Programming Project #2: Image Quilting

CS445: Computational Photography - Fall 2020

```
import cv2
In [1]:
        import numpy as np
        from scipy import signal
        import matplotlib.pyplot as plt
        %matplotlib inline
        import os
        from random import random
        import time
        # modify to where you store your project data including utils.py
        datadir = "C:/Users/jackt/Desktop/CS445/cs445quilt/"
        #utilfn = datadir + "utils.py"
        #!cp "$utilfn" .
        #samplesfn = datadir + "samples"
        #!cp -r "$samplesfn" .
        import utils
In [2]: | from utils import cut # default cut function for seam finding section
```

Part I: Randomly Sampled Texture (10 pts)

```
In [3]:
       def quilt random(sample, out size, patch size):
           Randomly samples square patches of size patchsize from sample in order to
        create an output image of size outsize.
           :param out size: int
                                       The width of the square output image
                                      The width of the square sample patch
           :param patch_size: int
           :return: numpy.ndarray
           im out = np.full((out size,out size,3), 0)
           h src = sample.shape[0]
           w src = sample.shape[1]
           for i in range(0, out_size - patch_size + 1, patch_size):
              for j in range(0, out_size - patch_size + 1, patch_size):
                  ry = np.random.randint(0, h src - patch size)
                  rx = np.random.randint(0, w_src - patch_size)
                  im_out[i:i+patch_size, j:j+patch_size] = sample[ry:ry+patch_size,
       rx:rx+patch size]
           return im out
```

```
In [4]:
        sample_img_fn = 'samples/text_small.jpg' # feet free to change
        sample img = cv2.cvtColor(cv2.imread(sample img fn), cv2.COLOR BGR2RGB)
        plt.imshow(sample_img)
        plt.show()
        out_size = 200 # change these parameters as needed
        patch size = 40
        res = quilt random(sample img, out size, patch size)
        if res is not None:
            plt.imshow(res)
            ut it becomes harder to lau
```

```
20 ound itself, at "this daily i
          ving rooms," as House Der
   40 escribed it last fall. He fail
          ut he left a ringing question
          ore years of Monica Lewin
   so inda Tripp?" That now seen
          Political comedian Al Fran
 100 ext phase of the story will
          ore years.
ore years lars of Monica Lewe left a rist "this da
nda Tripp, "ipp?" That now se years of I as House
colitical collegement Mall Entring." It fall the
left a rit nging que;t "this da's of Monibed it las
years of Monica Leas House pp?" That left a rin
Tripp?" That now tell. the comediar wave of I
this daily of Monica that da ooms, "as ging ques
11. He fail of Monica this da ooms, "as ging ques
11. He fail that nas House dit last falonica Le
125 self, at "this of Monicons, "itself, at "of Monico
tilast fahl comedia left a rin
ound itselfacomes had left a rin House De of Monic
```

und itselfecomes hie left a ricHouse De of Monic 175 ing roomsitself, at 'years of fall. He fap?" That n cribed it looms," as Tripp?" 'ng questicomedian 100 150

Part II: Overlapping Patches (30 pts)

```
Computes the cost of sampling each patch based on the
            SSD of the overlapping regions of the existing and patches of the image
            Assumes image is colored
            :param template: existing template
            :param sample: texture sample image
            :param x, y: top-left coordinates of next patch in existing image
            :param patch size: patch size
            :param overlap: width of overlap region
            # generate mask for left, top, or top-left overlap region
            mask = np.full((patch size, patch size, 3), 0)
            if mode == 0:
                # top
                mask[0:overlap,:,:] = 1
            elif mode == 1:
                # Left
                mask[:,0:overlap,:] = 1
            elif mode == 2:
                # top-left
                mask[0:overlap,:,:] = 1
                mask[:,0:overlap,:] = 1
            else:
                mask[:,:,:] = 1
            # compute cost using SSD
            imsq = sample.astype(np.float64)**2
            ssd cost = cv2.filter2D(imsq, ddepth=-1, kernel=mask[:,:,0]).sum(axis=2)
            mt = mask*template
            ssd cost += (mt**2).sum()
            for c in range(3):
                s2 = cv2.filter2D(sample[:,:,c].astype(np.float64), ddepth=-1, kernel=
        mt[:,:,c])
                ssd_cost -= 2 * s2
            return ssd_cost
In [6]: def choose_sample(cost_img, sample, phs, tol):
            ind = np.unravel_index(np.argsort(cost_img, axis=None), cost_img.shape)
            r = np.random.randint(0,tol)
```

In [5]: def ssd patch(template, sample, patch size, overlap, mode):

y = ind[0][r]
x = ind[1][r]

return sample[y:y+patch_size, x:x+patch_size]

```
In [7]:
        def quilt_simple(sample, out_size, patch_size, overlap, tol):
            Randomly samples square patches of size patchsize from sample in order to
         create an output image of size outsize.
            Feel free to add function parameters
            :param sample: numpy.ndarray
            :param out size: int
            :param patch_size: int
            :param overlap: int
            :param tol: float
            :return: numpy.ndarray
            im_out = np.zeros((out_size, out_size, 3), dtype=np.uint8)
            step = patch_size - overlap
            num_patches = (out_size - patch_size + overlap) // step
            phs = patch_size // 2
            for r in range(0, num_patches):
                for c in range(0, num_patches):
                     if r == 0 and c == 0:
                         im_out[:patch_size, :patch_size] = sample[:patch_size, :patch_
        size]
                     else:
                         mode = 0
                         if r == 0 and c > 0:
                            mode = 1
                         elif r > 0 and c > 0:
                            mode = 2
                         template = im_out[r*step:r*step+patch_size, c*step:c*step+patc
        h_size]
                         cost img = ssd patch(template, sample, patch size, overlap, mo
        de)
                         im_out[r*step:r*step+patch_size, c*step:c*step+patch_size] = c
        hoose_sample(
                             cost_img[phs:-phs, phs:-phs], sample, patch_size, tol)
            return im out
```

```
In [8]: sample_img_fn = 'samples/text_small.jpg'
    sample_img = cv2.cvtColor(cv2.imread(sample_img_fn), cv2.COLOR_BGR2RGB)
    plt.imshow(sample_img)
    plt.show()

out_size = 300  # change these parameters as needed
    patch_size = 25
    overlap = 11
    tol = 3
    res = quilt_simple(sample_img, out_size, patch_size, overlap, tol) #feel free
    to change parameters to get best results
    if res is not None:
        plt.figure(figsize=(10,10))
        plt.imshow(res)
```

ut it becomes harder to lau
ound itself, at "this daily o
ving rooms," as House Der
cscribed it last fall. He fail
ut he left a ringing question
ore years of Monica Lewin
inda Tripp?" That now seen
colitical comedian Al Fran

ut it now seconicz lexy ta LwinginGripp?"all . last fall. He fallt last fall oothanica he ore yeast nowf Morrearooms, a rip ring ques harder a ring rskit "traen 11 1251 Becomés närður", 8, "op / ed 11 1251 i Induna "t' He as Hö omat "thirf bearing at to 16: " " " " " " " " " " " and to write a ripe time of ring of a rest fall 50 d it las Heutifan a Left constitall, lef daiat now left a noyoir a ringing. ft a ripginging tre yeg roun Tripp?'t a medripp?" Thaooms," as 11.: left a ng ding irinhis da Tripa Trippis diberat "this dacribomed it last fore years o nicars cihe lefti itseft a ringing ms," asHouse Ing rooms, Tringindascribe 100 it becrire wast quomy of Moed it laut il it last last fall .: ribed i . H r to la në lëtr ringing në lëf. 35 That filoust last ist i i filitinat hoës fiques daily aurant Magna waat fadira. Likaringan tahun dina Lifaka Laue Di inging a rive ye.v. a ripg quest forebesondut now, coing ow, at ars ow see 150 - roomsound itself, at "tore uging a rithis dai Tripp?fallt lat a ringing qu he lorityinhabomat "ng r of roolast fahd itse luesng queteft a ringing roo it ibon at use lasft a riblome lelf, gin' as Hed itfalkis que of a Tripp?" " ound it" asft a ringing quelf, atof Monica ues'hatms;," as Hat "ms," as H native it les of Morriecomer', at" The time Lectast last relified it line fail ng randtakion?!! 7task itset.gr./las Snict nergye giperting-typeloftæytgatég ribinald itsela, ving rooms taripatnen talkunditselitaditsellegow se he lying rooms," as," as loopusal cooms Frooiring rooupoms," as," A1 Fr quescribedef last lastiboms "this equeut he left a' as our last last f: That of a Lit hed ift at last fat fa," a . H" as Hobec yeaunct a ringudascri comes has low becomes harder t it lastit lait lit he let a ring rooms on inditself, at " Ald itself, at "this cit a ring a rit a ore yeas oficribed it last g rooms," as ; 50 100 150 200 250

```
In [9]: def quilt cut(sample, out size, patch size, overlap, tol):
            Samples square patches of size patchsize from sample using seam finding in
        order to
            create an output image of size outsize.
            Feel free to add function parameters
            :param sample: numpy.ndarray
            :param out size: int
            :param patch size: int
            :param overlap: int
             :param tol: float
             :return: numpy.ndarray
            im out = np.zeros((out size, out size, 3), dtype=np.uint8)
            step = patch size - overlap
            num_patches = (out_size - patch_size + overlap) // step
            phs = patch size // 2
            for r in range(0, num_patches):
                for c in range(0, num patches):
                     if r == 0 and c == 0:
                         im_out[:patch_size, :patch_size] = sample[:patch_size, :patch_
        size]
                     else:
                         mode = 0
                         if r == 0 and c > 0:
                             mode = 1
                         elif r > 0 and c > 0:
                             mode = 2
                         template = im_out[r*step:r*step+patch_size, c*step:c*step+patc
        h size]
                         cost_img = ssd_patch(template, sample, patch_size, overlap, mo
        de)
                         chosen = choose_sample(cost_img[phs:-phs, phs:-phs], sample, p
        atch_size, tol)
                         if r == 0 and c == 0:
                             im_out[r*step:r*step+patch_size, c*step:c*step+patch_size]
        = chosen
                         elif r == 0 and c > 0:
                             diff = im out[r*step:r*step+patch size, c*step:c*step+over
        lap] - chosen[:patch_size,:overlap]
                             ssd = np.square(diff[:,:,0]) + np.square(diff[:,:,1]) + np
         .square(diff[:,:,2])
                             res mask = cut(ssd.T).T
                             for y in range(0, patch size):
                                 for x in range(0, patch_size):
                                     if x < overlap and res_mask[y,x] == 0:</pre>
                                         continue
                                     im out[r*step+y, c*step+x] = chosen[y,x]
```

```
elif r > 0 and c == 0:
                    diff = im_out[r*step:r*step+overlap, c*step:c*step+patch_s
ize] - chosen[:overlap,:patch_size]
                    ssd = np.square(diff[:,:,0]) + np.square(diff[:,:,1]) + np
.square(diff[:,:,2])
                    res_mask = cut(ssd)
                    for y in range(0, patch_size):
                        for x in range(0, patch_size):
                            if y < overlap and res mask[y,x] == 0:
                                continue
                            im_out[r*step+y, c*step+x] = chosen[y,x]
                else:
                    diff_v = im_out[r*step:r*step+patch_size, c*step:c*step+ov
erlap] - chosen[:patch size,:overlap]
                    ssd_v = np.square(diff_v[:,:,0]) + np.square(diff_v[:,:,1
]) + np.square(diff_v[:,:,2])
                    res mask v = cut(ssd v.T).T
                    diff_h = im_out[r*step:r*step+overlap, c*step:c*step+patch
_size] - chosen[:overlap,:patch_size]
                    ssd_h = np.square(diff_h[:,:,0]) + np.square(diff_h[:,:,1
]) + np.square(diff_h[:,:,2])
                    res_mask_h = cut(ssd_h)
                    for y in range(0, patch_size):
                        for x in range(0, patch_size):
                            if y < overlap and res_mask_h[y,x] == 0:</pre>
                                continue # mask doesn't allow
                            if x < overlap and res mask v[y,x] == 0:
                                continue # mask doesn't allow
                            # otherwise add to output
                            im out[r*step+x, c*step+y] = chosen[x, y]
   return im out
```

```
In [10]: sample_img_fn = 'samples/text_small.jpg'
sample_img = cv2.cvtColor(cv2.imread(sample_img_fn), cv2.COLOR_BGR2RGB)
plt.imshow(sample_img)
plt.show()

out_size = 300  # change these parameters as needed
patch_size = 25
overlap = 11
tol = 5
res = quilt_cut(sample_img, out_size, patch_size, overlap, tol)
if res is not None:
    plt.figure(figsize=(15,15))
    plt.imshow(res)
```

ut it becomes harder to lau
tound itself, at "this daily or ving rooms," as House Der
touched it last fall. He fail ut he left a ringing question ore years of Monica Lewin and Tripp?" That now seen colitical comedian Al France of the story will

ut it beccribca Ecart it becd it He it lall. He it last:. Ht bed io lf, at" The ound it he haveeround itses ringing it pecoha ringa in Lufflaus ."awed nat of non-ALEVillait Lieu of Monus dating, he left ally mundianulef e yearre progress teft a xing quiw self , at user jixe yainlitica ovribenong ou $^{\circ}$ ' Thair's rind itselfian Alica Houg ropit li last cef Mousie yeart novear ing queft aw seribe liert a itse darribeorg rim seselo". Al. Heft is roomse I onist how self have frag roomself, as of Montas had self, at high Al A l itsne left tog quesic lubecone res."ap?" Thalit 'as Homs," asse DHous 100 Joons," as House De land itsee \dot{y} t acts \dot{H} s, at,"I innyt fabl itst ta \dot{H} o fall. \dot{H} ed it last fall. Here wen, room Timpons " asst fa Turda rippe falitieans." véarst faktus.Nomsdjúribedólitágal ocopnquest fall, itisal czeß. Heft a Housethd itselrint he l'halouse Dra rir Halla Linginis dailyft ars croœars of 150 ll. Hitatinooms," 2mg 20pp?": Léwel fall. Heftliliouse Denouese If, at " groongisoomif, used for Longuestinging dittast fal Heft and Leval itself ibed it bed itas, "Hrouse De. A Led töllmir sellang infarmes hardecolusinom e lefthd dai it lefall! He fasel Les dai it l'Evast finging i lettis, "as Höuse veaing qu'ift a ging ques a rig. 200. a L'enouw ring (Moribed et last fall. He Monicalf notonical hereritat now at Alaf Modhane leviour fid is que edian lasft a Hhatleft aond iund i Ht a yjopliffr sedak years of," as H. He owisingious riligis dailed it lequests of leus of n itseAl . He as "as ," wars of Al Fydonica Mouse Phat now seep?". Tirat oms," as Houtus daiast rall. yzays of to nola Touanita Al Est nomediani triust fall. Heuse forgin," : a Tripp?". lailseliticríbed ithrodian Ælf , a áis a ring quafúeit fall bit last s of it cool it Fringt he left as He fasoms, "out Frionicat "thit ging ring que

150

```
In [11]: def texture transfer(sample, patch size, overlap, tol, guidance im, alpha):
             Samples square patches of size patchsize from sample using seam finding in
         order to create an output
             image of size outsize.
             Feel free to modify function parameters
             :param sample: numpy.ndarray
             :param patch size: int
             :param overlap: int
             :param tol: float
             :param quidance im: target overall appearance for the output
             :param alpha: float 0-1 for strength of target
             :return: numpy.ndarray
             im out = np.zeros(guidance im.shape, dtype=np.uint8)
             step = patch_size - overlap
             phs = patch size // 2
             for row in range(0, guidance_im.shape[0] - patch_size + 1, step):
                 for col in range(0, guidance im.shape[1] - patch size + 1, step):
                      if row == 0 and col == 0:
                          im_out[:patch_size, :patch_size] = sample[:patch_size, :patch_
         size]
                      else:
                         mode = 0
                         if row == 0 and col > 0:
                             mode = 1
                         elif row > 0 and col > 0:
                             mode = 2
                         template = im_out[row:row+patch_size, col:col+patch_size]
                          ssd overlap = ssd patch(template, sample, patch size, overlap
         , mode)
                         guide_template = guidance_im[row:row+patch_size, col:col+patch
         _size]
                         ssd_transfer = ssd_patch(guide_template, sample, patch_size, o
         verlap, 3)
                         cost img = alpha * ssd overlap + (1 - alpha) * ssd transfer
                          chosen = choose_sample(cost_img[phs:-phs, phs:-phs], sample, p
         atch_size, tol)
                         if row == 0 and col == 0:
                              im out[row:row+patch size, col:col+patch size] = chosen
                         elif row == 0 and col > 0:
                              diff = im out[row:row+patch size, col:col+overlap] - chose
         n[:patch size,:overlap]
                              ssd = np.square(diff[:,:,0]) + np.square(diff[:,:,1]) + np
         .square(diff[:,:,2])
                              res_mask = cut(ssd.T).T
                              for y in range(0, patch size):
                                  for x in range(0, patch size):
                                      if x < overlap and res_mask[y,x] == 0:</pre>
```

```
continue
                            im_out[row+y, col+x] = chosen[y,x]
                elif row > 0 and col == 0:
                    diff = im out[row:row+overlap, col:col+patch size] - chose
n[:overlap,:patch_size]
                    ssd = np.square(diff[:,:,0]) + np.square(diff[:,:,1]) + np
.square(diff[:,:,2])
                    res mask = cut(ssd)
                    for y in range(0, patch size):
                        for x in range(0, patch_size):
                            if y < overlap and res mask[y,x] == 0:
                                continue
                            im_out[row+y, col+x] = chosen[y,x]
                else:
                    diff_v = im_out[row:row+patch_size, col:col+overlap] - cho
sen[:patch size,:overlap]
                    ssd_v = np.square(diff_v[:,:,0]) + np.square(diff_v[:,:,1
]) + np.square(diff_v[:,:,2])
                    res mask v = cut(ssd v.T).T
                    diff_h = im_out[row:row+overlap, col:col+patch_size] - cho
sen[:overlap,:patch_size]
                    ssd_h = np.square(diff_h[:,:,0]) + np.square(diff_h[:,:,1
]) + np.square(diff_h[:,:,2])
                    res_mask_h = cut(ssd_h)
                    for y in range(0, patch_size):
                        for x in range(0, patch size):
                            if y < overlap and res_mask_h[y,x] == 0:</pre>
                                continue # mask doesn't allow
                            if x < overlap and res_mask_v[y,x] == 0:</pre>
                                continue # mask doesn't allow
                            # otherwise add to output
                            im out[row+x, col+y] = chosen[x, y]
    return im_out
```

```
In [12]: # load/process appropriate input texture and guidance images
         texture_img_fn = 'samples/toast_cropped.jpg'
         texture_img = cv2.cvtColor(cv2.imread(texture_img_fn), cv2.COLOR_BGR2RGB)
         plt.imshow(texture_img)
         plt.show()
         guidance_img_fn = 'samples/feynman.tiff'
         guidance_img = cv2.cvtColor(cv2.imread(guidance_img_fn), cv2.COLOR_BGR2RGB)
         plt.imshow(guidance_img)
         plt.show()
         patch_size = 25
         overlap = 11
         tol = 3
         alpha = 0.5
         res = texture_transfer(texture_img, patch_size, overlap, tol, guidance_img, al
         pha)
         plt.figure(figsize=(15,15))
         plt.imshow(res)
         plt.show()
```





