四川大學

本科生毕业设计（学术论文）

题　　目　分数阶图像增强在计算神经科学中的应用

学　　院　软件学院（吴玉章）

专　　业　计算生物交叉试验班专业

学生姓名　李宬睿

学　　号　2016141021030　年级　2016

指导教师　蒲亦非，邓伟

教务处制表

二０一八年五月二十日

摘　要

计算生物交叉试验班

学生　李宬睿　　指导老师　蒲亦非，邓伟

[摘要] 随着近年来计算神经科学的发展，越来越多的计算工具被应用在认知神经科学的研究当中。分数阶微积分作为当前科研工程当中新兴的手段，在计算神经科学方面的应用也有巨大潜力以待挖掘。同时，分数阶微积分所具备的独特的性质，使其在图像增强方面显现出天然的优势。因此，本文将会研究最新的分数阶微积分图像增强算法，并将其应用在计算神经科学当中，以尝试发掘二者的内在关联，从而揭示分数阶微积分图像增强的良好性质与大脑视觉认知的本质关系。

首先，文章着重研究了最新的分数阶全变分 Retinex 图像增强算法，并针对其存在的计算时间过长、计算方式选取等问题，利用较底层的编程思想提出了改进算法。一系列比较实验证明，该算法至少存在三倍的提速空间；且在计算方式的选取上，使用二范数和 PU-2 算子所定义的一阶微分在通常情况下是更优的选择。之后，文章比较了人视觉认知与分数阶微分掩模的相似性。通过对比研究发现，视神经系统中两级细胞间的侧向抑制现象与分数阶微分掩模所起的作用高度一致，从而从认知神经科学的角度解释了为何分数阶微积分在图像增强方面具备其天然的优势。这一观点在智能图像增强算法的发展和认知神经科学的研究上具有启发性意义。文章最后将分数阶微积分图像增强应用于脑影像图的增强上。通过实验分析得出，分数阶全变分 Retinex 算法与 PU-2 算子分数阶掩模滤波方法对于脑影像图的增强各有优劣；而如果将二者结合，则能得到更好的增强影像，作为辅助图像供相关研究者参考。

[主题词] 分数阶微积分，图像增强，视觉认知，计算神经科学

ABSTRACT

Computational Biology

Student： LI Cheng-rui　　Adviser: PU Yi-fei, DENG Wei

[Abstract] With the developing of computational neuroscience, an increasing number of computational tools are applied to the research of cognitive neuroscience. As an emerging tool in current research and engineering, fractional calculus also has its potential in computational neuroscience that need to be explored. Moreover, the unique properties of the fractional calculus make it has a natural advantage in image enhancement. Therefore, in this article, we will study the latest fractional-calculus-based image enhancing algorithms and apply it to the computational neuroscience, and then try to find their internal relationships for revealing the essential relation between the good properties of fractional calculus in image enhancement and the visual cognition in our brain.

First, we focus on the latest fractional-order variational framework for Retinex. For its existing problems such as long computing time and the selection of its computation method, we propose an improved algorithm based on ground-floor programing strategy. A series of comparative experiment proves that the computing time of this algorithm could be reduced to about 1/3 of the original by improving it; and with respect to the selection of computation method, 2-norm and the first-order derivative defined by PU-2 operator are better choices in general conditions. Then, we compare the similarity of human visual cognition and fractional differential mask. Through a detailed comparison, we find that the high consistency between the lateral inhibition phenomenon of the bipolar cells in the optic nervous system and the function of the fractional differential mask explains why fractional calculus has its natural advantages in image enhancement from the perspective of cognitive neuroscience. This view provides a heuristic insight for the development of intelligent image enhancement algorithm and the advancement of cognitive neuroscience studies. In the last part, we apply the fractional-calculus-based image enhancement to the task of brain image enhancement. From the experimental analysis, we obtain that both the fractional-order variational framework for Retinex and the fractional mask filtering method of PU-2 operator have advantages and disadvantages; and if we combine them, it would give a better enhancement effect, which can be worked as auxiliary images for relevant researchers as references.

[Key Words] fractional calculus, image enhancement, visual cognition, and computational neuroscience

目 录

1 绪 论

1.1 引言

随着计算科学的发展，数学和计算模型越来越多的被应用到各类科研与工程领域，并衍生出了众多交叉学科，如将数学模型与计算机工具应用在生物科学上的计算生物学，将计算模型应用在金融领域的金融工程等。脑与认知神经科学作为神经科学、心理学等众多学科的融合学科，已经越来越多的开始将实验与数学模拟的辅助结合在了一起[1]，如用神经网络模型模拟人视觉的感知学习[2]。自从有了最早的人工智能，计算机科学家一直想接受更多来自心理学家和神经生物学家研究结果的启发，神经科学家和心理学家也一直在不断尝试使用最新的计算工具来辅助研究人的大脑。然而，脑与认知神经科学和人工智能之间，仍然存在实质性的区别。发展统一的理论将二者更加紧密的关联起来，一直是很多科学家不懈努力的方向之一。

在计算机科学中，一个与人最重要的感官——眼睛相关联的领域就是计算机视觉与图像处理。多年来人们一直不断努力想让计算机拥有人的高级视觉认知和辨析图像的能力。由此发展出了众多类型的算法，用于处理和分析图像。分数阶微积分是数学分析的一个分支，虽然其已经有很长的发展历史，但是直到最近才被少数科学家和工程师认识并利用起来。有了分数阶微积分这一重要数学工具，在科学计算、工程实践、现代信号处理及图像处理中发挥了重要作用[3]。用分数阶微积分处理图像是其众多应用领域之一，并且这种方法具有其天然的优势。首先，分数阶微积分把整数阶的微积分推广到了非整数阶，甚至复数域，是一种更普适的算子。此外，分数阶微积分天生具有一些特殊的性质，包括但不限于记忆性、非局部性、弱奇异性等[4]。因此，使用分数阶微积分处理图像，可以在保留图像细节的同时，更加充分的利用周围像素点的信息，从而得到更好的图像处理结果。

考虑到图像处理中比较重要的一类问题的图像增强，且对图像增强算法的进一步研究有利于启发计算神经科学中有关视觉认知的一些问题，并能够将图像增强这一工具用于脑影像学中，本文试图研究分数阶全变分 Retinex 图像增强算法（FR）。除了对其进行改进外，还尝试将其应用在计算神经科学的研究当中，发掘这种图像处理算法与人眼视觉认知的联系和区别。

1.2 相关研究进展

1.2.1 图像增强算法

图像平滑和图像增强是数字图像处理中两类重要的处理目的[5]。从最终的处理效果来看，图像平滑的过程和图像增强的过程可以认为是朝两个相反的方向对数字图像进行处理。在图像增强中，一类比较新的算法是 Retinex 算法[6]。Retinex 算法借用了 Retinex 理论中对人视觉认知的一种解释[7]，即我们看到的图像的本质是反射光，而反射光由入射光和物象本身固有的反射率相乘决定。因此，如果能正确的估计出入射光，就可以计算固有反射率。之后再根据我们的需求加上一特定的入射光，得到增强/修正后的结果图。

对于 Retinex 算法，除了有比较老的 SSR[8]、MSR[9]算法外，还有 Ng, Wang 二人提出的用于 Retinex 的全变分算法[10]。其利用全变分法估计入射光后，对图像进行增强处理。这种方法相比于较老的 SSR, MSR 算法，效果更好，但计算复杂度较高。Pu 等人最近将 Ng, Wang 二人提出的整数阶 Retinex 全变分算法推广到了分数阶。即

利用了上述分数阶微积分的一些良好性质，更好的估算入射光，以达到比整数阶全变分算法更好的图像增强结果。当然，分数阶计算的复杂度更高。

对于分数阶全变分 Retinex 算法[11]，其核心主要用了分数阶偏微分方程、分数阶梯度下降法、以及 PU-2 分数阶微分算子/掩模。分数阶偏微分方程主要是利用全变分法，从数学上解决对入射光估计的理论公式；分数阶梯度下降法将上述的公式通过数值算法实现；而 PU-2 分数阶微分算子提供了计算分数阶微分的高收敛性掩模。三者结合起来，再根据 Retinex 的理论以及 Gamma 校正理论，就可以得到分数阶图像增强处理的结果图了。

文献中指出，虽然分数阶 Retinex 算法处理得到的增强图像效果最佳，但是其计算量相比于其整数阶 Retinex 算法大很多。如果与传统的 SSR, MSR 算法相比，则有数量级上的差距。其根本原因在于，除了要用时间步进法迭代求解分数阶极小值点外，对图像矩阵多次求其分数阶微积分的计算量也异常的大。但是，如果仔细观察则会发现，虽然由公式得出的理论复杂度较高，但是在实际操作中，有很大的优化空间。也就是说，算法可以在描述及实现两个角度进一步优化，以大大提高计算速度。

同时，图像处理也都面临一个普遍问题，即对图像边缘的处理。一般情况下，图像处理尽管涉及到边缘，不同的边缘处理方式也不会带来显著的差距。有时候，甚至可以忽略对边缘像素点的处理，如平滑去噪[5]。然而，对于分数阶 Retinex 算法来说，对一个像素点求其分数阶微积分本身就要利用周围很多的像素点。而且，如果忽略边缘像素点或处理不得当，误差会随时间步进法的迭代过程不断累积并向图像中央方向传播。因此，给出一个更严谨的处理图像边缘问题的方法，不仅对本算法有意义，对其他图像处理也有一定的指导性作用。此外，该算法还有很多可以研究部分。比如，算法在计算八个方向上的分数阶微分范数时，采用不同范数的区别、在计算分数阶微分时掩模尺寸取多大最适宜等。

1.2.2 视觉认知理论

在众多认知神经科学家已经在他们不断地努力下，视觉认知理论已经越来越成熟[12-14]。从神经元到神经网络系统、从编码到解码、从感受到认知、从眼睛到大脑，不同层次和角度的视觉理论都有他们各自的观点。有些比较成熟，已经经过了实验的验证，而有些也只是基于数学模型进行的一种模拟和推测。从视网膜的细胞组成来看，较为底层的两极细胞就负责对照入眼中的光信号进行正差分感知（on-center bipolar cell）和负差分感知（off-center bipolar cell）[15]。之后又有神经节细胞、简单细胞、复杂细胞等进行一层层的延迟差分感知，最终在大脑的高级区域形成了对视觉的认知。这种层级不断对变化进行感知的过程，与微积分的效果非常类似。

这其中，和图像增强最相关的一种机制叫做侧向抑制[16]。其主要表现为，相邻视神经元的活动会抑制某些神经元。在这种机制下，当光照射到物象边缘的时候，由于边缘内外两侧的两极细胞活动之间产生了侧向抑制，二者的兴奋程度差距比实际的物理刺激差距更大，从而实现了提高对比度的效果。如果能将这种更复杂、更贴近于真实神经活动的处理方式应用在图像增强上，或借助现有分数阶图像增强理论进一步研究视觉认知，二者都会取得更进一步的理解。

1.3 论文主要工作

本文会在第二部分介绍文中用到的一些基础数学知识和基本算法，即数阶微积分的 G-L 离散定义式、

Gamma 极限和 Retinex 算法理论。在第三部分首先重点研究了最新的分数阶全变分 Retinex 图像增强算法，并分析其算法的复杂度和存在的问题；之后在算法的描述和实现两个方面提出改进方案。第四部分中，我们将会比较改进后算法与原文所用算法的计算速度，验证该算法的计算速度有很大的提升空间；并研究比较分数阶全变分 Retinex 算法中一些细节部分的实现对处理效果的区别。第五部分，会尝试将其应用在计算神经科学领域。最后的第六部分中则是对全文的讨论和总结。

2 背景知识简介

2.1 分数阶微积分

分数阶微积分的定义有许多种类。本文用到的是其最基本的定义式——Grünwald-Letnikov 定义[17]：可微积分函数$f(x)$的v阶分数阶微积分为

$$\mathrm{D}_x^v f(x)=\frac{\mathrm{d}^v f(x)}{[\mathrm{d}(x-a)]^v}=\lim_{N\to\infty}\left[\frac{\left(\frac{x-a}{N}\right)^{-v}}{\Gamma(-v)}\sum_{k=0}^{N-1}\frac{\Gamma(k-v)}{\Gamma(k+1)}f\left(x-k\left(\frac{x-a}{N}\right)\right)\right] \tag{2.1}$$

其中$[a,x]$是$f(x)$的持续区间，v不一定为整数，$\Gamma(\alpha)\coloneqq\int_0^\infty \mathrm{e}^{-x}x^{\alpha-1}\mathrm{d}x=(\alpha-1)!$。由于其定义方式是离散的，在图像处理这类离散问题中实现更方便。

注意到，对于(2.1)中的$\frac{\Gamma(k-v)}{\Gamma(v)}$，当$v\le 0, v\in\mathbb{Z}$时，有$\Gamma(v)=\infty$。然而其比率是有限的：$\forall n,N\in\mathbb{N}^+$，因为

$$\Gamma(-n)=\frac{\Gamma(-n+1)}{-n}=\frac{\Gamma(-n+2)}{(-n)(-n+1)}=\cdots=\frac{\Gamma(-n+n+1)}{(-n)(-n+1)(-n+2)\cdots(-n+n)}$$

所以有

$$\frac{\Gamma(-n)}{\Gamma(-N)}=\frac{(-N)(-N+1)\cdots(-N+N)}{(-n)(-n+1)\cdots(-n+n)}=\frac{(-1)^{N+1}N!}{(-1)^{n+1}n!}=(-1)^{N-n}\frac{N!}{n!} \tag{2.2}$$

2.2 图像处理中的 Retinex 算法

当前图像增强处理中，人们最常用到的是 Retinex 算法[6,7,18]。Retinex 算法理论认为，人眼接收到的图像本质为反射光，其实质为入射光照到物体表面经反射后所成的结果，反射光S与入射光L和反射率R的关系为

$$S=L\cdot R \tag{2.3}$$

对于一幅图像来说，图像所呈现出的是公式中(2.3)中的S。如果把一副 HSV 图像在(x,y)处的像素亮度 V 通道值记为$S(x,y)$，则$R(x,y)\in(0,1]$表示该位置物体本身的亮度特性（反射率），$L(x,y)$表示该位置的入射光强。因此，通过某种方式根据$S(x,y)$估计入射光强度$L(x,y)$，并将其从$S(x,y)$中除去，则可得到物体本身固有的反射率。最后将其与 Gamma 校正后的光强重新相乘，从而获得图像增强的结果。

3 算法或系统分析与设计

3.1 用于估计入射光l的分数阶全变分 Retinex 算法

3.1.1 分数阶全变分梯度下降法

对于一副图像，将其像素值（反射光）记为关于位置的函数$S(x,y)$。为方便计算，对公式(2.3)取其对数形式

$$s(x,y)=l(x,y)+r(x,y) \tag{3.1}$$

其中$s(x,y)=\ln S(x,y)$, $l(x,y)=\ln L(x,y)$, $r(x,y)=\ln R(x,y)$，并且满足$l\geq s, r\leq 0$。为估计l，构造其能量函数[11,19]

$$E(l)=\iint_{\Omega}[\|\mathrm{D}^{v_1}l\|^{v_2}+\alpha_1|l-s|^{v_2}+\alpha_2\|\mathrm{D}^{v_1}(l-s)\|^{v_2}]\mathrm{d}x\mathrm{d}y \tag{3.2}$$

其中$\Omega\subset\mathbb{R}^2$表示图像区域，$\mathrm{D}^{v_1}$为$v_1$阶梯度向量，$\|\cdot\|$表示取范数，$\alpha_1,\alpha_2>0$。被积函数中的第一项$\|\mathrm{D}^{v_1}l\|^{v_2}$表示入射光是空间上平滑且富有纹理细节的；第二项$\alpha_1|l-s|^{v_2}$表示入射光与反射是很接近的，即保真性；第三项$\alpha_2\|\mathrm{D}^{v_1}(l-s)\|^{v_2}$表示每个位置的反射率为分段稳定连续的常量，并富有细节。因此，需要在满足$l\geq s$的条件下寻找最小化$E(l)$的$l(x,y)$。也就是说，我们需要寻找这样一个关于$E(l)$的极值曲面l，其能够使得被积函数中三项各自的积分按$(1,\alpha_1,\alpha_2)$加权最小。

根据分数阶欧拉-拉格朗日方程，公式(3.2)中$E(l)$关于l的v_3阶导数[4,11,19]可以表示为

$$\frac{-\Gamma(1-v_1)}{\Gamma(-v_1)\Gamma(-v_3)}\sum_{k=0}^{\infty}\frac{\prod_{\tau=1}^{2k}(v_2-\tau+1)}{(2k)!}\left\{\begin{array}{l}\mathrm{D}_x^1\begin{bmatrix}\|\mathrm{D}^{v_1}l\|^{v_2-2k-2}\mathrm{D}_x^{v_1}l\\+\alpha_1|l-s|^{v_2-2k-2}(l-s)\\+\alpha_2\|\mathrm{D}^{v_1}(l-s)\|^{v_2-2k-2}\mathrm{D}_x^{v_1}(l-s)\end{bmatrix}\\+\mathrm{D}_y^1\begin{bmatrix}\|\mathrm{D}^{v_1}l\|^{v_2-2k-2}\mathrm{D}_y^{v_1}l\\+\alpha_1|l-s|^{v_2-2k-2}(l-s)\\+\alpha_2\|\mathrm{D}^{v_1}(l-s)\|^{v_2-2k-2}\mathrm{D}_y^{v_1}(l-s)\end{bmatrix}\end{array}\right\} \tag{3.3}$$

上式为0时即为$E(l)$关于l的v_3阶稳定点。当对上式取负并将$\frac{\Gamma(1-v_1)}{\Gamma(-v_1)}$化简为$-v_1$后，寻找$E(l)$关于$l$的$v_3$阶稳定点的梯度下降法可以表示为

$$\frac{\partial^{v_3}l}{\partial t^{v_3}}=\frac{-v_1}{\Gamma(-v_3)}\sum_{k=0}^{\infty}\frac{\prod_{\tau=1}^{2k}(v_2-\tau+1)}{(2k)!}\left\{\begin{array}{l}\mathrm{D}_x^1\begin{bmatrix}\|\mathrm{D}^{v_1}l\|^{v_2-2k-2}\mathrm{D}_x^{v_1}l\\+\alpha_1|l-s|^{v_2-2k-2}(l-s)\\+\alpha_2\|\mathrm{D}^{v_1}(l-s)\|^{v_2-2k-2}\mathrm{D}_x^{v_1}(l-s)\end{bmatrix}\\+\mathrm{D}_y^1\begin{bmatrix}\|\mathrm{D}^{v_1}l\|^{v_2-2k-2}\mathrm{D}_y^{v_1}l\\+\alpha_1|l-s|^{v_2-2k-2}(l-s)\\+\alpha_2\|\mathrm{D}^{v_1}(l-s)\|^{v_2-2k-2}\mathrm{D}_y^{v_1}(l-s)\end{bmatrix}\end{array}\right\} \tag{3.4}$$

3.1.2 算法的数值实现方法

在数字图像处理中，像素点之间的关系是离散的。因此，分数阶微分算子D本身可以由 Grünwald-Letnikov 定义（公式(2.1)）直接实现。然而，为了得到更好的精度和收敛速度，我们可以采用 PU-2 分数阶微分掩模的方式实现[20]。对于微分来说，公式(2.1)中的a不妨取0。在离散的数字图像信号$s(\cdot)$的某个方向，如x正方向上，对于

第x个像素$s(x)$来说，其非正方向上的$N+1$个像素为因果像素，记为

$$\begin{cases} s_N = s(0) \\ s_{N-1} = s\left(\frac{x}{N}\right) \\ \vdots \\ s_k = s\left(x - \frac{kx}{N}\right) \\ \vdots \\ s_1 = s\left(x - \frac{x}{N}\right) \\ s_0 = s(x) \end{cases} \tag{3.5}$$

而对于其正方向上的非因果像素，可以延拓性地定义为

$$\begin{cases} s_{-1} = s\left(\frac{x}{N}\right) \\ \vdots \\ s_{-k} = s\left(x + \frac{kx}{N}\right) \\ \vdots \\ s_{-(N-1)} = s\left(x + \frac{(N-1)x}{N}\right) \\ s_{-N} = s(2x) \end{cases} \tag{3.6}$$

此时，根据公式(2.1)中对分数阶微积分的离散定义，当N很大且微分情况下a不妨取0时，有

$$\mathrm{D}_x^v s(x) \approx \frac{x^{-v}N^v}{\Gamma(-v)} \sum_{k=0}^{N-1} \frac{\Gamma(k-v)}{\Gamma(k+1)} s\left(x - \frac{kx}{N}\right) \tag{3.7}$$

为了根据 PU-2 算子的构造方法[20]，人为地向$s\left(x - \frac{kx}{N}\right)$中加入一个与$v$有关的微小偏移量$+\frac{vx}{2N}$，得到

$$\mathrm{D}_x^v s(x) \approx \frac{x^{-v}N^v}{\Gamma(-v)} \sum_{k=0}^{N-1} \frac{\Gamma(k-v)}{\Gamma(k+1)} s\left(x + \frac{vx}{2N} - \frac{kx}{N}\right) \tag{3.8}$$

(3.8)中当$v \neq 0, \pm 2, \pm 4, \cdots$时，其实引入了非像素点$s\left(x + \frac{vx}{2N} - \frac{kx}{N}\right)$上的值。这里可以通过其临近的三个点$s\left(x + \frac{x}{N} - \frac{kx}{N}\right), s\left(x - \frac{kx}{N}\right), s\left(x - \frac{x}{N} - \frac{kx}{N}\right)$作拉格朗日三点插值

$$L_3(x) = \sum_{j=0}^{2} \prod_{i \neq j, i \in 0,1,2} \frac{x - x_i}{x_j - x_i} y_j \tag{3.9}$$

可以得出，对于$x - \frac{kx}{N}$附近的一点ξ，

$$\begin{aligned} s(\xi) \approx & \frac{\left(\xi - x + \frac{kx}{N}\right)\left(\xi - x + \frac{x}{N} + \frac{kx}{N}\right)}{\frac{2x^2}{N^2}} s\left(x + \frac{x}{N} - \frac{kx}{N}\right) \\ & - \frac{\left(\xi - x - \frac{x}{N} + \frac{kx}{N}\right)\left(\xi - x + \frac{x}{N} + \frac{kx}{N}\right)}{\frac{x^2}{N^2}} s\left(x - \frac{kx}{N}\right) \\ & + \frac{\left(\xi - x - \frac{x}{N} + \frac{kx}{N}\right)\left(\xi - x + \frac{kx}{N}\right)}{\frac{2x^2}{N^2}} s\left(x - \frac{x}{N} - \frac{kx}{N}\right) \end{aligned} \tag{3.10}$$

令$\xi = x + \frac{vx}{2N} - \frac{kx}{N}$并代入插值公式(3.9)中，得到

$$\begin{aligned} s\left(x + \frac{vx}{2N} - \frac{kx}{N}\right) &\approx \left(\frac{v}{4} + \frac{v^2}{8}\right) s\left(x + \frac{x}{N} - \frac{kx}{N}\right) + \left(1 - \frac{v^2}{4}\right) s\left(x - \frac{kx}{N}\right) + \left(\frac{v^2}{8} - \frac{v}{4}\right) s\left(x - \frac{x}{N} - \frac{kx}{N}\right) \\ &= \left(\frac{v}{4} + \frac{v^2}{8}\right) s_{k-1} + \left(1 - \frac{v^2}{4}\right) s_k + \left(\frac{v^2}{8} - \frac{v}{4}\right) s_{k+1} \end{aligned} \tag{3.11}$$

把(3.11)代入(3.8)得到

$$\mathrm{D}_x^v s(x) \approx \frac{x^{-v} N^v}{\Gamma(-v)} \sum_{k=0}^{N-1} \frac{\Gamma(k-v)}{\Gamma(k+1)} \left[s_k - \frac{v}{4}(s_{k-1} - s_{k+1}) + \frac{v^2}{8}(s_{k-1} - 2s_k + s_{k+1}) \right] \tag{3.12}$$

把图像上某个方向上相邻两个像素点定义为拉格朗日插值中所谓的相邻点，方向向量定义为$\boldsymbol{e}$。则对于一副数字图像$s(x,y)$，共有八个方向可以求其分数阶微分，分别为$\boldsymbol{e}_{x^-} = (-1,0), \boldsymbol{e}_{x^+} = (1,0), \boldsymbol{e}_{y^-} = (0,-1), \boldsymbol{e}_{y^+} = (0,1), \boldsymbol{e}_{x^+y^-} = (1,-1), \boldsymbol{e}_{x^-y^+} = (-1,1), \boldsymbol{e}_{x^-y^-} = (-1,-1), \boldsymbol{e}_{x^+y^+} = (1,1)$。那么有$s$在$\boldsymbol{m} = (x,y)$处$\boldsymbol{e}$上的方向分数阶导数为

$$\begin{aligned} \frac{\partial^v s(\boldsymbol{m})}{\partial \boldsymbol{e}^v} = &\left(\frac{v}{4} + \frac{v^2}{8}\right) s(\boldsymbol{m} + \boldsymbol{e}) \\ &+ \left(1 - \frac{v^2}{2} - \frac{v^3}{8}\right) s(\boldsymbol{m}) \\ &+ \frac{1}{\Gamma(-v)} \sum_{k=1}^{n-2} \begin{bmatrix} \frac{\Gamma(k-v-1)}{(k+1)!} \cdot \left(\frac{v}{4} + \frac{v^2}{8}\right) \\ + \frac{\Gamma(k-v)}{k!} \cdot \left(1 - \frac{v^2}{4}\right) \\ + \frac{\Gamma(k-v-1)}{(k-1)!} \cdot \left(-\frac{v}{4} - \frac{v^2}{8}\right) \end{bmatrix} s(\boldsymbol{m} - k\boldsymbol{e}) \\ &+ \left[\frac{\Gamma(n-v-1)}{(n-1)!\,\Gamma(-v)} \cdot \left(1 - \frac{v^2}{4}\right) + \frac{\Gamma(n-v-2)}{(n-2)!\,\Gamma(-v)} \cdot \left(-\frac{v}{4} + \frac{v^2}{8}\right)\right] s(\boldsymbol{m} - (n-1)\boldsymbol{e}) \\ &+ \frac{\Gamma(n-v-1)}{(n-1)!\,\Gamma(-v)} \cdot \left(-\frac{v}{4} + \frac{v^2}{8}\right) s(\boldsymbol{m} - n\boldsymbol{e}) \end{aligned} \tag{3.13}$$

其中，x正方向为图像向下的方向，y正方向为图像向右的方向。这里n通常为奇数。因此，从公式(3.13)中提出PU-2 分数阶微分算子掩模系数[20]如下

$$\begin{cases} C_{s_{-1}} = \frac{v}{4} + \frac{v^2}{8} \\ C_{s_0} = 1 - \frac{v^2}{2} - \frac{v^3}{8} \\ \vdots \\ C_{s_k} = \frac{\Gamma(k-v-1)}{(k+1)!} \cdot \left(\frac{v}{4} + \frac{v^2}{8}\right) + \frac{\Gamma(k-v)}{k!} \cdot \left(1 - \frac{v^2}{4}\right) + \frac{\Gamma(k-v-1)}{(k-1)!} \cdot \left(-\frac{v}{4} - \frac{v^2}{8}\right) \\ \vdots \\ C_{s_{n-1}} = \frac{\Gamma(n-v-1)}{(n-1)!\,\Gamma(-v)} \cdot \left(1 - \frac{v^2}{4}\right) + \frac{\Gamma(n-v-2)}{(n-2)!\,\Gamma(-v)} \cdot \left(-\frac{v}{4} + \frac{v^2}{8}\right) \\ C_{s_n} = \frac{\Gamma(n-v-1)}{(n-1)!\,\Gamma(-v)} \cdot \left(-\frac{v}{4} + \frac{v^2}{8}\right) \end{cases} \tag{3.14}$$

由公式(3.14)所示的掩模系数进一步得出八个方向上的掩模矩阵$\boldsymbol{W} \in \mathcal{M}_{(2n+1)\times(2n+1)}(\mathbb{R})$为

$$\boldsymbol{W}_{x^-}=\begin{bmatrix}\cdots & 0 & C_{s_n} & 0 & \cdots\\ \cdots & 0 & C_{s_{n-1}} & 0 & \cdots\\ & \vdots & \vdots & \vdots & \\ \cdots & 0 & C_{s_k} & 0 & \cdots\\ & \vdots & \vdots & \vdots & \\ \cdots & 0 & C_{s_0} & 0 & \cdots\\ \cdots & 0 & C_{s_{-1}} & 0 & \cdots\\ \cdots & 0 & 0 & 0 & \cdots\\ \iddots & \vdots & \vdots & \vdots & \ddots\end{bmatrix}\quad \boldsymbol{W}_{x^+}=\begin{bmatrix}\ddots & \vdots & \vdots & \vdots & \iddots\\ \cdots & 0 & 0 & 0 & \cdots\\ \cdots & 0 & C_{s_{-1}} & 0 & \cdots\\ \cdots & 0 & C_{s_0} & 0 & \cdots\\ & \vdots & \vdots & \vdots & \\ \cdots & 0 & C_{s_k} & 0 & \cdots\\ & \vdots & \vdots & \vdots & \\ \cdots & 0 & C_{s_{n-1}} & 0 & \cdots\\ \cdots & 0 & C_{s_n} & 0 & \cdots\end{bmatrix}$$

$$\boldsymbol{W}_{y^-}=\begin{bmatrix}\vdots & \vdots & & \vdots & & \vdots & \vdots & \iddots\\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots\\ C_{s_n}C_{s_{n-1}} & & \cdots C_{s_k} & & \cdots C_{s_0}C_{s_{-1}} & & 0 & \cdots\\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots\\ \vdots & \vdots & & \vdots & & \vdots & \vdots & \ddots\end{bmatrix}\quad \boldsymbol{W}_{y^+}=\begin{bmatrix}\ddots & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots\\ \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0\\ \cdots & 0 & C_{s_{-1}} & C_{s_0} & \cdots & C_{s_k} & \cdots & C_{s_{n-1}} & C_{s_n}\\ \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0\\ \iddots & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots\end{bmatrix}$$

$$\boldsymbol{W}_{x^+y^-}=\begin{bmatrix}&&&&&&\vdots&\iddots&\iddots\\ &&&&&\cdots&0&0&\iddots\\ &&&&\cdots&0&C_{s_{-1}}&0&\cdots\\ &&&\cdots&0&C_{s_0}&0&\vdots&\\ &&&\iddots&\iddots&0&\vdots&&\\ &&\vdots&\iddots&C_{s_k}&\iddots&\vdots&&\\ &\vdots&0&\iddots&\iddots&&&&\\ \cdots&0&C_{s_{n-1}}&0&\cdots&&&&\\ C_{s_n}&0&\cdots&&&&&&\end{bmatrix}\quad \boldsymbol{W}_{x^-y^+}=\begin{bmatrix}&&&&&&\cdots&0&C_{s_n}\\ &&&&&\cdots&0&C_{s_{n-1}}&0\\ &&&&\iddots&\iddots&0&\vdots&\\ &&&\iddots&C_{s_k}&\iddots&\vdots&&\\ &&\vdots&0&\iddots&\iddots&&&\\ &\vdots&0&C_{s_0}&0&\cdots&&&\\ \cdots&0&C_{s_{-1}}&0&\cdots&&&&\\ \iddots&0&0&\cdots&&&&&\\ \iddots&\iddots&\vdots&&&&&&\end{bmatrix}$$

$$\boldsymbol{W}_{x^-y^-}=\begin{bmatrix}C_{s_n}&0&\cdots&&&&&&\\ 0&C_{s_{n-1}}&0&\cdots&&&&&\\ \vdots&0&\ddots&\ddots&&&&&\\ &\vdots&\ddots&C_{s_k}&\ddots&\vdots&&&\\ &&\ddots&\ddots&0&\vdots&&&\\ &&\cdots&0&C_{s_0}&0&\vdots&&\\ &&&\cdots&0&C_{s_{-1}}&0&\cdots&\\ &&&&\cdots&0&0&\ddots&\\ &&&&&\vdots&\ddots&\ddots&\end{bmatrix}\quad \boldsymbol{W}_{x^+y^+}=\begin{bmatrix}\ddots&\ddots&\vdots&&&&&&\\ \ddots&0&0&\cdots&&&&&\\ \cdots&0&C_{s_{-1}}&0&\cdots&&&&\\ &\vdots&0&C_{s_0}&0&\cdots&&&\\ &&\vdots&0&\ddots&\ddots&&&\\ &&&\vdots&\ddots&C_{s_k}&\ddots&\vdots&\\ &&&&\ddots&\ddots&0&\vdots&\\ &&&&\cdots&0&C_{s_{n-1}}&0&\\ &&&&&\cdots&0&C_{s_n}&\end{bmatrix}$$

因此，对于点(x,y)，记$\boldsymbol{S_m}\in\mathcal{M}_{(2n+1)\times(2n+1)}(\mathbb{R})$

$$\boldsymbol{S_m}\coloneqq\begin{bmatrix}s(x-n,y-n)&\cdots&s(x-n,y-1)&s(x-n,y)&s(x-n,y+1)&\cdots&s(x-n,y+n)\\ \vdots&\ddots&\vdots&\vdots&\vdots&\iddots&\vdots\\ s(x-1,y-n)&\cdots&s(x-1,y-1)&s(x-1,y)&s(x-1,y+1)&\cdots&s(x-1,y+n)\\ s(x,y-n)&\cdots&s(x,y-1)&s(x,y)&s(x,y+1)&\cdots&s(x,y+n)\\ s(x+1,y-n)&\cdots&s(x+1,y-1)&s(x+1,y)&s(x+1,y+1)&\cdots&s(x+1,y+n)\\ \vdots&\iddots&\vdots&\vdots&\vdots&\ddots&\vdots\\ s(x+n,y-n)&\cdots&s(x+n,y-1)&s(x+n,y)&s(x+n,y+1)&\cdots&s(x+n,y+n)\end{bmatrix}$$

则其$\boldsymbol{e}$方向上的分数阶微积分为

$$\mathrm{D}_{\boldsymbol{e}}^{v}s(x,y)=\frac{\partial^v s(\boldsymbol{m})}{\partial \boldsymbol{e}^v}=\sum_{i=1}^{2n+1}\sum_{j=1}^{2n+1}\boldsymbol{S_m}(i,j)\cdot\boldsymbol{W_e}(i,j)\tag{3.15}$$

公式(3.15)解决了公式(3.4)中数值计算离散点分数阶微分的问题。而想要数值计算公式(3.4)中的等号左边所描述的这种梯度下降法，则需要采用分数阶梯度下降法（最速下降法）[21]。其利用了时间步进法，得到了第$n+1$次迭代和第n次迭代之间的递推关系。记$l_{x,y}^n=l(x,y,n\Delta t),s_{x,y}^0=s(x,y,0\Delta t)=s(x,y)$。不妨设迭代初始值

$l_{x,y}^{-1}=l_{x,y}^{0}=1.05s_{x,y}^{0}$，则迭代公式为

$$l_{x,y}^{n+1}=P\left(l_{x,y}^{n}\right)\Delta t^{v_3}+l_{x,y}^{n}-\frac{2\mu}{\Gamma(3-v_3)}\left(l_{x,y}^{n-1}-l_{x,y}^{n}\right)^{2}\left(l_{x,y}^{n}\right)^{-v_3} \tag{3.16}$$

其中μ为收敛速率。在离散图像上，公式(3.4)所描述的分数阶微分可以拓展到八个方向。所以在简化计算的情况下，如果只考虑公式(3.4)中$k=0,1$的情况，则$P\left(l_{x,y}^{n}\right)$为

$$P\left(l_{x,y}^{n}\right)=\frac{-v_1}{\Gamma(-v_3)}\sum_{k=0}^{1}\frac{\prod_{\tau=1}^{2k}(v_2-\tau+1)}{(2k)!}\left\{\sum_{\boldsymbol{e}}\mathrm{D}_{\boldsymbol{e}}^{1}\begin{bmatrix}\left\|\mathrm{D}^{v_1}l_{x,y}^{n}\right\|^{v_2-2k-2}\mathrm{D}_{\boldsymbol{e}}^{v_1}l_{x,y}^{n}\\+\alpha_1\left|l_{x,y}^{n}-s_{x,y}^{0}\right|^{v_2-2k-2}\left(l_{x,y}^{n}-s_{x,y}^{0}\right)\\+\alpha_2\left\|\mathrm{D}^{v_1}\left(l_{x,y}^{n}-s_{x,y}^{0}\right)\right\|^{v_2-2k-2}\mathrm{D}_{\boldsymbol{e}}^{v_1}\left(l_{x,y}^{n}-s_{x,y}^{0}\right)\end{bmatrix}\right\} \tag{3.17}$$

在后续的实验中，不妨取$\mu=0.1,\alpha_1=0.05,\alpha_2=0.1,\Delta t=0.002$，并取$N=6$为总迭代次数。为满足条件$l\geq s$，可以在每次迭代后，取$l_{x,y}^{n+1}=\max\left\{l_{x,y}^{n+1},s_{x,y}^{0}\right\}$。考虑到迭代过程中可能会出现$\left\|\mathrm{D}^{v_1}l_{x,y}^{n}\right\|=0$或$\left\|\mathrm{D}^{v_1}\left(l_{x,y}^{n}-s_{x,y}^{0}\right)\right\|=0$的情况，当$\left\|\mathrm{D}^{v_1}l_{x,y}^{n}\right\|<\varepsilon_1$时，取$\left\|\mathrm{D}^{v_1}l_{x,y}^{n}\right\|=\varepsilon_1$；当$\left\|\mathrm{D}^{v_1}\left(l_{x,y}^{n}-s_{x,y}^{0}\right)\right\|<\varepsilon_1$，取$\left\|\mathrm{D}^{v_1}\left(l_{x,y}^{n}-s_{x,y}^{0}\right)\right\|=\varepsilon_1$。迭代过程中还可能会出现$l_{x,y}^{n}=0$的情况，当$l_{x,y}^{n}<\varepsilon_2$时，取$l_{x,y}^{n}=\varepsilon_2$。后续实验中不妨取$\varepsilon_1=0.006,\varepsilon_2=10^{-5}$。此外，$\|\mathrm{D}^{v_1}\cdot\|$在八个方向上的一范数为$\sum_{\boldsymbol{e}}\left|\mathrm{D}_{\boldsymbol{e}}^{v_1}\cdot\right|$，二范数为$\sqrt{\sum_{\boldsymbol{e}}\left(\mathrm{D}_{\boldsymbol{e}}^{v_1}\cdot\right)^{2}}$，无穷范数为$\max_{\boldsymbol{e}}\left|\mathrm{D}_{\boldsymbol{e}}^{v_1}\cdot\right|$。

3.1.3 算法步骤

第一步：将需要处理的图像转换到HSV空间。通常，HSV色彩空间中的V值取值范围为[0,1]，因此，将V放缩到[0,255]的范围后，记其为$S(x,y)$。此外，为了避免取对数后值为$-\infty$，令$s(x,y)=\ln[S(x,y)+1]$。

第二步：设置算法中所有的参数值。除上述提到的参数设定外，取$v_1=1.25,v_2=2.25,v_3=0.9$。按上述算法估算出$l(x,y)$，并恢复成估算的入射光$L(x,y)=\mathrm{e}^{l(x,y)}$。

第三步：根据估算的入射光$L(x,y)$和反射光$S(x,y)$计算出反射率，$R(x,y)=\frac{S(x,y)}{L(x,y)}$。由于这种方法通常会得到一个过度增强的图像[10]，所以这里对入射光按如下方法进行Gamma校正。在光强范围为[0,255]的情况下，最大光强$W=255$。则使用参数γ校正后的入射光L'为

$$L'(x,y)=W\cdot\left(\frac{L(x,y)}{W}\right)^{\frac{1}{\gamma}} \tag{3.18}$$

通常，$\gamma=2.2$。校正后，便可以根据估算出的反射率$R(x,y)$和校正后的入射光$L'(x,y)$计算出增强的反射光，

$$S'(x,y)=L'(x,y)\cdot R(x,y) \tag{3.19}$$

第四步：经过Gamma校正后的图像最好在sRGB标准下显示[22,23]。因此，以$\frac{S'(x,y)}{W}$作为HSV色彩空间中的V通道值，并将其转换成sRGB色彩空间，得到最终的增强结果图。

3.2 算法分析

3.2.1 边缘处理

在实际处理一副图像时，不可避免的要对图像的边缘点求分数阶微分，此时便需要用到图像外未知像素点的值。通常情况下，比较简单的方法是对边缘进行镜像翻折。还有一些比较简单图像处理算法，选择直接忽略图像边缘上的点，不对他们进行处理。然而，对于本文所述的图像增强算法，在估算$l(x,y)$时，采用了时间步进法。

也就是说，这种方法是在不断地迭代求解$l(x,y)$。此外，该算法充分利用了临近像素点之间的相关性。因此，如果对图像边缘延拓后得到的像素点与实际物象存在较大误差，这种误差会随迭代计算$l(x,y)$的过程不断积累，并不断向图像内部像素点传播。这也是大部分图像处理中普遍存在的一个根本性问题。

为了在迭代计算$l(x,y)$的过程中不积累传播上述误差，根据公式(3.17)，需要在每次迭代的过程中，实时对当前的$l(x,y), l(x,y)-s(x,y)=r(x,y)$进行边缘延拓。假设在图像边缘附近，入射光$l(x,y)$和反射率$r(x,y)$都为空间上连续平滑的量，尝试采用拉格朗日三点插值法对其边缘向外延拓。根据拉格朗日三点插值公式(3.9)，对于边缘内侧已知的三个连续点$s(0), s(1), s(2)$，可以写出如下插值公式

$$\hat{s}(x)=\frac{(x-1)(x-2)}{(0-1)(0-2)}s(0)+\frac{x(x-2)}{(1-0)(1-2)}s(1)+\frac{x(x-1)}{(2-0)(2-1)}s(2) \tag{3.20}$$

将$x=-1$代入上式，得到图像边缘外侧第一个未知点的估计值

$$\hat{s}(-1)=3s(0)-3s(1)+s(2) \tag{3.21}$$

由此发现，图像边缘可以通过拉格朗日三点插值进行延拓。为加快计算速度将上式用分配律重写为$\hat{s}(-1)=3[s(0)-s(1)]+s(2)$，从而每个点减少了一次乘法计算。并且，在边缘延拓次数较少的情况下，可以通过递推的方式进行计算。具体方法为，对于图像某个边外一点，用与该边垂直的边内三个点估计；而对于图像某个角外一点，则用该角内对角线方向上的三个点估计。当以上述方式得到延拓一圈的图像后，以延拓后的图像作为已知图像，再以上述延拓方式进行下一轮延拓。对于迭代总次数N，图像尺寸n来说，这种延拓方式大约需要进行$2\times N\times 4n$次加法运算，$N\times 4n$次乘法运算，关于n的复杂度为$\mathcal{O}(n)$，与镜像延拓的时间复杂度相同。

3.2.2 复杂度分析及算法优化方法

相比于整数阶全变分 Retinex 算法，分数阶全变分 Retinex 算法的计算量大很多。然而，仔细观察公式(3.17)可以发现，该算法虽然描述上很复杂，但是在实现过程中有很大的优化空间，并且在某些步骤的计算过程中，图像的像素点之间不存在计算先后顺序的依赖关系，因此可以进行并行计算。接下来着重讨论公式(3.17)中的优化方法。

首先看到，公式(3.17)中$\forall k$，$l_{x,y}^n-s_{x,y}^0$为一常量。因此，对于每次计算$P\left(l_{x,y}^n\right)$时，应先把两矩阵的差计算出来，作为一常量矩阵$r_{x,y}^n=l_{x,y}^n-s_{x,y}^0$。其次，$\forall k, \mathrm{D}_{\boldsymbol{e}}^{v_1} l_{x,y}^n, \mathrm{D}_{\boldsymbol{e}}^{v_1}\left(l_{x,y}^n-s_{x,y}^0\right)$为常量；$\forall k, \boldsymbol{e}, \left\|\mathrm{D}^{v_1} l_{x,y}^n\right\|, \left\|\mathrm{D}^{v_1}\left(l_{x,y}^n-s_{x,y}^0\right)\right\|$也为常量。最后，$\forall \boldsymbol{e}$，$\alpha_1\left|l_{x,y}^n-s_{x,y}^0\right|^{v_2-2k-2}\left(l_{x,y}^n-s_{x,y}^0\right), \left\|\mathrm{D}^{v_1} l_{x,y}^n\right\|^{v_2-2k-2}, \left\|\mathrm{D}^{v_1}\left(l_{x,y}^n-s_{x,y}^0\right)\right\|^{v_2-2k-2}$也都为常量。所以，在对不同$k, \boldsymbol{e}$的累加求和之前，应该首先计算出这些常量，再代入公式中的对应位置进行求解。

再考虑分数阶微分算子D^{v_1}。以$l(x,y)\in\mathcal{M}_{N_1\times N_2}$矩阵$\boldsymbol{e}=\boldsymbol{e}_{x^-}=(-1,0)$为例，并记向外延拓了$n$次的$l(x,y)$矩阵为$l_+(x,y)\in\mathcal{M}_{(N_1+2n)\times(N_2+2n)}$，记$l_+(x,y)$从第$i_1$行到第$i_2$行，第$j_1$列到第$j_2$列所构成的子矩阵为$l_+(i_1{:}\,i_2, j_1{:}\,j_2)$。观察到，对$l(x,y)$中每个像素点计算其分数阶微分时，不存在先后顺序的依赖关系。因此$l(x,y)$上所有点在某个方向上（如$\boldsymbol{e}_{x^-}$）的分数阶微积分矩阵$\mathrm{D}_{\boldsymbol{e}_{x^-}}^{v_1} l(x,y)$可以并行计算如下

$$\mathrm{D}_{\boldsymbol{e}_{x^-}}^{v_1} l(x,y)=\sum_{k=-1}^{n} C_k\cdot l_+\left((n+1-k){:}\,(n+N_1-k), (n+1){:}\,(N_2+n)\right) \tag{3.22}$$

其中C_k为分数阶掩模系数。其他方向、其他矩阵同理。

再观察公式(3.16)，其本身是在进行迭代计算，即在已知$l_{x,y}^{n-1}, l_{x,y}^n$的情况下计算$l_{x,y}^{n+1}$。如果采用“增量”的

思想，公式(3.16)也有改进优化空间。记$(\Delta l)_{x,y}^{n}=l_{x,y}^{n}-l_{x,y}^{n-1}$，将公式(3.16)中等号右边的$l_{x,y}^{n}$移到等号左边，并用$(\Delta l)_{x,y}^{n+1}$代替$l_{x,y}^{n+1}-l_{x,y}^{n}$，则有

$$(\Delta l)_{x,y}^{n+1}=P\left(l_{x,y}^{n}\right)\Delta t^{v_3}-\frac{2\mu}{\Gamma(3-v_3)}\left[(\Delta l)_{x,y}^{n}\right]^{2}\left(l_{x,y}^{n}\right)^{-v_3} \tag{3.23}$$

所以，对于每一次迭代来说，在已知$l_{x,y}^{n}$和$(\Delta l)_{x,y}^{n}$的条件下。可以采用由上式先计算出$(\Delta l)_{x,y}^{n+1}$，再将更新后的$(\Delta l)_{x,y}$代入下式计算$l_{x,y}^{n+1}$

$$l_{x,y}^{n+1}=l_{x,y}^{n}+(\Delta l)_{x,y}^{n+1} \tag{3.24}$$

这样做不仅简化了计算，还节约了存储空间。

4 实验与分析

本部分将在4.1小节中对比改进算法与原算法，证明分数阶全变分Retinex算法本身有很大的计算提速空间。之后将会在4.2小节中讨论算法实现中不同的取范数方式和一次差分做法的优劣。而在本章最后一部分中，我们将综合比较改进前算法与经过本章分析后最终确定的算法的效果。

4.1 改进算法与原算法的速度比较

为了证明第三部分所述的改进后的分数阶全变分Retinex算法与原算法相比，在算法描述和算法实现两个方面都有很大的优化空间，接下来将展示改进后算法在计算速度上有显著的提升。

为控制变量，在所有参数和计算方法的选择上，均采用原文中使用的参数和计算方式[19]，即$v_1 = 1.25, v_2 = 2.25, v_3 = 0.9, \Delta t = 0.002, \mu = 0.1, \alpha_1 = 0.05, \alpha_2 = 0.1, \varepsilon_1 = 0.006, \varepsilon_2 = 10^{-5}, \gamma = 2.2$，并取$7 \times 7$的掩模矩阵，迭代次数$N = 6$；在取范数时，采用二范数计算方法，每次迭代过程忽略l, s之间的大小关系。注意，在上述操作相同的情况下，改进前后的算法得到的结果是完全相同的。这里分别选取了大小为$100,300,700,1000,3000$的正方形图像，计算比较其核心算法部分所花费的时间。对于每种尺寸，重复计算20次后取其计算时间的平均值作为大致的计算时间。在一台CPU为Intel Core i5-6300U @ 2.4 GHz主频、8 GB内存、64位操作系统上，使用MATLAB R2016b在CPU占用率恒定在30%左右的情况下进行实验，结果见表4.1：

表4.1 改进前后不同尺寸算法的比较，单位为秒(s)

算法\尺寸	100	300	700	1000	3000
改进前	0.3514	3.1106	20.9496	41.6102	459.1827
改进后	0.1111	0.6882	7.6549	15.6970	141.6127

正如上表所见，相比于改进前，对于尺寸为$100,300,700,1000,3000$的正方形图像，改进后的计算时间分别为改进前时间的$31.62\%, 22.12\%, 36.54\%, 37.72\%, 30.84\%$。总体来看，改进后算法大约只需花费改进前时间的$1/3$，计算速度有了显著的提升。

4.2 参数的选取

由于在上一节的比较中，改进后的算法能够在和改进前算法得出相同计算结果的条件下，显著缩短计算时间，因此在之后的实验中，均使用改进后的算法。此外，对图像边缘的处理采取边缘插值的方案。

4.2.1 范数的取法

正如公式(3.17)所示，在迭代计算$P(l)$的过程中，需要对八个方向的分数阶微分取范数。工程应用中的大多数情况下，范数有三种取法，分别为一范数、二范数和无穷范数。而对于图像处理中八个方向上的范数来说，一

范数为$\|\mathrm{D}^{v_1}\cdot\|_1=\sum_e\left|\mathrm{D}_e^{v_1}\cdot\right|$，二范数为$\|\mathrm{D}^{v_1}\cdot\|_2=\sqrt{\sum_e\left(\mathrm{D}_e^{v_1}\cdot\right)^2}$，无穷范数为$\|\mathrm{D}^{v_1}\cdot\|_\infty=\max_e\left|\mathrm{D}_e^{v_1}\cdot\right|$。不同的取法得到的图像处理结果也各不相同。接下来将比较这三种分数阶范数的效果。

除分数阶范数取法不同外，其他参数和计算方式均与 4.1 节中所述的参数和计算方式相同。图 4.1 展示了其中某一次实验的效果比较。可以看到，图 4.1 中的(b)(c)(d)相较于(a)，都有比较好的图像增强效果，且(b)(c)(d)效果整体上看没有明显区别。但放大图片右侧 1/16 的部分后可以发现，(f)中红框所示区域内纹理细节体现程度较弱；而(h)中红框所示区域内存在一块块朝向一致并列排列的条纹（(h)红框内左侧部分），这造成了图片内容的失真。此外(h)中还出现了对比度增强过度以致产生类噪音的现象。其原因在于，无穷范数只考虑了该像素最大分数阶微分方向上的变化量，而舍弃了另外七个方向的信息，因此对变化非常敏感。在一段连续的区域里，最大分数阶微分的方向一致，从而出现了这种失真性条纹。而(f)与(g)相比，在纹理细节本该丰富的区域上丢失了一些原图中存在的纹理。(f)与(g)相比模糊的原因在于，(f)等价的考虑了八个方向上的分数阶微分，而(g)为八个方向上的欧式距离，物理意义更加明确。

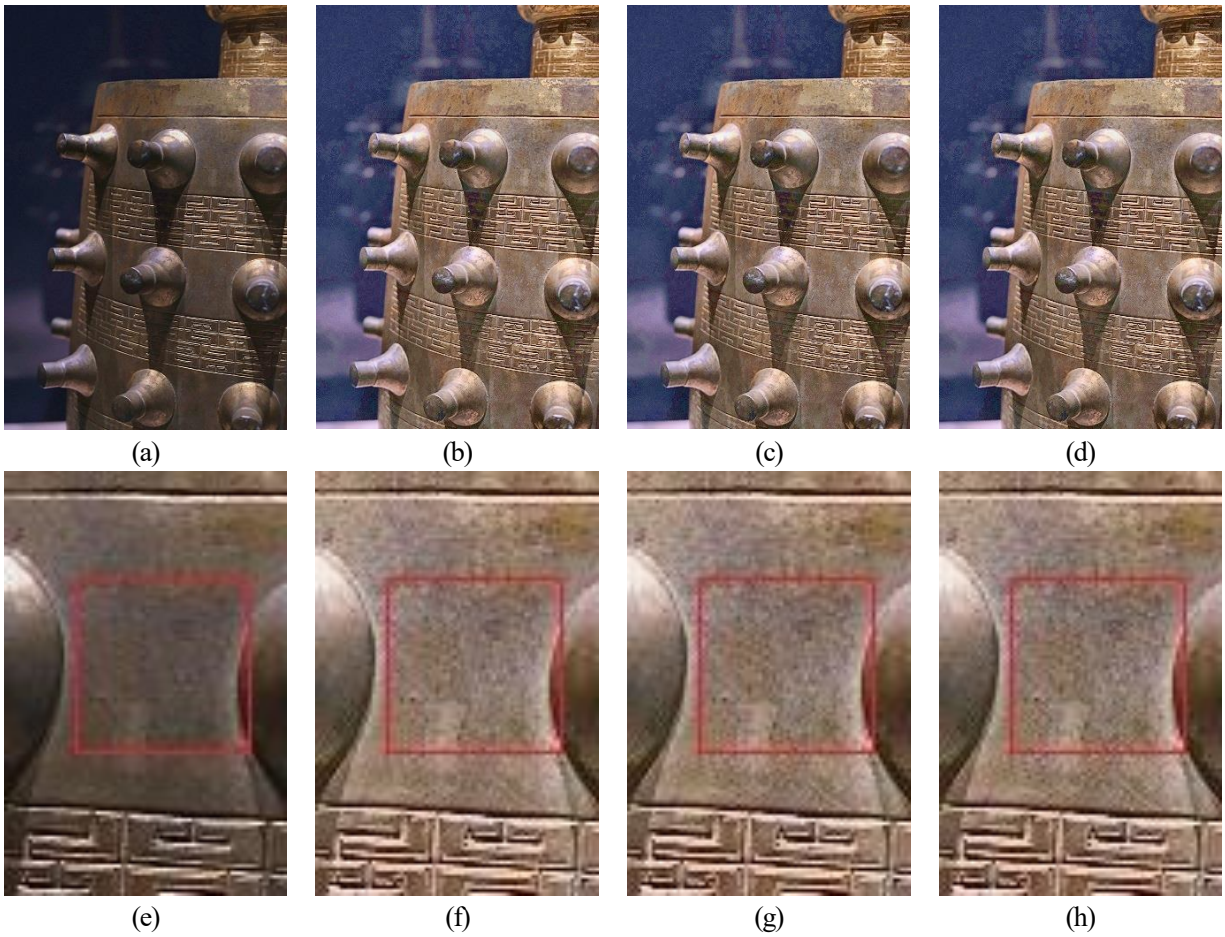


图 4.1 不同范数取法处理对比。(a)：原图，(b)：一范数处理图，(c)：二范数处理图，(d)：无穷范数处理图，(e)：(a)的右侧 1/16 的部分，(f)：(b)的右侧 1/16 的部分，(g)：(c)的右侧 1/16 的部分，(h)：(d)的右侧 1/16 的部分。

也就是说，对于一范数、二范数和无穷范数三者来说，二范数是一个比较合适的选择。当然，从数学的角度

考虑，二次以上且次数不大的范数也是可取的，效果应与二范数极其接近。但在计算量上考虑，取二次以上的范数会增加计算量。这种做法是没有必要的。如果为了节约计算时间，并且对非常细节的部分要求不高的情况下，可以选择无穷范数作为替代。在接下来的实验中，我们仍然选择使用二范数。

4.2.2 一阶微分的方式

原算法中，为了避免在v_1取非正整数时，PU-2 算子在程序实现的过程中出现∞/∞ 的现象，其一阶微分采用了与 PU-2 算子本身定义方式不同的一阶微分做法。原算法的实现中，采用了 Sobel 算子：对于点$\boldsymbol{m}=(x,y)$，其$\boldsymbol{e}$方向的一阶微分为

$$\begin{aligned}\mathrm{D}_{\boldsymbol{e}}^{1}s(\boldsymbol{m}) = &\frac{1}{2}\big(s(\boldsymbol{m}-\boldsymbol{e})-s(\boldsymbol{m}+\boldsymbol{e})\big)\\ &+\frac{1}{4}\big(s(\boldsymbol{m}+\boldsymbol{n}_1-\boldsymbol{e})-s(\boldsymbol{m}+\boldsymbol{n}_1+\boldsymbol{e})\big)\\ &+\frac{1}{4}\big(s(\boldsymbol{m}+\boldsymbol{n}_2-\boldsymbol{e})-s(\boldsymbol{m}+\boldsymbol{n}_2+\boldsymbol{e})\big)\end{aligned} \tag{4.1}$$

其中，$\boldsymbol{n}_1,\boldsymbol{n}_2$为与$\boldsymbol{e}$方向垂直并指向其临近一像素点的法向量，且$\boldsymbol{n}_1=-\boldsymbol{n}_2$。这种做法考虑到了被微分点关于微分方向两侧的变化情况，但对微分的因果方向和被微分像素点本身考虑不足。另一种可以考虑的方法是，按照原始的 PU-2 算子定义，并利用公式(2.2)得到一阶微分的 PU-2 算子掩模系数$C_{-1}=0.375, C_0=0.375, C_1=-0.875, C_2=0.125$，从而便可按照(3.15)的方法计算一阶微分。

下面比较两种一阶微分计算方法的效果。与上一小节一样，除一阶微分做法不同外，其他参数和计算方式均与 4.1 节中所述的参数和计算方式相同。在范数的选取问题上采用二范数。图 4.2 展示了经选择大量图片实验后其中一组结果。经过放大仔细对比后发现，这两种做法产生的结果没有实质性区别。无论在视觉上还是在图像后续处理的角度上考量，都是几乎一样的。而如果从算法实现的角度考虑，在两种做法一样的情况下，应该选择计算量小、形式统一的做法。因此，采用 PU-2 微分算子的原始定义，无论从数学角度还是从算法实现角度看，都是一种更优的选择。

(a)　　(b)　　(c)

图 4.2 采用不同的一阶微分方法效果对比。(a)：原图，(b)：采用公式(4.1)所示基于 Sobel 算子的一阶微分处理方法所得结果图，(c)：采用原始的 PU-2 算子一阶微分定义处理方法所得结果图。

4.3 综合比较

鉴于改进后算法纠正了原算法中存在的一些问题，并在速度、计算方法等方面对原算法进行了改进，这里

我们简要的对改进前后算法进行一次综合比较。

图 4.3 的原始图像选自 Pu 等人 2019 年的文章中[19]第一组实验图片——玉兔号月球车。从处理结果图像来看，改进后算法的纹理细节保留效果、对比度增强效果更好。而对于计算速度来说，在同等测试条件下，原算法核心部分运行时间为 8.5340 秒，而改进后算法核心部分运行时间为 3.2326 秒。这个结果与 4.2.1 节展示的结果相一致，即改进后算法的时间约为改进前的 1/3。值得注意的是，根据 Pu 等人 2017 年的文章[11]与 2019 年的文章[19]提到的关于整数阶全变分 Retinex 算法速度与分数阶全变分 Retinex 算法速度的比较，整数阶全变分算法所花时间约为分数阶全变分算法所花时间的 1/3 到 2/3。而如果以本文所描述的改进后算法来衡量，分数阶全变分 Retinex 算法至少可以达到与整数阶全变分 Retinex 算法相同数量级的计算速度，这对于实际应用中高速即时处理图像需求具有重大意义。

(a)　　(b)　　(c)

图 4.3 改进前后算法对玉兔号月球车图片增强处理效果对比。(a)：原图，(b)：改进前算法处理结果图，(c)：改进后算法处理结果图。

5 分数阶图像增强在计算神经科学中的应用

5.1 分数阶微积分对比度增强的神经学解释

分数阶图像增强的效果相比于整数阶来说，有其天然优势。这种优势主要源于分数阶微积分的记忆性与非局部性这两种性质。接下来，我们将从神经生物学的角度解释，应用于图像增强的分数阶微分掩模与人视觉系统之间的联系。

首先，图5.1展示了人视觉系统中的神经细胞在侧向抑制作用下产生的感知效果的示意图。可以看到，虽然图5.1的下半部分是按照虚线所示的函数产生的光强，但读者却感觉到在渐变与非渐变的交界处产生了程度更深的亮斑（约位于$x = 0.6$）和暗斑（约位于$x = 0.4$），这与马赫带效应所反映的视觉认知现象相同。这种感知效果产生的原因是，在物理刺激变化较大的地方，神经细胞活跃的地方会更为明显的抑制周围不活跃的神经细胞。这种效应致使在明暗变化的位置，神经细胞的活跃程度差距更大，从而达到对比度增强的效果[12,24]。也就是说，在物象的交界处，我们感知到的强度变化比实际的物理变化差异更大。视觉系统利用这种方法帮助我们辨析各种物象的边缘、交界。而分数阶微积分体现的效果刚好与侧向抑制表现出的效果相吻合。

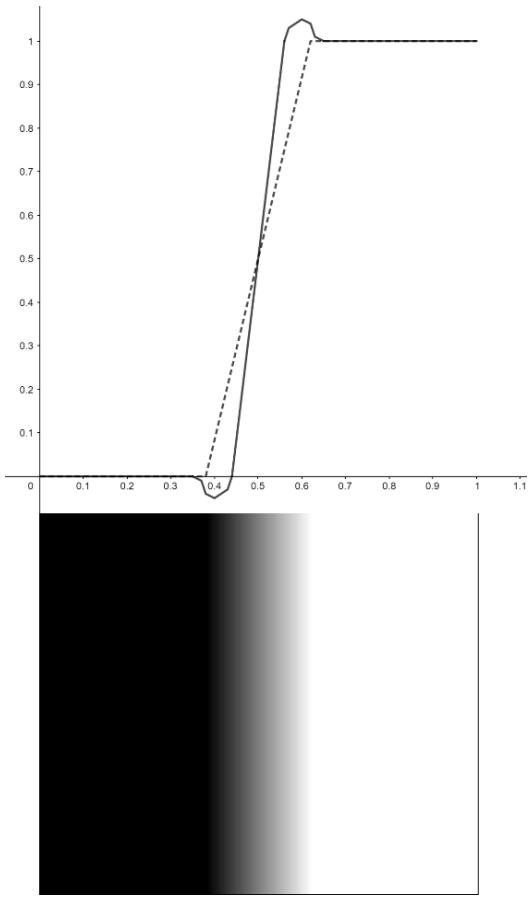


图5.1 侧向抑制作用下，物理光强刺激与神经细胞活动的对比示意图。下半部分为实际的物理光强刺激。上半部分中，虚线表示不同位置实际物理刺激的强度，而实线表示不同位置对应的神经细胞的活跃程度（注意：这种表示只是一种示意，属于定性表示而非定量表示）。

为证明分数阶微分的作用与侧向抑制相吻合，这里对图 5.1 中虚线所示的函数求半导数（图 5.2）。结果表明，图 5.2 中橙色曲线所表示的对物理光强的分数阶微分结果，与图 5.1 中实线所表达的神经细胞的活动高度一致。即，在物理刺激变化较大的地方，活动更强的神经细胞会侧向抑制活动程度本来就很弱的神经细胞，使交界处神经细胞的活动出现“强者更强，弱者更弱”的现象。事实上，对于分数阶微分阶数$0 < v < 1$，都会有这种侧向抑制的效果[20]。这也就从神经活动角度完备了$0 < v < 1$的分数阶掩模能够起到图像增强的效果。而从数学角度解释，$v = 0$表示函数本身，$v = 1$表示函数的变化，如果v介于0到1之间，则表示既在一定程度上包含了函数本身，同时也包含了被微分点周围的变化情况。并且，被微分点对于周围的变化是有记忆的。如果周围存在变化，这种变化会按一定的比例积累起来。这种积累与视神经网络中两级细胞、神经节细胞起到的作用相类似。它们之间互相整合、相互抑制，从而表达中视神经网络最终的活动状态，也就是大脑最终感知到的视觉影像。

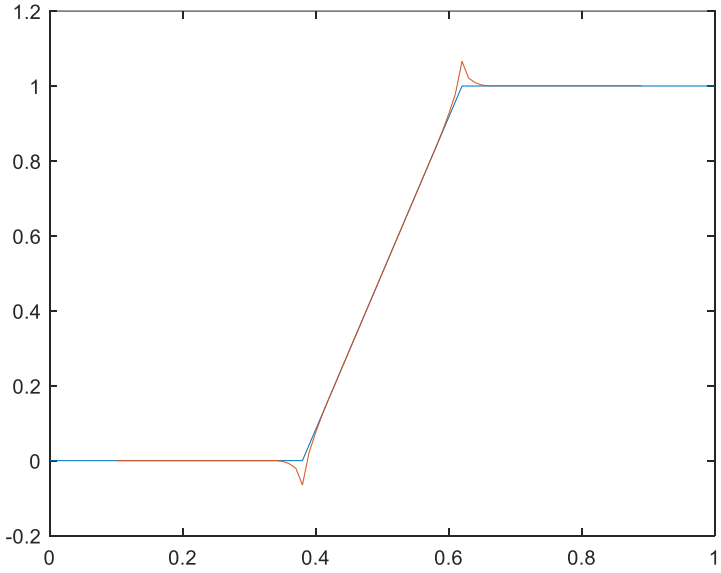


图 5.2 分数阶微分掩模作用在物理光强上的效果。其中蓝色曲线为物理光强函数，其定义与图 5.1 中虚线表示的函数定义相同；橙色曲线为对其向x负方向和x正方向进行 0.5 阶微分后等权重相加并放缩匹配到物理光强函数后的结果。

综上所述，分数阶微分本身具备的良好性质刚好能够表达视神经细胞之间侧向抑制的效果。因此，分数阶微分能够像视神经细胞一样，更好的起到增强对比度的作用。

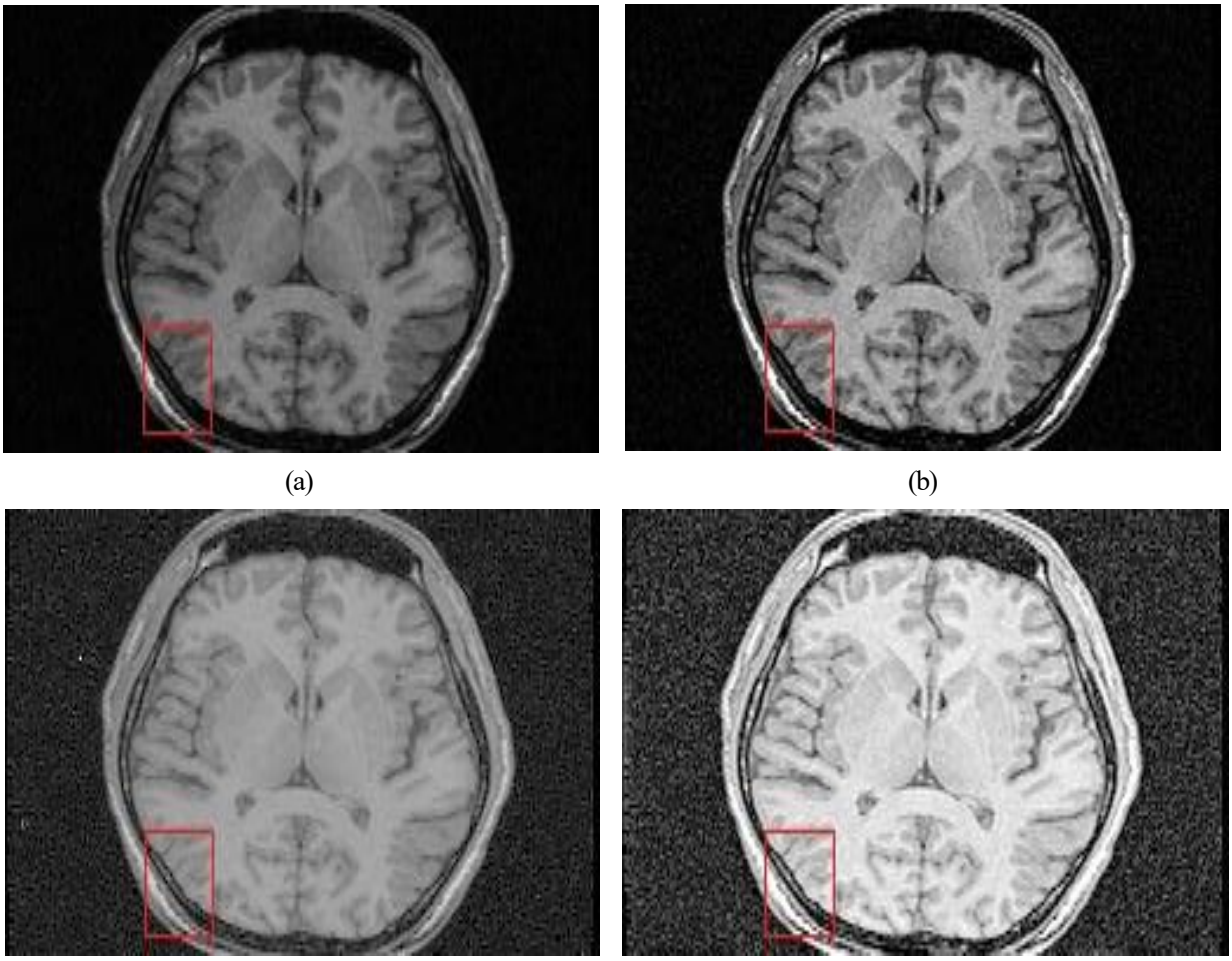
5.2 脑成像图的增强

本部分将利用分数阶全变分 Retinex 算法和 PU-2 分数阶掩模，对脑磁共振影像图进行增强处理，以尝试达到更好的观察效果。以下所使用脑影像图片均来自公共数据库。

首先我们用分数阶全变分 Retinex 算法和 PU-2 分数阶掩模分别处理一张脑影像图。其中，分数阶全变分 Retinex 算法中所有的参数和计算方式均采用前面所述最优的方法。PU-2 分数阶掩模微分阶数取$v = 0.2$，最终

结果取八个方向微分的等权和。处理结果见图 5.3 中(b)(c)。总的来说，(b)(c)所示的两种处理方法各有优劣。对于 PU-2 算子生成的分数阶掩模滤波增强法来说，整体的增强效果良好，噪音增强程度不大，并且能够使图像中的层次感更鲜明，如图 5.3(b)的中央区域所呈现的大脑沟、回。但是对于一些图像中隐藏的信息或细节部分（如图 5.3(b)中红框所示区域中黑暗背景上的白色条纹），挖掘的不够充分，对比度增强不明显。而对于分数阶全变分 Retinex 算法来说，对于图像中的高频区域，能够很好的发掘出细节纹理并增强（如图 5.3(c)中红框所示区域中黑暗背景上的白色条纹），但是对于医学影像这种信噪比很低且物象中存在很多空区域的图像来说，大量噪音对分数阶全变分 Retinex 算法的干扰很强。全局散布的大量噪音使得低频区域的图像增强后，层次感不再突出，大面积区域中的细节和纹理混淆在了一起。因此两者都不算是很理想的处理办法。

为此，这里尝试将两种方法结合起来。首先利用分数阶全变分法挖掘图像中高频区域的细节特征，之后利用 PU-2 算子的分数阶掩模滤波增强方法，增强图中的层次信息，如图 5.3(d)所示。可以很明显的看出，图 5.3(d)既发掘并增强了图像中的高频区域细节（如图 5.3(d)中红框所示区域中黑暗背景上的白色条纹），同时也保留并增强了低频区域的细节和层次感信息（如图 5.3(d)的中央区域所呈现的沟、回）。当然，这种做法也导致处在本该是黑暗的外周区域中的噪音被增强。然而，对噪音的增强没有影响到对图像细节和层次感区域的保留。



(a)　(b)

(c)　(d)

图 5.3 经过增强处理的脑磁共振影像图。(a)：原图，(b)：0.2阶微分掩模滤波增强图，(c)：分数阶全变分 Retinex 算法增强图，(d)：先经过(c)处理后再经过(b)处理的混合处理图。

当然，对于脑影像图的增强来说，对于研究者而言是一种辅助工具而非必要工具。也就是说，增强的图片虽然存在一定的缺点，但可以作为原始图像的额外辅助图像，帮助研究者更好的观察影像图。比如，可以将图5.3 的(a)和(d)一起呈现出来。从而，研究者可以在确保有原始影像图作为参照的情况下，借助增强图(d)观察原始影像图(a)中呈现较差的部分。

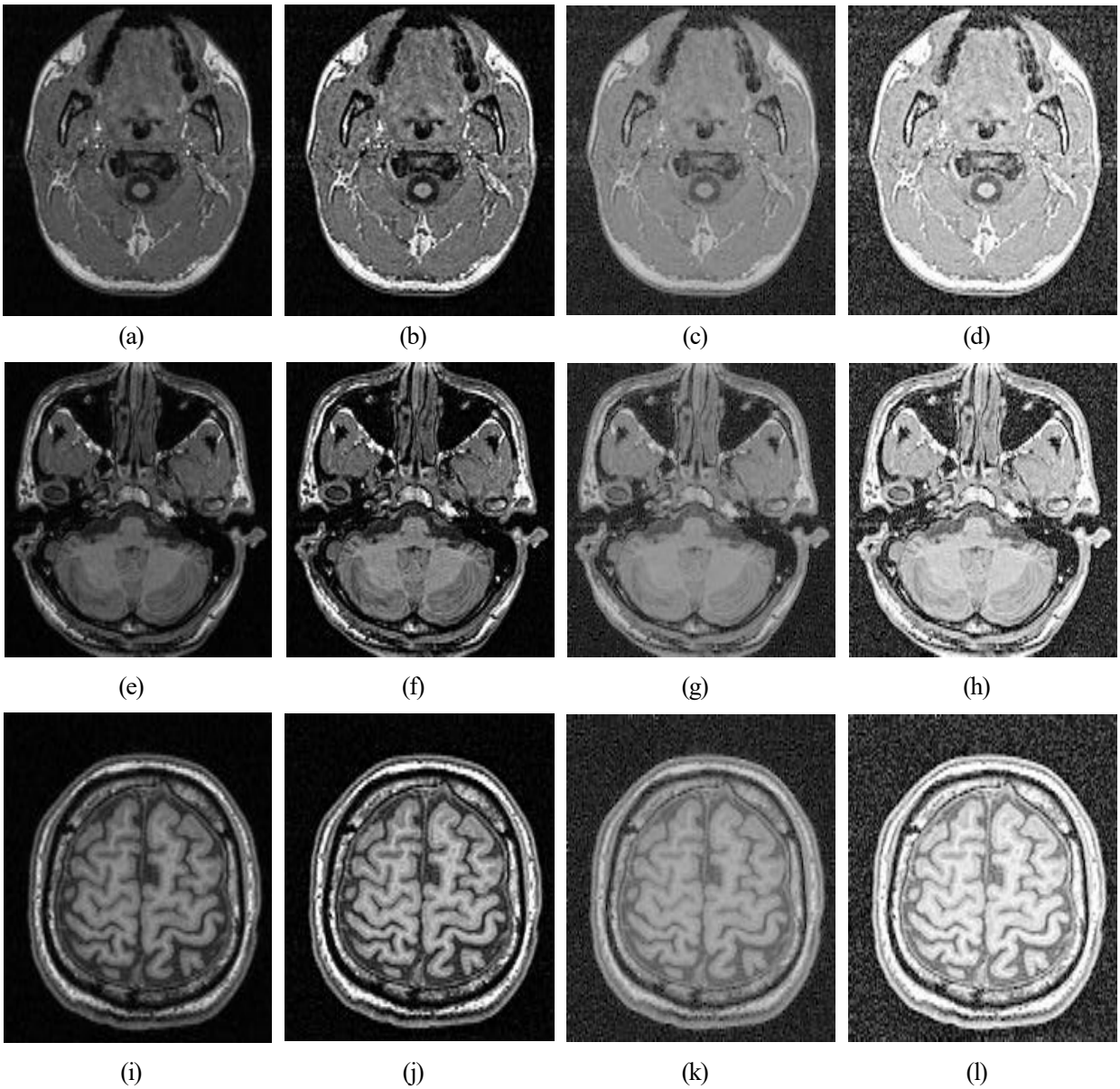


图 5.4 大脑不同高度水平切面增强图。其中每行为同一高度水平切面的原始影像以及不同处理方法的结果图。其中(a)(e)(i)为原始影像，(b)(f)(j)为0.2阶微分掩模滤波增强图，(c)(g)(k)为分数阶全变分 Retinex 算法增强图，(d)(h)(l)为两种方法混合处理的增强图。

图 5.4 展示了不同高度水平切面脑影像图的增强结果。与上述分析类似。分数阶掩模法对脑影像图这类信噪比第，噪音均匀散布，层次感强的图像有更好的增强效果（如图 5.4(b)(f)(j)）。而分数阶全变分 Retinex 算法在

噪音干扰较大的情况下整体效果不好，但是可以挖掘保留高频区域的细节。如果将两种方法相结合，则可以互相取长补短，作为原始影像的辅助图像，供相关研究者参考。

6 工作总结和心得体会

6.1 工作总结

作为计算生物学交叉专业的学生，毕业论文的项目也应以学科间交叉创新为主旨。因此本文在选题方面经与两位导师深入交流后，选择了这一既融合多学科知识、又具有创新内容的主题。在广泛涉猎多学科的基础上，确立了《分数阶图像增强在计算神经科学中的应用》这一主题。

首先，本文以数学模型为工具，从算法分析和系统级编程的角度深入分析了最新的分数阶全变分 Retinex 算法，并重点研究了算法计算复杂度。为尽可能削减分数阶全变分 Retinex 算法计算量大的弊端，本文从算法描述和实现两个角度共同优化了该算法。比如，在计算$P(l)$的过程中，免去了很多重复性的计算，并充分利用了时间局部性、空间局部性原则；在迭代计算l的过程中，采用了图形学中增量的思想，大大节约了时间和空间。实验证明，改进后的算法与改进前算法相比，所画时间大约为原算法的 1/3。在比较改进前后算法的速度后，文章还提出了使用拉格朗日三点插值法对图像边缘点进行延拓的方法，完善了分数阶全变分 Retinex 算法的定义。

之后，本文又选择了算法中的两个细节部分——范数的取法和一阶微分的做法，比较了不同的范数取法、不同一阶微分取法对处理效果的影响，得出二范数和 PU-2 算子一阶微分在通常情况下是比较好的计算方式。当然，算法中还有其他细节问题值得讨论和比较，比如，如何根据图像选取合适的v_1, v_2, v_3参数。

接下来，文章尝试将分数阶微分的记忆性、非局域性与认知神经科学中视觉感知的相关理论进行统一。用视网膜上两级细胞的侧向抑制效应与神经节细胞在视神经网络中的系统作用解释分数阶微分在图像增强中具有优势的原因。具体来说，PU-2 分数阶掩模对物理刺激的响应与两级细胞对物理光刺激产生的细胞活动响应具有高度的相似性，也就是侧向抑制。这种对变化的感知、记忆与响应无论在真实的视觉感知中还是在图像处理中，都一致性的起到了增强对比度、增强细节变化信息的功能。将两者放在一起对比，对分数阶微积分、图像增强算法和视神经网络的研究都具有启发性意义。

最后，为了更好的利用图像增强这一强有力的工具，我们将分数阶图像增强应用在了脑磁共振影像图上。增强后的影像图可以作为辅助图像，供相关领域的研究者参考。当然，由于医学影像具有信噪比低、噪音遍布广、图像中存在大量所谓“空区域”等特点，当前采用的图像增强方法对脑影像图的增强效果，不如对实际物象图像的增强效果好。对于这类图像，是否存在更好的增强方法，有待进一步的研究。

总体来讲，本毕业设计全部工作中，既包含了数学模型、计算方法和计算机图像处理的知识，也融合了神经生物学、脑与认知科学、心理学中的有关概念与研究方法。这种学科间交叉的研究主题对相关学科的发展有很大的启发意义。未来应该进一步从神经科学的角度，实验研究神经系统的编码与分数阶微积分之间的联系。

受 COVID-19 疫情影响，这次的毕业设计基本采取的是线上交流指导方式。在这种情况下，完成这种交叉性很强的毕业论文是一个很大的挑战。在前半期我独立完成了《The fractional calculus theory and applications of differentiation and integration to arbitrary order》一书前半部分的学习，并整理了中文笔记，为之后的工作打下了有关数学基础。之后在两位导师的指导下，我完成了文献阅读、代码编写，并收集材料进行了系列性实验。其中虽然遇到一些困难，但都通过查阅文献、请教老师等方式得以解决。在文章的最后部分，我充分发挥了交叉学科的优势，将分数阶图像增强与认知神经科学结合，这无疑是本文最大的创新点。接下来，我将继续发挥交叉学科的优势，继续将计算科学与认知神经科学结合，研究学习计算神经科学的理论和应用。

6.2 心得体会

总体来说，这次毕业设计的全部工作中，挑战与创新并存，理论与实践并存。完成这些工作让我对科研有了更深入的理解与体会。计算生物学作为川大的交叉创新专业，确实为学生发展提供了一个优秀的平台。大学四年的学习使我对自己的兴趣和特长有了更清晰的认识，也帮助我建立了未来更加明确的奋斗目标。令我十分满意的一点是，毕业论文所做的工作与我的研究兴趣非常契合，这些工作也锻炼了我的综合能力。

随着知识体系的不断完善，我越来越清楚地认识到，数学、计算科学作为一切的基本方法和基础工具，在各领域研究中发挥了重要的作用。与已经被数学语言充分描述的经典物理、化学不同，过于复杂的生物系统很难被数学公式精准的表达。因此，当前大部分和认知神经科学相关的工作都只是统计性结果。像使用数学建模与计算机研究计算生物学一样，计算认知神经科学也应该被量化、系统化、算法化地研究。另一方面，计算机视觉、人工智能与数字逻辑分别启发自人类的视觉、智能和逻辑。所以，只有更精准的理解我们大脑中的神经机制，我们才能设计出与真实大脑更接近的机器智能。

大学的知识如大海般无垠。大学这个平台开阔了我的视野，帮助我构建了自己的知识体系，也教会了我思辨能力。目前，我已经获得了美国理想学校的 Ph.D. offer，这对我是极大的成就与鼓励。我将继续秉持严谨的科学态度和积极的热情，在博士期间取得更优异的成绩。

参考文献

[1] O'reilly R C, Munakata Y, Frank M, et al. Computational cognitive neuroscience[M]. PediaPress, 2012.
[2] Wenliang L K, Seitz A R. Deep neural networks for modeling visual perceptual learning[J]. Journal of Neuroscience, 2018, 38(27): 6028-6044.
[3] Machado J T, Kiryakova V, Mainardi F. Recent history of fractional calculus[J]. Communications in Nonlinear Science
Numerical Simulation, 2011, 16(3): 1140-1153.
[4] Pu Y-F. Fractional-order euler-Lagrange equation for fractional-order variational method: A necessary condition for fractional-order fixed boundary optimization problems in signal processing and image processing[J]. IEEE Access, 2016, 4: 10110-10135.
[5] Gonzalez R C, Woods R E, Eddins S L. Digital image processing using MATLAB[M]. Pearson Education India, 2004.
[6] Provenzi E, De Carli L, Rizzi A, et al. Mathematical definition and analysis of the Retinex algorithm[J]. JOSA A, 2005, 22(12): 2613-2621.
[7] Land E H, Mccann J J. Lightness and retinex theory[J]. Josa, 1971, 61(1): 1-11.
[8] Provenzi E, Fierro M, Rizzi A, et al. Random spray retinex: a new retinex implementation to investigate the local properties of the model[J]. IEEE Transactions on Image Processing, 2006, 16(1): 162-171.
[9] Petro A B, Sbert C, Morel J-M. Multiscale retinex[J]. Image Processing On Line, 2014: 71-88.
[10] Ng M K, Wang W. A total variation model for Retinex[J]. SIAM Journal on Imaging Sciences, 2011, 4(1): 345-365.
[11] Pu Y-F, Siarry P, Chatterjee A, et al. A fractional-order variational framework for retinex: fractional-order partial differential equation-based formulation for multi-scale nonlocal contrast enhancement with texture preserving[J]. Transactions on Image Processing, 2017, 27(3): 1214-1229.
[12] Kalat J W. Biological psychology[M]. Nelson Education, 2015.
[13] Gazzaniga M S. The cognitive neurosciences[M]. MIT press, 2009.
[14] Gibson J J. The perception of the visual world[J], 1950.
[15] Delcomyn F. Foundations of neurobiology[M]. WH Freeman, 1998.
[16] Wagman I, Hartline H, Milne L. Excitability changes of single visual receptor cells following flashes of light of intensity near threshold[C]. Federation Proceedings, 1949: 159-160.
[17] Oldham K, Spanier J. The fractional calculus theory and applications of differentiation and integration to arbitrary order[M]. Elsevier, 1974.
[18] Land E H. The retinex[J]. American Scientist, 1964, 52(2): 247-264.
[19] Pu Y-F, Zhang N, Wang Z-N, et al. Fractional-Order Retinex for Adaptive Contrast Enhancement of Under-Exposed Traffic Images[J]. IEEE Intelligent Transportation Systems Magazine, 2019.
[20] Pu Y-F, Zhou J-L, Yuan X. Fractional differential mask: a fractional differential-based approach for multiscale texture enhancement[J]. IEEE transactions on image processing, 2009, 19(2): 491-511.
[21] Pu Y-F, Zhou J-L, Zhang Y, et al. Fractional extreme value adaptive training method: fractional steepest descent approach[J]. IEEE transactions on neural networks
learning systems, 2013, 26(4): 653-662.
[22] Anderson M, Motta R, Chandrasekar S, et al. Proposal for a standard default color space

for the internet—srgb[C]. Color and imaging conference, 1996: 238-245.
[23] Stokes M. A standard default color space for the internet-srgb[EB/OL]. http://www.color.org/contrib/sRGB.html.
[24] Samui P, Roy S S, Balas V E. Handbook of neural computation[M]. Academic Press, 2017.

声　明

本人声明所呈交本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

学位论文作者（签名）_____

论文指导教师（签名）_____

2020 年　5 月 27 日

致　谢

首先十分感谢蒲亦非老师在百忙之中抽出宝贵时间对我的毕业设计进行了指导与帮助。蒲亦非老师在我进入吴玉章荣誉学院、出国交换、博士申请上都给予了我充分的帮助。在学术上他也是一位令人敬佩的科学家，是我科研道路上的榜样。同时，也非常感谢邓伟老师在毕业设计以及大学四年中给予我的帮助支持。他是我计算神经科学上的启蒙导师，四年中也为我创造了很多学习机会，如医院实习、国际会议等，让我开阔了眼界。作为第二指导老师，他所具备的灵活思维和创新精神是我学习的榜样。此外，还要感谢毕业期间为我提供脑影像图的魏巍师兄，为我提供速度、参数对比实验图片的室友曾涵章同学。

大学四年的学习生涯中，除了一直支持我的蒲亦非老师和邓伟老师外，还有许多人给我带来了支持与感动。是父母、家人多年来对我的悉心培养，使我能够有机会接触到更广阔的世界，使我能够进入优秀的学校安心学习。在四川大学，有很多优秀的老师在学术方面给我带来了深远的影响。有在博士申请上长期给予我帮助的班主任席祯翔老师；有在数学建模学习上热心帮助我的胡朝浪老师、张世全老师、邓瑾老师；有在英语学习上帮助我的张露露老师；有在计算机学习上大力支持我的胡晓勤老师、程艳红老师、赵辉老师。同时还要感谢软件学院、生命科学学院和吴玉章学院中为计算生物学专业的建设付出努力、给予支持的老师们。在大三时，四川大学以及吴玉章学院为我提供了公费出国交换一年的机会。在美国田纳西大学诺克斯维尔分校中，Bruce J. MacLennan 教授指导我和徐璐、徐心玥同学组成的团队拿到了美国数学建模大赛特等奖提名的优异成绩。Aaron T. Buss 教授和 Caglar Tas 教授指导我在他们的实验室中学习，锻炼了我的科研课题领导能力。来自台湾的萧世綸教授给了我学术写作上的指导与生活上的帮助。孔子学院的沈洁老师一家人以及田纳西大学中国学生学者联谊会的同学们在生活上给了我极大的帮助和支持。当然，四年的大学生活中也很多同学一直在我身边时刻支持着我。我的室友（好友）余梓棋、熊沐钊、冯岚清、曾涵章、黄剑宇、连伟邵同学在大学四年期间给了我充分的包容与关爱。多年的挚友李智同学一直在我身后支持着我。我的女朋友胡海瑶同学在我学习、工作繁忙之时，给予了我充分的理解和鼓励。此外，生命科学学院的辅导员徐青锐老师在我的人生规划上给予了我充分的指导和帮助；燕丽娜学姐在我刚入学时给了我很多的大学经验；侯牧村同学四年来的实验与项目中与我组队不遗余力；程昊阳学长在我博士申请文书和专业知识上给予了很大的帮助……最后，作为计算生物学班的班长和院学生会的一员，感谢所有同学对我学生工作的认可与支持。

附录 1 关键算法代码

```
function [coefficients] = PU2Operator(v, N)
%PU2OPERATOR PU-2 分数阶微分算子掩模
%   给定阶数 v 和掩模尺寸 N = n + 2，其中 N >= 3，N 通常为奇数
%   返回掩模系数，从 C_{-1} 到 C_n

% 计算掩模系数
n = N - 2;
coefficients = zeros(1, n + 2); % 生成一空的行向量，用于存储掩模系数
temp = [v/4 + v^2/8, 1 - v^2/4, -v/4 + v^2/8];
for k = -1:(n - 2)
    coefficients(k + 2) = myGamma(k - v + 1, -v)/gamma(k + 2)*temp(1) ...
        + myGamma(k - v, -v)/gamma(k+1)*temp(2) ...
        + myGamma(k - v - 1, -v)/gamma(k)*temp(3);
end
coefficients(n + 1) = myGamma(n - v - 1, -v)/gamma(n) * temp(2) ...
    + myGamma(n - v - 2, -v)/gamma(n - 1) * temp(3);
coefficients(n + 2) = myGamma(n - v - 1, -v)/gamma(n) * temp(3);
end



function [D8, D] = cal8Deriv(A, C, norm)
%CAL8DERIV 计算八个方向上的分数阶偏导数
%   据系数向量，对输入进来的矩阵计算八个方向上的偏导数矩阵
% C 的长度为 n + 2
% D8 的第三个维度的排列顺序为 u, d, l, r, ld, ru, lu, rd

n = size(C, 2) - 2;
[sizex, sizey] = size(A);

% 镜像扩展矩阵，如何扩展矩阵是创新点
A_pad = pad(A, n, sizex, sizey);

% 分别计算八个方向的分数阶偏导数矩阵
D_u = 0;
for k = -1:n
    D_u = D_u + C(k + 2) * A_pad((n + 1 - k):(sizex + n - k), (n + 1):(n + sizey));
end

D_d = 0;
```

```
for k = -1:n
    D_d = D_d + C(k + 2) * A_pad((n + 1 + k):(sizex + n + k), (n + 1):(n + sizey));
end

D_l = 0;
for k = -1:n
    D_l = D_l + C(k + 2) * A_pad((n + 1):(n + sizex), (n + 1 - k):(sizey + n - k));
end

D_r = 0;
for k = -1:n
    D_r = D_r + C(k + 2) * A_pad((n + 1):(n + sizex), (n + 1 + k):(sizey + n + k));
end

D_ld = 0;
for k = -1:n
    D_ld = D_ld + C(k + 2) * A_pad((n + 1 + k):(sizex + n + k), (n + 1 - k):(sizey + n - k));
end

D_ru = 0;
for k = -1:n
    D_ru = D_ru + C(k + 2) * A_pad((n + 1 - k):(sizex + n - k), (n + 1 + k):(sizey + n + k));
end

D_lu = 0;
for k = -1:n
    D_lu = D_lu + C(k + 2) * A_pad((n + 1 - k):(sizex + n - k), (n + 1 - k):(sizey + n - k));
end

D_rd = 0;
for k = -1:n
    D_rd = D_rd + C(k + 2) * A_pad((n + 1 + k):(sizex + n + k), (n + 1 + k):(sizey + n + k));
end

% 将 8 个方向的分数阶偏导数矩阵整理到一个三维数组中
D8 = zeros(sizex, sizey, 8);
D8(:, :, 1) = D_u;
D8(:, :, 2) = D_d;
D8(:, :, 3) = D_l;
D8(:, :, 4) = D_r;
```

```
D8(:, :, 5) = D_ld;
D8(:, :, 6) = D_ru;
D8(:, :, 7) = D_lu;
D8(:, :, 8) = D_rd;

% 计算全微分
if norm == 1
    D = abs(D_u) + abs(D_d) + abs(D_l) + abs(D_r) + abs(D_ld) + abs(D_ru) + abs(D_lu) +
abs(D_rd); % 采用一范数的方法
elseif norm == 2
    D = sqrt(D_u.^2 + D_d.^2 + D_l.^2 + D_r.^2 + D_ld.^2 + D_ru.^2 + D_lu.^2 + D_rd.^2); % 采
用二范数的方法
else
    D = max(abs(D8), [], 3); % 采用最大绝对值的方法
end



function [P_l] = computeP(v_1, v_2, v_3, alpha_1, alpha_2, l, s_0, epsilon_1, C, norm)
%computeP 根据当前矩阵 l^n，利用给定参数 α_1, α_2，计算 P(l^n)
% 针对当前的 l 矩阵，分别计算 l 和 (l - s) 在八个方向上的分数阶偏导数矩阵
ls = l - s_0;

[D_l_8, D_l] = cal8Deriv(l, C, norm);
D_l(find(D_l < epsilon_1)) = epsilon_1;

[D_ls_8, D_ls] = cal8Deriv(ls, C, norm);
D_ls(find(D_ls < epsilon_1)) = epsilon_1;

% 对 k 进行遍历，将求和符号逐 k 累加到 sum_k
sum_k = 0;
for k = 0:1
    prod_tau = 1;
    for tau = 1:(2*k)
        prod_tau = prod_tau*(v_2 - tau + 1);
    end

    % 计算在当前 k 值的情况下，八个方向的偏微分的差分的和

    eight_terms = 0;
    D_l_power = D_l.^(v_2 - 2*k - 2);
```

```
    D_ls_power = D_ls.^(v_2 - 2*k - 2);
    second_term = alpha_1 * (abs(ls)).^(v_2 - 2*k - 2) .* ls;
    for layer = 1:8
        temp = D_l_power .* D_l_8(:, :, layer) + second_term + alpha_2 * D_ls_power .*
D_ls_8(:, :, layer);
        eight_terms = eight_terms + fstDiff(temp, layer);
    end

    % do fractional derivative
    sum_k = sum_k + prod_tau/factorial(2*k) * eight_terms;
end

P_l = -gamma(1 - v_1)/gamma(-v_1)/gamma(-v_3) * sum_k;
end


function [result] = fstDiff(A, direction)
%FSTDIFF 对一个矩阵在指定方向上做一次差分
%   对输入矩阵 A，向 direction 方向做一次差分
% direction 设置如下:
% [1, 2, 3, 4, 5, 6, 7, 8] := [D_u, D_d, D_l, D_r, D_ld, D_ru, D_lu, D_rd]

[sizex, sizey] = size(A);

method = 2;
if method == 1
    C = [-1/4, -1/2, 0, 1/2, 1/4];
    A_pad = pad(A, 1, sizex, sizey);
    if direction == 1 % D_u
        result = C(1)*A_pad(1:sizex, 1:sizey) + ...
            C(2)*A_pad(1:sizex, 2:(sizey + 1)) + ...
            C(1)*A_pad(1:sizex, 3:(sizey + 2)) + ...
            C(5)*A_pad(3:(sizex + 2), 1:sizey) + ...
            C(4)*A_pad(3:(sizex + 2), 2:(sizey + 1)) + ...
            C(5)*A_pad(3:(sizex + 2), 3:(sizey + 2));
    elseif direction == 2 % D_d
        result = C(5)*A_pad(1:sizex, 1:sizey) + ...
            C(4)*A_pad(1:sizex, 2:(sizey + 1)) + ...
            C(5)*A_pad(1:sizex, 3:(sizey + 2)) + ...
            C(1)*A_pad(3:(sizex + 2), 1:sizey) + ...
```

```
        C(2)*A_pad(3:(sizex + 2), 2:(sizey + 1)) + ...
        C(1)*A_pad(3:(sizex + 2), 3:(sizey + 2));
elseif direction == 3 % D_l
    result = C(1)*A_pad(1:sizex, 1:sizey) + ...
        C(2)*A_pad(2:(sizex + 1), 1:sizey) + ...
        C(1)*A_pad(3:(sizex + 2), 1:sizey) + ...
        C(5)*A_pad(1:sizex, 3:(sizey + 2)) + ...
        C(4)*A_pad(2:(sizex + 1), 3:(sizey + 2)) + ...
        C(5)*A_pad(3:(sizex + 2), 3:(sizey + 2));
elseif direction == 4 % D_r
    result = C(5)*A_pad(1:sizex, 1:sizey) + ...
        C(4)*A_pad(2:(sizex + 1), 1:sizey) + ...
        C(5)*A_pad(3:(sizex + 2), 1:sizey) + ...
        C(1)*A_pad(1:sizex, 3:(sizey + 2)) + ...
        C(2)*A_pad(2:(sizex + 1), 3:(sizey + 2)) + ...
        C(1)*A_pad(3:(sizex + 2), 3:(sizey + 2));
elseif direction == 5 % D_ld
    result = C(1)*A_pad(2:(sizex + 1), 1:sizey) + ...
        C(2)*A_pad(3:(sizex + 2), 1:sizey) + ...
        C(1)*A_pad(3:(sizex + 2), 2:(sizey + 1)) + ...
        C(5)*A_pad(1:sizex, 2:(sizey + 1)) + ...
        C(4)*A_pad(1:sizex, 3:(sizey + 2)) + ...
        C(5)*A_pad(2:(sizex + 1), 3:(sizey + 2));
elseif direction == 6 % D_ru
    result = C(5)*A_pad(2:(sizex + 1), 1:sizey) + ...
        C(4)*A_pad(3:(sizex + 2), 1:sizey) + ...
        C(5)*A_pad(3:(sizex + 2), 2:(sizey + 1)) + ...
        C(1)*A_pad(1:sizex, 2:(sizey + 1)) + ...
        C(2)*A_pad(1:sizex, 3:(sizey + 2)) + ...
        C(1)*A_pad(2:(sizex + 1), 3:(sizey + 2));
elseif direction == 7 % D_lu
    result = C(5)*A_pad(3:(sizex + 2), 2:(sizey + 1)) + ...
        C(4)*A_pad(3:(sizex + 2), 3:(sizey + 2)) + ...
        C(5)*A_pad(2:(sizex + 1), 3:(sizey + 2)) + ...
        C(1)*A_pad(2:(sizex + 1), 1:sizey) + ...
        C(2)*A_pad(1:sizex, 1:sizey) + ...
        C(1)*A_pad(1:sizex, 2:(sizey + 1));
else % D_rd
    result = C(1)*A_pad(3:(sizex + 2), 2:(sizey + 1)) + ...
        C(2)*A_pad(3:(sizex + 2), 3:(sizey + 2)) + ...
```

```
            C(1)*A_pad(2:(sizex + 1), 3:(sizey + 2)) + ...
            C(5)*A_pad(2:(sizex + 1), 1:sizey) + ...
            C(4)*A_pad(1:sizex, 1:sizey) + ...
            C(5)*A_pad(1:sizex, 2:(sizey + 1));
    end
else
    C = [0.375, 0.375, -0.875, 0.125];
    A_pad = pad(A, 2, sizex, sizey);
    if direction == 1 % D_u
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad((3 - k):(sizex + 2 - k), 3:(2 + sizey));
        end
    elseif direction == 2 % D_d
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad((3 + k):(sizex + 2 + k), 3:(2 + sizey));
        end
    elseif direction == 3 % D_l
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad(3:(2 + sizex), (3 - k):(sizey + 2 - k));
        end
    elseif direction == 4 % D_r
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad(3:(2 + sizex), (3 + k):(sizey + 2 + k));
        end
    elseif direction == 5 % D_ld
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad((3 + k):(sizex + 2 + k), (3 - k):(sizey + 2 -
k));
        end
    elseif direction == 6 % D_ru
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad((3 - k):(sizex + 2 - k), (3 + k):(sizey + 2 +
k));
        end
```

```
    elseif direction == 7 % D_lu
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad((3 - k):(sizex + 2 - k), (3 - k):(sizey + 2 -
k));
        end
    else % D_rd
        result = 0;
        for k = -1:2
            result = result + C(k + 2) * A_pad((3 + k):(sizex + 2 + k), (3 + k):(sizey + 2 +
k));
        end
    end
end
end


function [A] = pad(A, n, row, column)
%PAD 对图像矩阵的边缘进行拉格朗日插值延拓
%   A 为图像矩阵，n 为延拓次数，拉格朗日插值公式为 s(-1) = 3[s(0) - s(1)] + s(2)
% row, column 为 A 的行数和列数

for k = 1:n
    left = 3*(A(:, 1) - A(:, 2)) + A(:, 3);
    right = 3*(A(:, column) - A(:, column - 1)) + A(:, column - 2);
    top = 3*(A(1, :) - A(2, :)) + A(3, :);
    bottom = 3*(A(row, :) - A(row - 1, :)) + A(row - 2, :);
    top_left = 3*(A(1, 1) - A(2, 2)) + A(3, 3);
    top_right = 3*(A(1, column) - A(2, column - 1)) + A(3, column - 2);
    bottom_left = 3*(A(row, 1) - A(row - 1, 2)) + A(row - 2, 3);
    bottom_right = 3*(A(row, column) - A(row - 1, column - 1)) + A(row - 2, column - 2);
    A = [top_left, top, top_right; left, A, right; bottom_left, bottom, bottom_right];
    row = row + 1;
    column = column + 1;
end
end


%FR.m 主程序
% 在这里设置参数
```

```
v_1 = 1.25;
v_2 = 2.25;
v_3 = 0.90;
mu = 0.1;
alpha_1 = 0.05;
alpha_2 = 0.1;
Delta_t = 0.002;
n = 6;
epsilon_1 = 0.006;
epsilon_2 = 0.00001;
gamma_corr = 2.2;
N = 7;
norm = 2;

% step 1: 读图
[filename, pathname] = uigetfile('*.jpg');
img_filename = [pathname, filename];
RGB_img = imread(img_filename);
HSV_img = rgb2hsv(RGB_img);
V = HSV_img(:, :, 3); % 0 <= V <= 1
V_max = max(max(V));
V_min = min(min(V));
V = (V - V_min)/(V_max - V_min);
s = log(255*V + 1);
l_curr = 1.05*s;
l_curr(find(l_curr == 0)) = epsilon_2;
Delta_l = 0;
C = PU2Operator(v_1, N);

% step 2:
for t = 1:n
    P_l = computeP(v_1, v_2, v_3, alpha_1, alpha_2, l_curr, s, epsilon_1, C, norm);
    Delta_l = P_l*(Delta_t^v_3) - 2*mu/gamma(3 - v_3)*(Delta_l.^2).*(real(l_curr.^(-v_3)));
    l_curr = l_curr + Delta_l;
    l_curr(find(l_curr == 0)) = epsilon_2;
end
L = exp(l_curr);
HSV_img(:, :, 3) = V./((L/255).^(1 - 1/gamma_corr)); % gamma 校正
result_RGB_img = tosRGB(hsv2rgb(HSV_img));
imshow(result_RGB_img);
```

```
imwrite(result_RGB_img, '2.jpg');
```