Geometric Algorithms
Assignment 3

Laura Baakman (s1869140)

September 28, 2014

A

B

Line Segment Intersection

To find the intersection of the following two line segments

$$s_1 = \lambda_1 \cdot \mathbf{P_1} + (1 - \lambda_1) \cdot \mathbf{P_2} \qquad \text{for } 0 \leq \lambda_1 \leq 1 \tag{1}$$
$$s_2 = \lambda_2 \cdot \mathbf{P_3} + (1 - \lambda_2) \cdot \mathbf{P_4} \qquad \text{for } 0 \leq \lambda_2 \leq 1 \tag{2}$$

we need to solve the equation:

$$\begin{aligned} s_1 &= s_2 \\ \lambda_1 \cdot \mathbf{P_1} + (1 - \lambda_1) \cdot \mathbf{P_2} &= \lambda_2 \cdot \mathbf{P_3} + (1 - \lambda_2) \cdot \mathbf{P_4}. \end{aligned} \tag{3}$$

Equation 3 can be solved using the Mathematica code presented in Listing 1. This results in an expression for λ_1 (Equation 4) and one for λ_2 (Equation 5).

$$\lambda_1 = -\frac{-\mathbf{P_{2,x}P_{3,y}} + \mathbf{P_{2,x}P_{4,y}} + \mathbf{P_{2,y}P_{3,x}} - \mathbf{P_{2,y}P_{4,x}} - \mathbf{P_{3,x}P_{4,y}} + \mathbf{P_{3,y}P_{4,x}}}{q} \tag{4}$$

$$\lambda_2 = -\frac{-\mathbf{P_{1,x}P_{2,y}} + \mathbf{P_{1,x}P_{4,y}} + \mathbf{P_{1,y}P_{2,x}} - \mathbf{P_{1,y}P_{4,x}} - \mathbf{P_{2,x}P_{4,y}} + \mathbf{P_{2,y}P_{4,x}}}{q} \tag{5}$$

$$\begin{aligned} q = &-\mathbf{P_{1,x}P_{3,y}} + \mathbf{P_{1,x}P_{4,y}} + \mathbf{P_{1,y}P_{3,x}} - \mathbf{P_{1,y}P_{4,x}} + \mathbf{P_{2,x}P_{3,y}} - \mathbf{P_{2,x}P_{4,y}} \\ &- \mathbf{P_{2,y}P_{3,x}} + \mathbf{P_{2,y}P_{4,x}} \end{aligned} \tag{6}$$

q is the magnitude of the cross product of the vectors $\mathbf{v_1} = \mathbf{P_2} - \mathbf{P_1}$ and $\mathbf{v_2} = \mathbf{P_4} - \mathbf{P_3}$ when the vectors $\mathbf{P_1}$ through $\mathbf{P_4}$ are extended to three-dimensional space. If q is zero the vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ are parallel and the two line segments will thus never intersect. If they are not parallel the two line segments only intersect when $\lambda_1, \lambda_2 \in [0, 1]$.

Based on the presented equations we have defined the method `line_segments_intersect` that takes two line segements defined by their endpoints and return `None` if they do not intersect and the intersection point if they do intersect. The code of that method is presented in Listing 2.

Listing 1: Mathematica code used to solve Equation 3.

```
eq1 = lam1 p1x + (1 - lam1) p2x == lam2  p3x + (1 - lam2) p4x
eq2 = lam1 p1y + (1 - lam1) p2y == lam2  p3y + (1 - lam2) p4y
Solve[eq1 == eq2 {lam1, lam2}]
```

Listing 2: The method `line_segments_intersect()`.

```
"""Module with methods that handle things related to line segments."""
from __future__ import division


def line_segments_intersect(segment_1, segment_2):
    """
    Return the point of intersection of segment one and two or none.

    Input:
        segment: List of two points, where each point is a list
            with the x and y coordinate of an endpoint of the line segment.
    """
    [p1, p2] = segment_1
    [p3, p4] = segment_2
    q = (
        -(p1[1]*p3[0]) + p2[1]*p3[0] + p1[0]*p3[1] - p2[0]*p3[1]
        + p1[1]*p4[0] - p2[1]*p4[0] - p1[0]*p4[1] + p2[0]*p4[1]
    )
    if(q):
        lambda_1 = (
            (p2[1] * p3[0] - p2[0] * p3[1] - p2[1] * p4[0] + p3[1] * p4[0]
                + p2[0] * p4[1] - p3[0] * p4[1]) / q
        )
        if(lambda_1 >= 0 and lambda_1 <= 1):
            lambda_2 = (
                - (p1[1] * p2[0] - p1[0] * p2[1] - p1[1] * p4[0] + p2[1] * p4[0]
                    + p1[0] * p4[1] - p2[0] * p4[1]) / q
            )
            if(lambda_2 >= 0 and lambda_2 <= 1):
                x = lambda_1 * p1[0] + (1 - lambda_1) * p2[0]
                y = lambda_1 * p1[1] + (1 - lambda_1) * p2[1]
                return [
                    lambda_1 * p1[0] + (1 - lambda_1) * p2[0],
                    lambda_1 * p1[1] + (1 - lambda_1) * p2[1]
                ]
    return None
```

Projection of a Point on a Plane