# Empirical transformations in the Euclidean Plane

Juan M. Escamilla

December 19, 2013

**Abstract**

This is the documentation of the program `cartesian2affine.py` a software written in Python that transforms coordinates defined in the cartesian plane to an affine plane. In order to achieve this, a set of *Ground Control Points* (cgp.txt) is needed. The program calculates the parameters needed to perform the affine transformation.

## 1 Affine transformations in the plane

Affine transformations are functions from the plane into the plane that preserves parallel lines. Thus, the equivalence relation of "*being parallel to...* " is invariant under these type of transformations (subset of $Hom(R^2, R^2)$). Affine transformations can be seen as composition of *rotations, translations, scaling* and *shearing*. The general equations system for converting cartesian coordinates into affine coordinates is the following:

$$\begin{array}{rcl} X & = & a_0 + a_1 x + a_2 y \\ Y & = & b_0 + b_1 x + b_2 y \end{array} \tag{1.0.1}$$

Where $X, Y$ are the affine coordinates and:

$$\begin{array}{lll} a_0 = X_0 & a_1 = m_x Cos(\alpha) & a_2 = -m_x Sin(\alpha + \beta) \\ b_0 = Y_0 & b_1 = m_x Sin(\alpha) & b_2 = m_y Cos(\alpha + \beta) \end{array} \tag{1.0.2}$$

# 2 Workflow of the program

In order to solve this linear and coupled system we need an extra constraint (linear equation with defined values). Either we use at least six linear equations or use the matrix representation of the matrix extended into a 3x3 matrix.

## 2.1 Calculate the parameters with at least six GCP

As we can see in the equations above, six variables (parameters) are needed to construct an affine transformation. Given the fact that the system is linear (linear transformation) we need at least six linear equations to solve it: Three for X and three for Y. The use of the *Ground Control Points* gives constraints to the system and with these solves it. To estimate precise parameters it is convenient to use as much *control points* as possible. In this exercise the *Root Mean Square* error estimator was used as a measure of quality for the transformation.

i.e.

$$\begin{aligned} X_1 &= a_0 + a_1 x_1 + a_2 x_1 \\ X_2 &= a_0 + a_1 x_2 + a_2 x_2 \\ X_3 &= a_0 + a_1 x_3 + a_2 x_3 \end{aligned} \tag{2.1.1}$$

$$\begin{vmatrix} X_1 \\ X_2 \\ X_3 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix} \bullet \begin{vmatrix} a_0 \\ a_1 \\ a_2 \end{vmatrix}$$

The solution is the product of the inverse of the matrix by the vector $(X_1, X_2, X_3)$ 1.0.1 With this we obtain: $a_0, a_1, a_2, b_0, b_1, b_2$.
Finally, for the conversion between these parameters into $\alpha, \beta, X_0, Y_0, m_x, m_y$ it is needed to solve the equations on 1.0.2.

$$X_0 = a_0 \quad Y_0 = b_0$$

## 2.2 Pseudocode

1. Make 3-tuples of each pair of coordinates.
2. For each tuple do:
3. Matrix representation M of the system build onto a matrix data type

4. find the inverse of M and
5. multiply M by the actual coordinates.
6. convert the parameters
7. return the result

The program builds a parameter vector for every tuple. After that the RMS is calculated in order to find the best set of gcp.