```
/*
 * File:   trackerAnalysis.cpp
 * Author: mcaprio2
 *
 * Created on July 15, 2014, 1:39 PM
 */

#include "trackerAnalysis.h"

trackerAnalysis::trackerAnalysis() {
}

trackerAnalysis::trackerAnalysis(const trackerAnalysis& orig) {
}

trackerAnalysis::~trackerAnalysis() {
}


void trackerAnalysis::writeATrack(int index, int mAvr, double fps, char * baseFileName, char
*trackname) {
  vector<vector<PotentialBug> >::iterator tracked;
  vector<PotentialBug> OneBug;
  sprintf(baseFileName, "%s-%s", baseFileName,trackname);
  FILE *trackerFile;
  trackerFile = fopen(baseFileName, "w");
  printf("Writing track %u\n", index);
  fprintf(trackerFile, "Mean distance %7.4f, mean weighted degrees %7.4f, unweighted degrees %7.4f,
Total distance %7.4f, AverageOver %d, fps: %7.3f\n", getTrackAverageDistance(index, mAvr),
getTrackAverageDegreesTurned(index, mAvr), getTrackAverageUnweighterDegreesTurned(index,
mAvr), getTrackDistance(index, mAvr),mAvr, fps);

  OneBug = tracks[index];
  for (int j1 = 0; j1 < OneBug.size(); j1++) {
    fprintf(trackerFile, "%5.2f, %5.2f, %u, %6.3f, %6.3f\n", OneBug[j1].GetLocation().x,
OneBug[j1].GetLocation().y, OneBug[j1].GetLastFrame(),getPointDegreesTurned(index, j1, mAvr)
,getPointSignedDegreesTurned(index, j1, mAvr));
  }
  fclose(trackerFile);
}


void trackerAnalysis::forward(int trackNum, int editPosition, int *jump) {
  vector<PotentialBug> OneBug;
  OneBug = tracks[trackNum];
  int startFrame = OneBug[0].GetLastFrame();

  if (*jump > editPosition) {
```

```
   if (*jump >= OneBug.size())
     *jump = OneBug.size();

   while (editPosition < *jump  && editPosition < OneBug.size()-1) {
     line(*img, OneBug[editPosition].GetLocation(), OneBug[editPosition + 1].GetLocation(), Scalar(255, 0,
0), 2, 8);
     editPosition++;
   }
  } else {
   if (*jump < 0)
     *jump = 0;

   while (editPosition > *jump  && editPosition > 0) {
     line(*img, OneBug[editPosition - 1].GetLocation(), OneBug[editPosition].GetLocation(), Scalar(0, 0,
0), 2, 8);
     editPosition--;
   }
  }
  updateOutputImage(editPosition);
}

void trackerAnalysis::image2Update(Mat *i) {
  im_out = i;
}

void trackerAnalysis::updateOutputImage(int cCount) {
  char stringBuffer[100];
  Mat img2;

  img2 = im_out->clone();
  vector<PotentialBug> OneBug;
  OneBug = tracks[0];
  sprintf(stringBuffer, " Frame %u", OneBug[cCount].GetLastFrame());
  putText(img2, stringBuffer, Point2f(100, 100), CV_FONT_HERSHEY_SIMPLEX, 0.5, cv::Scalar(255), 1, 8,
false);
  sprintf(stringBuffer, " x =  %7.3f, y = %7.3f", OneBug[cCount].GetLocation().x,
OneBug[cCount].GetLocation().y);
  putText(img2, stringBuffer, Point2f(100, 125), CV_FONT_HERSHEY_SIMPLEX, 0.5, cv::Scalar(255), 1, 8,
false);
  sprintf(stringBuffer, " Index: %u", cCount);
  putText(img2, stringBuffer, Point2f(100, 150), CV_FONT_HERSHEY_SIMPLEX, 0.5, cv::Scalar(255), 1, 8,
false);

  //Now we overlay non-black point from img4 over img3 (the color image we see)
  uint8_t* pixelPtr = (uint8_t*) img2.data;
  uint8_t* img4Ptr = (uint8_t*) img->data;
  int cn = img2.channels();
  unsigned int tmp;
```

```
  Scalar_<uint8_t> bgrPixel, img4Pixel;
  for (unsigned int i = 0; i < img->cols; i++) {
    for (unsigned int u = 0; u < img->rows; u++) {
      tmp = u * img->cols * cn + i * cn;
      img4Pixel.val[0] = img4Ptr[tmp + 0]; // B
      img4Pixel.val[1] = img4Ptr[tmp + 1]; // G
      img4Pixel.val[2] = img4Ptr[tmp + 2]; // R
      if ((img4Pixel.val[0] != 0) || (img4Pixel.val[1] != 0) || (img4Pixel.val[2] != 0)) {
        tmp = u * img2.cols * cn + i * cn;
        pixelPtr[tmp + 0] = img4Pixel.val[0];
        pixelPtr[tmp + 1] = img4Pixel.val[1];
        pixelPtr[tmp + 2] = img4Pixel.val[2];
      }
    }
  }

  //resize(subtracted_frame, subtracted_frame_shrunk, Size(subtracted_frame.cols / 2,
subtracted_frame.rows / 2), 0, 0, INTER_AREA);
  //imshow("mainWin2", subtracted_frame_shrunk);
  //rectangle(img3, interiorLimit, Scalar(0, 255, 0));
  //resize(img3, im_out, Size(img3.cols / 2, img3.rows / 2), 0, 0, INTER_AREA);
  imshow("CurrentView", img2);
  waitKey(1);
}
```