```
String filename = "test";
String fileext = ".jpg";
String foldername = "./";

final static String pattern_prefix = "nyt/NYTimes-Dec1900-Jan1901_";
final static String file_ext = ".jpg";
final static int pattern_init = 3; // starting number
final static int pattern_length = 8; // how many images from the set
final static int pattern_size = 4; // number of digits

// choose method of mapping
int mode = ABS_MODE; // list below AVG_MODE, ABS_MODE,
DIST_MODE

int THR = 20; // higher value bigger rectangles (1..200)
int MINR = 8; // minimum block (4..200)

int number_of_iterations = 20; // more = more variety
int number_of_blocks = 50; // more = more search tries

// MODEs LIST
final static int AVG_MODE = 0; // worst matching, difference of avgs of
```

```
the luma
final static int ABS_MODE = 1; // difference of the luma each pixel
final static int DIST_MODE = 2; // best matching, distance between
pixels colors (vectors)

int max_display_size = 1000; // viewing window size (regardless image
size)

boolean do_blend = false; // blend image after process
int blend_mode = OVERLAY; // blend type

// working buffer
PGraphics buffer;

// image
PImage img;

String sessionid;

void setup() {
  sessionid = hex((int)random(0xffff),4);
  img = loadImage(foldername+filename+fileext);
```

```
    buffer = createGraphics(img.width, img.height);
    buffer.beginDraw();
    buffer.noStroke();
    buffer.smooth(8);
    buffer.background(0);
    buffer.endDraw();

    // calculate window size
    float ratio = (float)img.width/(float)img.height;
    int neww, newh;
    if(ratio < 1.0) { neww = (int)(max_display_size * ratio); newh =
max_display_size; } else { neww = max_display_size; newh =
(int)(max_display_size / ratio); } size(neww,newh); processImage(); }
void draw() { // fill for iterative processing } ArrayListimgsb = new
ArrayList();
HashMap<string, arraylist="" style="box-sizing: border-box; padding:
0px; margin: 0px;"> parts = new HashMap<string, arraylist=""
style="box-sizing: border-box; padding: 0px; margin: 0px;">();

class LImage {
    PVector[][] b;
    String name;
    int w, h;
}

class Part {
    int posx, posy, w, h;
    int x, y;

    String toString() {
        return "(" + posx + "," + posy + "," + w + "," + h + ") -> (" + x + ","
+ y + ")" ;
    }
}


void processImage() {
    buffer.beginDraw();

    println("Preparing data");
    prepare_image();
    prepare_patterns();
    segment(0, img.width-1, 0, img.height-1, 2);
    println("Layering");
    for (String key : parts.keySet ()) {
        ArrayListp = parts.get(key);
        PImage _img = loadImage(key);
        println("Parts from image: " + key);
        for (Part part : p) {
            buffer.image(_img.get(part.posx,   part.posy,   part.w,   part.h),
part.x, part.y);
        }
    }

    println("done");
    // END CODE HERE!

    if(do_blend)

buffer.blend(img,0,0,img.width,img.height,0,0,buffer.width,buffer.height,b
lend_mode);

    buffer.endDraw();
    image(buffer,0,0,width,height);
```

```
}

void keyPressed() {
   // SPACE to save
   if(keyCode == 32) {
      String fn = foldername + filename + "/res_" + sessionid +
hex((int)random(0xffff),4)+"_"+filename+fileext;
      buffer.save(fn);
      println("Image "+ fn + " saved");
   }
}

PVector[][] imgb;
void prepare_image() {
   imgb = new PVector[img.width][img.height];
   for (int x=0; x<img.width; x++)="" {="" for="" (int="" y="0;"
y<img.height;="" y++)="" int="" c="img.get(x," y);="" float="" r="map((c"
style="box-sizing: border-box; padding: 0px; margin: 0px;">>16)&0xff, 0,
255, 0, 1);
         float g = map((c>>8)&0xff, 0, 255, 0, 1);
         float b = map(c&0xff, 0, 255, 0, 1);
         PVector v = new PVector(r, g, b);
```

```
            imgb[x][y] = v;
        }
    }
}

void prepare_patterns() {
    for (int i=pattern_init; i< (pattern_init+pattern_length); i++) { String
suf = nf(i, pattern_size); String fname = pattern_prefix + suf + file_ext;
PImage _img = loadImage(fname); println(fname); LImage bi = new
LImage(); bi.b = new PVector[_img.width][_img.height]; bi.name = fname;
bi.w = _img.width; bi.h = _img.height; for (int x=0; x<_img.width; x++)
{ for (int y=0; y<_img.height; y++) { int c = _img.get(x, y); float r =
map((c>>16)&0xff, 0, 255, 0, 1);
                float g = map((c>>8)&0xff, 0, 255, 0, 1);
                float b = map(c&0xff, 0, 255, 0, 1);
                PVector v = new PVector(r, g, b);
                bi.b[x][y] = v;
            }
        }
        imgsb.add(bi);
    }
}
```

```
void find_match(int posx, int posy, int w, int h) {
   float br = 0;
   if (mode == AVG_MODE) {
      for (int x=posx; x< (posx+w); x++) { for (int y=posy; y< (posy+h);
y++) { br+= getLuma(imgb[x][y]); } } } float currdiff = 1.0e10; int currxx
= -1; int curryy = -1; LImage currimg = null; for (int i=0;
i<number_of_iterations; i++)="" {="" limage="" _img="imgsb.get("
(int)random(imgsb.size())="" );="" for="" (int="" iter="0;"
iter<number_of_blocks;="" iter++)="" int=""
xx="(int)random(_img.w-w-1);" yy="(int)random(_img.h-h-1);"
if(xx+w="" style="box-sizing: border-box; padding: 0px; margin: 0px;">=
_img.w || yy+h >= _img.h) break;

         float lbr = 0;
         for (int x=xx, xi=posx; x< (xx+w); x++, xi++) { for (int y=yy,
yi=posy; y< (yy+h); y++, yi++) { if(mode == DIST_MODE) lbr +=
_img.b[x][y].dist(imgb[xi][yi]); else if(mode == AVG_MODE) lbr +=
getLuma(_img.b[x][y]); else if(mode == ABS_MODE) lbr +=
abs(getLuma(_img.b[x][y])-getLuma(imgb[xi][yi])); } } float ldiff =
mode == AVG_MODE?abs(br-lbr):lbr; if (ldiff<currdiff) {=""
currdiff="ldiff;" currxx="xx;" curryy="yy;" currimg="_img;" }="" part=""
```

p="new" part();="" p.posx="currxx;" p.posy="curryy;" p.w="w;" p.h="h;" p.x="posx;" p.y="posy;" arraylistlist;="" if="" (parts.containskey(currimg.name))="" list="parts.get(currimg.name);" else="" arraylist();="" parts.put(currimg.name,="" list);="" list.add(p);="" println("matched:="" "="" +="" currimg.name="" ";="" p);="" void="" segment(int="" x1,="" int="" x2,="" y1,="" y2,="" obl)="" diffx="x2-x1;" diffy="y2-y1;" ((obl="" style="box-sizing: border-box; padding: 0px; margin: 0px;">0) || (diffx>MINR && diffy>MINR && godeeper(x1, x2, y1, y2))) {

```
      int midx = (int)random(diffx/2-diffx/4, diffx/2+diffx/4);
      int midy = (int)random(diffy/2-diffy/4, diffy/2+diffy/4);
      segment(x1, x1+midx, y1, y1+midy, obl-1);
      segment(x1+midx+1, x2, y1, y1+midy, obl-1);
      segment(x1, x1+midx, y1+midy+1, y2, obl-1);
      segment(x1+midx+1, x2, y1+midy+1, y2, obl-1);
   } else {
      find_match(x1, y1, diffx+1, diffy+1);
   }
}

final float getLuma(PVector v) {
   return v.x*0.3+0.59*v.y+0.11*v.z;
```

```
}

final int getLumaN(PVector v) {
    return (int)(255*getLuma(v));
}

boolean godeeper(int x1, int x2, int y1, int y2) {
    int[] h = new int[256];
    // top and bottom line
    for (int x=x1; x<=x2; x++) { h[getLumaN(imgb[x][y1])]++;
h[getLumaN(imgb[x][y2])]++; } // left and right, without corners for
(int y=y1+1; y<y2; y++)="" {="" h[getluman(imgb[x1][y])]++;=""
h[getluman(imgb[x2][y])]++;="" }="" int="" midx="x1+(x2-x1)/2;"
midy="y1+(y2-y1)/2;" horizontal,="" without="" endpoints="" for=""
(int="" x="x1+1;" x<x2;="" x++)="" h[getluman(imgb[x][midy])]++;=""
vertical,="" y="y1+1;" y<y2;="" h[getluman(imgb[midx][y])]++;=""
remove="" crossingpoint="" h[getluman(imgb[midx][midy])]--;=""
calculate="" mean="" float="" sum="0;" i="0;" i<256;="" i++)="" +="i"
*="" h[i];="" =="" sum;="" stddev="0;" return="" style="box-sizing:
border-box; padding: 0px; margin: 0px;"> THR;
}
```

```
//

final static int[] blends = {ADD, SUBTRACT, DARKEST, LIGHTEST,
DIFFERENCE, EXCLUSION, MULTIPLY, SCREEN, OVERLAY, HARD_LIGHT,
SOFT_LIGHT, DODGE, BURN};

// ALL Channels, Nxxx stand for negative (255-value)
// channels to work with
final static int RED = 0;
final static int GREEN = 1;
final static int BLUE = 2;
final static int HUE = 3;
final static int SATURATION = 4;
final static int BRIGHTNESS = 5;
final static int NRED = 6;
final static int NGREEN = 7;
final static int NBLUE = 8;
final static int NHUE = 9;
final static int NSATURATION = 10;
final static int NBRIGHTNESS = 11;

float getChannel(color c, int channel) {
    int ch = channel>5?channel-6:channel;
    float cc;

    switch(ch) {
        case RED: cc = red(c); break;
        case GREEN: cc = green(c); break;
        case BLUE: cc = blue(c); break;
        case HUE: cc = hue(c); break;
        case SATURATION: cc = saturation(c); break;
        default: cc= brightness(c); break;
    }

    return channel>5?255-cc:cc;
}
import ddf.minim.*;
AudioPlayer player;
Minim minim;
PImage p,b;
int r=200;
float ptheta=0.0;
color c1;
float lineRotate=PI;
```

```
int step=50;
int num=16;

void setup()
{
   fullScreen();
   colorMode(HSB,360,100,100,100);
   Album();
   minim=new Minim(this);
   player=minim.loadFile("1.mp3", step*num);
   player.play();
   frameRate(15);
}
void draw()
{
   translate(width/2,height/2);
   background(0);
   pushMatrix();
   rotate(ptheta);
   image(p,0,0);
   popMatrix();
   ptheta+=0.015;
```

```
    rotate(lineRotate);
    leftJump();
    rightJump1();
    rightJump2();
}

void leftJump()
{
    float l=400;
    noFill();
    strokeWeight(2);
    c1=color(random(0,255),random(0,255),random(0,255));
    stroke(c1, 100);
    PVector[] point=new PVector[num];
    float rTheta=2*PI/num;
    for (int i=0; i<num; i++)="" {="" point[i]="new" 
pvector((r+20)*cos(rtheta*i),(r+20)*sin(rtheta*i));="" }="" 
if(abs(player.left.get(0))="" style="box-sizing: border-box; padding: 0px; 
margin: 0px;">0.01)
    {
        l=l/2;
        for (int j=1;j<6;j++) { point[j]=new
```

```
PVector((r+20+abs(player.left.get(j*step))*l)*cos(rTheta*j),(r+20+a
bs(player.left.get(j*step))*l)*sin(rTheta*j));              }             }
if(abs(player.left.get(0))>0.2)
   {
      l=l/2;
      int j=9;
      point[j]=new
PVector((r+20+abs(player.left.get(j*step))*l)*cos(rTheta*j),(r+20+a
bs(player.left.get(j*step))*l)*sin(rTheta*j));
      j = 13;
      point[j]                              =                              new
PVector((r+20+abs(player.left.get(j*step))*l)*cos(rTheta*j),(r+20+a
bs(player.left.get(j*step))*l)*sin(rTheta*j));
   }


   if(abs(player.left.get(0))>0.3)
   {
      l=l/2;
      for(int           j=6;j<10;j++)            {            point[j]=new
PVector((r+20+abs(player.left.get(j*step))*l)*cos(rTheta*j),(r+20+a
bs(player.left.get(j*step))*l)*sin(rTheta*j));  }  for(int  j=13;j<16;j++)
{                                                              point[j]=new
```

```
PVector((r+20+abs(player.left.get(j*step))*l)*cos(rTheta*j),(r+20+a
bs(player.left.get(j*step))*l)*sin(rTheta*j)); } int j=0; point[j]=new
PVector((r+20+abs(player.left.get(j*step))*l)*cos(rTheta*j),(r+20+a
bs(player.left.get(j*step))*l)*sin(rTheta*j));            }            if
(abs(player.left.get(0))>0.5)
  {
    l=l/2;
    for(int          j=10;j<13;j++)          {          point[j]=new
PVector((r+20+abs(player.left.get(j*step))*l)*cos(rTheta*j),(r+20+a
bs(player.left.get(j*step))*l)*sin(rTheta*j)); } } beginShape(); for
(int  i=0;i<num;i++)  {=""  curvevertex(point[i].x,point[i].y);=""  }=""
for=""  (int=""  i="0;i<3;i++)"  endshape();=""  void=""  rightjump1()=""
float=""        l="300;"        nofill();=""        strokeweight(2);=""
c1="color(random(0,255),random(0,255),random(0,255));"
stroke(c1,=""  100);=""  pvector[]=""  point="new"  pvector[num];=""
rtheta="2*PI/num;"      i<num;=""      i++)=""      point[i]="new"
pvector((r+20)*cos(rtheta*i),(r+20)*sin(rtheta*i));=""
if(abs(player.right.get(0))=""  style="box-sizing:  border-box;  padding:
0px; margin: 0px;">0.01)
  {
    l=l/2;
    for        (int        j=1;j<6;j++)        {        point[j]=new
```

```
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j));            }            }
if(abs(player.right.get(0))>0.2)
  {
     l=l/2;
     int j=9;
     point[j]=new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j));
     j = 13;
     point[j]                          =                          new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j));
  }

  if(abs(player.right.get(0))>0.3)
  {
     l=l/2;
     for(int          j=6;j<10;j++)          {          point[j]=new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j)); } for(int j=13;j<16;j++)
{                                                       point[j]=new
```

PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+ abs(player.right.get(j*step))*l)*sin(rTheta*j)); } int j=0; point[j]=new PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+ abs(player.right.get(j*step))*l)*sin(rTheta*j)); } if (abs(player.right.get(0))>0.5)

{

l=l/2;

for(int j=10;j<13;j++) { point[j]=new PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+ abs(player.right.get(j*step))*l)*sin(rTheta*j)); } } beginShape(); for (int i=0;i<num;i++) {="" curvevertex(point[i].x,point[i].y);="" }="" for="" (int="" i="0;i<3;i++)" endshape();="" void="" rightjump2()="" float="" l="150;" nofill();="" strokeweight(2);="" c1="color(random(0,255),random(0,255),random(0,255));" stroke(c1,="" 100);="" pvector[]="" point="new" pvector[num];="" rtheta="2*PI/num;" i<num;="" i++)="" point[i]="new" pvector((r+20)*cos(rtheta*i),(r+20)*sin(rtheta*i));="" if(abs(player.right.get(0))="" style="box-sizing: border-box; padding: 0px; margin: 0px;">0.01)

{

l=l/2;

for (int j=1;j<6;j++) { point[j]=new

```
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j));          }          }
if(abs(player.right.get(0))>0.2)
   {
      l=l/2;
      int j=9;
      point[j]=new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j));
      j = 13;
      point[j]                    =                    new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j));
   }

   if(abs(player.right.get(0))>0.3)
   {
      l=l/2;
      for(int          j=6;j<10;j++)          {          point[j]=new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j)); } for(int j=13;j<16;j++)
{                                                          point[j]=new
```

```
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j)); } int j=0; point[j]=new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j));          }          if
(abs(player.right.get(0))>0.5)
  {
    l=l/2;
    for(int          j=10;j<13;j++)          {          point[j]=new
PVector((r+20+abs(player.right.get(j*step))*l)*cos(rTheta*j),(r+20+
abs(player.right.get(j*step))*l)*sin(rTheta*j)); } } beginShape(); for
(int  i=0;i<num;i++)  {=""  curvevertex(point[i].x,point[i].y);=""  }=""
for=""  (int=""  i="0;i<3;i++)"  endshape();=""  void=""  album()=""
imagemode(center);=""                              p="loadImage("2.jpg");"
b="loadImage("Black.jpg");"  p.resize(2*r,2*r);=""  b.resize(2*r,2*r);=""
color();=""      b.filter(invert);=""      p.mask(b);=""      color()=""
c1="p.get(0,0);"  }<=""  pre=""  style="box-sizing:  border-box;  padding:
0px; margin: 0px;">
float angle = 0.0;
float speed = 0.01;
float r = 200;
float sx = 3;
float sy = 1;
float t = 30;

void setup()
{
   size(500, 500);
   noStroke();
   frameRate(30);
   background(0);
}

void draw()
{
   angle += speed;
   float sinval = sin(angle);
   float cosval = cos(angle);

   for (int x = 40; x < 460; x += 9) { float y = height/2 + (sinval * r);
fill(random(255)); ellipse(x, y, t, t); } for (int y = 40; y < 460; y += 8)
{ float x = width/2 + (sinval * r); fill(random(255)); ellipse(x, y, t, t); }
for (int x2 = 40; x2 < 460; x2 += 8) { float y2 = height/2 + (sinval * -r);
fill(random(255)); ellipse(x2, y2, t, t); } for (int y2 = 40; y2 < 460;
y2 += 8) { float x2 = width/2 + (sinval * -r); fill(random(255));
ellipse(x2, y2, t, t); } }
```