Method for Translating a Point in One Plane to Another Plane, Given a Set of Corresponding Control Points in Each Plane

Justin A. Parr
q3ej7ejsjb@snkmail.com
http://justinparrtech.com
Version 1.0, 9/24/2016

Executive Summary

Assuming that we start with a set of control points in plane *A*, and a corresponding set of control points in plane *B*, this document describes a method for translating an arbitrary point *P* from plane *A* to its corresponding location in plane *B*.

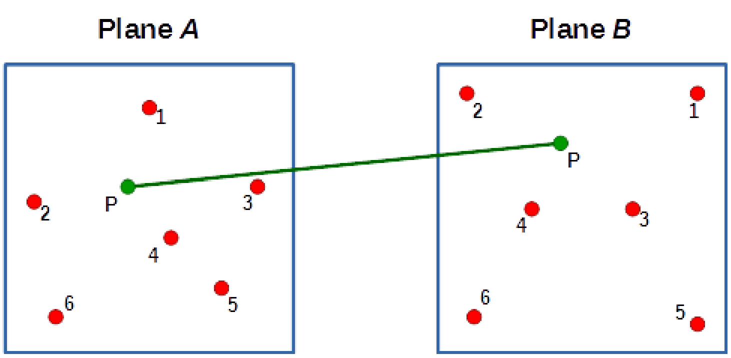


Illustration 1: Using a set of points in Plane A, and a corresponding set of points in Plane B, Point P in Plane A is translated to its new position in Plane B

Notation

Through the course of this document, we will be using the following notation.

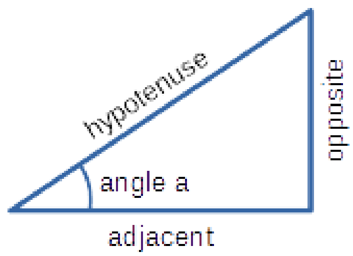
Notation	Meaning
__ A	Line A
__ PQ	Line segment formed by points P and Q
/ a	Angle a
/ PQR	Angle formed by line segments __ PQ and __ QR
P vs P’	Point P in Plane A will be notated as P, while the same point in Plane B will be notated as P’
sqrt()	Square Root function. sqrt(4) = 2
A * B	A multiplied by B
A / B	A divided by B

Corrected Arctangent Function

Some calculators and computers only return a result for ArcTangent in the first quadrant.

What is ArcTangent?

Tangent is one of the three basic trig functions.



For a given angle a:

sin(a)	$\frac{\text{opposite}}{\text{hypotenuse}}$
cos(a)	$\frac{\text{adjacent}}{\text{hypotenuse}}$
tan(a)	$\frac{\text{opposite}}{\text{adjacent}}$

ArcTangent is the inverse of the tangent function:

atan(opp / adj) = a

Given a y offset and an x offset, we find the angle with:

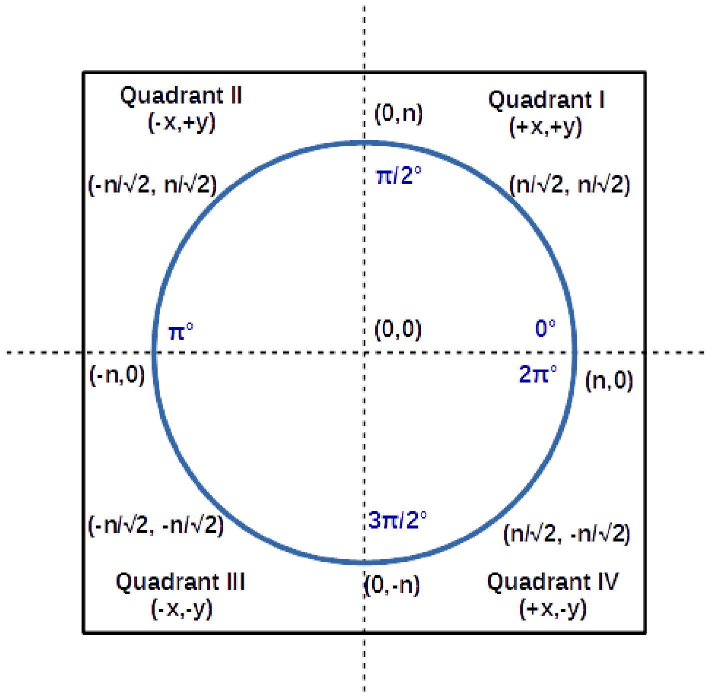
a = atan (y/x)

If we want to find the angle of a line connecting two points, p_1 and p_2:

$a = \text{atan}((y_2 - y_1) / (x_2 - x_1))$

The Problem With Atan and Quadrants

A circle has four quadrants:



In quadrant I, x and y are positive. In quadrant III, x and y are negative.

Positive divided by positive yields a positive result. Likewise, a negative divided by a negative yields a positive result.

This means that, without additional information, atan(y/x) would have the same result in both quadrant I and III. Likewise, a result in quadrant II can't be distinguished from quadrant IV.

This means that many calculator and compiler implementations won't return a result in quadrants II or III – they simply return a result in the range of +/- (π/2). Worse, some compilers only operate in quadrant I, returning 0.. π/2

Building a Corrected Atan Function

To correct the quadrant problem, we basically have to add a bunch of if..then logic:

```
a = atan(y/x)     // Return a result in quadrant I:  0..π/2
```

```
if y < 0 then
        // Quadrants III and IV
        if x < 0 then
                // Quadrant III -x,-y;  Shift result 2 quadrants
                a = a + π
        else
                // Quadrant IV +x,-y;  Shift result 3 quadrants
                a = a + 3π/2
        endif
else
        // Quadrants I and II
        if x < 0 then
                // Quadrant II  -x,+y;  Shift result 1 quadrant
                a = a + π/2
        else
                // Quadrant I  +x, +y;   Don't do anything
        endif
endif
```

This returns an angle a in the range of 0.. 2π

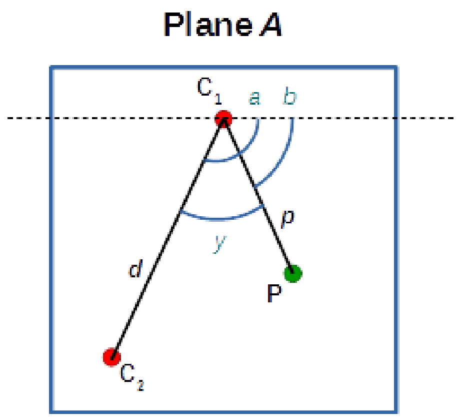
We will call our corrected function atanc – short for atan (corrected)

We will use this form to invoke our corrected atan function:

a = atanc(y,x)

Process Overview

1. **In Plane A:**



a) Find the two closest control points to point P, and label them C_1 and C_2.

b) C_1 will be assigned as the pivot.

c) Find the distance *d* between the two control points.

d) Find the distance *p* between P and the pivot, C_1.

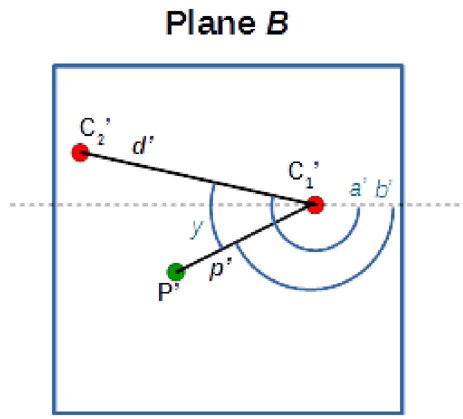
e) Compute *r* as the ratio of *p:d*.

f) Find the absolute angle / *a* of line __ C_1C_2.

g) Find the absolute angle / *b* of line __ C_1P (C_1 is the pivot).

h) Compute relative angle / *y* as / *b* minus / *a*. (Note that / *y* might be negative or positive, and represents the angular offset)

2. **In Plane B:**



a) Assign C_1’ and C_2’ as elements from Plane B corresponding to C_1 and C_2, respectively.

b) Find the distance *d’* between C_1’ and C_2’.

c) Compute *p’* as *d’* times *r.*

d) Find absolute angle / *a’* of line __ C_1’C_2’.

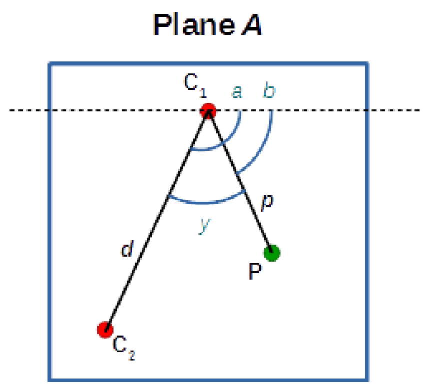
e) Compute angle / *b’* as / *a’* + / *y.*

f) Compute P’ offset from C_1’ using a vector whose magnitude is *p*’ and angle is / *b*’.

Detailed Process

Approach: We will compute *y* and *r* in Plane A and carry them over to Plane B. Then, we will use *y*, *r*, *d'* and *a'* to calculate a vector (*p' @ b'*) which determines P' as an offset of C_1'.

1. Plane A



Variable	Explanation
C_1	First Control point, and also the "pivot" point. C_1 is located at (X_1,Y_1)
C_2	Second Control point, located at (X_2,Y_2)
P	Arbitrary point, located at (X_P,Y_P)
d	Distance from C_1 to C_2 (same as $d_{1,2}$)
p	Distance from C_1 to P
a	Angle from normal through C_1 to __ C_1C_2
b	Angle from normal through C_1 to C_1P
y	+/- Angle offset from / *a* to / *b*
r	Ratio of p:d
S{}	Set of all control points in Plane A

a) In the set of all control points S{}, find the two closest control points to P, C_1 and C_2.

For each element S_k in $S_{1..n}$, compute $d_k = \text{sqrt}((X_P – X_k)^2 + (Y_P – Y_k)^2)$

Assign the elements of S corresponding to the two smallest values of $d_{1..n}$ to C_1 and C_2, respectively.

So if d_k and d_m are the smallest, then $C_1 = S_k$ and $C_2 = S_m$

After this point, we can effectively ignore the rest of S{}

b) C_1 (also, S_k) is the pivot.

c) Compute d

$d_{1,2} = \text{sqrt}((X_2 – X_1)^2 + (Y_2 – Y_1)^2)$

d) p has already been computed: $p = d_k$

e) Compute r

$r = p / d$

Note: r is constant from Plane A to Plane B

f) Compute / *a*, the angle between C_1 and C_2

$a = \text{atanc}((Y_2 – Y_1) , (X_2 – X_1))$

g) Compute / *b*, the angle between C_1 and P

$b = \text{atanc}((Y_P – Y_1) , (X_P – X_1))$

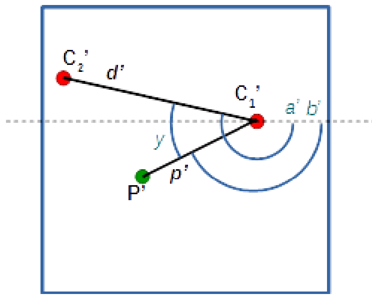
h) Compute / *y*, the offset of angle / *a* from angle / *b*.

$y = b - a$

Note: y is constant from Plane A to Plane B

2. Plane B

Plane *B*



Variable	Explanation
C_1’	First Control point, and also the “pivot” point. C_1’ is located at (X_1’,Y_1’)
C_2’	Second Control point, located at (X_2’,Y_2’)
P’	The translation of P, and arbitrary point, from Plane A to Plane B
d’	Distance from C_1’ to C_2’ (same as $d_{1,2}$’)
p’	Distance from C_1’ to P’
a’	Angle from normal through C_1’ to __ C_1’C_2’
b’	Angle from normal through C_1’ to C_1’P’
y	+/- Angle offset from / *a* to / *b* in Plane A, and / *a’* to / *b’* in Plane B
r	Ratio of p:d in Plane A, and p’:d’ in Plane B
S’{}	Set of all control points in Plane B. Each $S_{1..n}$ in Plane A corresponds to $S_{1..n}$’ in Plane B

a) Assign corresponding elements from S’{} from Plane B to C_1’ and C_2’.

Recalling that

$C_1 = S_k$ and $C_2 = S_m$

Then,

$C_1' = S_k'$

and

$C_2' = S_m'$

b) Compute d’

$$d_{1,2}' = \text{sqrt}(\ (X_2' - X_1')^2 + (Y_2' - Y_1')^2\ )$$

c) Compute *p’* using *r*, the ratio of d:p

$$p' = d' * r$$

d) Compute *a’*

$$a' = \text{atanc}(\ (Y_2' - Y_1')\ ,\ (X_2' - X_1')\ )$$

e) Compute *b’* from *a’* and *y*. Note that *y* is an angular offset, +/- from *a*.

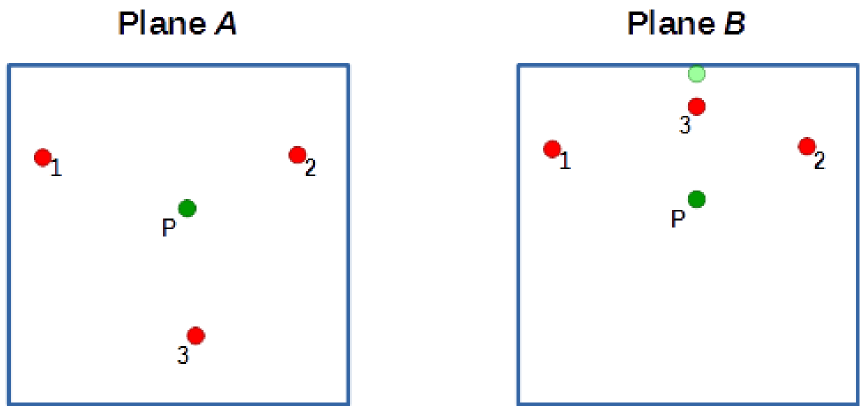
$$b' = a' + y$$

f) Compute P’ from C_1 plus a vector whose magnitude is *p’* and angle is / *b*’

$$X_P' = X_1' + p' * \cos(b')$$
$$Y_P' = Y_1' + p' * \sin(b')$$

P’ is located at (X_P', Y_P')

Limitations



In this example, the two closest control points to P don't move, but a third control point moves significantly. The resulting position for P' in Plane B is identical to the position of P, but doesn't take in to account the final position of control point C_3.

In this situation, knowing that C_3' is closer in Plane B than either C_1' or C_2', we could go back to Plane A and average the result of translating P using each set:

- C_1, C_2
- C_2, C_3
- C_1, C_3

Further, having many more control points yields more accurate translation results.

Conclusion

This method provides a simple method to translate an arbitrary point from Plane A to Plane B using a network of control points.

A practical application for this process, is to dynamically translate an arbitrary location between virtual and physical coordinates based on a set of landmarks.