**Q1. TOY PROBLEM**

**Code**

|  |
| --- |
| % Read Image  img = imread('data/toy\_problem.png');  [height, width, chan] = size(img);  img = double(img);    % height and width of gradiant image  grad\_h = height - 1;  grad\_w = width - 1;    im2var = zeros(height, width, 'uint32');  im2var(1:height\*width) = 1:height\*width;  im2grad = zeros(grad\_h, grad\_w, 'uint32');  im2grad(1:grad\_h\*grad\_w) = 1:grad\_h\*grad\_w;      A = zeros(grad\_h\*grad\_w + 1, height\*width);  b = zeros(grad\_h\*grad\_w + 1, 1);  for y = 1 : height - 1  for x = 1 : width - 1  A(im2grad(y, x), im2var(y+1, x)) = -1;  A(im2grad(y, x), im2var(y, x+1)) = -1;  A(im2grad(y, x), im2var(y, x)) = 2;  b(im2grad(y, x)) = 2 \* img(y, x, 1) - ...  img(y, x+1, 1) - img(y+1, x, 1);  end  end  % image intensity of top left corner  A(grad\_h\*grad\_w + 1, im2var(1, 1)) = 1;  b(grad\_h\*grad\_w + 1) = img(1,1,1);    v = A \ b;    % Write out the result  img\_out = reshape(v, height, width, chan);  imwrite(img\_out / 256., 'data/toy\_problem\_res.png'); |

**Result**

****

**Description**

Firstly, I will treat x and y axis at once. Therefore, the kernel of image filtering is .

I made a gradient image of source(“toy\_problem.png”) and matrix looks like

. The last row is for the third equation, color of the top left corner pixel.

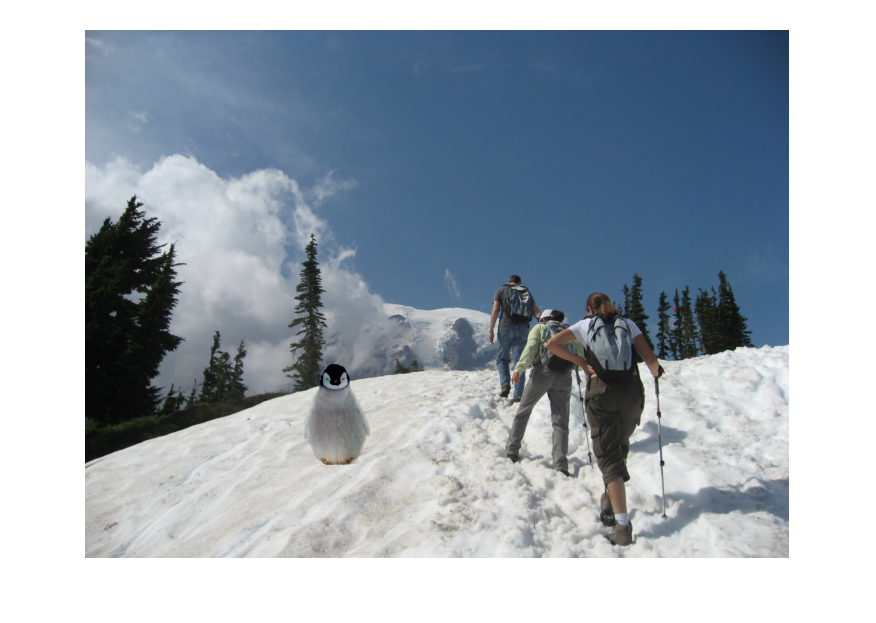
And then calculate to minimize , where A is the matrix and b is the gradient image.

**Q2. POISSON BLENDING**

**Code**

|  |
| --- |
| % Read Image  img\_back = imresize(imread('data/hiking.jpg'), 0.25, 'bilinear');  img\_obj = imresize(imread('data/penguin-chick.jpeg'), 0.25, 'bilinear');    % Get Align mask  objmask = getMask(img\_obj);  [img\_s, mask\_s] = alignSource(img\_obj, objmask, img\_back);    img\_back = double(img\_back);  img\_obj = double(img\_obj);  img\_source = img\_back(326:450, 201:301, :);  imshow(img\_source / 256.);    [height, width, chan] = size(img\_obj);  grad\_h = height - 2;  grad\_w = width - 2;    % Make matrix  im2var = zeros(height, width, 'uint32');  im2var(1:height\*width) = 1:height\*width;  im2grad = zeros(grad\_h, grad\_w, 'uint32');  im2grad(1:grad\_h\*grad\_w) = 1:grad\_h\*grad\_w;  A = zeros(grad\_h\*grad\_w, height\*width);  b\_r = zeros(grad\_h\*grad\_w, 1);  b\_g = zeros(grad\_h\*grad\_w, 1);  b\_b = zeros(grad\_h\*grad\_w, 1);  for y = 2:height-1  for x = 2:width-1  if objmask(y, x)  A(im2grad(y-1, x-1), im2var(y, x)) = 4;  A(im2grad(y-1, x-1), im2var(y-1, x)) = -1;  A(im2grad(y-1, x-1), im2var(y, x-1)) = -1;  A(im2grad(y-1, x-1), im2var(y+1, x)) = -1;  A(im2grad(y-1, x-1), im2var(y, x+1)) = -1;  b\_r(im2grad(y-1, x-1)) = img\_obj(y, x, 1) \* 4 - img\_obj(y-1, x, 1) - ...  img\_obj(y, x-1, 1) - img\_obj(y+1, x, 1) - img\_obj(y, x+1, 1);  b\_g(im2grad(y-1, x-1)) = img\_obj(y, x, 2) \* 4 - img\_obj(y-1, x, 2) - ...  img\_obj(y, x-1, 2) - img\_obj(y+1, x, 2) - img\_obj(y, x+1, 2);  b\_b(im2grad(y-1, x-1)) = img\_obj(y, x, 3) \* 4 - img\_obj(y-1, x, 3) - ...  img\_obj(y, x-1, 3) - img\_obj(y+1, x, 3) - img\_obj(y, x+1, 3);  else  A(im2grad(y-1, x-1), im2var(y, x)) = 1;  b\_r(im2grad(y-1, x-1)) = img\_source(y, x, 1);  b\_g(im2grad(y-1, x-1)) = img\_source(y, x, 2);  b\_b(im2grad(y-1, x-1)) = img\_source(y, x, 3);  end  end  end  % Get result  v\_r = A \ b\_r;  v\_g = A \ b\_g;  v\_b = A \ b\_b;  vimg\_r = reshape(v\_r, height, width);  vimg\_g = reshape(v\_g, height, width);  vimg\_b = reshape(v\_b, height, width);    % Copy to Original Image  for y = 2:height-1  for x = 2:width-1  if objmask(y, x)  img\_back(y+325, x+200, 1) = vimg\_r(y,x);  img\_back(y+325, x+200, 2) = vimg\_g(y,x);  img\_back(y+325, x+200, 3) = vimg\_b(y,x);  end  end  end    imshow(img\_back / 256.); |

**Result**



** ->**



**Description**

I reduce the image with scale 1/4 because when I used the original image it took too much time.

Same as first question, I make the gradient image and transform matrix. To get gradient image, I used laplacian filter.

**Q3. BLENDING WITH MIXED GRADIENTS**

**Code**

|  |
| --- |
| ...    for y = 2:height-1  for x = 2:width-1  if objmask(y, x)  A(im2grad(y-1, x-1), im2var(y, x)) = 4;  A(im2grad(y-1, x-1), im2var(y-1, x)) = -1;  A(im2grad(y-1, x-1), im2var(y, x-1)) = -1;  A(im2grad(y-1, x-1), im2var(y+1, x)) = -1;  A(im2grad(y-1, x-1), im2var(y, x+1)) = -1;  grad\_s = img\_obj(y, x, :) \* 4 - img\_obj(y-1, x, :) - ...  img\_obj(y, x-1, :) - img\_obj(y+1, x, :) - img\_obj(y, x+1, :);  grad\_t = img\_source(y, x, :) \* 4 - img\_source(y-1, x, :) - ...  img\_source(y, x-1, :) - img\_source(y+1, x, :) - img\_source(y, x+1, :);  if abs(mean(grad\_s)) > abs(mean(grad\_t))  b\_r(im2grad(y-1, x-1)) = grad\_s(1);  b\_g(im2grad(y-1, x-1)) = grad\_s(2);  b\_b(im2grad(y-1, x-1)) = grad\_s(3);  else  b\_r(im2grad(y-1, x-1)) = grad\_t(1);  b\_g(im2grad(y-1, x-1)) = grad\_t(2);  b\_b(im2grad(y-1, x-1)) = grad\_t(3);  end  else  A(im2grad(y-1, x-1), im2var(y, x)) = 1;  b\_r(im2grad(y-1, x-1)) = img\_source(y, x, 1);  b\_g(im2grad(y-1, x-1)) = img\_source(y, x, 2);  b\_b(im2grad(y-1, x-1)) = img\_source(y, x, 3);  end  end  end  ... |

**Result**

****

** -> **

**Description**

Not just using source gradient, calculate the magnitude of gradient of source image and target image and use the larger one.

The penguin’s feature has been blended transparently, but it doesn’t seems a living one.