COMPSCI 527 Homework 5

Cody Lieu, Yixin Lin

November 18, 2015

Problem 1(a)

Points tracked well: 1, 3, 4, 5

Points lost during tracking: 2

Points tracked but don't correspond to fixed points in the world: 6

Point 2 was lost during tracking, which is clearly not satisfactory because during reconstruction, that point cannot be found in the point cloud and the 3d information is lost.

Point 6 was tracked but don't correspond to a fixed point in the world, which would be problematic in reconstruction because the point cloud would be constructed incorrectly.

Problem 1(b)

The last frame in which all features are present is frame 5.

Feature	cond(H)	$\sigma_m in(H)$
1	3.19	49.82
2	7.14	0.14
3	5.06	55.04
4	7.54	41.91
5	3.45	161.35
6	10.88	11.20

Problem 1(c)

The $\sigma_{min}(H)$ of feature 2 is very close to 0.

TODO: explain why

Problem 1(d)

	Newton	gradient descent	grid
ssd evals	169	837	5124

Problem 1(e)

TODO

Problem 1(f)

The camera moved forward, since the tracking points look like they got closer (the image got slightly larger, and the points "expanded").

Problem 2(a)

Gradient:

$$\begin{bmatrix} -2a + 4bx_1^3 - 4bx_1x_2 + 2x_1 \\ 2b(x_2 - x_1^2) \end{bmatrix}$$

Hessian:

$$\begin{bmatrix} 12bx_1^2 - 4bx_2 + 2 & -4bx_1 \\ -4bx_1 & 2b \end{bmatrix}$$

Problem 2(b)

$\mathbf{x}^* =$

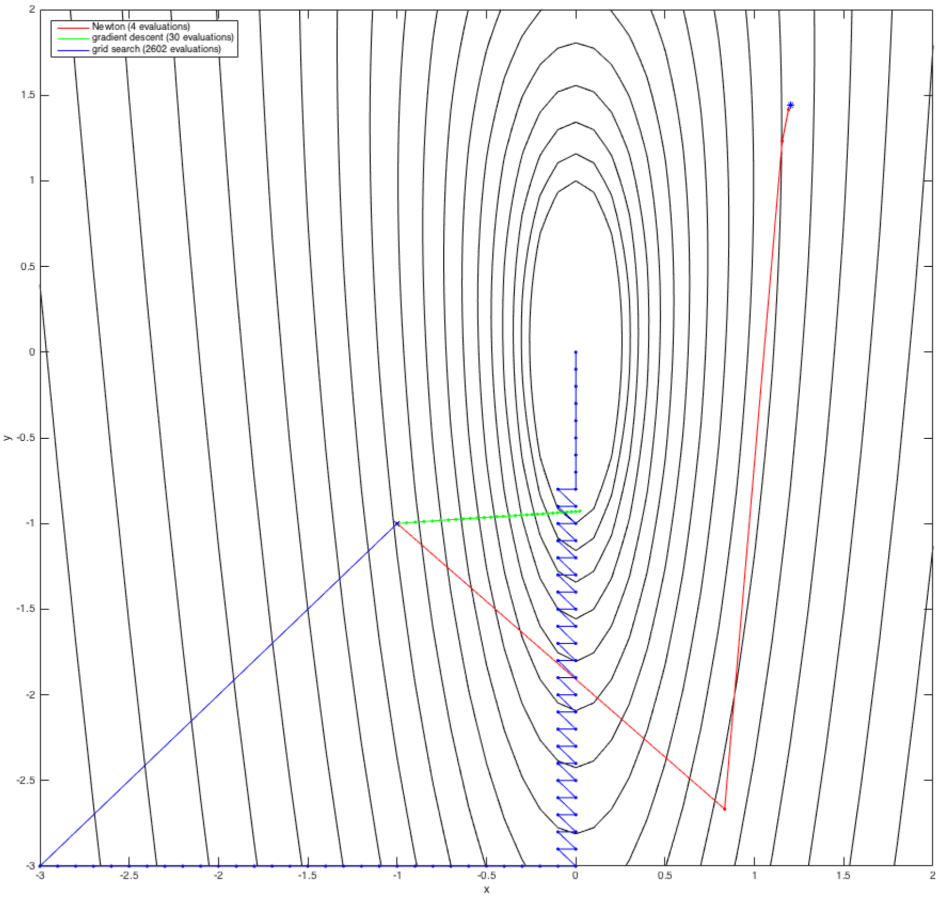
$$\begin{bmatrix} a \\ a^2 \end{bmatrix}$$

$f(\mathbf{x}^*) = 0$

Problem 2(c)

```
function cost = bananaCost(a, b)

cost.f = @banana;
cost.OK = @OK;
cost.a = a;
cost.b = b;
```

```
    function point = banana(x, cost, order)
        % YOUR CODE HERE

        if order > 0 && ~isvector(x)
            error('if order is nonzero, x must be a column vector')
        end

        x = double(x);
        cost.a = double(cost.a);
        cost.b = double(cost.b);

        point.x = x;
        point.y = (cost.a * - x(1, :)).^2 + cost.b * (x(2, :) - x(1, :).^2).^2;

                if order>=1
                        point.g(1,1) = -2 * cost.a + 4 * cost.b * x(1)^3 - 4 * cost.b * x(1) * x(2) + 2 *
                        point.g(2,1) = 2 * cost.b * (x(2) - x(1)^2);
                        if order>=2
                                point.H(1,1) = 12 * cost.b * x(1)^2 + 2;
                                point.H(1,2) = -4 * cost.b * x(1);
                                point.H(2,1) = point.H(1,2);
                                point.H(2,2) = 2 * cost.b;
                        end
                end

    end

    function good = OK(~, ~, ~)
        % YOUR CODE HERE
        good = true;
    end
end
```


Newton (4 evaluations)
gradient descent (30 evaluations)
grid search (2602 evaluations)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
y
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x