ПОИСК ОКРУЖНОСТЕЙ
Лекция 11.

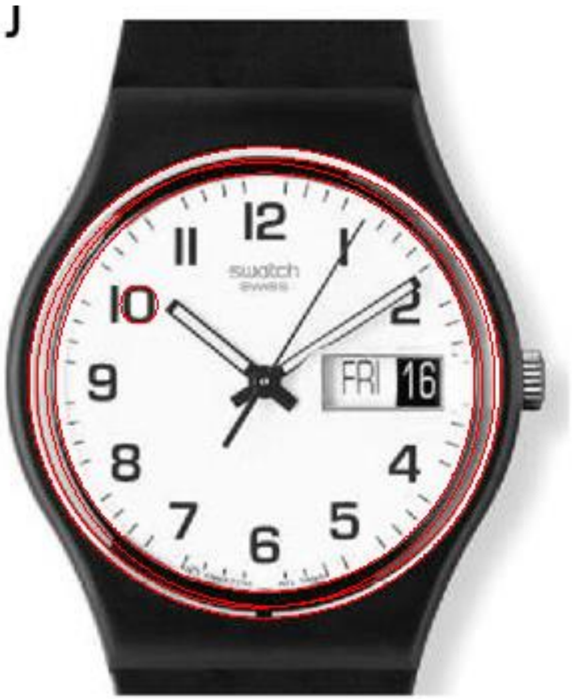
Преподаватель: Сибирцева Елена

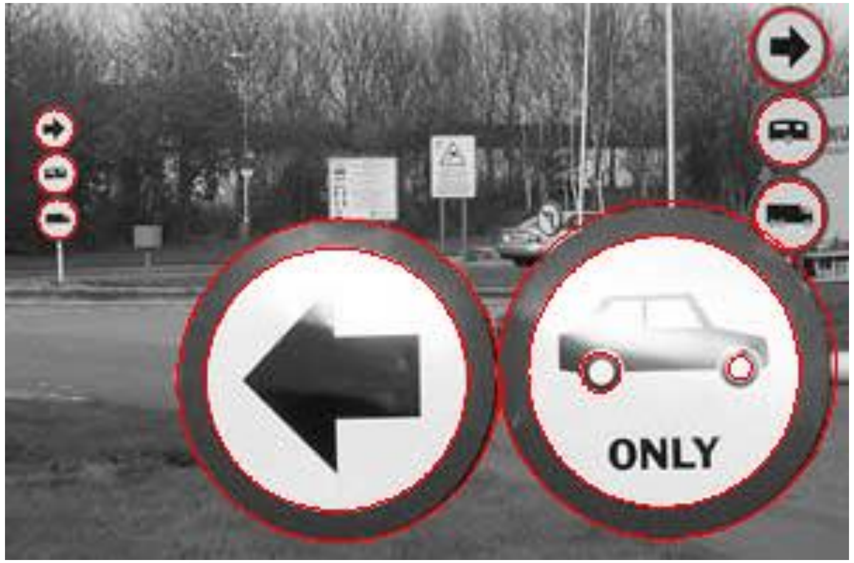
elsibirtseva@gmail.com

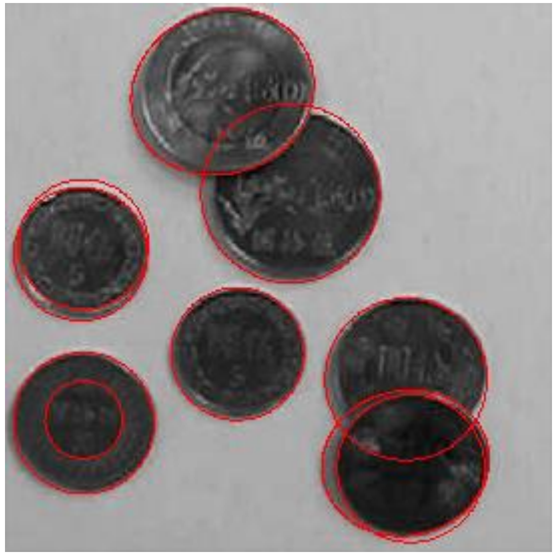
Refs

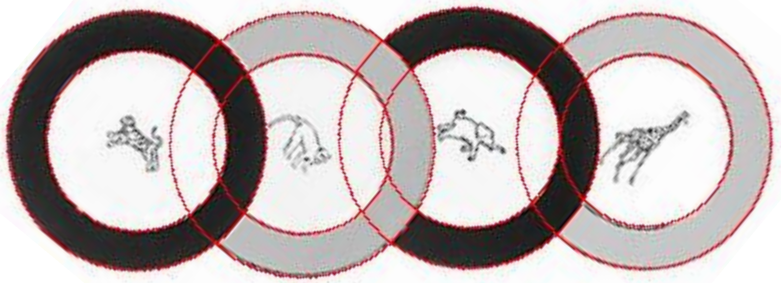
1. C. Akinlar and C. Topal, EDCircles: Real-time Circle Detection by Edge Drawing (ED), International Conference on Acoustics, Speech and Signal Processing (ICAASP), 2012.
2. C. Topal, C. Akinlar, and Y. Genc, Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection, Proceedings of the ICPR, pp. 2424-2427, August 2010.
3. C. Akinlar and C. Topal, EDPF: A Real-time Parameter-free Edge Segment Detector with a False Detection Control, International Journal of Pattern Recognition and Artificial Intelligence, 2012.
4. C. Akinlar and C. Topal, EDLines: A real-time line segment detector with a false detection control, Pattern Recognition Letters, 32(13), 2011.
5. C. Akinlar and C. Topal, EDLines: Real-Time Line Segment Detection by Edge Drawing, International Conference on Image Processing (ICIP), 2011.
6. A. Desoulneux, L. Moisan, and J.M. Morel, Gestalt theory and Computer Vision, bool chapter in Seeing, Thinking and Knowing, pp. 71-101, Kluwer Academic Publishers, 2004.
7. A. Desolneux, L. Moisan, and J.M. Morel, From Gestalt Theory to Image Analysis: A Probabilistic Approach, Springer, 2008.
8. A. Desolneux, L. Moisan, and J.M. Morel, Meaningful Alignments, International Journal of Computer Vision, vol. 40, no. 1, pp. 7-23, 2000.
9. K.L. Chung, Y.H. Huang, S.M. Shen, A.S. Krylov, D.V. Yurin, and E.V. Semeikina, “Efficient sampling strategy and refinement strategy for randomized circle detection,” Pattern Recognition, vol. 45, pp. 252-263, 2012.
10. V. Ayala-Ramirez, C.H. Garcia-Capulin, A. Peres-Garcia, and R.E. Sanchez-Yanez, “Circle detection on images using genetic algorithms,” Pattern Recognition Letters, vol. 27, no. 6, pp. 652-657, 2006

Output









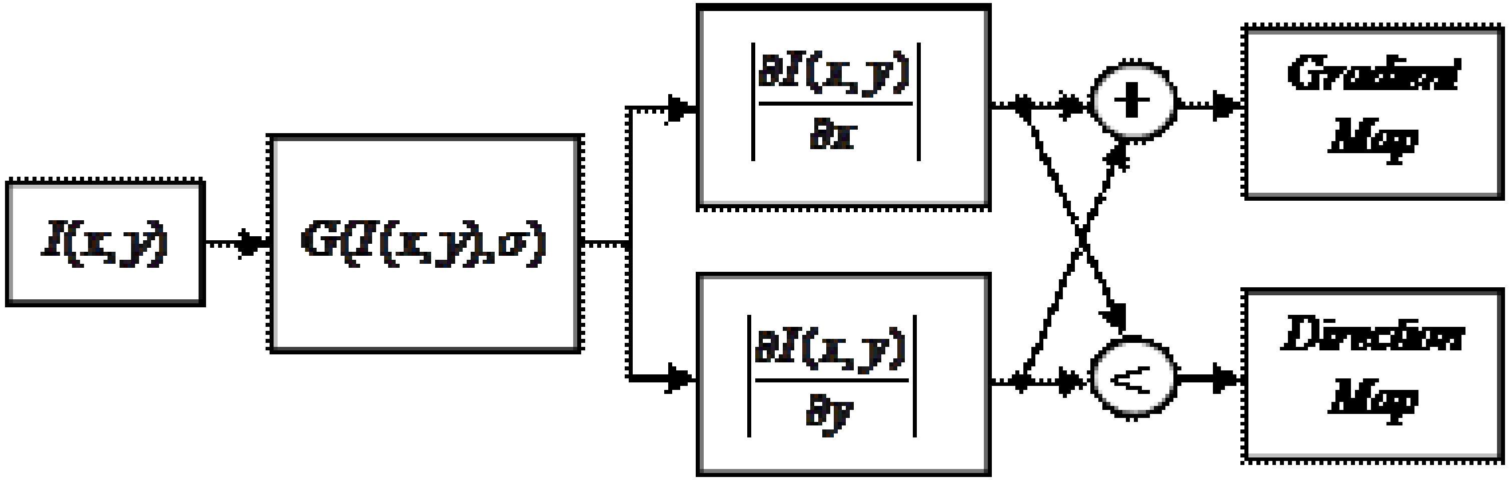
Алгоритм (верхушка айсберга)

- I. Detect edge segments by EDPF and extract complete circles and ellipses
- II. Convert the remaining edge segments into line segments
- III. Detect arcs by combining line segments
- IV. Join arcs to detect circle candidates
- V. Join the remaining arcs to detect near-circular ellipse candidates
- VI. Validate the candidate circles/ellipses using the Helmholtz principle
- VII. Output the valid remaining circles/ellipses

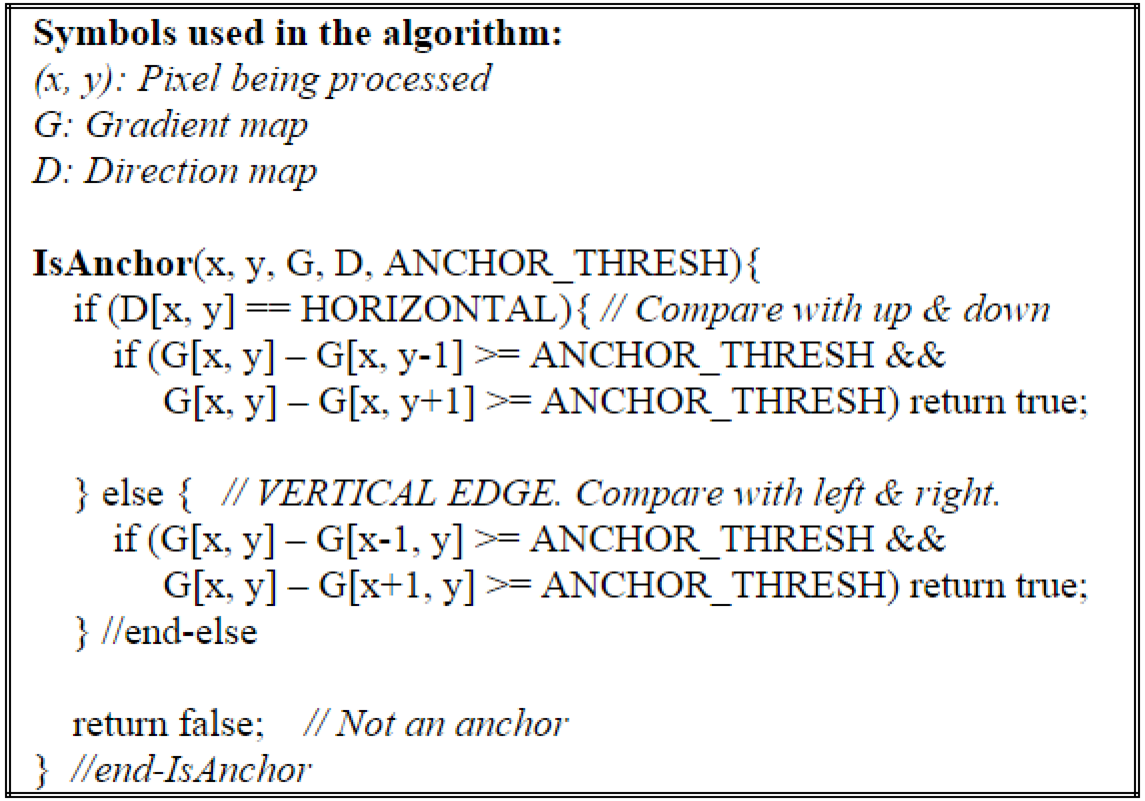
I. EDPF

- http://www.youtube.com/watch?v=-Bpb_OLfOts#t=52

- (1) Suppression of noise by Gaussian filtering,
- (2) Computation of the gradient magnitude and edge direction maps,
- (3) Extraction of the anchors (peaks of the gradient map),
- (4) Linking of the anchors by smart routing to compute the final edge map.


$I(x,y)$
$G(I(x,y),\sigma)$
$\left|\frac{\partial I(x,y)}{\partial x}\right|$
$\left|\frac{\partial I(x,y)}{\partial y}\right|$
+
<
Gradient Map
Direction Map

I. EDPF. Anchors extraction



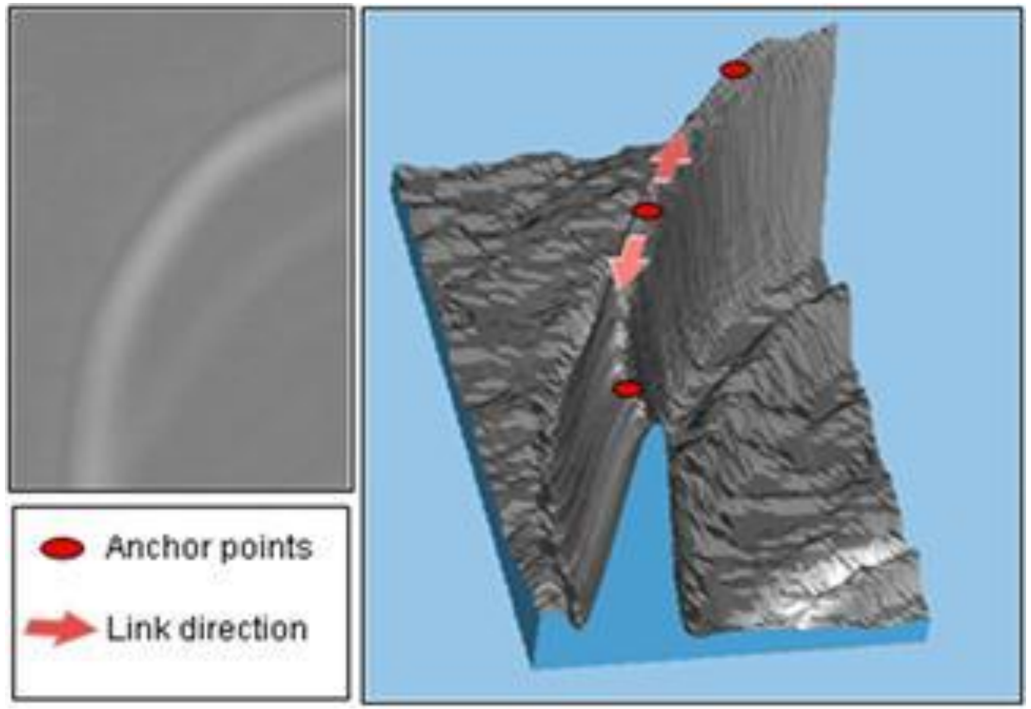
Symbols used in the algorithm:

(x, y): Pixel being processed

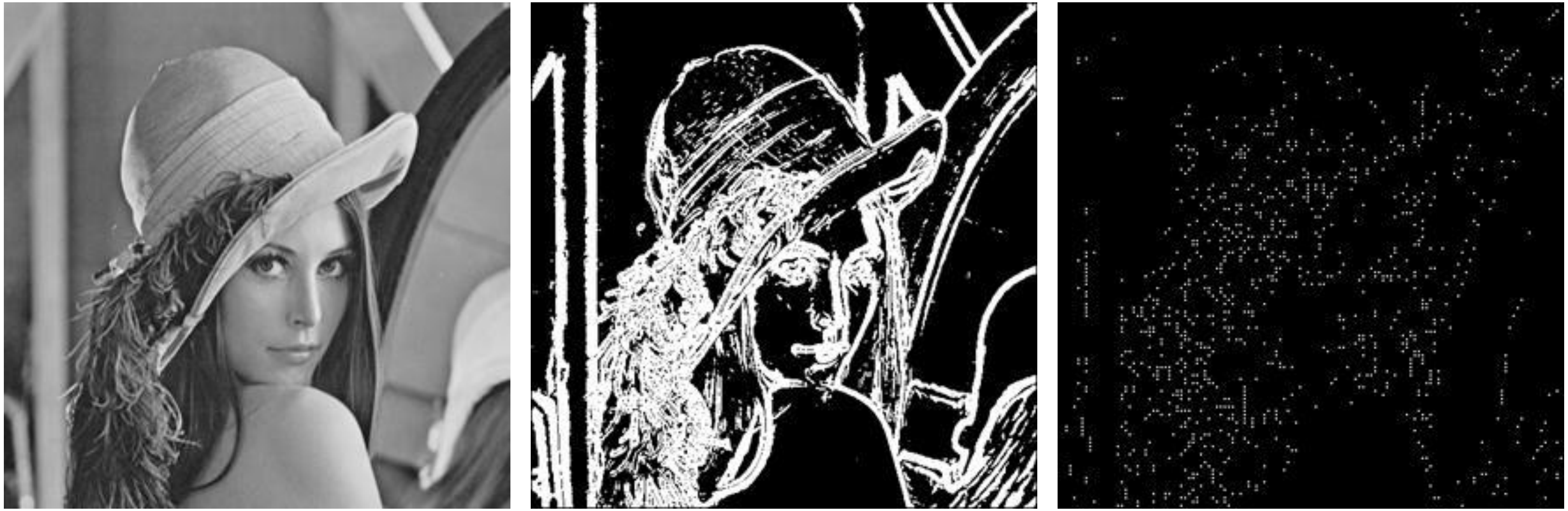
G: Gradient map

D: Direction map

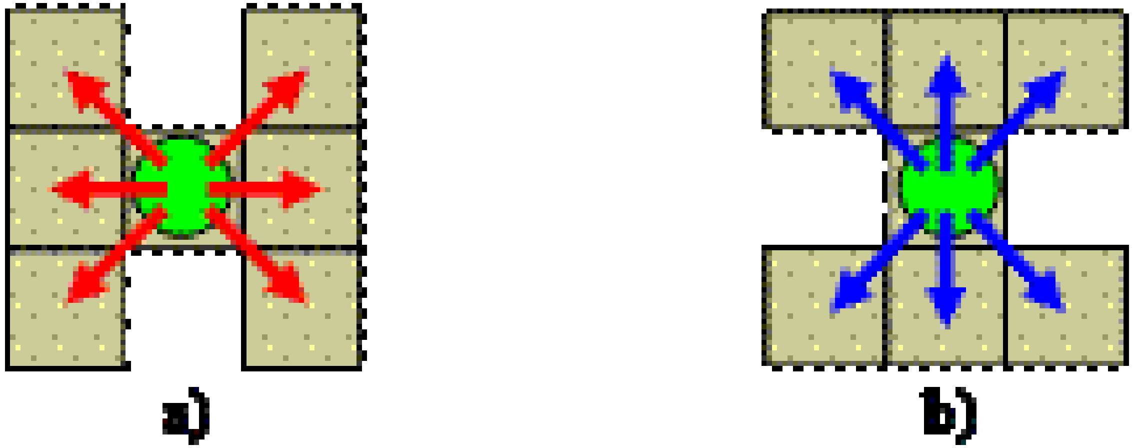
```
IsAnchor(x, y, G, D, ANCHOR_THRESH){
   if (D[x, y] == HORIZONTAL){ // Compare with up & down
      if (G[x, y] – G[x, y-1] >= ANCHOR_THRESH &&
          G[x, y] – G[x, y+1] >= ANCHOR_THRESH) return true;

   } else {   // VERTICAL EDGE. Compare with left & right.
      if (G[x, y] – G[x-1, y] >= ANCHOR_THRESH &&
          G[x, y] – G[x+1, y] >= ANCHOR_THRESH) return true;
   } //end-else

   return false;    // Not an anchor
}  //end-IsAnchor
```



I. EDPF. Anchors extraction

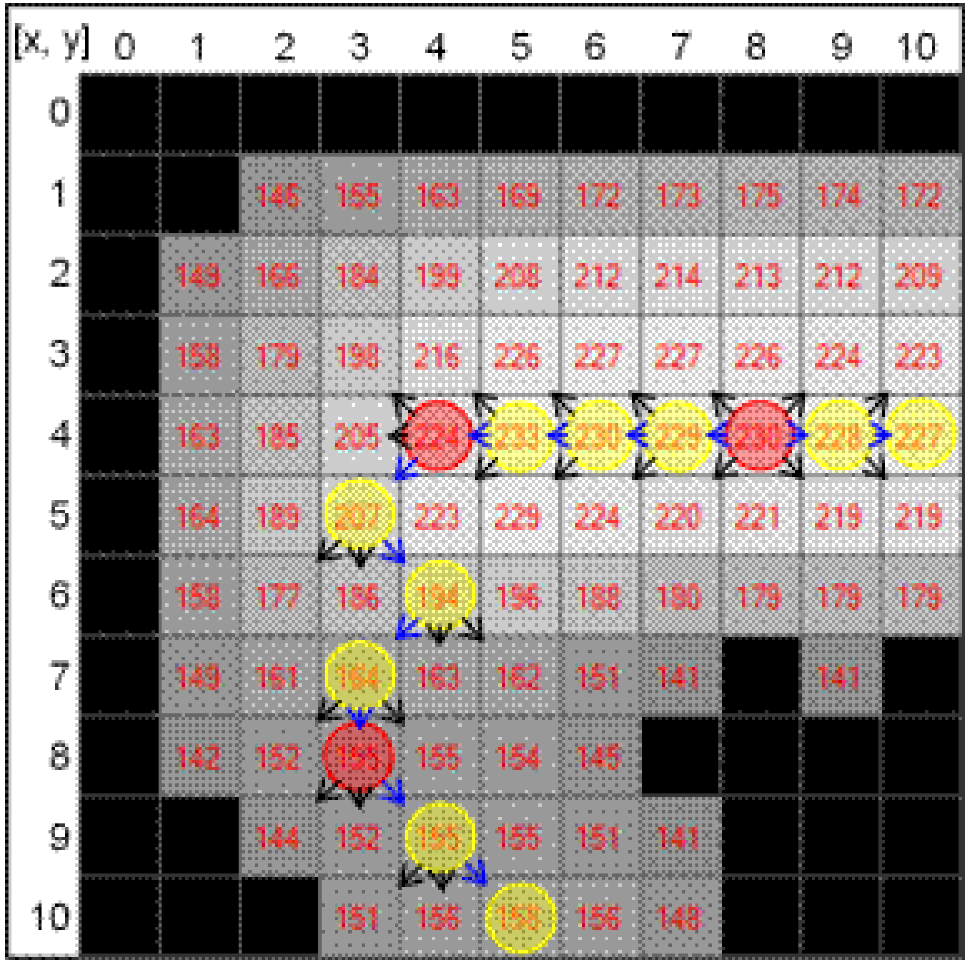


I. EDPF. Smart Routing



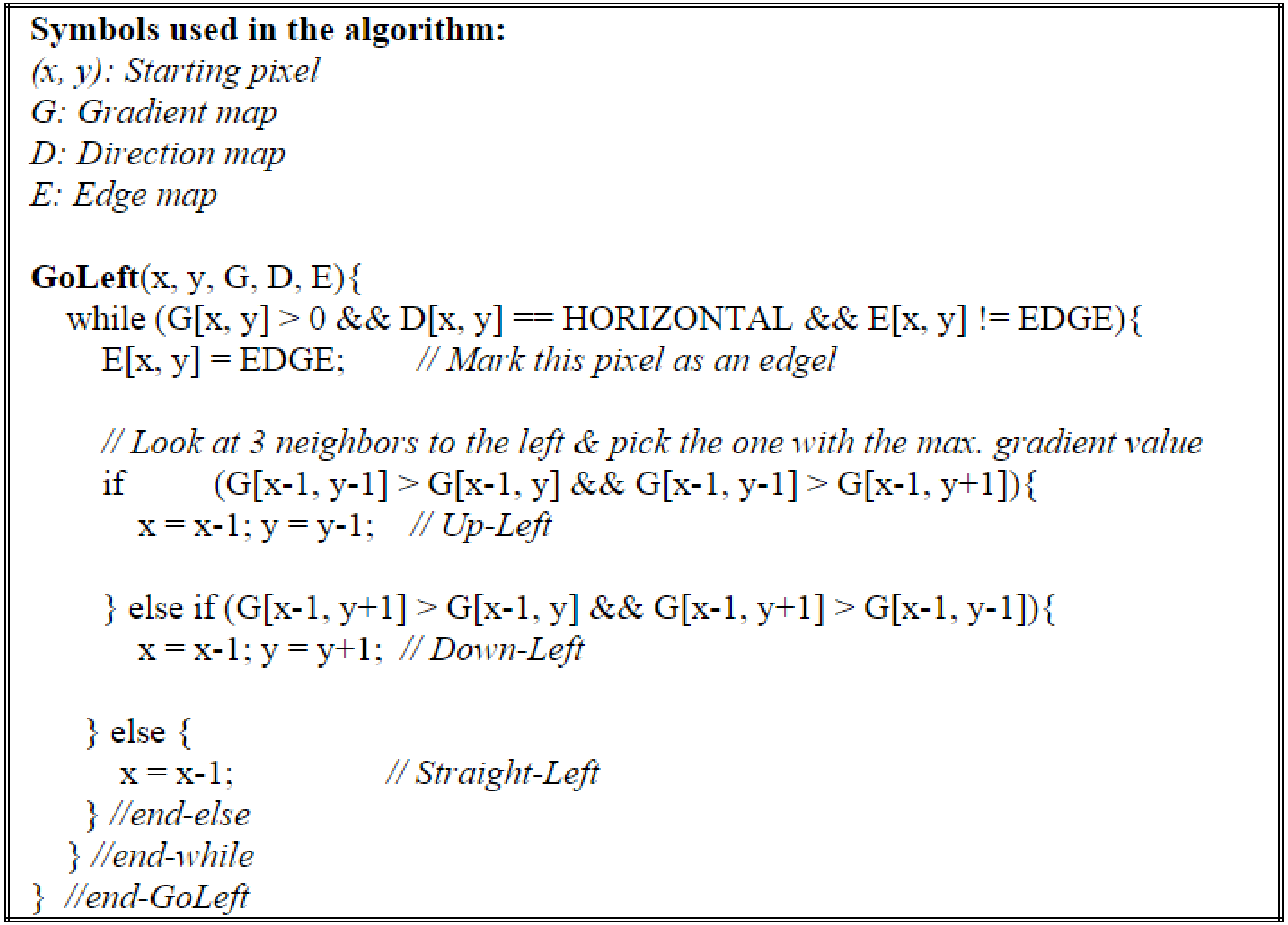
(a) Horizontal, (b) Vertical walks in smart routing.

I. EDPF. Smart Routing



[x, y]	0	1	2	3	4	5	6	7	8	9	10
0											
1			146	155	163	169	172	173	175	174	172
2		149	166	184	199	208	212	214	213	212	209
3		158	179	198	216	226	227	227	226	224	223
4		163	185	205	224	233	230	229	230	228	227
5		164	189	207	223	229	224	220	221	219	219
6		158	177	186	194	196	188	180	179	179	179
7		149	161	164	163	162	151	141		141	
8		142	152	158	155	154	145				
9			144	152	155	155	151	141			
10				151	156	158	156	148			

I. EDPF. Smart Routing

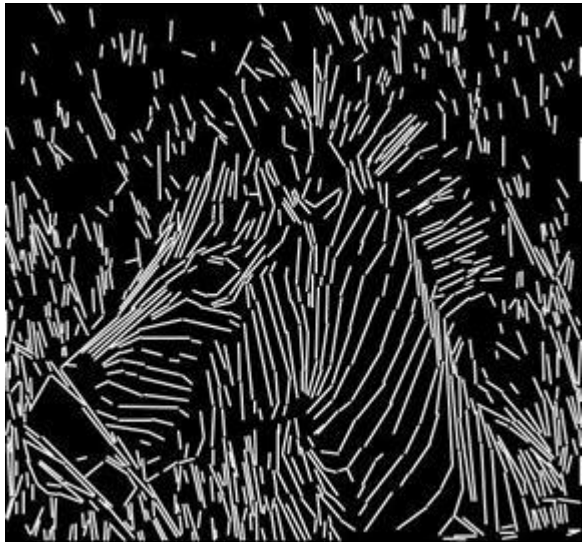


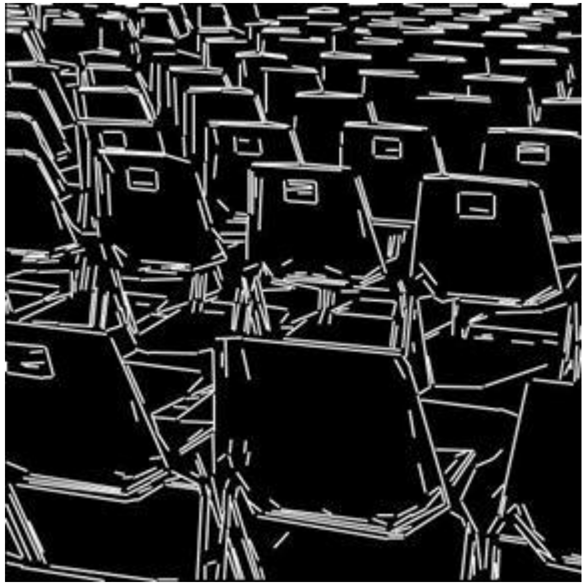
Symbols used in the algorithm:

(x, y): Starting pixel
G: Gradient map
D: Direction map
E: Edge map

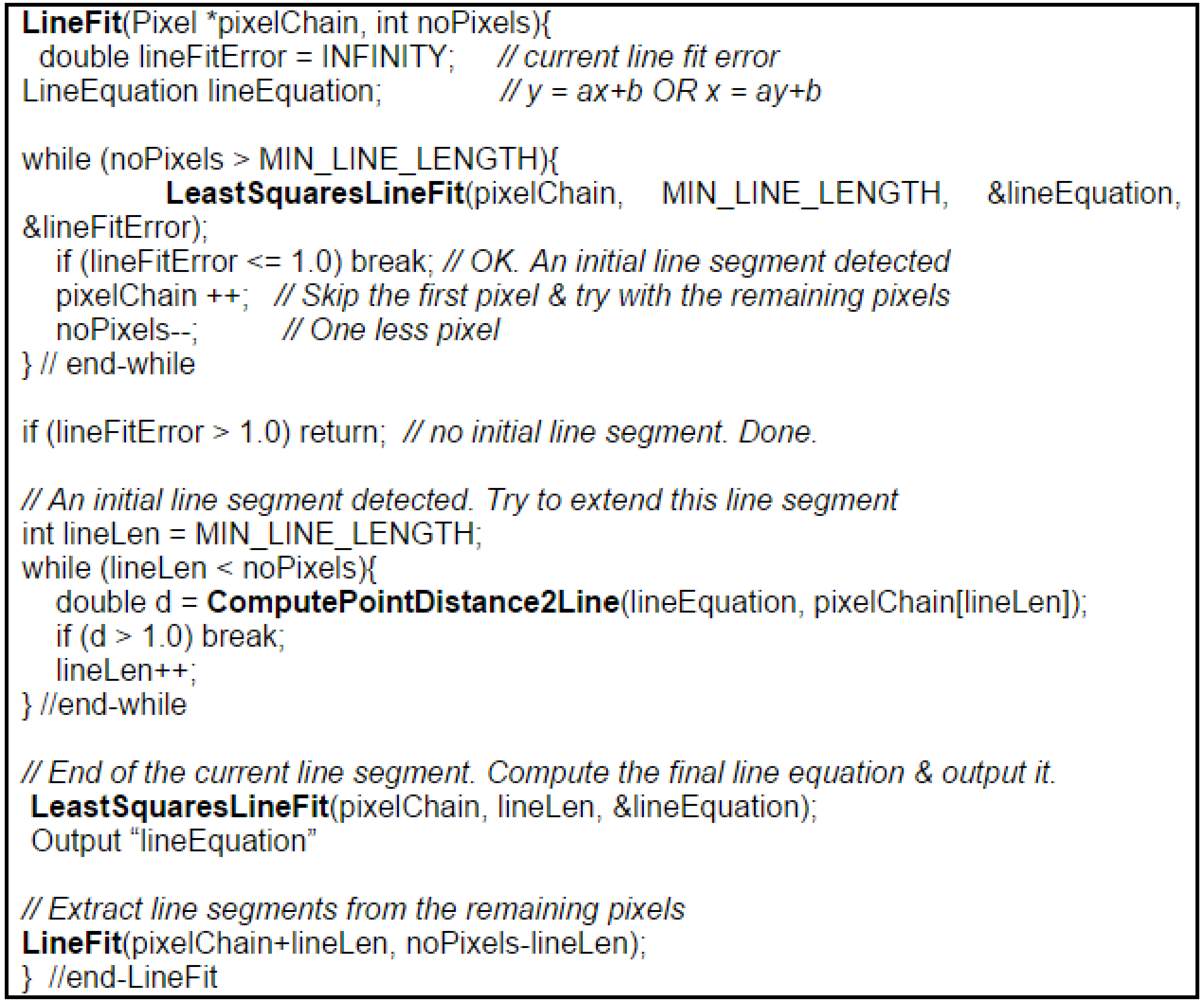
```
GoLeft(x, y, G, D, E){
  while (G[x, y] > 0 && D[x, y] == HORIZONTAL && E[x, y] != EDGE){
    E[x, y] = EDGE;      // Mark this pixel as an edgel

    // Look at 3 neighbors to the left & pick the one with the max. gradient value
    if       (G[x-1, y-1] > G[x-1, y] && G[x-1, y-1] > G[x-1, y+1]){
      x = x-1; y = y-1;    // Up-Left

    } else if (G[x-1, y+1] > G[x-1, y] && G[x-1, y+1] > G[x-1, y-1]){
      x = x-1; y = y+1;  // Down-Left

   } else {
     x = x-1;              // Straight-Left
   } //end-else
  } //end-while
} //end-GoLeft
```

II. Line fitting



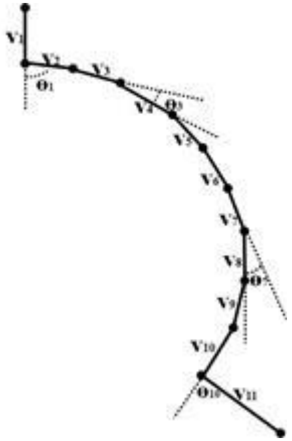


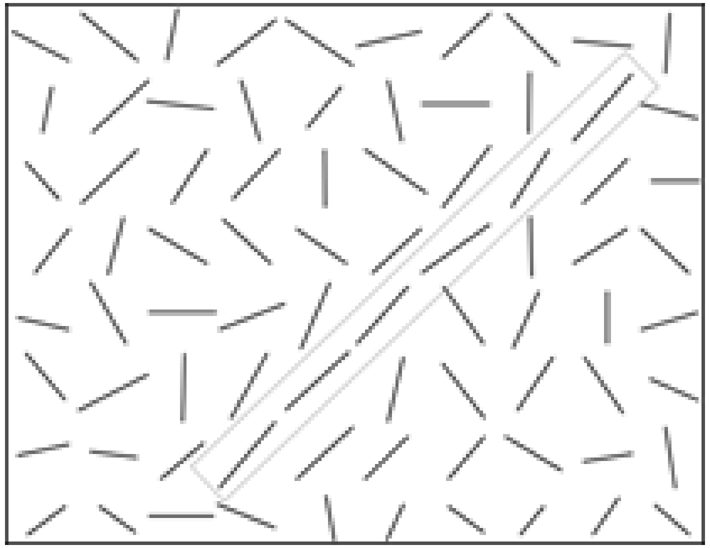
II. Line fitting

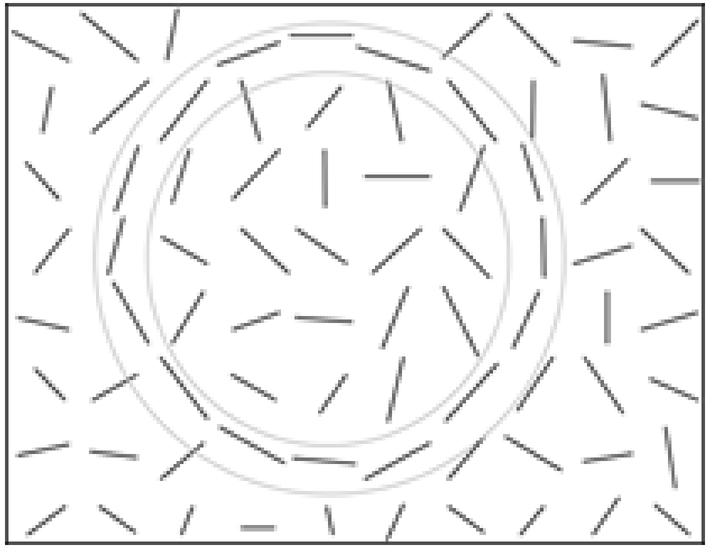


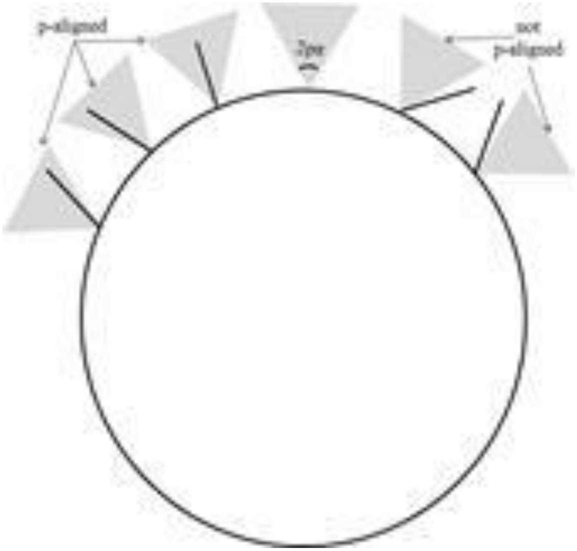
```
LineFit(Pixel *pixelChain, int noPixels){
  double lineFitError = INFINITY;     // current line fit error
LineEquation lineEquation;            // y = ax+b OR x = ay+b

while (noPixels > MIN_LINE_LENGTH){
                LeastSquaresLineFit(pixelChain,   MIN_LINE_LENGTH,   &lineEquation,
&lineFitError);
    if (lineFitError <= 1.0) break; // OK. An initial line segment detected
    pixelChain ++;   // Skip the first pixel & try with the remaining pixels
    noPixels--;         // One less pixel
} // end-while

if (lineFitError > 1.0) return;  // no initial line segment. Done.

// An initial line segment detected. Try to extend this line segment
int lineLen = MIN_LINE_LENGTH;
while (lineLen < noPixels){
    double d = ComputePointDistance2Line(lineEquation, pixelChain[lineLen]);
    if (d > 1.0) break;
    lineLen++;
} //end-while

// End of the current line segment. Compute the final line equation & output it.
 LeastSquaresLineFit(pixelChain, lineLen, &lineEquation);
 Output "lineEquation"

// Extract line segments from the remaining pixels
LineFit(pixelChain+lineLen, noPixels-lineLen);
}  //end-LineFit
```

III. Circle fitting

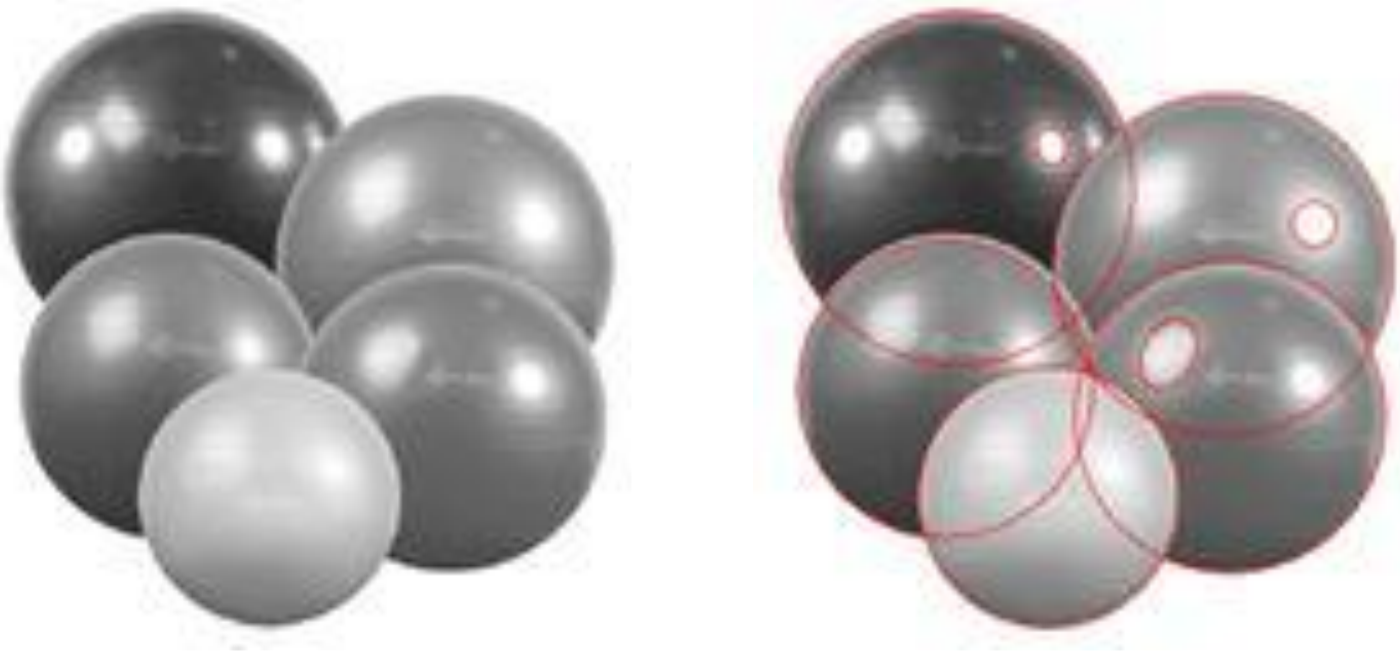




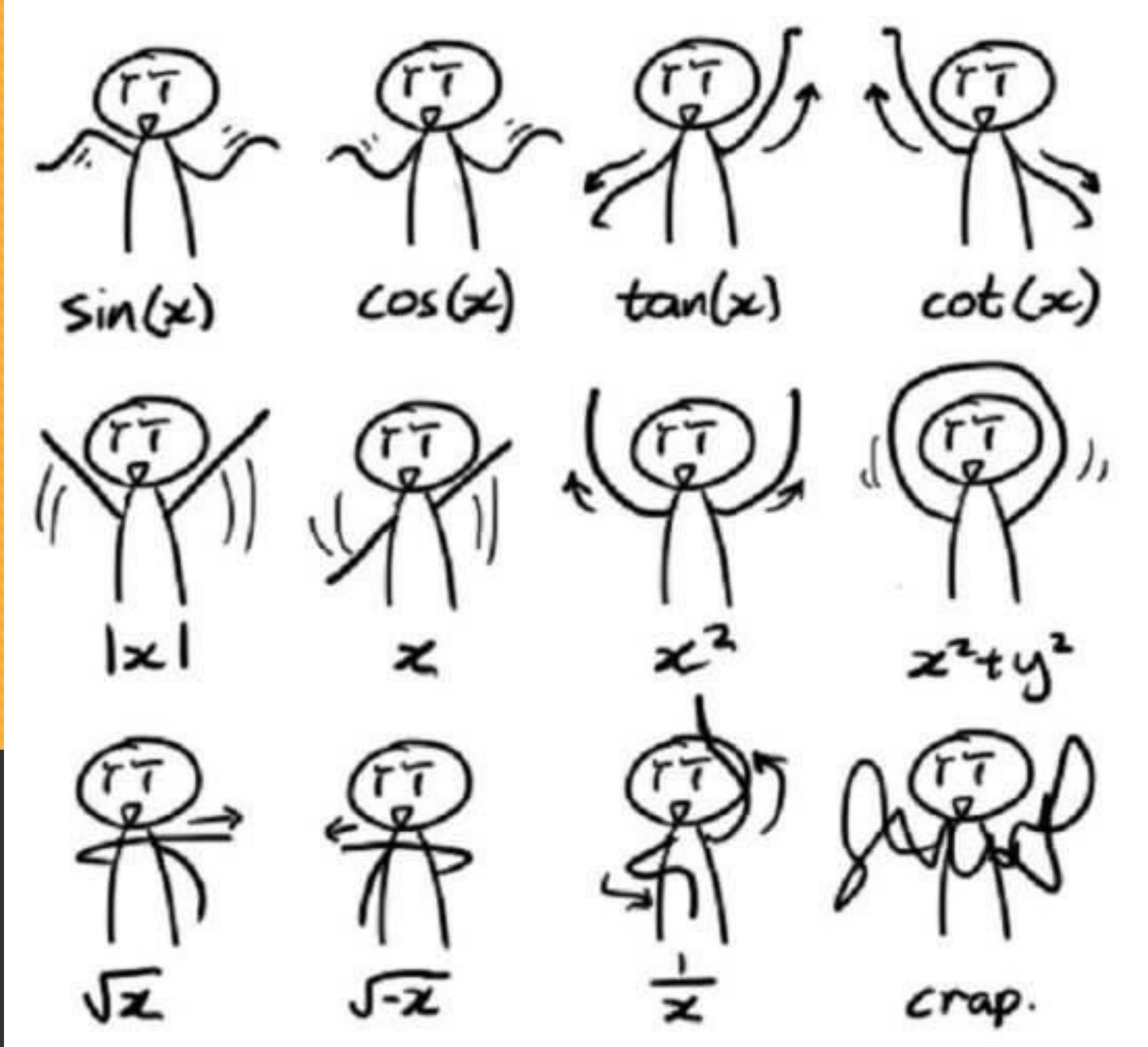




Results



В следующих сериях…


sin(x)
cos(x)
tan(x)
cot(x)
|x|
x
x²
x²+y²
√x
√-x
1/x
crap.