Homework 02

Deepak Sharma
CSCI 731 Advance Computer Vision

July 2, 2017

Part A, Supervised Watershed Segmentation

Solution:
As professor has discussed in lecture, strong edge in the a image work as regularizer and similarity between pixels guide watershed algorithm to iteratively expend. Though Gaussian smoothing can enhance similarity between pixels related to same object but it will also reduce edge strength. In order to preserve the edge strength and for performing smoothing, I decided to apply bilateral filters during the pre-processing step [1]. Then I conducted experiment to evaluate the quality of watershed segmentation with our without performing bilateral transformation. For most of the cases bilinear transformation has improved quality of segmentation.

Figure 1: FearlessGirlTinySegmentSKIN

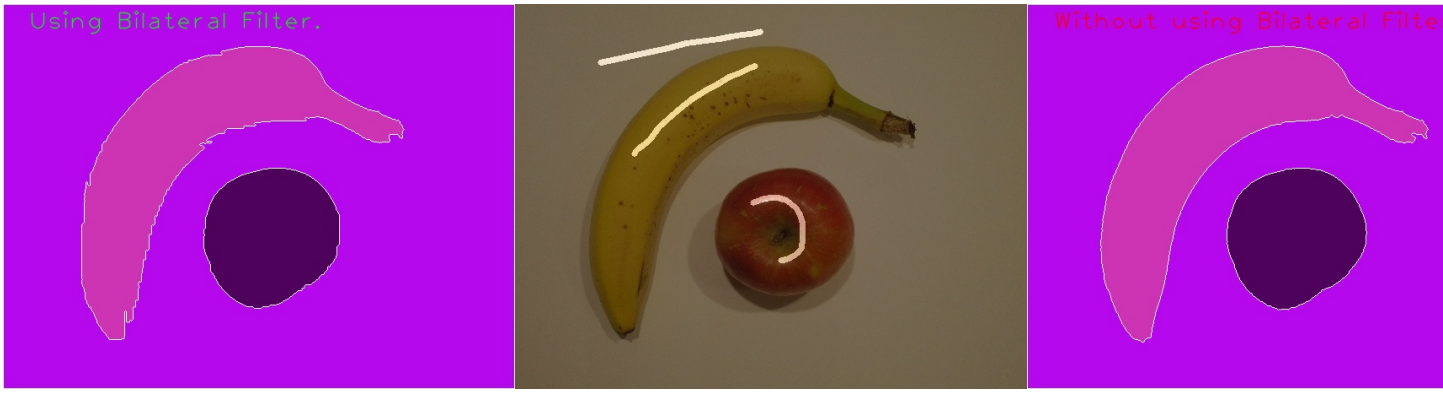

Figure 2: FruitSegmentTruitTiny

Figure 3: IMG8665SegmentPlantsTiny

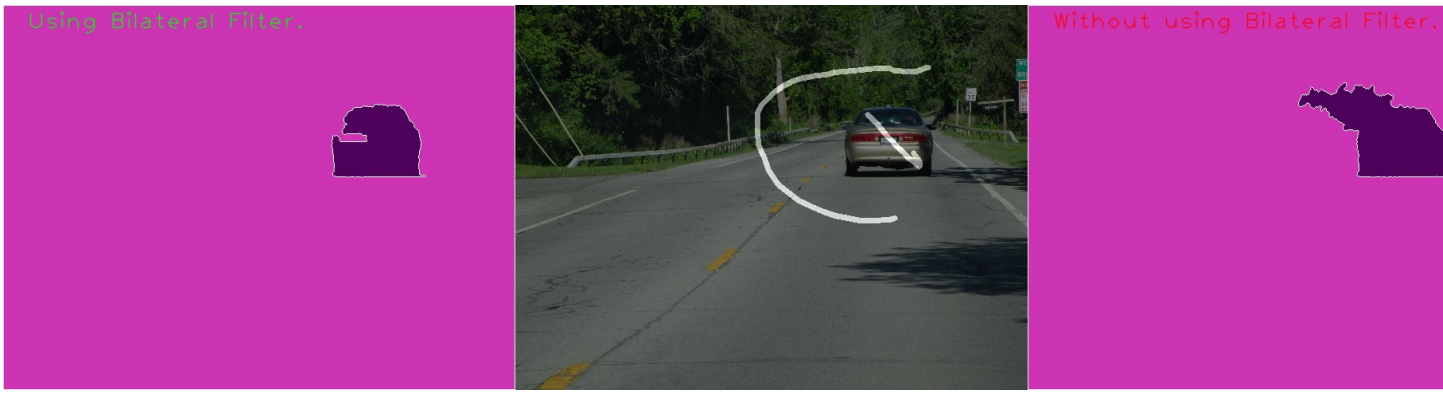

Figure 4: IMG8670SegmentCarTiny

Figure 5: IMG8735SegmentLicensePlateTiny

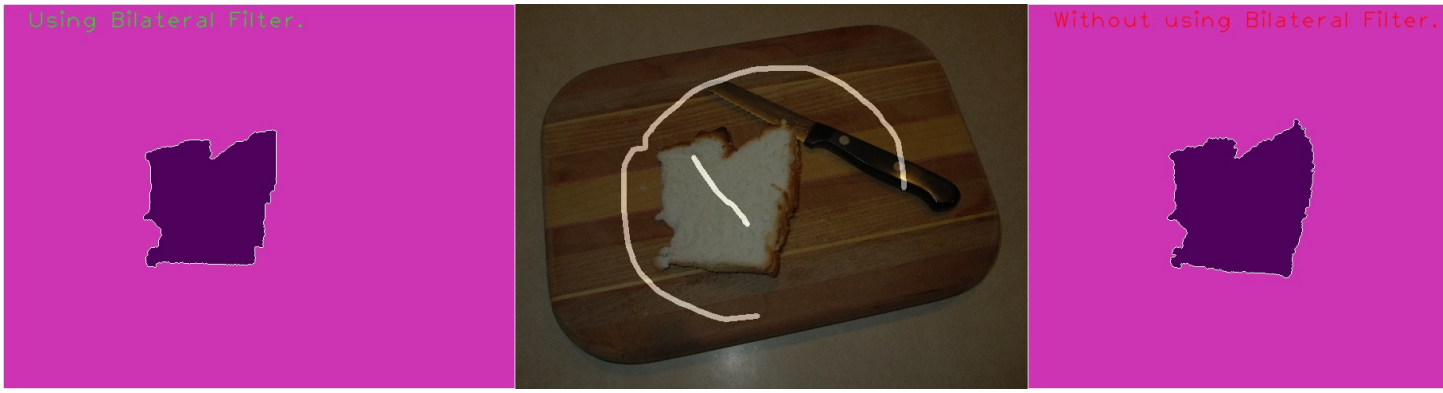

Figure 6: IMG8773SegmentFoodTiny

Figure 7: IMG9105BirdsSegmentBothBirdsTiny

Figure 8: IMGDEERSegmentDeerTiny

Figure 9: IMGSEDIMENTSegmentDirtTiny

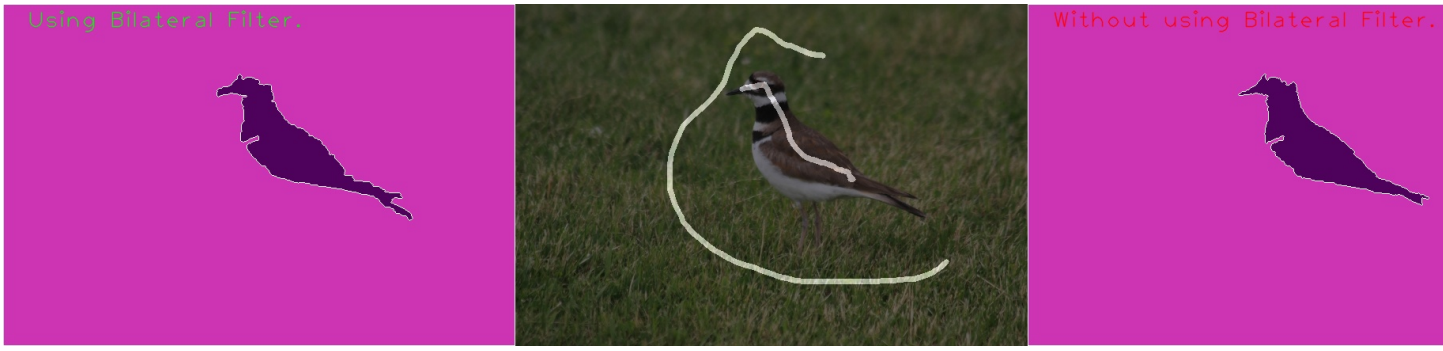

Figure 10: IMG9021MOTHERBIRDSegmentBirdTiny

Code explanation

Mouse event handling

```
        /*
* @brief Mouse event handler, response to the mouse actions.
* @param event, integer type event_id corresponding to the action taken by user
* @param x, y integer coordinate of curser position
* Code
   Source/Help:https://github.com/opencv/opencv/blob/master/samples/cpp/watershed.cpp
*/
static void onMouse( int event, int x, int y, int flags, void* )
{
    if( x < 0 || x >= image.cols || y < 0 || y >= image.rows )
        return;
    // if left button of mouse has been clicked up
    if( event == EVENT_LBUTTONUP || !(flags & EVENT_FLAG_LBUTTON) )
        prevPt = Point(-1,-1);
    // if left button of mouse has been clicked down
    else if( event == EVENT_LBUTTONDOWN )
        prevPt = Point(x,y);
    // if mouse has been moved and if this is the target movement
    else if( event == EVENT_MOUSEMOVE && (flags & EVENT_FLAG_LBUTTON) )
    {
        Point pt(x, y); // created Point object with mouse click location,
        //before it started movement
        if( prevPt.x < 0 )
            prevPt = pt; // if selected point is out of frame then set it to
        //previous location
        //created a line segment between current and previous location of
        //the mouse in the binary image markermask(Global Variable).
        line( markerMask, prevPt, pt, Scalar::all(255), 5, 8, 0 );
        //For showing the mouse movement to the user, also overlayed line
        //segment on original image.
        line( image, prevPt, pt, Scalar::all(255), 5, 8, 0 );
        prevPt = pt;
        imshow("image", image);
    }
}
```

[6]

Mask Generation

```
        vector<vector<Point> > contours;
         //Contains all countorus by holding contour points for
         //each countor, output of findContours method will be stored
         //in this container.
         //Contour represents a curve using set of points.
         //findContours generate these set of points for a given
         //binary image.
         vector<Vec4i> hierarchy;
```

```
// Contain contour hierachy(releationship between contours)
//In the presence of mutiple contours, for defining the
//relationship between contours, findcontor povide relationship
//between contorus in hierachy container.

findContours(markerMask, contours, hierarchy,
       RETR_CCOMP, CHAIN_APPROX_SIMPLE);
if( contours.empty() ){
    cout<<"No point is present in contoures"<<endl;
    continue;}
Mat markers(markerMask.size(), CV_32S);
markers = Scalar::all(0);
int idx = 0;
for( ; idx >= 0; idx = hierarchy[idx][0], compCount++ )
    drawContours(markers, contours, idx,
           Scalar::all(compCount+1), -1, 8, hierarchy, INT_MAX);
//creating a mask image where countours have been given
// different integer numbers, this number has been used as
// intensity value for all the points associated with that contour.
```

[6]

Source code credit

1. Mouse event handling and connected component mask generation [6].
2. Image smoothing, bilateral transform, edge detection, binarization, put text, image stitching [1, 4].
3. Exception Handling [5, 8].
4. Codding style and guidelines [2].
5. Image channel extraction [3].
6. Assertion of a file path [7].

References

[1] Gary Bradski Adrian Kaehler. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library 1st.* O'Reilly Media, Inc., 2 edition, 2016. ISBN 1491937998 978149193799.

[2] Errin Fulp. C++ coding guidelines, 2006. URL `http://csweb.cs.wfu.edu/~fulp/CSC112/codeStyle.html`.

[3] Haris. how do i separate the channels of an rgb image and save each one, using the 2.4.9 version of opencv?, 2015.

[4] *The OpenCV Reference Manual.* Itseez, 3.2 edition.

[5] Nsanders Terry Li, John Dibling. How to throw a c++ exception, 2012. URL https://stackoverflow.com/questions/8480640/how-to-throw-a-c-exception.

[6] avdmitry Valery Tyumen, Steven Puttemans. Demo implementation of the watershed segmentation algorithm in opencv, 2013. URL http://docs.opencv.org/trunk/d8/da9/watershed_8cpp-example.html.

[7] Pherric Oxide Vincent, Pevik. Fastest way to check if a file exist using standard c++/c++11/c?, 2012.

[8] Jonathan Wakely. Throw exception if the file does not exist in constructor and try/catch it when creating an object in main(), if good - start using the object, 2016.