่งซึ่งไม่ก่อให้เกิดความเสียหายได้ ๆ กับภาพเดิม เนื่องจากเป็นการซ่อมแซมโดยการใช้ขั้นตอนวิธีเชิงตัวเลขบนภาพดิจิตัลซึ่งเป็นสำเนาของภาพเดิม ด้วยเหตุผลดังกล่าว ผู้วิจัยได้เลือกใช้วิธีการซ่อมแซมภาพจิตรกรรมไทยโบราณมีความจำเป็นเร่งด่วน เนื่องจากภาพที่ได้รับการซ่อมแซมด้วยการต่อเติมภาพสามารถนำไปใช้ประกอบการตัดสินใจเพื่อวางแผนก่อนการลงมือซ่อมแซมภาพเขียนจริงได้ นอกจากนี้ ขั้นตอนวิธีการต่อเติมภาพสามารถนำไปใช้สร้างแอปพลิเคชันโทรศัพท์เคลื่อนที่เพื่อในไปใช้เป็นข้อมูลในการเข้าชมภาพเขียนเดิมที่ยัง

ไม่ได้รับการซ่อมแซมและภาพเขียนที่ได้รับการซ่อมแซมโดยวิธีการทางคณิตศาสตร์จากแอปพลิเคชันที่พัฒนาขึ้น

รูปที่ 1.2 แสดงตัวอย่างภาพจิตรกรรมไทย¹ ที่ต้องได้รับการซ่อมแซมบนบริเวณแขนสีอ่อนของรูปปานาผู้ชายที่มีส่วนของสีแดงเดิมหลุดหายไป ทั้งนี้ในการซ่อมแซมภาพโดยการต่อเติมภาพ เราจะเริ่มด้วยการสร้างโคลเมนต่อเติมบนบริเวณสีพื้นผิวปูนที่แขนเสื้อ จากนั้นจึงนำขั้นตอนวิธีการต่อเติมภาพเพื่อซ่อมแซมภาพบริเวณนั้นให้เป็นสีแดง



รูปที่ 1.2: ภาพจิตรกรรมไทยที่วัดภูมินทร์ อำเภอเมือง จังหวัดน่าน

1.2 การลับบทบรรยายบนอนิเมะ

อนิเมะคือวิดีโอภาพวดการ์ตูนสต็อกญี่ปุ่นซึ่งเป็นที่นิยมของเยาวชนไทย ใน การรับชมอนิเมะ แม้ว่าเยาวชนไทยสามารถรับชมด้วยบทพากย์เสียงภาษาไทย แต่ก็สูญเสียอรรถรสของการรับชมจากบทบรรยายแบบแข็ง² (hardsub) ที่เป็นภาษาต่างประเทศในบริเวณด้านล่างของจอภาพ ในการซ่อมแซมอนิเมะด้วยการลับบทบรรยายภาษาต่างประเทศจึงเป็นงานที่ยุ่งยากและท้าทายมาก เนื่องจาก

- (1) อนิเมะเป็นวิดีโอซึ่งแสดงผลประมาณ 24 เฟรม(ภาพ)ต่อวินาที
- (2) แต่ละเฟรมอาจมีหรืออาจไม่มีบทบรรยายก็ได้
- (3) แต่ละเฟรมอาจมีหรืออาจไม่มีบทบรรยายเดียวกันก็ได้
- (4) แต่ละเฟรมเป็นการแสดงภาพสีที่มีระดับความคมชัดสูง (high definition) ขนาดมากถึง 1920×1080 พิกเซล

¹ภาพถ่ายที่วัดภูมินทร์ อำเภอเมือง จังหวัดน่าน; ภาพจาก <http://topicstock.pantip.com/camera/topicstock/2009/02/O7514399/O7514399.html> สืบค้นเมื่อวันที่ 23 กันยายน 2561

²บทบรรยายที่ไม่สามารถปิดหรือเปิดได้

ด้วยความท้าทายข้างต้น การพัฒนาขั้นตอนวิธีการต่อเติมภาพที่สามารถกำหนดโดยเมนูต่อเติมเชิงอัตโนมัติให้กับแต่ละเฟรมและประมวลผลได้แม่นยำจากการลับบทบรรยายสามารถทำงานได้แบบเรียลไทม์จึงเป็นสิ่งจำเป็นที่หลีกเลี่ยงไม่ได้

รูปที่ 1.3 แสดงตัวอย่าง 1 เฟรมของอนิเมะที่มีบทบรรยายแบบแข็ง³ ที่ต้องซ้อมแซมด้วยการลับบทบรรยายออก ทั้งนี้ในการลับบทบรรยายออกจากเฟรมโดยใช้การต่อเติมภาพ เราจะเริ่มด้วยการสร้างโดยเมนูต่อเติมแบบอัตโนมัติในบริเวณบทบรรยาย จากนั้นจึงนำขั้นตอนวิธีการต่อเติมภาพแบบเร็วเพื่อลับบทบรรยายออกจากเฟรม



รูปที่ 1.3: 1 เฟรมของอนิเมะที่มีบทบรรยายแบบแข็ง

โครงการวิจัยนี้ ผู้วิจัยมีเป้าหมายสำคัญคือการพัฒนาขั้นตอนวิธีการต่อเติมภาพแบบเร็วและแม่นยำ ชนิดใหม่เพื่อนำไปใช้สำหรับซ้อมแซมภาพจิตกรรมไทยและการลับบทบรรยายออกจากอนิเมะ

2 วรรณกรรมและทฤษฎีบทที่เกี่ยวข้อง

ในการกล่าวถึงขั้นตอนวิธีการต่อเติมภาพ จะเริ่มต้นด้วยการกล่าวบททวนเกี่ยวกับการต่อเติมภาพเฉดสีเทา (grayscale image) ก่อน ดังนี้

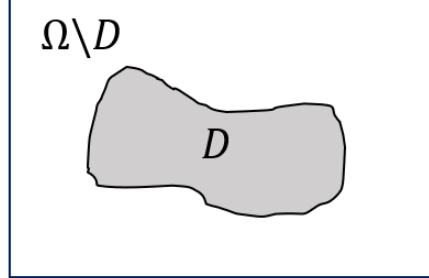
ให้ $\Omega \subset \mathbb{R}^2$ แทนโดเมนภาพ (image domain) $D \subset \mathbb{R}^2$ แทนโดเมนต่อเติม (ดูรูปที่ 2.1) และ $V \subset [0, \infty)$

ให้ $u : \Omega \rightarrow V$, $z : \Omega \rightarrow V$ แทนภาพที่ได้รับการซ้อมแซมและภาพที่ต้องการซ้อมแซม ตามลำดับ

ในที่นี้ $x = (x, y) \in \Omega$ แทนพิกัดทางกายภาพ (physical position) ของภาพ และ $u(x) \in V$ แทนระดับความเข้มของภาพ (image intensity) ที่ x และ Ω มีรูปร่างสี่เหลี่ยม

นอกจากนี้ความสามารถสมมติได้โดยไม่เสียหลักการสำคัญว่า $\Omega = [1, n]^2$ และ $V = [0, 1]$ เมื่อ $n > 0$ เป็นจำนวนเต็มบวก ทั้งนี้ เราจะเรียกภาพ u , z ที่นิยามข้างต้นว่าภาพเฉดสีเทา

³ภาพจาก <https://www.samehadaku.tv/2018/07/grand-blue-episode-1-subtitle-indonesia.html> สืบคันเมื่อวันที่ 23 กันยายน 2561



รูปที่ 2.1: D แทนโดเมนต่อเติม

2.1 ตัวแบบการต่อเติมภาพเขตสีเทาที่ใช้การแปรผันรวม

ในการต่อเติมภาพเขตสีเทา Chan และ Shen [1] ได้นำเสนอตัวแบบเชิงการแปรผัน (variational model) ที่ใช้รากวิลาร์เรซ์เซชันแบบการแปรผันรวม (Total variation based regularization) โดยพัฒนาต่อจากตัวแบบ ROF สำหรับการกำจัดสัญญาณรบกวน [2] ซึ่งตัวแบบเชิงการแปรผันนี้กำหนดโดย

$$\min_u \{ \mathcal{J}(u) = \frac{1}{2} \int_{\Omega} \lambda(u - z)^2 d\Omega + \int_{\Omega} |\nabla u| d\Omega \} \quad (2.1)$$

เมื่อ

$$\lambda = \lambda(\mathbf{x}) = \begin{cases} \lambda_0, & x \in \Omega \setminus D \\ 0, & x \in D \end{cases} \quad (2.2)$$

แทนพารามิเตอร์รักษาความเรียบ (regularization parameter) และ $\lambda_0 > 0$

โดยแคลคูลัสของการแปรผัน (Calculus of variations) จะได้สมการอย่างลักษณะที่เกี่ยวข้องกับ (2.1) เป็น

$$\begin{cases} -\nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) + \lambda(u - z) = 0, & \mathbf{x} \in (1, n)^2 \\ \frac{\partial u}{\partial \mathbf{n}} = 0, & x \in \partial\Omega \end{cases} \quad (2.3)$$

เมื่อ \mathbf{n} แทนเวกเตอร์หน่วยที่ตั้งฉากกับของของภาพ

ต่อไปจะกล่าวทบทวนวิธีการเชิงตัวเลขสำหรับแก้สมการเชิงอนุพันธ์อย่างใน (2.3)

(1) วิธีการเดินเวลาแบบชัดแจ้ง (explicit time marching method)

คณะวิจัย [2] ได้แนะนำวิธีการเชิงตัวเลขสำหรับการกำจัดสัญญาณรบกวนโดยใช้วิธีการเดินเวลาแบบชัดแจ้ง ซึ่งสามารถประยุกต์เป็นวิธีเชิงตัวเลขสำหรับการต่อเติมภาพได้ดังนี้

เริ่มจากการแนะนำตัวแปรเวลาสังเคราะห์ (time artificial variable) จากนั้นหาคำตอบแบบสภาวะคงตัว (steady-state solution) ในขณะที่ $t \rightarrow \infty$ ของสมการเชิงอนุพันธ์อย่างไม่เป็นเชิงเส้น

ที่ขึ้นอยู่กับเวลา

$$u(\mathbf{x}, t_{k+1}) = u(\mathbf{x}, t_k) + \tau \left(\nabla \cdot \left(\frac{\nabla u(\mathbf{x}, t_k)}{|\nabla u(\mathbf{x}, t_k)|} \right) + \lambda(\mathbf{x})(u(\mathbf{x}, t_k) - z(\mathbf{x})) \right), \quad u(\mathbf{x}, t_0) = z \quad (2.4)$$

เมื่อ $t_k = t_0 + k\tau$ ($\tau > 0$) แทนขั้นเวลาที่ k และ $t_0 = 0$ แทนขั้นเวลาเริ่มต้น

(2) วิธีการทำซ้ำแบบจุดตรึง (fixed-point iteration method)

คณะวิจัย [3] ได้แนะนำวิธีการเชิงตัวเลขสำหรับการทำจัดสัญญาณรบกวนโดยใช้วิธีการทำซ้ำแบบจุดตรึง ซึ่งสามารถประยุกต์เป็นวิธีเชิงตัวเลขสำหรับการทำจัดสัญญาณรบกวนได้ดังนี้

เริ่มจากการแนะนำดังนี้การทำซ้ำแบบจุดตรึง $\nu = 0, 1, 2, \dots$ และนิยามรูปแบบการทำซ้ำโดย

$$-\nabla \cdot \left(\frac{\nabla u^{[\nu+1]}}{|\nabla u|^{[\nu]}} \right) + \lambda(u^{[\nu+1]} - z) = 0, \quad u^{[0]} = z \quad (2.5)$$

เนื่องจาก $\frac{1}{|\nabla u|} = \frac{1}{\sqrt{u_x^2 + u_y^2}} \rightarrow \infty$ ในบริเวณที่ u มีความเข้มสีเป็นเอกพันธ์ ($u(\mathbf{x}) = \text{ค่าคงตัว}$)
เพื่อหลีกเลี่ยงปัญหาเชิงตัวเลขจะเกิดขึ้นใน (2.4) และ (2.5) เราจะใช้

$$|\nabla u| \approx |\nabla u|_\beta = \sqrt{u_x^2 + u_y^2 + \beta}, \quad 0 < \beta \ll 1$$

จึงทำให้สามารถเขียน วิธีเดินเวลาแบบขั้ดแจ้งสำหรับภาพ缺หายเป็นขั้นตอนวิธีได้ดังนี้

Algorithm 1: Explicit time marching gray-scale solver

Input:

u is image which is damaged image

D is image which is image of inpaint domain

λ is positive rational number which is regularization parameter

β is positive rational number which is used to avoid divide by zero

τ is positive rational number which is marching parameter

N is positive integer which is number of maximum loop

ε is positive rational number which is expected relative error

Output: inpainted image

Function **ExplicitTimeMarchingInpaint**($u, D, \lambda, \tau, \beta, N, \varepsilon$):

initialize $i = 0$

$\Lambda = \lambda \cdot D$,

$z = u$,

while $i < N$ and $err > \varepsilon$ **do**

$$u = u + \tau \left(\nabla \cdot \left(\frac{\nabla u}{\sqrt{u_x^2 + u_y^2 + \beta}} \right) + \Lambda(u - z) \right)$$

$$err = \frac{\|u - z\|}{\|u\|}$$

$i = i + 1$

end

return u

และจะสามารถเขียน ขั้นตอนวิธีจุดตรึงสำหรับภาพเนดเทาเป็นขั้นตอนวิธีได้ดังนี้

Algorithm 2: Fixed point gray-scale solver

Input:

u is image which is damaged image
 D is image which is image of inpaint domain
 λ is positive rational number which is regularization parameter
 β is positive rational number which is use to avoid devide by zero
 g is positive integer which is number of gauss seidel iteration
 N is positive interger which is number of maximum loop
 ε is positive rational number which is expected relative error

Output: inpainted image

Function $\text{FixedPointInpaint}(u, D, \lambda, \beta, g, N, \varepsilon)$:

```

while  $i < N$  and  $err > \varepsilon$  do
     $u = \text{InnerFixedPointInpaint}(u, D, \lambda, \beta, g)$ 
     $u = \left( \nabla \cdot \left( \frac{\nabla u}{\sqrt{u_x^2 + u_y^2 + \beta}} \right) + (z - u) \right)$ 
     $err = \frac{\|u - z\|}{\|u\|}$ 
     $i = i + 1$ 
end
return  $u$ 

```

Function $\text{InnerFixedPointInpaint}(u, D, \lambda, \beta, g, N, \varepsilon)$:

```

initialize  $\Lambda = \lambda \cdot D$ ,  $height = \text{height of } u$ ,  $width = \text{width of } u$ ,  $z = u$ ,
 $k = 0$ ,  $d = \frac{1}{\sqrt{u_x^2 + u_y^2 + \beta}}$ ,  $h = 1$ 
for  $k = 0$ ;  $k < g$ ;  $k = k + 1$  do
    for  $i = 0$ ;  $i < height$ ;  $i++$  do
        for  $j=0$ ;  $j < width$ ;  $j++$  do
             $u(i, j) = \frac{\Lambda_{i,j} z_{i,j} + \frac{1}{h^2} (d_{i,j} (u_{i+1,j} + u_{i,j+1}) + d_{i-1,j} u_{i-1,j} + d_{i,j-1} u_{i,j-1})}{\Lambda_{i,j} + \frac{1}{h^2} (2d_{i,j} + d_{i-1,j} + d_{i,j-1})}$ 
        end
    end
end
return  $u$ 

```

จาก (2.4) และ (2.5) เรายกบัวว่า β มีค่าน้อยลงมากขึ้นเท่าไหร่ ความแม่นยำของตัวแบบ (2.1) ยิ่งมีมากขึ้นเท่านั้น นอกจากนี้ เรายังพบอีกว่า การแก้สมการ (2.4) และ (2.5) ยิ่งมีความยุ่งยากมากขึ้นสำหรับ β ที่มีค่าน้อยๆ

เพื่อเอาชนะความยากเชิงตัวเลขนี้ คณะวิจัยโดย [4] ได้แนะนำวิธีการสปริทเบรกແມນซึ่งสามารถกล่าวถึงพ้องสังเขป ดังนี้

(3) วิธีการสปริตเบรกแม่น (Split Bregman method)

เริ่มจากการແນະນຳເວກເຕອບສໍາເລັດ \mathbf{w} ພາຣາມີເຕອບເບຣກແມນ (Bregman parameter) \mathbf{b} ແລະ ພາຣາມີເຕອບພັນລື້ (penalty parameter) $\theta > 0$ ແລະ ເປີຍິນ (2.1) ໃໝ່ ດັ່ງນີ້

$$\min_{u, \mathbf{w}} \{\mathcal{J}(u, \mathbf{w}) = \frac{1}{2} \int_{\Omega} \lambda(u - z)^2 d\Omega + \int_{\Omega} |\nabla \mathbf{w}| d\Omega + \frac{\theta}{2} \int_{\Omega} (\mathbf{w} - \nabla u + \mathbf{b}) d\Omega\} \quad (2.6)$$

ສໍາຫຼັບການຫາຄຳຕອບຂອງ (2.6) ເຮັດວຽກໃຊ້ວິທີການຫາຄ່າຕໍ່ສຸດແບບສລັບ (alternating minimization method) ໂດຍເຮັ່ມຈາກການຕັ້ງ \mathbf{w}^{old} ແລະ \mathbf{b}^{old} ຈາກນັ້ນແກ້ປຸ່ມຫຍ່ຍ່ອຍ

$$u^{\text{New}} = \arg \min_u \{\mathcal{J}_1(u) = \frac{1}{2} \int_{\Omega} \lambda(u - z)^2 d\Omega + \frac{\theta}{2} \int_{\Omega} (\mathbf{w}^{\text{old}} - \nabla u + \mathbf{b}^{\text{old}}) d\Omega\} \quad (2.7)$$

ຈາກນັ້ນໃຊ້ u^{New} ທີ່ໄດ້ຈາກການແກ້ປຸ່ມຫຍ່ຍ່ອຍໃນ (2.7) ເພື່ອແກ້ປຸ່ມຫຍ່ຍ່ອຍ

$$\mathbf{w}^{\text{New}} = \arg \min_{\mathbf{w}} \{\mathcal{J}_2(\mathbf{w}) = \int_{\Omega} |\nabla \mathbf{w}| d\Omega + \frac{\theta}{2} \int_{\Omega} (\mathbf{w} - \nabla u^{\text{New}} + \mathbf{b}^{\text{old}}) d\Omega\} \quad (2.8)$$

ສຸດທ້າຍຈຶ່ງປະບົບປຽບພາຣາມີເຕອບເບຣກແມນ

$$\mathbf{b}^{\text{New}} = \mathbf{b}^{\text{old}} + \nabla u^{\text{New}} - \mathbf{w}^{\text{New}} \quad (2.9)$$

ດຳເນີນການເຂັ້ນນີ້ຈະຮ່ວມມື $\|u^{\text{new}} - u^{\text{old}}\| < \epsilon_1$ ອີ່ວນ $\text{New} > \epsilon_2$ ເນື້ອ $\epsilon_1, \epsilon_2 > 0$

จะสามารถเขียน ขั้นตอนวิธีสปริทเบรกแม่นสำหรับภาพเขตเทาเป็นขั้นตอนวิธีได้ดังนี้

Algorithm 3: Split-bergman gray-scale solver

Input:

u is image which is damaged image

D is image which is image of inpaint domain

λ is positive rational number which is regularization parameter

θ is positive rational number which is panelty parameter that shouldn't too big or too small

N is positive interger which is number of maximum loop

g is positive integer which is number of gauss seidel iteration

ε is positive rational number which is expected relative error

Output: inpainted image

Function SplitBergmanInpaint($u, D, \lambda, \theta, g, N, \varepsilon$):

initialize $i = 0, b = \vec{0}, w = \vec{0}, z = u$

while $i < N$ and $err > \varepsilon$ **do**

$v = u$

$w = wSubproblemSolver(u, b, \theta)$

$u = uSubproblemSolber(u, w, v, b, D, \lambda, \theta, g)$

$b = b + -w$

$err = \frac{\|u-v\|}{\|u\|}$

$i = i + 1$

end

return u

Function uSubproblemSolver($(u, w, z, b, D, \lambda, \theta, gn)$):

initialize $b = \vec{0}, w = \vec{0}, v = u, width = \text{width of } u, height = \text{height of } u,$

$d = \left(\nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) + \Lambda(u - z) \right) \Lambda = \lambda \cdot D, h = 1$

for $k = 0; k < g; i++$ **do**

for $i = 0; i < height; i++$ **do**

for $j = 0; j < width; j++$ **do**

$L = \frac{u_{i,j-1}+u_{i,j+1}+u_{i-1,j}+u_{i+1,j}}{h^2}$

$B = \Lambda_{i,j}z_{i,j} - \theta d_{i,j} + \theta L_{i,j}$

$u_{i,j} = \frac{h^2}{\Lambda_{i,j} * h^2 + 4 * \theta} B$

end

end

end

return u

Function wSubproblemSolver(u, b, θ):

$w = \max(|\nabla b| - \frac{1}{\theta}, 0)$

return w

2.2 ตัวแบบการต่อเติมภาพสีที่ใช้การแปรผันรวม

ต่อไปเราจะพิจารณาภาพสีในระบบ RGB นั่นคือ เราสมมติว่า

$$\mathbf{u} = (u_1, u_2, u_3)^\top, \mathbf{z} = (z_1, z_2, z_3)^\top : \Omega \rightarrow V^3$$

เมื่อ $u_1, u_2, u_3 : \Omega \rightarrow V$ และ $z_1, z_2, z_3 : \Omega \rightarrow V$ แทนภาพในเขตสีแดง สีเขียว และสีฟ้าเงินของ \mathbf{u}, \mathbf{z} ตามลำดับ

ในทำนองเดียวกันกับตัวแบบการต่อเติมภาพเขตสีเทาที่ใช้การแปรผันรวม ตัวแบบการต่อเติมภาพสีที่ใช้การแปรผันรวมสามารถเขียนได้ดังนี้

$$\min_{\mathbf{u}} \{\bar{\mathcal{J}}(\mathbf{u}) = \bar{\mathcal{D}}(\mathbf{u}, \mathbf{z}) + \bar{\mathcal{R}}(\mathbf{u})\} \quad (2.10)$$

เมื่อ

$$\bar{\mathcal{D}}(\mathbf{u}, \mathbf{z}) = \frac{1}{2} \int_{\Omega} \lambda(u_1 - z_1)^2 d\Omega + \frac{1}{2} \int_{\Omega} \lambda(u_2 - z_2)^2 d\Omega + \frac{1}{2} \int_{\Omega} \lambda(u_3 - z_3)^2 d\Omega$$

และ

$$\bar{\mathcal{R}}(\mathbf{u}) = \int_{\Omega} |\nabla u_1| d\Omega + \int_{\Omega} |\nabla u_2| d\Omega + \int_{\Omega} |\nabla u_3| d\Omega$$

ดังนั้นเพื่อต่อเติมภาพสี จะเป็นการแก้ปัญหาการหาค่าต่ำที่สุดต่อไปนี้

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3} \{ \bar{\mathcal{J}}(\mathbf{u}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) &= \bar{\mathcal{D}}(\mathbf{u}, \mathbf{z}) + \sum_{l=1}^3 \int_{\Omega} |\mathbf{w}_l| d\Omega \\ &+ \frac{\theta_l}{2} \sum_{l=1}^3 \int_{\Omega} (\mathbf{w}_l - \nabla u_l - \mathbf{b}_l)^2 d\Omega \}, \quad \theta_l > 0 \end{aligned} \quad (2.11)$$

ซึ่งสามารถประยุกต์ใช้ริบภาพ缺เดา มาใช้งานกับภาพสีได้ดังต่อไปนี้
สำหรับวิธีการเดินเวลา สามารถประยุกต์จากขั้นตอนสำหรับภาพ缺เดาเป็นขั้นตอนสำหรับภาพสีได้ดังนี้

Algorithm 4: Explicit time marching color image solver

Input:

u is image which is damaged image
 D is image which is image of inpaint domain
 λ is positive rational number which is regularization parameter
 β is positive rational number which is use to avoid devide by zero
 τ is positive rational number which is marching parameter
 N is positive interger which is number of maximum loop
 ε is positive rational number which is expected relative error

Output: inpainted image

Function *ExplictTimeMarchingColorInpaint*($u, D, \lambda, \tau, \beta, N, \varepsilon$):
 foreach $l, l \in \{1, 2, 3\}$ **do**
 $| u_l = ExplictTimeMarchingInpaint(u_l, D, \lambda, \tau, \beta, N, \varepsilon)$
 end
 return u

ในทำนองเดียวกัน จะได้ว่าขั้นตอนการต่อเติมภาพสีสำหรับวิธีการตรึงจุด คือ

Algorithm 5: Fixed point color solver

Input:

u is image which is damaged image
 D is image which is image of inpaint domain
 λ is positive rational number which is regularization parameter
 β is positive rational number which is use to avoid devide by zero
 g is positive integer which is number of gauss seidel iteration
 N is positive interger which is number of maximum loop
 ε is positive rational number which is expected relative error

Output: inpainted image

Function *FixedPointColorInpaint*($u, D, \lambda, \beta, g, N, \varepsilon$):
 foreach $l, l \in \{1, 2, 3\}$ **do**
 $| u_l = FixedPointInpaint(u_l, D, \lambda, \tau, \beta, N, \varepsilon)$
 end
 return u

Algorithm 6: Split-bergman Color solver

Input:

- u is image which is damaged image
- D is image which is image of inpaint domain
- λ is positive rational number which is regularization parameter
- θ is positive rational number which is panelty parameter that shouldn't too big or too small
- N is positive interger which is number of maximum loop
- g is positive integer which is number of gauss seidel iteration
- ε is positive rational number which is expected relative error

Output: inpainted image

Function `SplitBergmanColorInpaint($u, D, \lambda, \theta, g, N, \varepsilon$)`:

```

foreach  $l, l \in \{1, 2, 3\}$  do
    |  $u_l = SplitBergmanInpaint(u_l, D, \lambda, \tau, \beta, N, \varepsilon)$ 
end
return  $u$ 

```

2.3 การวัดประสิทธิภาพของภาพที่ผ่านกระบวนการต่อเติม

หลังจากการต่อเติมภาพแล้วจำเป็นต้องพิจารณาว่าการวัดคุณภาพของภาพที่ผ่านการต่อเติมดีมากน้อยเพียงใด โดยในวิจัยนี้จะสนใจคุณภาพของค่าในแต่ละพิกเซลที่ใกล้เคียงกับภาพต้นฉบับ และโครงสร้างโดยรวมที่ใกล้เคียงกับภาพต้นฉบับ โดยการวัดค่าดังต่อไปนี้

2.3.1 Peak Signal Noise Ratio: PSNR

Peak signal-to-noise ratio (PSNR) [6] ใช้สำหรับวัดคุณภาพของภาพโดยเปรียบเทียบจากพิกเซลแต่ละพิกเซล โดยภาพที่มีความคล้ายต้นฉบับจะมีค่า PSNR เข้าใกล้ล้านนัต หรือก็คือยิ่งมีค่ามากยิ่งคุณภาพดี ซึ่งสามารถคำนวณได้โดย

$$PSNR = 10 \cdot \log_{10} \left(\frac{\text{peak}^2}{\sqrt{MSE}} \right)$$

เมื่อ MSE คือ mean square error และ $peak$ คือค่าสูงสุดโดยประเภทของภาพ ซึ่งสำหรับงานที่จะพูดถึงต่อไปนี้ จะพิจารณาภาพเป็นฟังก์ชันที่มีความเข้มของภาพอยู่ในช่วง $[0, 1]$ จึงได้ว่า $peak$ มีค่าเป็น 1

2.3.2 Structural Similarity: SSIM

Structural similarity (SSIM) [7] ใช้สำหรับวัดคุณภาพของภาพจากโครงสร้างของภาพ โดยพิจารณาว่าภาพนั้นมีโครงสร้างแตกต่างหรือคล้ายคลึงกับภาพต้นฉบับมากน้อยเพียงใด โดยมีค่าอยู่ระหว่าง 0 ถึง 1 หากทั้งสองภาพมีความคล้ายคลึงกันมากค่า SSIM จะเข้าใกล้กับค่า 1 ซึ่ง SSIM นั้นสามารถคำนวณได้โดย

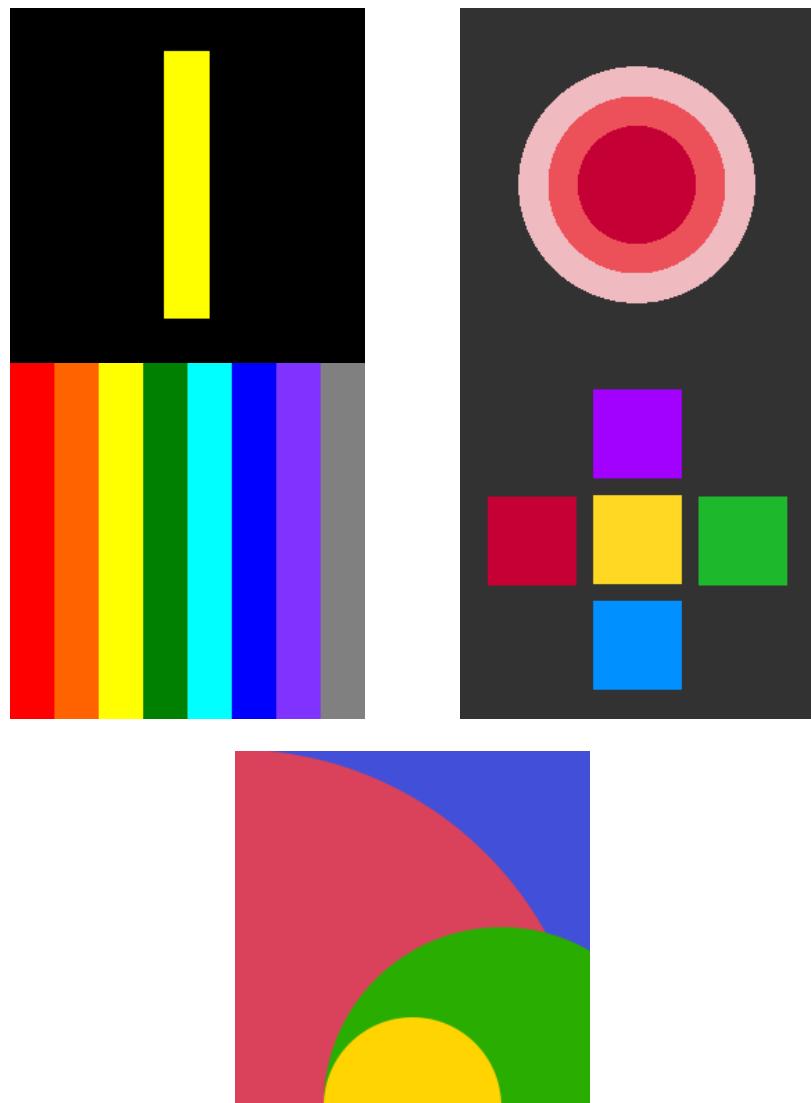
$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

เมื่อ x, y คือภาพที่ต้องการ 비교เทียบ μ คือค่าเฉลี่ยของภาพ σ^2 คือค่าความแปรปรวนของภาพ σ_{xy} คือความแปรปรวนร่วม $c_1 = (0.01L)^2, c_2 = (0.03L)^2$ และ L คือค่าสูงสุดโดยประเภทของภาพ ซึ่งสำหรับงานที่จะพูดถึงต่อไปนี้ จะพิจารณาภาพเป็นฟังก์ชันที่มีความเข้มของภาพอยู่ในช่วง $[0, 1]$ จึงได้ว่า L มีค่าเป็น 1

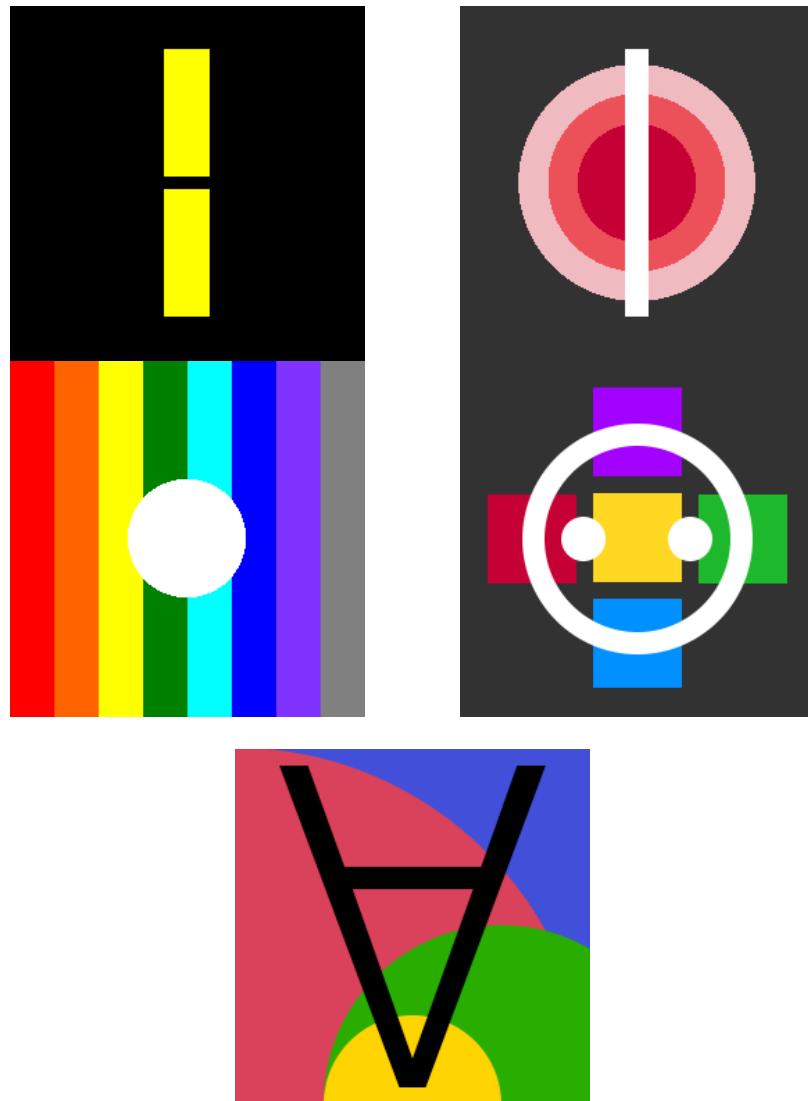
3 ผลการดำเนินงานเบื้องต้น

3.1 การซ้อมแซมภาพจิตรกรรมไทยโบราณ

สำหรับการซ้อมแซมจิตรกรรมไทยโบราณ ก่อนอื่นจะทำการปรับปรุงขั้นตอนวิธีเชิงตัวเลขที่มีอยู่แล้วเดิมให้ดีขึ้นเสียก่อน โดยระหว่างการปรับปรุงวิธีเชิงตัวเลข จะใช้ภาพสีที่ได้สร้างขึ้นทั้งสิ้น 5 ภาพ โดยแต่ละภาพมีขนาด 256×256 พิกเซล ซึ่งมีดังนี้



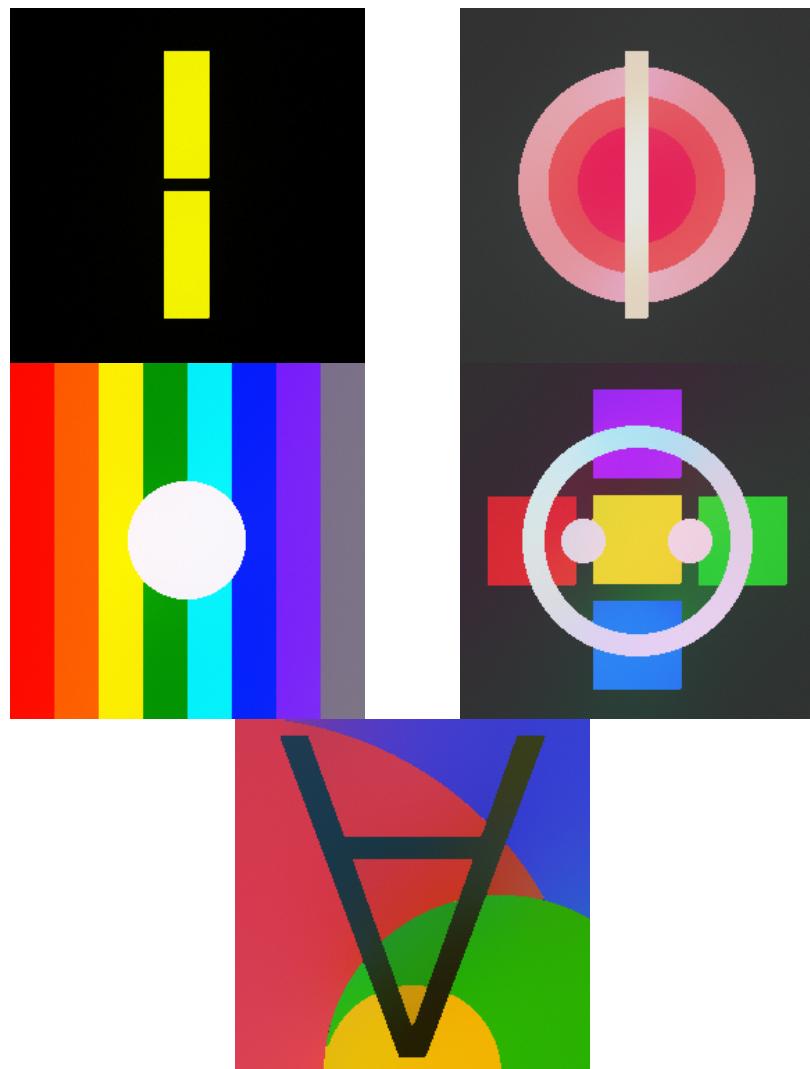
รูปที่ 3.1: ภาพต้นฉบับ



รูปที่ 3.2: ภาพที่จะทำการซ่อมแซม

3.1.1 การเปรียบเทียบประสิทธิภาพขั้นตอนวิธีเชิงตัวเลขที่มีอยู่สำหรับตัวแบบต่อเติมภาพสีที่ใช้การแปรผันรวม

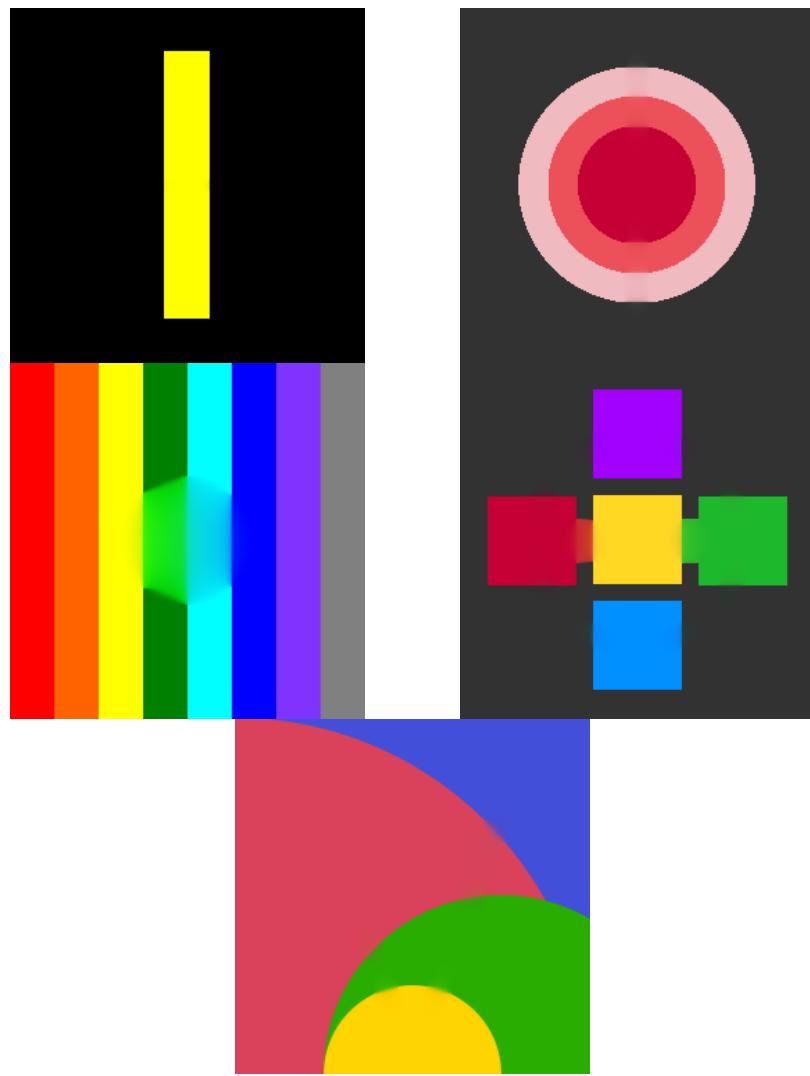
โดยจะทำการเปรียบเทียบความเร็วของวิธีการแก้การแปรผันรวมที่มีอยู่เดิม ดังที่ได้กล่าวไว้ในหัวข้อ 2.1 และหัวข้อ 2.2 โดยวิธีการทั้ง 3 วิธี จะทำการทำซ้ำจนกระทั่งภาพในการทำซ้ำรอบปัจจุบัน กับภาพการทำซ้ำในครั้งก่อนหน้า มีค่าความคลาดเคลื่อนสัมพัทธ์ (relative error) ต่างกันไม่เกิน 0.0001 หรือ ทำซ้ำเกิน 10,000 รอบ ซึ่งได้ผลลัพธ์เฉลี่ยของรูปภาพที่ใช้ทดสอบดังนี้



รูปที่ 3.3: ผลลัพธ์จากการเดินเวลา

รูปภาพ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
1	82.40	25.17	0.9997
2	127.36	17.92	0.9980
3	116.39	13.33	0.9941
4	160.59	12.40	0.9927
5	116.66	14.79	0.9958
เฉลี่ย	120.68	16.72	0.9960

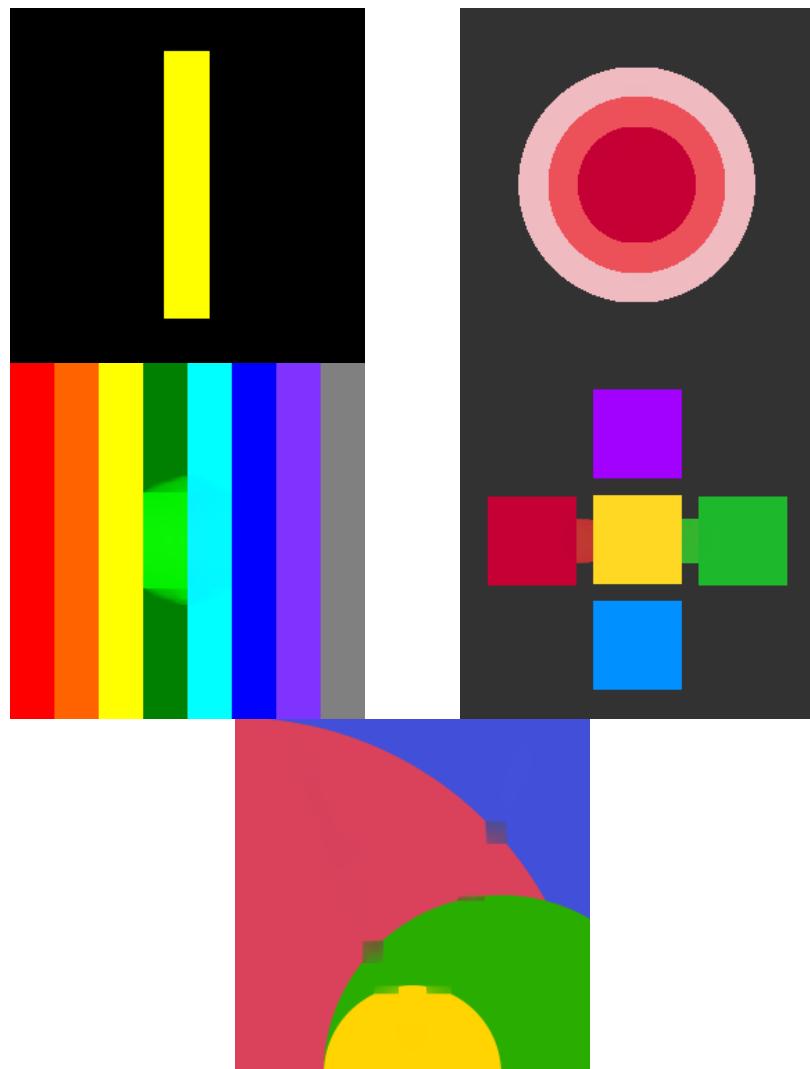
ตารางที่ 1: ผลลัพธ์ของวิธีการเดินเวลาสำหรับการต่อเติมภาพ



รูปที่ 3.4: ผลลัพธ์จากการจุดตรีง

รูปภาพ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
1	24.97	60.95	1.0000
2	53.06	37.69	1.0000
3	190.64	25.17	0.9997
4	50.63	28.81	0.9999
5	54.74	40.73	1.0000
เฉลี่ย	74.81	38.67	0.9999

ตารางที่ 2: ผลลัพธ์ของวิธีจุดตรีงสำหรับการต่อเติมภาพ



รูปที่ 3.5: ผลลัพธ์จากวิธีการสเปริทเบรกแมกน์

รูปภาพ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
1	3.39	71.54	1.0000
2	10.74	37.08	1.0000
3	24.50	26.08	0.9997
4	15.80	29.61	0.9999
5	15.85	32.78	1.0000
เฉลี่ย	14.06	39.42	0.9999

ตารางที่ 3: ผลลัพธ์ของวิธีสเปริทเบรกแมกน์สำหรับการต่อเติมภาพ

เมื่อเปรียบเทียบผลลัพธ์เฉลี่ยจากทั้ง 3 วิธีได้ดังตารางนี้

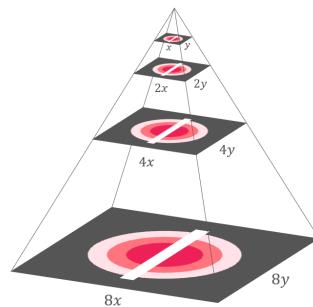
วิธีการ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
การเดินเวลา	120.68	16.72	0.9960
การทำซ้ำจุดตรึง	74.81	38.67	0.9999
การสปริทเบรกแม่น	14.06	39.42	0.9999

ตารางที่ 4: แสดงผลลัพธ์เฉลี่ยของวิธีการเชิงตัวเลขสำหรับการต่อเติมภาพ

ซึ่งจากทั้ง 3 วิธีที่ได้ทดสอบ จะเห็นได้ว่าวิธีการสปริทเบรกแม่นใช้เวลาอย่างกว่าวิธีอื่น และมีคุณภาพซึ่งพิจารณาจากค่า PSNR และค่า SSIM มากกว่าวิธีอื่น ซึ่งผู้จัดจะใช้วิธีสปริทเบรกแม่นมาใช้ในการพัฒนาวิธีต่อเติมภาพเป็นลำดับถัดไป

3.1.2 ขั้นตอนวิธีการสำหรับต่อเติมภาพชนิดใหม่

จากวิธีการสปริทเบรกแม่นนั้นจะใช้วิธีการทำคำตอบโดยวิธีการทำซ้ำจนกระทั่งถูกเข้าทางผู้ศึกษาจึงสนใจที่ทำคำตอบเริ่มต้นสำหรับการทำซ้ำที่ดีขึ้น เพื่อทำให้การทำซ้ำถูกคำตอบได้เร็วขึ้น โดยการทำงานกับรูปภาพที่เล็กกว่า จากนั้นจึงทำการขยายผลลัพธ์ที่ได้ขึ้นมา กับภาพใหญ่ ซึ่งวิธีนี้เรียกว่าวิธีพิริมิดรูปภาพ (pyramid methods) [5] โดยผู้จัดจะทำการย่อขนาดรูปลงครึ่งหนึ่งโดยใช้วิธี Bilinear Interpolation ทั้งสิ้น 4 ครั้ง จากนั้นเริ่มทำการต่อเติมภาพขนาดเล็ก จากนั้นนำผลลัพธ์ที่ได้จากการลดขนาดเล็ก ทำการขยายภาพขึ้นสองเท่าโดยใช้ Bilinear Interpolation ก่อนจะนำเฉพาะส่วนที่อยู่ในโดเมนต่อเติมของภาพที่ถูกขยายมาทำการต่อเติมเพื่อให้ส่วนที่ถูกขยายขึ้นมาเป็นคำตอบเริ่มต้นสำหรับการต่อเติมภาพในชั้นถัดไป



รูปที่ 3.6: วิธีการพิริมิดรูปภาพ

ซึ่งขั้นตอนวิธีสำหรับการทำพิริมิดรูปภาพสำหรับการต่อเติมภาพแบบสปริทเบรกแม่นเพื่อให้ประสิทธิภาพได้เร็วขึ้นนั้นสามารถทำได้ดังนี้

Algorithm 7: Split-bergman Color solver with Image Pyramid (Multi Resolution)

Input:

u is image which is damaged image

D is image which is image of inpaint domain

λ is positive rational number which is regularization parameter

θ is positive rational number which is panelyt parameter that shouldn't too big or too small

N_0 is positive interger which is number of maximum iteration on smallest size image

N_1 is positive interger which is number of maximum iteration on image that isn't the biggest neither smallest

N_2 is positive interger which is number of maximum iteration on biggest size image

g is positive integer which is number of gauss seidel iteration

ε is positive rational number which is expected relative error

Output: inpainted image

Function

ColorInpaint($u, D, \lambda, \theta, g, N_0, N_1, N_2, \varepsilon, c, m$):

Initialize $height =$ height of u , $width =$ width of u

if $c < M$ **then**

$x = \text{BilinearResize}(u, \lfloor width * 0.5 \rfloor, \lfloor height * 0.5 \rfloor)$

$y = \text{BilinearResize}(D, \lfloor width * 0.5 \rfloor, \lfloor height * 0.5 \rfloor)$

$r =$

$\text{MultiSplitBergmanColorInpaint}(x, y, \lambda, \theta, g, N_0, N_1, N_2, \varepsilon, c + 1, m)$

$R = \text{BilinearResize}(r, width, height)$

$u = \text{CopyByDomain}(u, D, R)$

end

if $c \neq 1$ **then**

if $1 = m$ **then**

 | $N_0 = N_1$

else

 | $N_0 = N_2$

end

end

return $\text{SplitBergmanColorInpaint}(u_l, D, \lambda, \tau, \beta, N_0, \varepsilon)$

โดยสำหรับขั้นตอนวิธีนี้ จะมีขั้นตอนวิธีอย่างสำหรับช่วยอีกสองขั้นตอน คือ

1. ขั้นตอนสำหรับการคัดลอกข้อมูลที่อยู่ในโดเมนต่อเติมจากรูป v ไปรูป u

Algorithm 8: Copy image inside inpaint domain from v to u

Input:

u is image which is damaged image
 D is image which is image of inpaint domain
 v is image which will copy to u image

Output: image

Function $\text{CopyByDomain}(u, D, v)$:

```

Initialize  $height = \text{height of } u$ ,  $width = \text{width of } u$ 
for  $i = 0; i < height; i++$  do
    for  $j = 0; j < i; j++$  do
        if  $D_{i,j} \neq 0$  then
            |  $u_{i,j} = v_{i,j}$ 
        end
    end
end
return  $u$ 

```

2. ขั้นตอนสำหรับปรับขนาดภาพด้วย bilinear interpolation

Algorithm 9: Bilinear Interpolation for Image resizing

Input:

u is image which need to resize
 x is positive integer which is new image height
 y is positive integer which is new image width

Output: image

Function $\text{BilinearResize}(I, x, y)$:

```

Initialize  $J$  is image that height  $x$  and width  $y$ ,  

 $v = \text{height of } I$  and  $w = \text{width of } I$ ,  

 $S_R = \frac{c}{a}, S_C = \frac{d}{b}, r = 1, 2, \dots, v, c = 1, 2, \dots, w$ ,  

 $r' = 1, 2, \dots, x, c' = 1, 2, \dots, y$ ,  

 $r_f = \lfloor r' \cdot S_R \rfloor$   

 $c_f = \lfloor c' \cdot S_C \rfloor$   

 $\Delta r = r_f - r$   

 $\Delta c = c_f - c$   

 $J(r', c') = I(r, c) \cdot (1 - \Delta r) \cdot (1 - \Delta c)$   

 $+ I(r + 1, c) \cdot \Delta r \cdot (1 - \Delta c)$   

 $+ I(r, c + 1) \cdot (1 - \Delta r) \cdot \Delta c$   

 $+ I(r + 1, c + 1) \cdot \Delta r \cdot \Delta c$ 
return  $J$ 

```

โดยจะทำการเบรียบเทียบจำนวนครั้งในขั้นที่รูปภาพมีขนาดเล็ก จนไปถึงขั้นที่มีขนาดใหญ่ ตัวอย่าง เช่น 10/3/3/10000 หมายถึงในขั้นเล็กสุดซึ่งขนาดเป็น 32x32 พิกเซลจะทำซ้ำไม่เกิน 10 ครั้ง ขั้นถัดมาขนาด เป็น 64x64 จะทำซ้ำไม่เกิน 3 ครั้ง และขั้นถัดมา ขั้นถัดมาขนาดเป็น 128x128 จะทำซ้ำ 3 ครั้ง และสุดท้าย ขนาด 256x256 จะทำซ้ำไม่เกิน 10,000 ครั้งหรือจนค่าความคลาดเคลื่อนสัมพัทธ์ต่างกันไม่เกิน 0.0001 ซึ่ง เมื่อทำการทดสอบแล้วได้ผลลัพธ์ดังตารางนี้

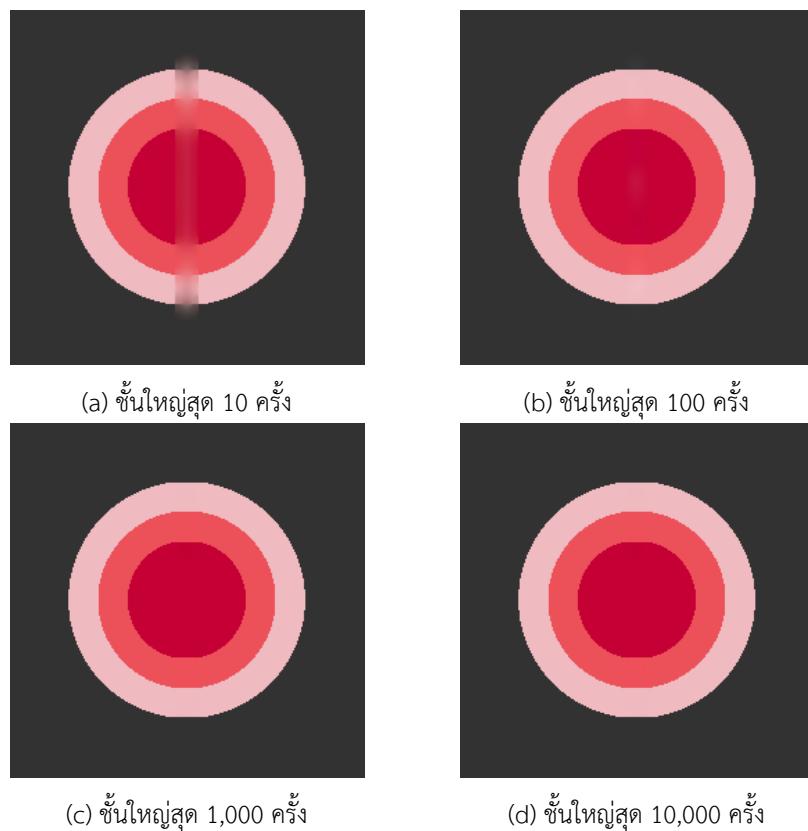
รูปแบบการทำซ้ำ	รูปภาพ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
ไม่ใช้พิรมิตรูปภาพ	1	4.49	71.54	1.0000
	2	13.16	37.08	1.0000
	3	29.46	26.08	0.9997
	4	20.50	29.61	0.9999
	5	19.32	32.78	1.0000
10/1/1/10000	1	2.44	69.59	1.0000
	2	11.31	37.04	1.0000
	3	23.48	27.34	0.9998
	4	16.60	29.42	0.9999
	5	13.75	33.53	1.0000
10/3/3/10000	1	2.24	69.96	1.0000
	2	10.91	37.05	1.0000
	3	21.99	27.66	0.9998
	4	12.70	29.35	0.9999
	5	11.49	33.69	1.0000
10/10/10/10000	1	1.83	71.58	1.0000
	2	7.83	37.05	1.0000
	3	16.75	28.62	0.9998
	4	11.89	29.32	0.9999
	5	8.00	34.26	1.0000
100/1/1/10000	1	1.43	67.63	1.0000
	2	7.17	37.10	1.0000
	3	20.86	27.70	0.9998
	4	12.80	29.64	0.9999
	5	9.17	33.14	1.0000
100/3/3/10000	1	1.68	71.18	1.0000
	2	7.41	37.11	1.0000
	3	21.08	28.00	0.9998
	4	13.28	29.38	0.9999
	5	7.96	33.34	1.0000
100/10/10/10000	1	1.76	71.56	1.0000
	2	7.32	37.04	1.0000
	3	16.62	28.65	0.9998
	4	13.18	29.39	0.9999
	5	7.45	33.94	1.0000

ตารางที่ 5: แสดงผลลัพธ์ของการใช้พิรมิตรูปภาพในการต่อเติมในขั้นที่ต่างกัน

รูปแบบการทำข้า	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
ไม่ใช้พิริเมติรูปภาพ	17.38	39.42	0.9999
10/1/1/10000	13.52	39.38	0.9999
10/3/3/10000	11.86	39.54	0.9999
10/10/10/10000	9.26	40.17	0.9999
100/1/1/10000	10.28	39.04	0.9999
100/3/3/10000	10.28	39.80	0.9999
100/10/10/10000	9.27	40.12	0.9999

ตารางที่ 6: แสดงผลลัพธ์เฉลี่ยของการใช้พิริเมติภาพในการต่อเติมในชั้นที่ต่างกัน

จากตารางจะสังเกตว่า ยิ่งจำนวนการทำข้าในชั้นที่รูปภาพมีขนาดเล็กจำนวนมากครั้ง จะยิ่งทำให้เวลาประมาณลดลง ซึ่งทำให้การทำการต่อเติมภาพ ใช้เวลาอยู่ในช่วง 10-100 วินาที ผู้วิจัยยังได้สังเกตอีกว่า การทำข้าชั้น จะถูกเข้าเร็วในช่วงแรก จากนั้นความเร็วในการถูกเข้าจะลดลง ซึ่งทำให้การทำการต่อเติมภาพเพียงไม่กี่ครั้งในรูปภาพขนาดใหญ่สุด มีผลลัพธ์ใกล้เคียงกับภาพต้นฉบับได้



รูปที่ 3.7: พิริเมติที่ลำดับการทำข้าเป็น 10/10/10 และชั้นใหญ่สุดทำในจำนวนครั้งที่ต่างกัน

โดยผู้วิจัยจึงกำหนดให้การทำข้าในรูปภาพขนาดใหญ่สุดมีการทำข้าเพียง 10 ครั้ง และพบว่าได้ผลลัพธ์ดังนี้

รูปแบบการทำข้า	รูปภาพ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
ไม่ใช้พีระมิดรูปภาพ	1	0.30	26.71	0.9998
	2	0.39	18.39	0.9982
	3	0.38	13.66	0.9944
	4	0.40	12.86	0.9934
	5	0.38	14.69	0.9956
10/1/1/10	1	0.29	40.10	1.0000
	2	0.41	31.28	0.9999
	3	0.46	16.51	0.9970
	4	0.47	26.56	0.9998
	5	0.39	28.25	0.9998
10/3/3/10	1	0.28	42.53	1.0000
	2	0.36	32.91	1.0000
	3	0.35	16.88	0.9972
	4	0.34	27.06	0.9998
	5	0.34	29.76	0.9999
10/10/10/10	1	0.31	50.06	1.0000
	2	0.41	34.01	1.0000
	3	0.38	18.19	0.9980
	4	0.39	27.50	0.9998
	5	0.40	33.05	1.0000
100/1/1/10	1	0.27	43.97	1.0000
	2	0.37	31.28	0.9999
	3	0.36	24.98	0.9997
	4	0.36	28.05	0.9998
	5	0.36	29.24	0.9999
100/3/3/10	1	0.29	45.08	1.0000
	2	0.36	32.36	0.9999
	3	0.40	0.40	0.9996
	4	0.38	27.88	0.9998
	5	0.37	30.28	0.9999
100/10/10/10	1	0.28	50.05	1.0000
	2	0.41	33.25	1.0000
	3	0.42	23.51	0.9995
	4	0.42	27.78	0.9998
	5	0.39	32.38	0.9999

ตารางที่ 7: แสดงผลลัพธ์ของการใช้พีระมิดภาพในการต่อเติมโดยกำหนดให้ขั้นรูปภาพขนาดใหญ่สุดทำข้าเพียง 10 ครั้ง

รูปแบบการทำซ้ำ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
ไม่ใช้พิระมิดรูปภาพ	0.37	17.26	0.9963
10/1/1/10	0.40	28.54	0.9993
10/3/3/10	0.33	29.83	0.9994
10/10/10/10	0.38	32.56	0.9995
100/1/1/10	0.34	31.50	0.9999
100/3/3/10	0.36	27.20	0.9999
100/10/10/10	0.38	33.39	0.9998

ตารางที่ 8: แสดงผลลัพธ์เฉลี่ยของการใช้พิระมิดภาพในการต่อเติมโดยกำหนดให้ชั้นรูปภาพขนาดใหญ่สุดทำซ้ำเพียง 10 ครั้ง

จากการจะเห็นว่า การทำซ้ำในชั้นที่รูปภาพมีขนาดเล็กมากจำนวนมาก ไม่ช่วยให้การประมวลผลได้เร็วขึ้น ผู้วิจัยจึงเลือกใช้การทำซ้ำแบบ 10/3/3/10 ในการต่อเติมภาพ

3.1.3 การทดสอบประสิทธิภาพในการซ่อมแซมภาพจิตกรรมไทยโบราณ

ชิ้นภาพจิตกรรมที่ใช้ทดสอบ มีทั้งสิ้น 5 ภาพ โดยแต่ละภาพเป็นภาพสีที่มีขนาด 256×256 พิกเซล ซึ่งทั้ง 5 ภาพได้แก่ ภาพที่ 3.8a⁴ และภาพที่ 3.8b⁵ คือ จิตกรรมฝาผนังวัดแก้วไพทุรย์ ภาพที่ 3.8c⁶ คือ จิตกรรมฝาผนังวัดพระยืนพุทธบາทყຸคล ภาพที่ 3.8d⁷ คือ จิตกรรมฝาผนังวัดคงคaram และภาพที่ 3.8e⁸ คือ จิตกรรมฝาผนังวัดท่าถนน โดยจะทำให้ข้อมูลข้างทั้ง 5 ภาพเกิดความเสียหาย โดยใช้ร้อยความเสียหายจากภาพพระเจ้าสร้างอดัม

⁴ภาพถ่ายที่วัดแก้วไพทุรย์; ภาพจาก [https://commons.wikimedia.org/wiki/File:จิตกรรมฝาผนัง_วัดแก้วไพทุรย์_\(7\).jpg](https://commons.wikimedia.org/wiki/File:จิตกรรมฝาผนัง_วัดแก้วไพทุรย์_(7).jpg) สืบคันเมื่อวันที่ 23 กันยายน 2561

⁵ภาพถ่ายที่วัดแก้วไพทุรย์; ภาพจาก [https://commons.wikimedia.org/wiki/File:จิตกรรมฝาผนัง_วัดแก้วไพทุรย์_\(2\).jpg](https://commons.wikimedia.org/wiki/File:จิตกรรมฝาผนัง_วัดแก้วไพทุรย์_(2).jpg) สืบคันเมื่อวันที่ 23 กันยายน 2561

⁶ภาพถ่ายที่วัดพระยืนพุทธบາทყຸคล; ภาพจาก https://commons.wikimedia.org/wiki/File:Wat_Phra_Yuen_Phutthabat_Yukhon_01.jpg สืบคันเมื่อวันที่ 23 กันยายน 2561

⁷ภาพถ่ายที่วัดคงคaram; ภาพจาก https://commons.wikimedia.org/wiki/File:จิตกรรม_อุบลสกัดคงคaram.JPG สืบคันเมื่อวันที่ 23 กันยายน 2561

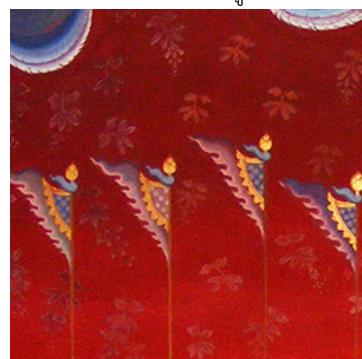
⁸ภาพถ่ายที่วัดท่าถนน; ภาพจาก https://commons.wikimedia.org/wiki/File:Wat_Tha_Thanon_05.JPG สืบคันเมื่อวันที่ 23 กันยายน 2561



(a) วัดแก้วไพทุรย์



(b) วัดแก้วไพทุรย์



(c) วัดพระยืนพุทธบาทยุคล



(d) วัดคงคาราม



(e) วัดท่าถนน



(f) รอยความเสียหาย

รูปที่ 3.8: ภาพต้นฉบับสำหรับใช้ในการทดสอบ



รูปที่ 3.9: ภาพที่ทำให้เสียหาย

จากนั้นทำการทดสอบการต่อเติมภาพทั้ง 5 โดยทดสอบวิธีสปริทเบรกแมน และวิธีที่พัฒนาขึ้นโดยใช้วิธีการสปริทเบรกแมนพร้อมทั้งการใช้พิระมิดรูปภาพที่มีการทำข้าแต่ละชั้นเป็น 10/3/3/10 ได้ผลลัพธ์ออกมาเป็นดังนี้



รูปที่ 3.10: ผลลัพธ์จากการสปีดเบรกแม่น

รูปภาพ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
1	2.95	33.92	1.0000
2	2.64	37.33	1.0000
3	3.49	37.21	1.0000
4	2.70	29.47	1.0000
5	15.85	32.78	1.0000
เฉลี่ย	2.72	34.89	1.0000

ตารางที่ 9: ผลลัพธ์การซ่อมแซมภาพศิลปะไทยจากการสปีดเบรกแม่น



รูปที่ 3.11: ผลลัพธ์จากการที่พัฒนาขึ้น

รูปภาพ	เวลาประมวล (วินาที)	PSNR (dB)	SSIM
1	0.40	34.13	1.0000
2	0.40	38.18	1.0000
3	0.39	37.73	1.0000
4	0.38	29.38	1.0000
5	0.39	37.11	1.0000
เฉลี่ย	0.39	35.30	1.0000

ตารางที่ 10: ผลลัพธ์การซ่อมแซมภาพศิลปะไทยจากวิธีที่พัฒนาขึ้น

ซึ่งทั้งสองวิธี ได้ผลลัพธ์การซ้อมแซมศิลปะไทยเฉียบคมมากดังนี้

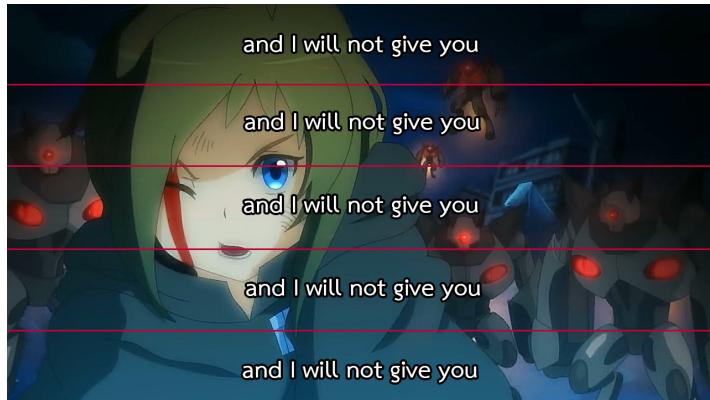
วิธีการ	เวลาประมวล (วินาที)	PSNR (dB)	SSIM
สเปรทเบรกแม่น	2.72	34.89	1.0000
วิธีการที่พัฒนาขึ้น	0.39	35.30	1.0000

ตารางที่ 11: แสดงผลลัพธ์เฉลี่ยของการซ้อมแม่มาพศิลปะไทย

จากตารางจะเห็นได้ว่า วิธีที่พัฒนาขึ้นนั้นสามารถทำงานได้เร็วกว่าวิธีสปริทเบรกเมนเดิม และยังมีคุณภาพที่ดีขึ้นด้วย

3.2 การลับบทบรรยายบนอนิเมะ

สำหรับการลับบทบรรยายอนิเมะ จะใช้วิดีโอ Anime Festival Asia Special Video - feat. Inori Aizawa ซึ่งผลิตโดย Collateral Damage Studios โดยจะตัดวิดีโอด้วยตัวเอง 1 นาทีแรกสำหรับการทดลอง โดยวิดีโอดังกล่าวขนาด 1280 x 720 พิกเซล แต่เนื่องจากโดยปกติแล้ว อนิเมะมักมีบรรยาย 1 ถึง 2 บรรทัด จึงทำการแบ่งวิดีโອอกอีกเป็น 5 ส่วนได้ขนาดเป็น 1280 x 144 พิกเซลก่อนนำไปทดสอบในลำดับถัดไป และสำหรับบทบรรยายที่จะใช้ทดสอบนั้น เนื่องจากวิดีโอ Anime Festival Asia Special Video - feat. Inori Aizawa ไม่มีคำพูดใดๆ จึงใช้บทความ lorem ipsum เป็นบทบรรยาย โดยจะทำการแสดงบทบรรยาย 1 บรรทัด ความยาว 3 วินาที ทุก 2 วินาที นั่นคือในวิดีโอดังกล่าวจะมีบทบรรยายทั้งสิ้น 20 บรรทัด



รูปที่ 3.12: การแบ่งไฟล์วิดีโอเป็น 5 ส่วนสำหรับใช้เป็น 5 ชุดทดสอบ

3.2.1 การหาบทบรรยายบนอนิเมะ

ก่อนจะจบบทบรรยายนั้น จะเป็นต้องหาบทบรรยายในภาพให้ได้เสียก่อน โดยบทบรรยายของอนิเมะนั้น มักจะขึ้นบริเวณด้านล่างของหน้าจอ และนอกจากนี้ บทบรรยายอนิเมะมักจะใช้ขอบของตัวอักษรเป็นสีดำ ถ้าหากตัวอักษรที่ใช้สีขาว หรือสีฟ้า ก็จะไม่สามารถมองเห็นได้ แต่ในบทบรรยายของอนิเมะนี้ ทางผู้สร้างได้ใช้สีฟ้าที่สามารถมองเห็นได้ ทำให้เราสามารถอ่านได้สะดวกมากขึ้น

Algorithm 10: Subtitle Finding Technique

Input: u is image which is part of anime frame**Output:** inpainting domain (subtitle)**Function** `FindSubtitle(u)`: $D =$ thresholding color in subtitle's edge color range of u $D =$ inverse D to select region inside subtitle edge $D =$ remove region that attach to edge of image D $D =$ remove region that too big to be subtitle from D $D =$ remove region that too small to be subtitle from D $D =$ dilate with subtitle's edge width**return** D

ซึ่งวิธีการหาบทบรรยายที่กล่าวไปข้างต้น จะทำการทดสอบกับบทความ lorem ipsum ที่ถูกแปลเป็นภาษาไทย ภาษาอังกฤษ และภาษาจีน โดยมีความสามารถในการหาโดยเม้นต์อเติมในบทบรรยายภาษาต่างๆ ดังนี้

ภาษา	วีดีโอ	จำนวนพิกเซลในโดเมน	จำนวนพิกเซลที่ตรวจพบ	จำนวนพิกเซลที่ผิดพลาด	ร้อยละการผิดพลาด
ไทย	1	23,190,522	24,044,004	2,108,772	9.09
	2	23,232,287	24,026,820	2,204,025	9.49
	3	23,189,082	24,300,589	2,081,340	8.98
	4	23,277,706	23,796,276	2,126,004	9.13
	5	23,221,502	24,247,935	2,185,864	9.41
อังกฤษ	1	27,281,185	28,631,063	3,477,960	12.75
	2	27,269,671	28,513,248	3,514,859	12.89
	3	27,325,148	28,611,300	3,815,082	13.96
	4	27,191,136	28,527,105	3,854,121	14.17
	5	27,326,584	28,709,405	3,909,582	14.31
จีน	1	28,509,908	30,058,101	3,953,067	13.87
	2	28,534,363	30,023,923	3,565,609	12.50
	3	28,537,968	30,015,047	3,553,128	12.45
	4	28,579,778	30,065,985	3,961,319	13.86
	5	28,558,848	30,354,275	3,671,730	12.86

ตารางที่ 12: แสดงความคาดเคลื่อนของการหาโดยเม้นต์อเติม ในบทบรรยายภาษาต่างๆ

ภาษา	จำนวนพิกเซลในโดเมน	จำนวนพิกเซลที่ตรวจพบ	จำนวนพิกเซลที่ผิดพลาด	ร้อยละการผิดพลาด
ไทย	23,222,220	24,083,125	2,141,201	9.22
อังกฤษ	27,278,745	28,598,424	3,714,321	13.62
ญี่ปุ่น	28,544,173	30,103,466	3,740,971	13.11

ตารางที่ 13: แสดงความคาดเคลื่อนเฉลี่ยของการหาโดเมนต่อเติม ในบทบรรยายภาษาต่างๆ

จากการทดลองทั้ง 3 ภาษาพบว่าวิธีการหาคำบรรยายนี้ มีร้อยละการผิดพลาดเฉลี่ยอยู่ที่ 11.98 ซึ่ง การทดลองจากนี้จะใช้วิธีการหาคำบรรยายนี้ในการหาโดเมนต่อเติมแบบอัตโนมัติ

3.2.2 การลบคำบรรยายจากบทอนิเมะ

สำหรับอนิเมะนั้น แต่ละเฟรมจะเป็นรูปภาพ เราจึงสามารถประยุกต์ใช้วิธีการซ่อมแซมภาพจิตรกรรมไทย มาใช้ในการลบคำบรรยายได้ แต่ผู้วิจัยก็ได้สังเกตว่า สำหรับอนิเมะที่เป็นวิดีโอแล้ว ในขณะที่ประมวลผลวิดีโอ เราสามารถใช้ผลการต่อเติมภาพจากภาพที่แล้ว มาใช้เป็นคำตออบเริ่มต้น โดยจะขอเรียกวิธีทั้งสามวิธีที่คิดขึ้นว่า วิธียืมเฟรม วิธีข้ามเฟรม และวิธียืมและข้ามเฟรม ซึ่งมีขั้นตอนวิธีดังนี้

ขั้นตอนวิธี การยืมเฟรม

Algorithm 11: Borrow Frame Technique

Input:

u is image which is current frame
 v is image which is previous frame
 D is image which is image of inpaint domain

Output: inpainted frame

Function BorrowFrameInpaint(u, D, v):

$U = \text{CopyByDomain}(u, D, \vec{0})$ $V = \text{CopyByDomain}(v, D, \vec{0})$ $s = \text{SSIM}$ of U and V if $s > 0.9$ then $u = \text{CopyByDomain}(u, D, v)$ end return $\text{MultiSplitBergmanColorInpaint}(u, D, \lambda, \theta, g, N_0, N_1, N_2, \varepsilon, 1, m)$

ขั้นตอนวิธี การข้ามเฟรม

Algorithm 12: Skip Frame Technique

Input:

u is image which is current frame
 v is image which is previous frame
 D is image which is image of inpaint domain

Output: inpainted frame

Function $\text{SkipFrameInpaint}(u, D, v)$:

```

     $U = \text{CopyByDomain}(u, D, \vec{0})$ 
     $V = \text{CopyByDomain}(v, D, \vec{0})$ 
     $s = \text{SSIM}$  of  $U$  and  $V$ 
    if  $s > 0.95$  then
        | return  $\text{CopyByDomain}(u, D, v)$ 
    end
    return
         $\text{MultiSplitBergmanColorInpaint}(u, D, \lambda, \theta, g, N_0, N_1, N_2, \varepsilon, 1, m)$ 
```

ขั้นตอนวิธี การข้ามและยืมเฟรม

Algorithm 13: Skip and Borrow Frame Technique

Input:

u is image which is current frame
 v is image which is previous frame
 D is image which is image of inpaint domain

Output: inpainted frame

Function $\text{BorrowSkipFrameInpaint}(u, D, v)$:

```

     $U = \text{CopyByDomain}(u, D, \vec{0})$ 
     $V = \text{CopyByDomain}(v, D, \vec{0})$ 
     $s = \text{SSIM}$  of  $U$  and  $V$ 
    if  $s > 0.95$  then
        | return  $\text{CopyByDomain}(u, D, v)$ 
    else if  $u > 0.9$  then
        |  $u = \text{CopyByDomain}(u, D, v)$ 
    end
    return
         $\text{MultiSplitBergmanColorInpaint}(u, D, \lambda, \theta, g, N_0, N_1, N_2, \varepsilon, 1, m)$ 
```

ชิ้นผลลัพธ์เบรี่ยบเทียบระหว่างแบบใช้วิธี ยึมเฟรม ใช้วิธีข้ามเฟรม และวิธีข้ามและยึมเฟรม ได้ผลดังตาราง

วิธีการ	วิดีโอ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
ไม่ใช้	1	130.03	32.19	0.9528
	2	135.17	29.98	0.9488
	3	142.11	30.54	0.9485
	4	151.42	30.79	0.9494
	5	147.70	33.48	0.9556
ยึมเฟรม	1	127.77	33.13	0.9701
	2	137.54	30.21	0.9590
	3	124.71	31.43	0.9620
	4	136.71	31.66	0.9614
	5	137.16	34.56	0.9748
ข้ามเฟรม	1	104.55	27.10	0.9429
	2	78.07	27.17	0.9351
	3	73.35	29.21	0.9393
	4	116.20	29.91	0.9423
	5	74.28	31.95	0.9442
ข้ามและยึมเฟรม	1	68.11	27.24	0.9424
	2	73.91	27.22	0.9386
	3	77.34	29.36	0.9437
	4	81.98	30.35	0.9483
	5	77.45	32.46	0.9540

ตารางที่ 14: แสดงผลลัพธ์ของการยึมเฟรมและข้ามเฟรม

วิธีการ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
ไม่ใช้	141.2850811	31.3936596	0.951021
ยึมเฟรม	132.7787761	32.1969618	0.9654718
ข้ามเฟรม	89.29141836	29.0667072	0.9407602
ยึมเฟรมและข้ามเฟรม	75.75894242	29.326696	0.9453868

ตารางที่ 15: แสดงผลลัพธ์เฉลี่ยของการยึมเฟรมและข้ามเฟรม

จากนั้นทำการทดสอบการต่อเดิมวิดีโอทั้ง 5 โดยวิธีที่คิดค้นขึ้นใช้วิธีการสปริทเบรกแม่นพร้อมทั้งการใช้พีระมิดรูปภาพที่มีการทำข้ามแต่ละชั้นเป็น 10/3/3/10 พร้อมทั้งใช้การข้ามเฟรมและยึมเฟรม ได้ผลลัพธ์ออกเป็นดังตารางนี้

วิธีการ	เวลาประมาณ (วินาที)	PSNR (dB)	SSIM
สปริทเบรกแม่น ที่คิดค้นขึ้น	*	*	*

ตารางที่ 16: แสดงผลลัพธ์เฉลี่ยของการลบบทบรรยายจากอนิเมะ

สำหรับวิธีสปริทเบรกแม่น เนื่องจากใช้เวลา 1 ชั่วโมงแล้วยังประมาณผลวิดีโอชุดทดสอบแรกไม่เสร็จ ทางผู้พัฒนาจึงตัดสินใจยุติการทดลอง เนื่องจากอาจต้องใช้เวลาการประมาณผลเป็นเวลาหลายชั่วโมงสำหรับ วิดีโอมีความยาว 1 นาที ส่วนวิธีที่คิดค้นขึ้น พบว่าสำหรับวิดีโอมีความยาว 1 นาที สามารถทำงานได้เสร็จ อุ่ย่างรวดเร็ว โดยใช้เวลาเพียง 75 วินาที

4 แผนการดำเนินงานวิจัย

แผนการดำเนินงานตลอดทั้งโครงการสามารถสรุปได้โดยย่อจากตารางต่อไปนี้

แผนการดำเนินงาน	เดือนที่											
	1	2	3	4	5	6	7	8	9	10	11	12
ศึกษาตัวแบบและขั้นตอนวิธีการต่อเติมภาพที่ใช้การแปลงรูปในเชิงลึก พัฒนาขั้นตอนวิธีสำหรับการต่อเติมภาพที่ใช้การแปลงรูปชนิดใหม่ ทดสอบขั้นตอนวิธีการต่อเติมภาพที่พัฒนาขึ้นโดยโปรแกรมคอมพิวเตอร์บนภาษาฟังก์ชันที่และภาษาจาวา ออกแบบผลลัพธ์ที่ได้จากการทดลองเชิงตัวเลข สรุปผลการดำเนินงานวิจัยและจัดทำรูปเล่มฉบับสมบูรณ์	x	x	x	x	x	x						

5 บรรณานุกรม

- [1] T.F. Chan and J. Shen , “Mathematical models of local non-texture inpaintings”, SIAM Journal on Applied Mathematics, vol. 62, no. 3, pp. 1019–1043, 2001.
- [2] L. I. Rudin, S. Osher, E. Fatemi, “Nonlinear total variation based noise removal algorithms”, Physica D: Nonlinear Phenomena, vol 60, issues 1–4, pp. 259-268, 1992.
- [3] C.R. Vogel and M.E. Oman,“Iterative methods for total variation denoising”, SIAM Journal on Scientific Computing. vol. 17, pp. 227-238, 1996.
- [4] T. Goldstein and S. Osher,“The Split Bregman Method for L1-Regularized Problems”, SIAM Journal on Imaging Sciences. vol. 2, issue 2, pp. 323-343, 2009.
- [5] E.H. Andelson and C.H. Anderson and J.R. Bergen and P.J. Burt and J.M. Ogden. ”Pyramid methods in image processing”. 1984
- [6] David Salomon. Data Compression: The Complete Reference (4 ed.). Springer. pp. 281. 2007.

- [7] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh and Eero P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, 2004.