Final Report on Variational Models and Numerical Methods for Image Processing

2020 NCTS-USRP-Group 6

Members: Jia-Wei Liao (NTNU), Chun-Hsien Chen (NCCU), Chen-Yang Dai (NCTU)

Advisor: Suh-Yuh Yang (NCU) TA: Chen-Yan Wei (NCU)

September 4, 2020

Contents

Chapter 1

Introduction

1.1 Image processing

Sometimes, we are unsatisfied with the result of our image. So, most people think many ways to change the image to best. There are many important issues in image processing and computer vision. The following topics are commonly encountered in image processing:

1. Image denoising
2. Image contrast enhancement
3. Image stitching
4. Image segmentation

In this research program, we will focus on topics 1, 2, and 3.

1.2 Preliminaries

Our goal is to design some variational models that can reduce the image noise or enhance the image contrast, or integrate two or more source images of the same scene into a single fused image with better visual quality than the original images. So we're going to introduce the following mathematical knowledge that will be helpful later.

1.2.1 Total variation

Definition 1.2.1 (Total variation of one variable function)**.**
Let $\Omega = (a, b) \subseteq \mathbb{R}$ *and* $\mathcal{P}_n = \{a = x_0, x_1, \cdots, x_{n-1}, x_n = b\}$ *be an arbitrary partition of* $\overline{\Omega}$*. The total variation of a real-valued function* $u : \Omega \to \mathbb{R}$ *is defined as the quantity,*

$$\|u\|_{TV(\Omega)} = \sup_{\mathcal{P}_n} \sum_{i=1}^{n} |u(x_i) - u(x_{i-1})| .$$

Theorem 1.2.1.
If u is a smooth function, then we have

$$\|u\|_{TV(\Omega)} = \int_\Omega |u'(x)|dx.$$

Proof.

$$\begin{aligned}\|u\|_{TV(\Omega)} &= \sup_{\mathcal{P}_n} \sum_{i=1}^{n} |u(x_i) - u(x_{i-1})| \\ &= \sup_{\mathcal{P}_n} \sum_{i=1}^{n} \left| \frac{u(x_i) - u(x_{i-1})}{\Delta x_i} \right| \Delta x_i \\ &= \int_\Omega |u'(x)|dx.\end{aligned}$$

□

Definition 1.2.2 (Total variation of two variable function)**.**
Let Ω be an open set of $\mathbb{R}^2$ and $u \in L^1(\mathbb{R})$. The total variation of u in Ω is defined as

$$\|u\|_{TV(\Omega)} = \sup \left\{ \int_\Omega u \operatorname{div} \varphi \, d\boldsymbol{x} : \varphi \in C_c^1\left(\Omega, \mathbb{R}^2\right), \|\varphi\|_{L^\infty(\Omega)} \leq 1 \right\},$$

where $C_c^1(\Omega, \mathbb{R}^n)$ is the set of continuously differentiable vector functions of compact support contained in Ω, and $\|\cdot\|_{L^\infty(\Omega)}$ is the essential supremum norm.

Theorem 1.2.2. *If u is a smooth function, then we have*

$$\|u\|_{TV(\Omega)} = \int_\Omega |\nabla u| d\boldsymbol{x}.$$

Definition 1.2.3 (Bounded variation)**.** *If $\|u\|_{TV(\Omega)} < \infty$, then we say that u is a function of bounded variation. Moreover, the space of functions of bounded variation $BV(\Omega)$ is defined as $u \in L^1(\Omega)$ such that the total variation is finite, i.e.,*

$$BV(\Omega) = \left\{ u \in L^1(\Omega) : \|u\|_{TV(\Omega)} < \infty \right\}.$$

Remark. *$BV(\Omega)$ is a Banach space with the norm*

$$\|u\|_{BV(\Omega)} = \|u\|_{L^1(\Omega)} + \|u\|_{TV(\Omega)}.$$

1.2.2 Euler-Lagrange equation

We know that if a single variable differentiable function has an extremum at some point, then its derivative must be zero at that point. Now if we are dealing with a functional, then there exists a function satisfying the Eular-Lagrange equation. Before introducing the equation, we need a lemma.

Lemma 1.2.1.
If F is a continuous function on an open interval (a,b), and satisfies $\int_a^b F(x)G(x)dx = 0$ for all compactly supported smooth functions G on (a,b), then F is identically zero on (a,b). If (a,b) is replaced by the closed interval $[a,b]$, then we require only that G vanishes at the endpoints a and b.

Proof. Suppose $F(c) \neq 0$ for some $c \in (a,b)$. Without loss of generality, assume that $F(c) > 0$, then there exists $[\alpha,\beta] \subseteq (a,b)$ such that $F(x) > 0$ for all $x \in [\alpha,\beta]$. Let $G(x)$ be a smooth function vanishing on (a,b) expect $[\alpha,\beta]$ in which $G(x)$ is positive. Then we have $\int_a^b F(x)G(x)dx \neq 0$. This is a contraction. Hence $F(x) = 0$ for all $x \in (a,b)$. Now if we change the open interval into closed interval, then the proof is similar, we only require that $G(x)$ vanishes at the endpoints a and b. □

Theorem 1.2.3 (Euler-Lagrange Equation for one dimension)**.**
Let $[a,b] \subseteq \mathbb{R}$ be a given interval. We consider the functional

$$\int_a^b L(x, v(x), v'(x))dx,$$

where we assume that $v \in C^2([a,b])$ and $L \in C^2$ with respect to its arguments x, v and v'. If u is an extremum of the functional E, then we have

$$\frac{\partial L}{\partial u} - \frac{d}{dx}\left(\frac{\partial L}{\partial u'}\right) = 0 \quad \text{in } (a,b).$$

Proof. Assume that $E[v] = \int_a^b L(x,v,v')dx$ has a local minimum at u, and $\eta(x)$ be a smooth function vanishes at a and b, then we have

$$E[u] \leq E[u + \varepsilon\eta]$$

as ε close to 0. Define $\Phi(\varepsilon) = E[u + \epsilon\eta]$, which is in the variable ε. Then we have

$$\Phi'(0) = \frac{d\Phi}{d\varepsilon}\bigg|_{\varepsilon=0} = \int_a^b \left(\frac{dL}{d\epsilon}\right)\bigg|_{\varepsilon=0} dx = 0.$$

Note that $v = u + \varepsilon\eta$ and $v' = u' + \varepsilon\eta'$. So,

$$\frac{dL}{d\varepsilon} = \frac{\partial L}{\partial v}\cdot\frac{\partial v}{\partial \varepsilon} + \frac{\partial L}{\partial v'}\cdot\frac{\partial v'}{\partial \varepsilon} = \frac{\partial L}{\partial v}\eta + \frac{\partial L}{\partial v'}\eta'.$$

Then

$$\begin{aligned}\int_a^b \left(\frac{dL}{d\varepsilon}\right)\bigg|_{\varepsilon=0} dx &= \int_a^b \left(\frac{\partial L}{\partial u}\eta + \frac{\partial L}{\partial u'}\eta'\right)dx \\ &= \int_a^b \left(\frac{\partial L}{\partial u}\eta - \eta\frac{d}{dx}\left(\frac{\partial L}{\partial u'}\right)\right)dx \\ &= \int_a^b \eta\left(\frac{\partial L}{\partial u} - \frac{d}{dx}\left(\frac{\partial L}{\partial u'}\right)\right)dx = 0.\end{aligned}$$

By Lemma 1.2.1, we have

$$\frac{\partial L}{\partial u} - \frac{d}{dx}\left(\frac{\partial L}{\partial u'}\right) = 0.$$

□

Theorem 1.2.4 (Euler-Lagrange Equation for two dimensions)**.**
Let $\Omega \subseteq \mathbb{R}^2$ be an open set. We consider the functional

$$E(v) = \int_\Omega L(x, y, v, v_x, v_y) d\boldsymbol{x},$$

where we assume that $v \in C^2(\Omega)$ and $L \in C^2$ with respect to its arguments x, y, v, v_x and v_y. If u is an extremum of the functional E, then we have

$$\frac{\partial L}{\partial u} - \nabla \cdot \left(\frac{\partial L}{\partial u_x}, \frac{\partial L}{\partial u_y}\right) = 0 \quad \text{in } \Omega.$$

Chapter 2

Image denoising

There are many mathematical ways to reduce image noise such as

- Fourier transform
- Heat-type equation
- Machine learning
- Variational method (energy functional)

In this chapter, we will introduce the Rudin-Osher-Fatemi (ROF) model for image denoising.

2.1 ROF variational model

ROF model (Physica D, 1992).
Let $f : \overline{\Omega} \subseteq \mathbb{R}^2 \to \mathbb{R}$ be a given noisy image. Rudin, Osher, and Fatemi proposed the following model for image denoising:

$$\min_{u \in BV(\Omega) \cap L^2(\Omega)} \left(\|u\|_{TV(\Omega)} + \frac{\lambda}{2} \int_\Omega (u - f)^2 d\boldsymbol{x} \right),$$

where $\lambda > 0$ is a tuning parameter which controls the regularization strengthen.

Remark.

1. *A smaller value of λ will lead to a more regular solution.*
2. *The space of functions with bounded variation can help remove spurious oscillations (noise) and preserve sharp signals (edges).*
3. *The* TV *term allows the solution to have discontinuities.*

The energy functional of ROF model is convex, which guarantees the existence of minimizer of the minimization problem.

Lemma 2.1.1. *Let the energy functional of ROF model be* $E[u] = \int_\Omega \left(|\nabla u| + \frac{\lambda}{2}(u-f)^2 \right) d\boldsymbol{x}$. *Then E is convex, that is, for any u and v, we have*

$$E[tu + (1-t)v] \leq tE[u] + (1-t)E[v], \quad \forall\, 0 \leq t \leq 1.$$

Proof. By the triangle inequality, we have

$$\begin{aligned} E[tu + (1-t)v] &= \int_\Omega |\nabla(tu + (1-t)v)| + \frac{\lambda}{2}[(tu + (1-t)v) - (tf + (1-t)f)]^2 d\boldsymbol{x} \\ &\leq \int_\Omega t|\nabla u| + (1-t)|\nabla v| + \frac{\lambda}{2}[t(u-f) + (1-t)(v-f)]^2 d\boldsymbol{x}. \end{aligned}$$

Note that

$$\begin{aligned} [t(u-f) + (1-t)(v-f)]^2 &= t^2(u-f)^2 + (1-t)^2(v-f)^2 + 2t(1-t)(u-f)(v-f) \\ &= t^2(u-f)^2 + (1-t)^2(v-f)^2 + t(1-t)[(u-f)^2 + (v-f)^2] \\ &= t(u-f)^2 + (1-t)(v-f)^2. \end{aligned}$$

So, we obtain

$$\begin{aligned} E[tu + (1-t)v] &\leq \int_\Omega t|\nabla u| + (1-t)|\nabla v| + \frac{\lambda}{2}[t(u-f)^2 + (1-t)(v-f)^2] d\boldsymbol{x} \\ &= t\int_\Omega \left(|\nabla u| + \frac{\lambda}{2}(u-f)^2 \right) d\boldsymbol{x} + (1-t)\int_\Omega \left(|\nabla v| + \frac{\lambda}{2}(v-f)^2 \right) d\boldsymbol{x} \\ &= tE[u] + (1-t)E[v]. \end{aligned}$$

This completes the proof. □

The uniqueness of minimizer can be ensured easily.

Lemma 2.1.2 (Uniqueness)**.**
If u_1 *and* u_2 *are two minimizers of ROF model, then we have* $u_1 = u_2$.

Proof. Let $E[u] = \int_\Omega \left(|\nabla u| + \frac{\lambda}{2}(u-f)^2 \right) d\boldsymbol{x}$. By the triangle inequality, we have

$$\begin{aligned}
E\left[\frac{u_1+u_2}{2}\right] &= \int_\Omega \left| \nabla \left(\frac{u_1+u_2}{2} \right) \right| + \frac{\lambda}{2} \left(\frac{u_1+u_2}{2} - f \right)^2 d\boldsymbol{x} \\
&\leq \int_\Omega \frac{1}{2}(|\nabla u_1| + |\nabla u_2|) + \frac{\lambda}{2} \left(\frac{u_1+u_2}{2} - f \right)^2 d\boldsymbol{x} \\
&= \int_\Omega \frac{1}{2}(|\nabla u_1| + |\nabla u_2|) + \frac{\lambda}{2} \left[\frac{(u_1-f)^2}{2} + \frac{(u_2-f)^2}{2} - \left(\frac{(u_1-f)-(u_2-f)}{2} \right)^2 \right] d\boldsymbol{x} \\
&= \frac{1}{2} \int_\Omega \left(|\nabla u_1| + \frac{\lambda}{2}(u_1-f)^2 \right) d\boldsymbol{x} + \frac{1}{2} \int_\Omega \left(|\nabla u_2| + \frac{\lambda}{2}(u_2-f)^2 \right) d\boldsymbol{x} \\
&\qquad - \frac{\lambda}{2} \int_\Omega \left(\frac{(u_1-f)-(u_2-f)}{2} \right)^2 d\boldsymbol{x} \\
&= \frac{1}{2}E[u_1] + \frac{1}{2}E[u_2] - \int_\Omega \left(\frac{(u_1-f)-(u_2-f)}{2} \right)^2 d\boldsymbol{x} \\
&\leq E[u_1] - \frac{\lambda}{2} \int_\Omega \left(\frac{u_1-u_2}{2} \right)^2 d\boldsymbol{x}.
\end{aligned}$$

If $u_1 \neq u_2$, then the above equation gives a contradiction that u_1 is not a minimizer. Therefore, $u_1 = u_2$. □

2.2 Discretization

- **ROF model:**

$$\min_{u \in BV(\Omega)} \left(\|u\|_{TV(\Omega)} + \frac{\lambda}{2} \int_\Omega (u-f)^2 d\boldsymbol{x} \right)$$

- **Discretization:**

$$\min_u \left(\sum_{i,j} \left| (\nabla u)_{i,j} \right| + \frac{\lambda}{2} \sum_{i,j} \left(u_{i,j} - f_{i,j} \right)^2 \right)$$

- **Constraint:**

$$\min_{d,u} \left(\sum_{i,j} |d_{i,j}| + \frac{\lambda}{2} \sum_{i,j} \left(u_{i,j} - f_{i,j} \right)^2 \right) \quad \text{subject to } d_{i,j} = \nabla u_{i,j}.$$

- **Bregman iteration:**

$$\min_{d,u} \left(\sum_{i,j} |d_{i,j}| + \frac{\lambda}{2} \sum_{i,j} \left(u_{i,j} - f_{i,j} \right)^2 + \frac{\gamma}{2} \sum_{i,j} \left| d_{i,j} - \nabla u_{i,j} - b_{i,j} \right|^2 \right)$$

2.2.1 Split Bregman iterative scheme

- ***u*-subproblem:**

With d fixed, we solve

$$u^{(k+1)} = \arg\min_u \left(\frac{\lambda}{2} \sum_{i,j} (u_{i,j} - f_{i,j})^2 + \frac{\gamma}{2} \sum_{i,j} \left| d_{i,j}^{(k)} - \nabla u_{i,j} - b_{i,j}^{(k)} \right|^2 \right).$$

Then consider the minimization problem

$$\min_u \int_\Omega \left(\frac{\lambda}{2}(u-f)^2 + \frac{\gamma}{2}|d - \nabla u - b|^2 d\boldsymbol{x} \right).$$

By the Euler-Lagrange equation, we have

$$\lambda(u-f) - \gamma\left[\nabla \cdot (\nabla u - d + b)\right] = 0,$$

or equivalently,

$$\lambda u - \gamma \Delta u = \lambda f - \gamma \nabla \cdot (d - b).$$

Note that

$$\begin{aligned} \Delta u_{i,j} &= (u_{i,j-1} + u_{i,j+1} - 2u_{i,j}) + (u_{i-1,j} + u_{i+1,j} - 2u_{i,j}) \\ &= u_{i-1,j} + u_{i,j-1} + u_{i,j+1} + u_{i+1,j} - 4u_{i,j}. \end{aligned}$$

So, we have

$$(\lambda + 4\gamma) u_{i,j} = c_{i,j} + \gamma \left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} \right),$$

where $c_{i,j} = (\lambda f - \gamma \nabla \cdot (d-b))_{i,j}$.

$$(\lambda + 4\gamma) u_{i,j} = c_{i,j} + \gamma \left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} \right),$$

which is a symmetric and strictly diagonally dominant linear system, by the Jacobi iterative method:

$$u_{i,j}^{(k+1)} = \left[c_{i,j}^{(k)} + \gamma \left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} \right) \right] / (\lambda + 4\gamma).$$

- ***d*-subproblem:**

With u fixed, we solve

$$d^{(k+1)} = \arg\min_d \left(\sum_{i,j} \left| d_{i,j} \right| + \frac{\gamma}{2} \sum_{i,j} \left| d_{i,j} - \nabla u_{i,j}^{(k+1)} - b_{i,j}^{(k)} \right|^2 \right).$$

Note that consider the simple 1-D case,

$$\begin{aligned} \arg\min_x \left(\tau|x| + \frac{\rho}{2}(x-y)^2 \right) &= \begin{cases} y - \tau/\rho, & y > \tau/\rho \\ 0, & |y| \le \tau/\rho \\ y + \tau/\rho, & y < -\tau/\rho \end{cases} \\ &= \frac{y}{|y|} \max\left\{ |y| - \tau/\rho, 0 \right\}. \end{aligned}$$

Then we have

$$d_{i,j}^{(k+1)} = \frac{\nabla u_{i,j}^{(k+1)} + b_{i,j}^{(k)}}{\left|\nabla u_{i,j}^{(k+1)} + b_{i,j}^{(k)}\right|} \max\left\{\left|\nabla u_{i,j}^{(k+1)} + b_{i,j}^{(k)}\right| - \frac{1}{\gamma}, 0\right\}.$$

- **Updating b:** $b_{i,j}^{(k+1)} = b^{(k)} + \nabla u^{(k+1)} - d^{(k+1)}$.
- **Algorithm:**

Input: f, λ, γ

Output: u

$u \leftarrow f, \quad b \leftarrow 0, \quad d \leftarrow 0$

while $\frac{\|u - u_{prev}\|}{\|u_{prev}\|} > tolerance$ **do**

  for $n = 1$ *to* $maxstep$ **do**

    Solve the ***u*-subproblem**

    Solve the ***d*-subproblem**

    $b \leftarrow b + \nabla u - d$

  end

end

2.3 Some indices for measuring image quality

Let $\widetilde{u}$ be the clean image, $\bar{u}$ be the mean intensity of the clean image, and u be the produced image. We introduce the following three indices for measuring image quality:

(1) **Mean square error:** $\mathrm{MSE} = \dfrac{1}{nm}\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{m}(\widetilde{u}_{i,j} - u_{i,j})^2$

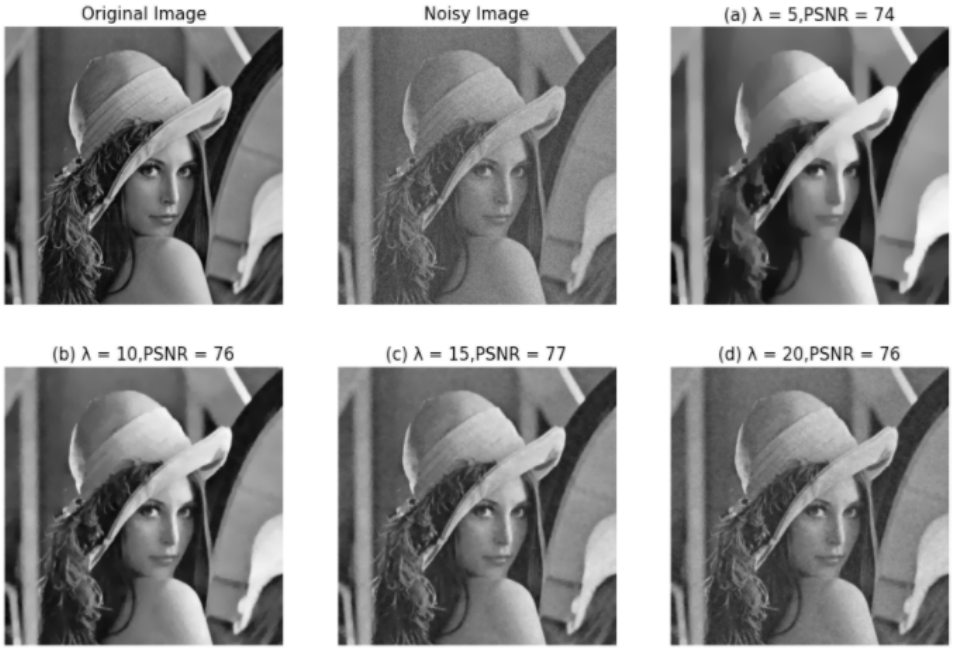
(2) **Peak signal to noise ratio:** $\mathrm{PSNR} = 10\log\left(\dfrac{255^2}{\mathrm{MSE}(\widetilde{u}, u)}\right)$

(3) **Signal to noise ratio:** $\mathrm{SNR} = 10\log\left(\dfrac{\mathrm{MSE}(\widetilde{u}, \bar{u})}{\mathrm{MSE}(\widetilde{u}, u)}\right)$

2.4 Numerical experiments

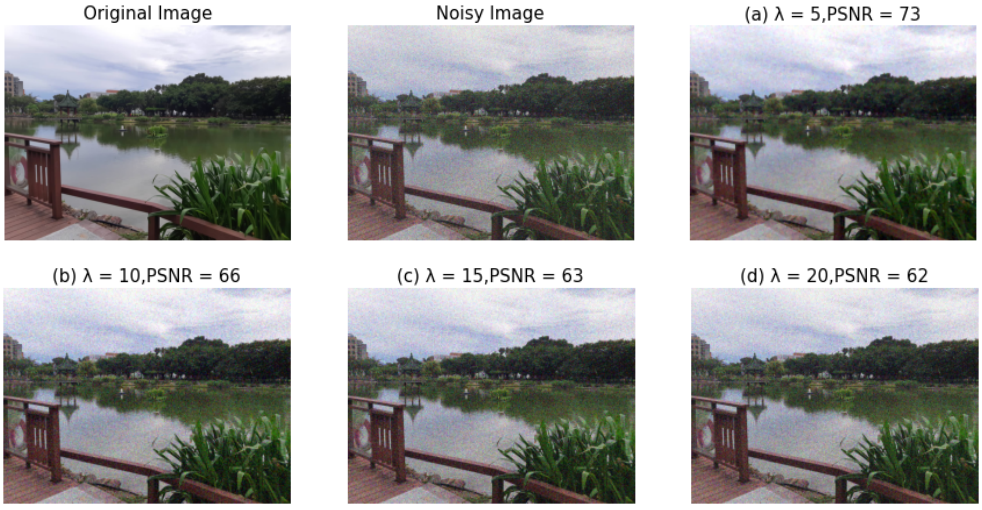
Below, we report two numerical examples using the ROF variational model for image denoising.

2.4.1 Grayscale image



Numerical results of ROF model on gray scale image, where the noisy level $\sigma = 10$ and the denoising parameter which gives the best PSNR is $\lambda = 15$. (a) $\lambda = 5$, $PSNR = 74$, (b)$\lambda = 10$, $PSNR = 76$, (c) $\lambda = 15$, $PSNR = 77$, (d) $\lambda = 20$, $PSNR = 76$.

2.4.2 Color image



Numerical results of ROF model on color image, where the noisy level $\sigma = 10$ and the denoising parameter which gives the best PSNR is $\lambda = 5$. (a) $\lambda = 5$, $PSNR = 73$, (b) $\lambda = 10$, $PSNR = 66$, (c) $\lambda = 15$, $PSNR = 63$, (d) $\lambda = 20$, $PSNR = 62$. Noted that for a color image ,we apply the ROF model among all three channels r,g,b with the same denoising parameter.

Chapter 3

Image contrast enhancement

3.1 Histogram equalization (HE)

Histogram equalization is a technique for adjusting image intensities to enhance contrast. It changes the histogram of an image to one that possesses a uniform distribution using as a transformation function the accumulative histogram of the image.

.

Let f be a given image on Ω. Define the normalized histogram of f with a bin for each possible intensity

$$p(k) = \frac{\text{number of pixels with intensity } k}{\text{total number of pixels}},$$

where $k = 0, 1, \cdots, 255$. The histogram equalized image u will be defined by

$$u(\boldsymbol{x}) = \left[255 \times \sum_{k=0}^{f(x)} p(k) \right],$$

where $\boldsymbol{x} \in \Omega$ and $[\cdot]$ is represented as floor.

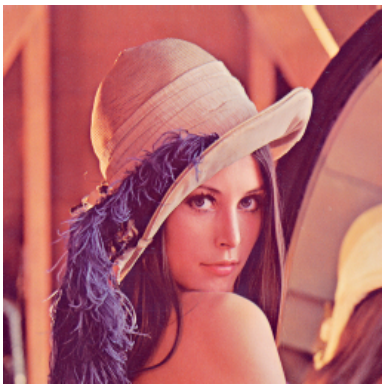
Here are some histogram equalization results:

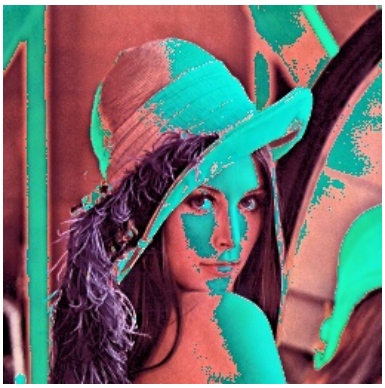
- Grayscale image:

Origin image

HE's image

- Color image:


Origin image


HE's image

Note that the quality of color image becomes really bad after HE processing. So, we must try other methods.

3.2 Variation model

Morel-Petro-Sbert model (IPOL 2014).
Let $f : \overline{\Omega} \to \mathbb{R}$ be a given grayscale image. The Morel-Petro-Sbert proposed the model for image contrast enhancement:

$$\min\left(\frac{1}{2}\int_\Omega |\nabla u - \nabla f|^2\, d\boldsymbol{x} + \frac{\lambda}{2}\int_\Omega (u-\overline{u})^2\, d\boldsymbol{x}\right),$$

where $\overline{u} = \frac{1}{|\Omega|}\int_\Omega u\, dx$ is the mean value of u over Ω and $\lambda > 0$ balances between detail preservation and variance reduction.

The data fidelity term preserves image details presented in f and the regularizer reduces the variance of u to eliminate the effect of nonuniform illumination. The original model is simple but difficult to solve due to the $\overline{u}$ term. So, they improved the model as follows.

Petro-Sbert-Morel model (MAA 2014).
Petro-Sbert-Morel further improved their model by using the L^1 norm to obtain sharper edges:

$$\min_u\left(\int_\Omega |\nabla u - \nabla f|\, d\boldsymbol{x} + \frac{\lambda}{2}\int_\Omega (u-\overline{f})^2\, d\boldsymbol{x}\right).$$

Requiring the desired image u being close to a pixel-independent constant $\overline{f}$ highly contradicts the requirement of ∇u being close to ∇f and restrains the parameter λ to be very small.

We consider another improvement. First, we define

$$\Omega_d = \{\boldsymbol{x} \in \bar{\Omega} : f(\boldsymbol{x}) \leq \bar{f}\} \quad \text{and} \quad \Omega_b = \{\boldsymbol{x} \in \bar{\Omega} : f(\boldsymbol{x}) > \bar{f}\}$$

as the dark part and the bright part of the image Ω. Second, we define the adaptive functions

$$g(\boldsymbol{x}) = \begin{cases} \alpha\bar{f}, & \boldsymbol{x} \in \Omega_d \\ f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega_b \end{cases} \quad \text{and} \quad h(\boldsymbol{x}) = \begin{cases} \beta f(x), & \boldsymbol{x} \in \Omega_d \\ f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega_b \end{cases},$$

where $\alpha > 0$ and $\beta > 1$.

Hsieh-Shao-Yang model (SIIMS 2020).
Hsieh-Shao-Yang proposed two adaptive functions g and h to replace $\overline{f}$ and the original input image f,

$$\min_u \left(\int_\Omega |\nabla u - \nabla h| d\boldsymbol{x} + \frac{\lambda}{2} \int_\Omega (u-g)^2 d\boldsymbol{x} + \chi_{[0,255]}(u) \right),$$

where the characteristic function is defined as

$$\chi_{[0,255]}(u) = \begin{cases} 0, & \text{range}(u) \subseteq [0,255], \\ \infty, & \text{otherwise.} \end{cases}$$

3.3 Discretization

- **Model:**

$$\min_u \left(\int_\Omega |\nabla u - \nabla h| d\boldsymbol{x} + \frac{\lambda}{2} \int_\Omega (u-g)^2 d\boldsymbol{x} + \chi_{[0,255]}(u) \right)$$

- **Discretization:**

$$\min_u \sum_{i,j} \left(\left| (\nabla u)_{i,j} - (\nabla h)_{i,j} \right| + \frac{\lambda}{2} \left(u_{i,j} - g_{i,j} \right)^2 \right) + \chi_{[0,255]}(u)$$

- **Constraint:**

$$\min_u \sum_{i,j} \left(\left| d_{i,j} \right| + \frac{\lambda}{2} \left(u_{i,j} - g_{i,j} \right)^2 \right) + \chi_{[0,255]}(v) \quad \text{subject to } d = \nabla u - \nabla h \text{ and } v = u.$$

- **Bregman iteration:**

$$\min_{u,d,v} \sum_{i,j} \left(\left|d_{i,j}\right| + \frac{\lambda}{2} \left(u_{i,j} - g_{i,j}\right)^2 + \frac{\gamma}{2} \left|d_{i,j} - (\nabla u)_{i,j} + (\nabla h)_{i,j} - b_{i,j}\right|^2 + \frac{\delta}{2} \left(v_{i,j} - u_{i,j} - c_{i,j}\right)^2 \right) + \chi_{[0,255]}(v)$$

3.3.1 Split Bregman iterative scheme

- ***u*-subproblem:**

  With d and v fixed, we solve

$$u^{(k+1)} = \arg\min_{u} \sum_{i,j} \left(\frac{\lambda}{2} \left(u_{i,j} - g_{i,j}\right)^2 + \frac{\gamma}{2} \left| d_{i,j}^{(k)} - (\nabla u)_{i,j} + (\nabla h)_{i,j} - b_{i,j}^{(k)} \right|^2 + \frac{\delta}{2} \left(v_{i,j}^{(k)} - u_{i,j} - c_{i,j}^{(k)} \right)^2 \right).$$

  Then consider the minimization problem

$$\min_{u} \int_{\Omega} \left(\frac{\lambda}{2}(u-g)^2 + \frac{\gamma}{2}|d - \nabla u + \nabla h - b|^2 d\boldsymbol{x} + \frac{\delta}{2} \left(v - u - c\right)^2 \right).$$

  By the Euler-Lagrange equation, we have

$$\lambda(u-g) - \delta(v-u-c) - \gamma \left[\nabla \cdot (\nabla u - d - \nabla h + b)\right] = 0,$$

  or equivalently,

$$(\lambda + \delta)u - \gamma \Delta u = \lambda g - \gamma \nabla \cdot (d + \nabla h - b) + \delta(v - c).$$

  Note that

$$\begin{aligned} \Delta u_{i,j} &= (u_{i,j-1} + u_{i,j+1} - 2u_{i,j}) + (u_{i-1,j} + u_{i+1,j} - 2u_{i,j}) \\ &= u_{i-1,j} + u_{i,j-1} + u_{i,j+1} + u_{i+1,j} - 4u_{i,j}. \end{aligned}$$

  So, we have

$$(\lambda + \delta + 4\gamma)u_{i,j} = c_{i,j} + \gamma \left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right),$$

  where $c_{i,j} = (\lambda g - \gamma \nabla \cdot (d + \nabla h - b))_{i,j} + \delta(v - c)$, which is a symmetric and strictly diagonally dominant linear system, by the Jacobi iterative method, we have

$$u_{i,j}^{(k+1)} = \left[c_{i,j}^{(k)} + \gamma \left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} \right) \right] / (\lambda + \delta + 4\gamma).$$

- ***d*-subproblem:**

  With u and v fixed, we solve

$$d^{(k+1)} = \arg\min_{d} \left(\sum_{i,j} \left| d_{i,j} \right| + \frac{\gamma}{2} \sum_{i,j} \left| d_{i,j} - \nabla u_{i,j}^{(k+1)} + (\nabla h)_{i,j} - b_{i,j}^{(k)} \right|^2 \right).$$

  Note that consider the simple 1-D case,

$$\arg\min_{x} \left(\tau |x| + \frac{\rho}{2}(x-y)^2 \right) = \frac{y}{|y|} \max\left\{ |y| - \tau/\rho, 0 \right\}.$$

  Then we have

$$d_{i,j}^{(k+1)} = \frac{\nabla u_{i,j}^{(k+1)} - (\nabla h)_{i,j} + b_{i,j}^{(k)}}{\left| \nabla u_{i,j}^{(k+1)} - (\nabla h)_{i,j} + b_{i,j}^{(k)} \right|} \max\left\{ \left| \nabla u_{i,j}^{(k+1)} - (\nabla h)_{i,j} + b_{i,j}^{(k)} \right| - \frac{1}{\gamma}, 0 \right\}.$$

- ***v*-subproblem:**

  With u and d fixed, we solve

$$v^{(k+1)} = \arg\min_{v} \left(\sum_{i,j} \frac{\delta}{2} \left(v_{i,j} - u_{i,j}^{(k+1)} - c_{i,j}^{(k)} \right)^2 \right) + \chi_{[0,255]}(v).$$

  Then we obtain

$$v_{i,j} = \min\{\max\{u_{i,j} + c_{i,j}, 0\}, 255\}.$$

- **Updating *b*:** $b^{(k+1)} = b^{(k)} + \nabla u^{(k+1)} - \nabla h - d^{(k+1)}$.
- **Updating *c*:** $c^{(k+1)} = c^{(k)} + u^{(k+1)} - v^{(k+1)}$.
- **Algorithm:**

Input: $f, \lambda, \gamma, \delta$

Output: u

$u \leftarrow h, \quad v \leftarrow h, \quad b \leftarrow 0, \quad c \leftarrow 0, \quad d \leftarrow 0$

while $\frac{\|u - u_{prev}\|}{\|u_{prev}\|} > tolerance$ **do**

  for $n = 1$ *to* $maxstep$ **do**

    Solve the ***u*-subproblem**

    Solve the ***d*-subproblem**

    Solve the ***v*-subproblem**

    $b \leftarrow b + \nabla u - \nabla h - d$

    $c \leftarrow c + u - v$

  end

end

3.4 Color RGB images

The domain division for color RGB images denoted by (f_R, f_G, f_B) is conducted as follows. First, we define the maximum image as

$$f_{\max}(\boldsymbol{x}) = \max\{f_R(\boldsymbol{x}), f_G(\boldsymbol{x}), f_B(\boldsymbol{x})\}, \forall \boldsymbol{x} \in \overline{\Omega}.$$

For example,

65	27	100
22	31	47
112	54	78

58	21	10
145	213	48
132	2	9

15	122	200
189	32	45
12	52	79

↓

65	122	200
189	213	48
132	54	79

Let $\overline{f}_{\max} = \frac{1}{|\Omega|}\int_\Omega f_{\max}\, d\boldsymbol{x}$. Then we divide the image domain Ω into two parts:

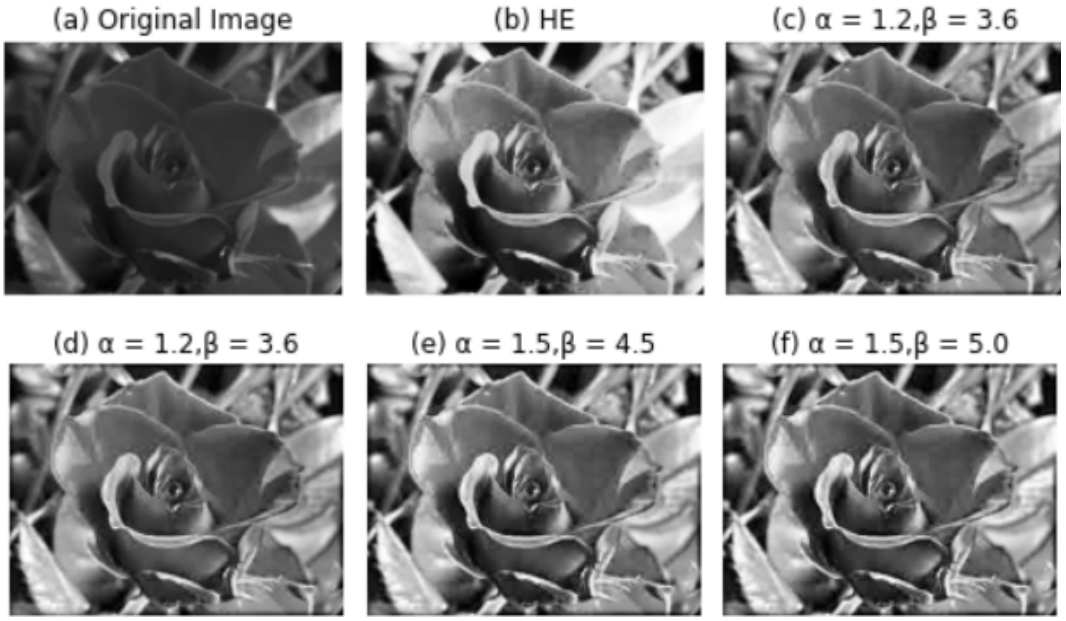
$$\Omega_d = \{\boldsymbol{x} \in \overline{\Omega} : f_{\max}(\boldsymbol{x}) \leq \overline{f}_{\max}\},$$
$$\Omega_b = \{\boldsymbol{x} \in \overline{\Omega} : f_{\max}(\boldsymbol{x}) > \overline{f}_{\max}\}.$$


Blue part represents Ω_d and yellow part represents Ω_b

3.5 Numerical experiments

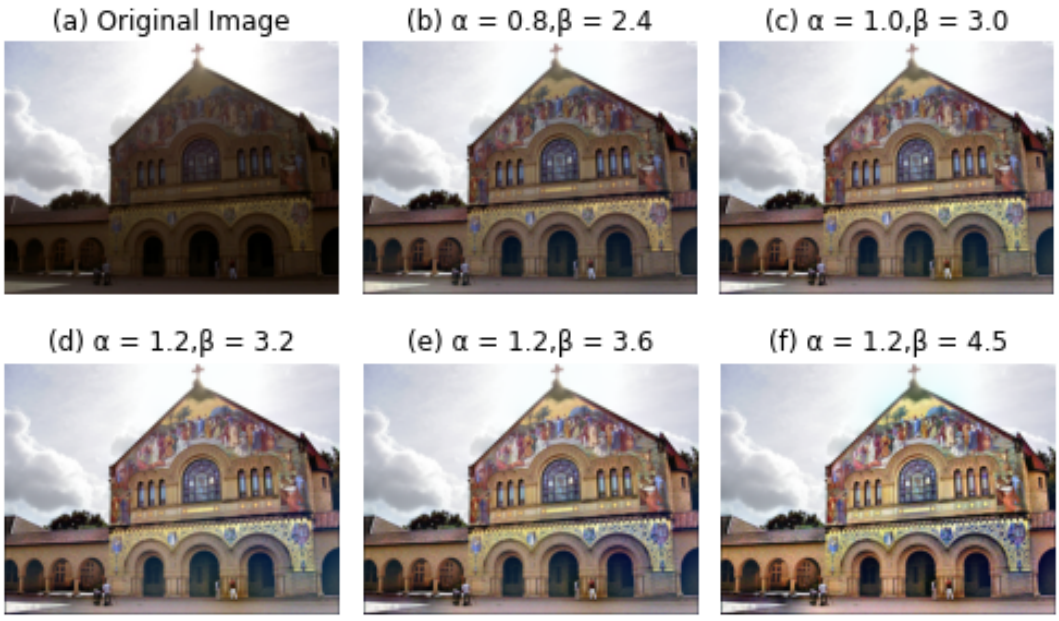
In this section, we report two numerical examples.

3.5.1 Grayscale image



Numerical results of the Hsieh-Shao-Yang model for gray scale image: (a) original image, (b) Histogram equilization, (c)-(f) minimizers of the model with various α and β, where $\lambda = 0.0005$.

3.5.2 Color image



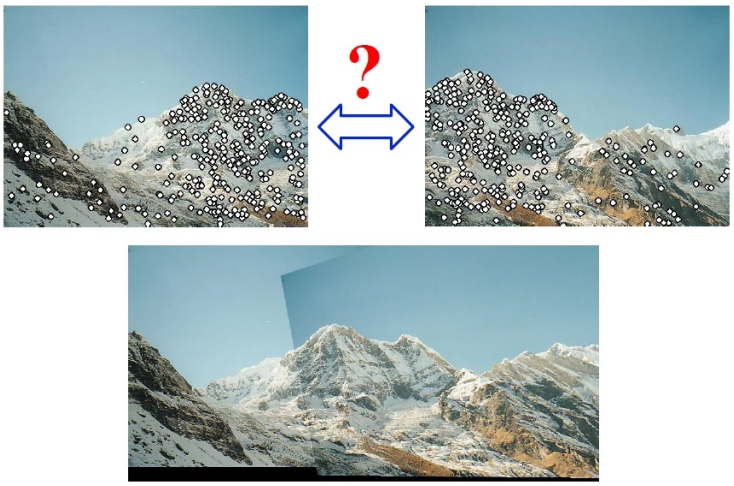
Numerical results of the Hsieh-Shao-Yang model for color image: (a) original low-light image, (b)-(f) images with various model parameter α and β, we gradually increase α and β to enhance the luminance of the shadow area in (a) and make the hidden details clearer. And it is recommended in [4] that in general the model has good performance by taking $\beta = 3\alpha$.

Chapter 4

Image stitching

Image stitching consists of the following two parts:

- Image alignment
- Image blending



4.1 Image alignment

In order to find image features, we employ the famous feature detection algorithm, the scale-invariant feature transform (SIFT) (cf. [1, 5, 8, 9]).

4.1.1 Scale-invariant feature transform (SIFT)

David Lowe first proposed the SIFT algorithm. SIFT's main purpose is to extract feature points from an image, which can also be said to be local feature blocks of the image and have the scale, rotation, luminosity invariance, and stability to viewing angle changes and noise.

The SIFT algorithm can be divided into four major steps. The first one is the scale space extreme value detection. The second is the positioning of key points. The direction is determined in the third step, and the final step is the description of key points. Before starting to describe these steps, let us introduce the Gaussian pyramid.

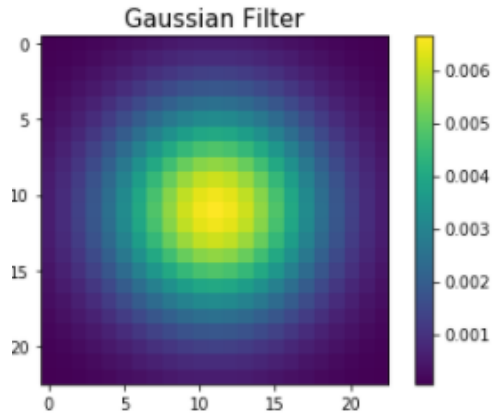
Gaussian blur is a filter and an indispensable tool for making blurred images. We use a two-dimensional normal distribution function to create a Gaussian blur template, which will be used to convolve the image to achieve the effect of blurring the image. The following is the formula of the Gaussian kernel of dimension N:

$$G(r) = \frac{1}{\sqrt{2\pi\sigma^2}^N} e^{-\frac{r^2}{2\sigma^2}},$$

where σ is the standard deviation of the normal distribution, so the larger the value, the more blurred the image; r is the blur radius, which is the distance from the template element to the center of the template. As a result, a two-dimensional of size $m \times n$ with the element being (x, y) corresponding to the value would be

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(\frac{x-m}{2})^2 + (\frac{y-n}{2})^2}{2\sigma^2}}.$$

To put it simply, the points farther from the center are assigned less weights, while points closer to the center are assigned more weights, as shown in the figure below. In the discrete approximation of the function, distance farther than 3σ can be regarded as having no effect. Therefore, we only use a template of $(6\sigma + 1) \times (6\sigma + 1)$ when dealing with images, which is enough.



After briefly introducing the Gaussian blur, let's enter the part of the Gaussian pyramid. First, let's explain intuitively why we need to use the Gaussian pyramid. The Gaussian pyramid is a pyramid stacked by images of different sizes and different degrees of blur. Since the blurring method is obtained by convolution with a Gaussian

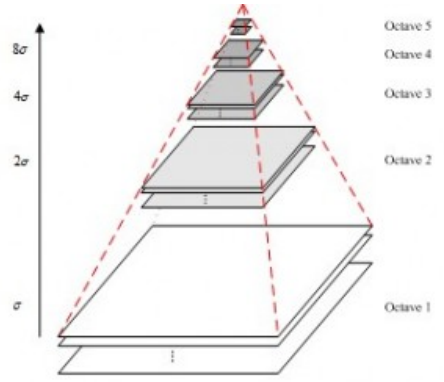
blur template, we call it a Gaussian pyramid. And since the Gaussian pyramid collects images of different scales and different degrees of blur, we can find some salient points through the differences between them. These points may be the feature point candidates. That is to say, the Gaussian pyramid allows us to look at the feature points roughly, and at the same time carefully, which will enable us to find the points with scale invariance regardless of the scale of our original image.

Let $L(x, y, \sigma)$ be the convolution of a Gaussian function $G(x, y, \sigma)$ with different scale and the original image $I(x, y)$. That is,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

where * means convolution, $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(\frac{x-m}{2})^2+(\frac{y-n}{2})^2}{2\sigma^2}}$, $m = n = 6\sigma + 1$, (x, y) is the pixel position, and σ is the scale factor. The larger the σ, the more blurred the image, which is the larger scale; the smaller the σ, the clearer the image, that is, the smaller scale. We use the Gaussian pyramid to realize the scale-space (it can be imagined as a container with pictures of different scales), which requires two steps: first, do Gaussian blur of different scales on the image. Second, down-sampling the picture.

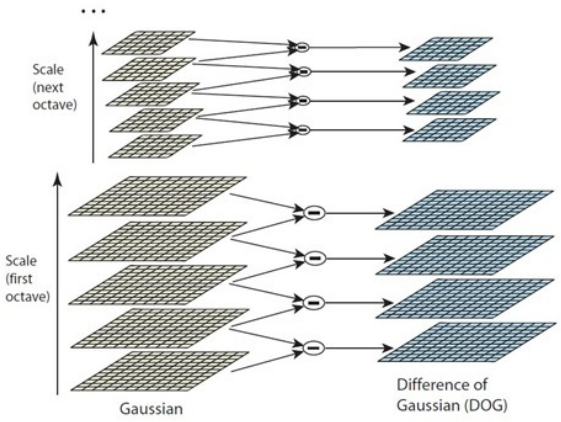
The main body of the Gaussian pyramid is composed of "octave". It can be seen as layers of a pyramid; note that each octave is composed of many images, which we call "interval". The difference between group and group (octave) is the image size; the difference between layer and layer (interval) is the scale (degree of blur). For example, the size of the first octave of images in the pyramid is $M \times N$, but each interval in this octave is obtained by convolution of Gaussian blur templates with different σ. To be precise, if the first interval uses σ, then the second interval uses kσ , the third interval uses $k^2\sigma$, and so on. The first-interval image of the second octave is extracted from the third interval counted back from the image of the first octave, and then subjected it to $\frac{1}{2}$ down-sampling. Therefore, the size of the first interval in the second octave is $\frac{M}{2} \times \frac{N}{2}$. The rest follows the same rule, and we successfully constructed a Gaussian pyramid.



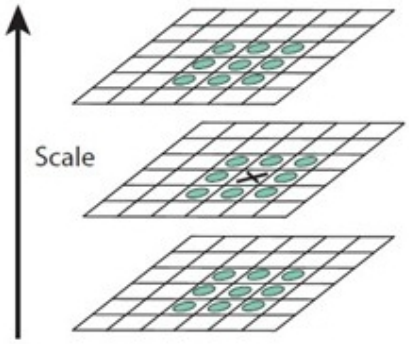
Next, let's begin the first step of SIFT.

To find the image's feature points, we first subtracted the images of the same octave of Gaussian pyramids to obtain another pyramid, which we call the "differential of Gaussian pyramid" or simply DOG. It can be imagined that the information left in the DOG is almost the object's contour, or we call it the feature. To find out the key point, we need to find the extreme value because the feature points of an image are particularly different from the surroundings. After subtraction, these extreme points can be more revealed. Note that DOG can also be presented in the form

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma). \end{aligned}$$



What's more remarkable is that when we look for extreme points in the difference pyramid, we will use the information at different intervals in the same octave. In other words, the upper and lower layers will also be considered, that is, 26 points surrounding the candidate, as shown in the figure below:



After finding these extreme points, we have completed the first step. Note that this is just the candidate for feature points. We still have to eliminate some points.

A poorly defined differential Gaussian operator will make the edge stand out, so we must eliminate some edge effects. The characteristic of these edges is that there is high principal curvature in the horizontal direction, but almost none in the vertical direction, and vice versa. For example, assume there is a vertical line, which changes drastically in the horizontal (x-axis) direction, but hardly changes in the vertical (y-axis) direction. In the situation, we used the Hessian matrix as a medium to solve this problem. The Hessian matrix looks like:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix},$$

where α, β be the eigenvalues of H, which are also the gradient of x and y. Therefore, we have

$$\begin{aligned} \mathrm{tr}(H) &= D_{xx} + D_{yy} = \alpha + \beta, \text{ and} \\ \det(H) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \end{aligned}$$

Without loss of generality, assume $\alpha = \gamma\beta$ and $\gamma \geq 1$, we have

$$\frac{\mathrm{tr}(H)^2}{\det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(\gamma\beta+\beta)^2}{\gamma\beta^2} = \frac{(\gamma+1)^2}{\gamma}.$$

Note that if γ is large, then the gradient is much different, which is the point we want to eliminate. Therefore, we can use the certain threshold to eliminate some points, as follows:

$$\frac{\mathrm{tr}(H)^2}{\det(H)} \geq \frac{(\gamma+1)^2}{\gamma}.$$

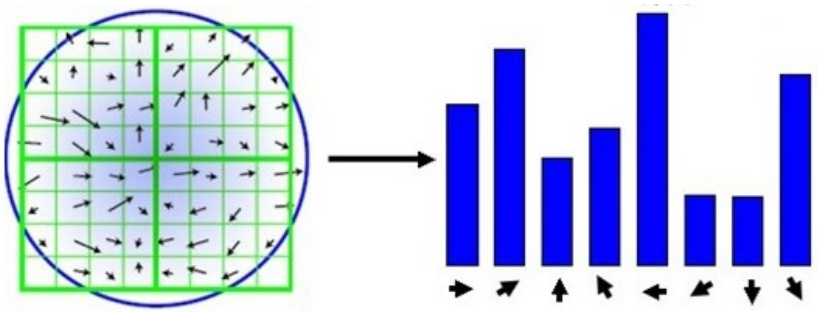
And it is suggested that the value of γ be 10.

Now we can execute the third step of SIFT. This step aims to turn the whole image into a specific direction corresponding to the key point. So when facing two images in different directions, we can find the same key point and keep them in the same direction. This step guarantee rotation invariance. Moreover, it is very helpful to describe the key points in the next step.

First, we define the direction and magnitude of a pixel (x, y).

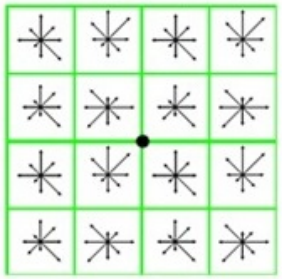
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2},$$
$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right).$$

Use weight and value to calculate the information, and then use the histogram to accumulate the data. We divide the directions into thirty-six groups: one group for every ten degrees and the highest -value direction as the main direction.



Next, we make the main direction of the key point be the x-axis, and the other surrounding points also rotates relative to the key point, so that rotation invariance can be achieved.

In the last step of SIFT, we want to describe the key points. In other words, we only need to know how to express this key point uniquely. The neighborhood of the key point is divided into 16 sub-regions. Each sub-region is a seed point. Let each subpoint have eight directions (and their magnitudes). Simply calculating, a key point will be described by a 16× 8=128-dimensional vector. The eight directions are obtained by weighting the information around each seed point, and we won't tell in more detail here.



At present, we have successfully found the key points and the corresponding descriptors and explained why SIFT could guarantee scale invariance and rotation invariance. Now we can normalize the 128-dimensional descriptor. Then eliminate the influence caused by the difference in luminosity, and truly complete the SIFT, finding out key points and give each of them a descriptor.

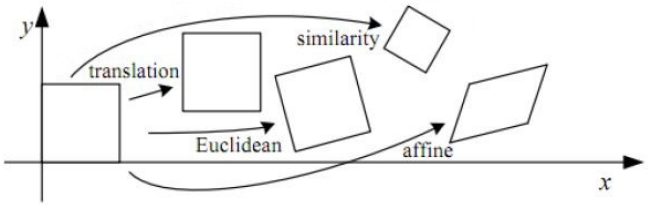
We will use SIFT at image stitching, putting two graphics together. Because SIFT has scale invariance, rotation invariance, and luminosity normalization, the overlap area of two images can be found using the same key points and descriptors. The closer the descriptors are, the more likely they are a pair of key points.

4.1.2 Homography

To fit the features, we must use a transformation to match all the features.

Definition 4.1.1 (Affine transformation)**.**
A 2D affine transformation is composed of a linear transformation by $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbb{R}^{2\times 2}$ *and a translation by a vector* $\begin{bmatrix} e \\ f \end{bmatrix} \in \mathbb{R}^2$*, given by*

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}.$$



We convert $\begin{bmatrix} x \\ y \end{bmatrix}$ to the homogeneous image coordinates $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

Remark (Homogeneous expression of 2D affine transformation)**.**
The homogeneous expression of the affine transformation is given by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Remark.
The affine transformation has 6 degree of freedom.

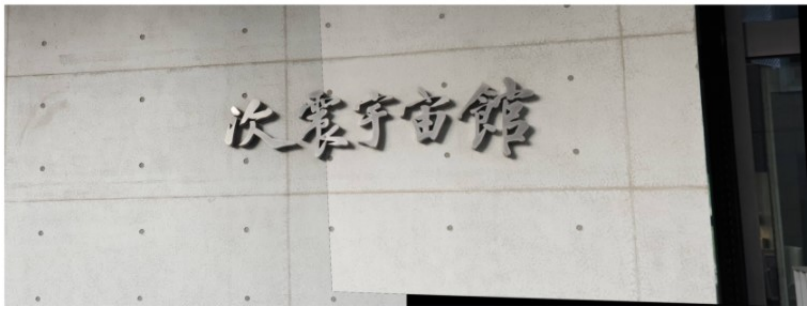
Definition 4.1.2 (Homography)**.**
The homogeneous expression of the homography is given by

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Remark.
The homogeneous expression of the homography has 8 degree of freedom (9 parameters, but scale is arbitrary).

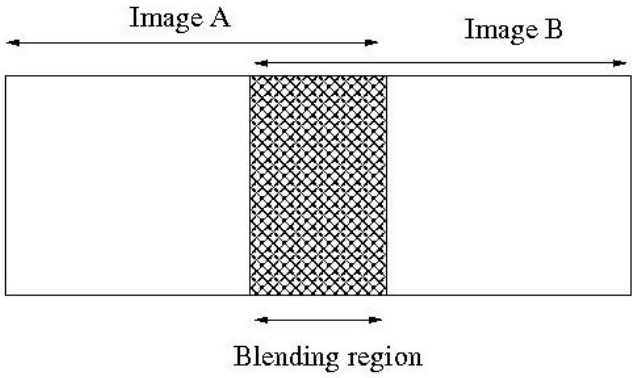
4.2 Image blending

After we aligned the images, there is still one step we need to do. When different images are stitched together, for various reasons (e.g., different lighting conditions of images, etc.), the adjacent pixel intensities may differ enough to produce artifacts (e.g., seam) shown in the following image.



The procedure to remove artifacts is image blending. There are many ways to perform blending. Here, we use a simple approach called linear blending to remove the seam between the overlapping images. Sometimes the simple approach may not work, but for this example, images were taken at the same time, so the simple algorithm gave rise to good results.

We applied the formula below on the overlapping region (blending region).

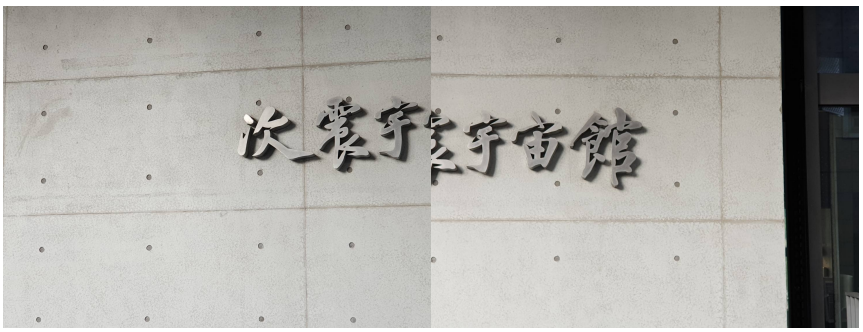


$$\text{Pano}(i,j) = (1 - w(i,j)) * \text{Image A}(i,j) + w(i,j) * \text{Image B}(i,j),$$

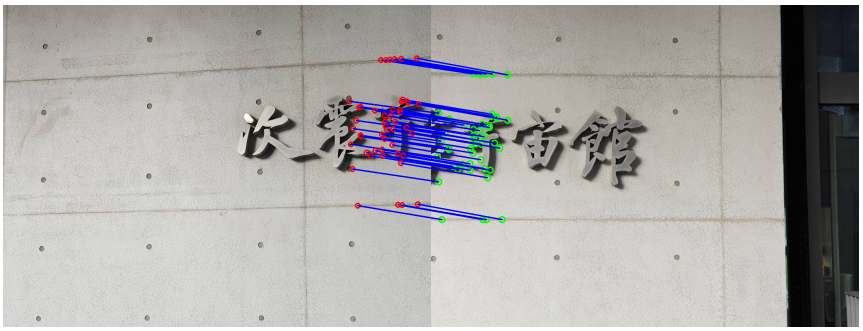
where $\text{Pano}(i,j)$ is the i,j^{th} pixel of the panoramic image, Image $\text{A}(i,j)$ is the i,j^{th} pixel of Image A, Image $\text{B}(i,j)$ is the i,j^{th} pixel in the panoramic image corresponds to which in Image B after transformed by homography, and the closer the weight w is to Image B, the large w is, $0 \leq w \leq 1$.

4.3 Numerical experiments

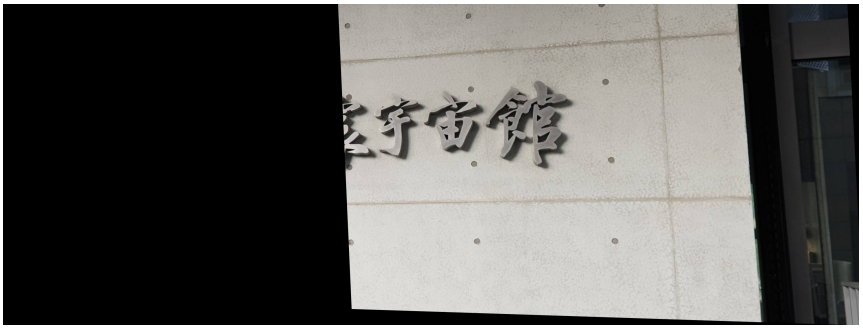
(a) Image1 (left) and Image2 (right) we want to stitch together



(b) Matches of the feature points in the image1 and image2



(c) Image2 after being transform by a homography



(d) Images aligned according to a homography

(a) is the picture that we stacked two images we want to stitch together left and right. The first step of image stitching is finding the pictures' feature points and each feature point's descriptors individually. Then we can match the feature points by calculating the distance between descriptors. In (b), we visualize the matches by connecting them with green lines and label the

feature points out. After matching the feature points, one can solve the homography we mentioned above, which maps the feature points from one picture to another. (c) is the right picture after being transformed by the homography we found in the previous step. (d) Images aligned according to a tomography. We applied linear blending on the overlapping region of twp images to make it join up smoothly.

Figure: Drunken moon lake at NTU

The numerical experiment of image stitching and contrast enhancement: (a) We use linear blending strategy to yield a seamless panorama. But, still, we notice the same changes in illumination. (b) We use the contrast enhancement strategy mentioned in Section3 to eliminate the changes in illumination.

Chapter 5

Conclusion and future works

In this research program, we have learned some topics for mathematical image processing, including the bounded variation function space BV, the basics of calculus of variation and the Euler-Lagrange equation, the Rudin-Osher-Fatemi model for image denoising, an adaptive contrast enhancement model, some knowledge for image alignment, and the split Bregman iterative scheme for solving the associated minimization problems. We have also studied the famous feature detection algorithm, called the scale-invariant feature transform (SIFT). Moreover, we have successfully realized some image stitching problems using Matlab and Python.

We find that some issues deserve to be further studied, including how to find a more simple algorithm for image feature matching and how to find a more efficient energy functional to achieve image blending.

Finally, we provide some Python codes on the Github for the interested readers:

(1) Image denoising: https://github.com/SeanChenTaipei/ImageProcessing/blob/master/Adaptive-Model.ipynb

(2) Contrast enhancement: https://github.com/SeanChenTaipei/ImageProcessing/blob/master/Contrast_Enhencement.ipynb

(3) Image stitching: https://github.com/SeanChenTaipei/ImageProcessing/blob/master/Image_Stitching.ipynb

Bibliography

[1] M. Brown and D. G. Lowe, Recognising panoramas, *Proceedings of Ninth IEEE International Conference on Computer Vision*, Nice, France, 2 (2003), pp. 1218-1225. doi: 10.1109/ICCV.2003.1238630.

[2] T. Goldstein and S. Osher, The split Bregman method for L1-regularized problems, *SIAM Journal on Imaging Sciences*, 2 (2009), pp. 323-343.

[3] P.-W. Hsieh, P.-C. Shao, and S.-Y. Yang, A regularization model with adaptive diffusivity for variational image denoising, *Signal Processing*, 149 (2018), pp. 214-228.

[4] P.-W. Hsieh, P.-C. Shao, and S.-Y. Yang, Adaptive variational model for contrast enhancement of low-light images, *SIAM Journal on Imaging Sciences*, 13 (2020), pp. 1-28.

[5] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, 60 (2004), pp. 91-110.

[6] L. I. Rudin, S. Osher, and E. Fatemi, Nonlinear total variation based noise removal algorithms, *Physica D*, 60 (1992), pp. 259-268.

[7] S.-Y. Yang, Lecture notes at http://www.math.ncu.edu.tw/~syyang/research/2020NCTS_USRP.pdf, August 2020.

[8] https://blog.csdn.net/dcrmg/article/details/52561656.

[9] https://blog.csdn.net/zddmail/article/details/7521424.