Edinburgh
Mouse
Atlas
Project

MRC Medical Research Council
Human Genetics Unit

EMAP Technical Report

A 3D Paint Program for the Mouse Atlas and Gene Expression Database: Full Affine Version

Author: **Richard Baldock**
Email: R.Baldock@hgu.mrc.ac.uk

Date: **25th March 2004**
Distribution: **Open distribution is permitted**
Status: **Final**
Source: **Report/000012-MAPaint_affine**
Project: **DG401**
Report number: EMAP/Report/000012
Keywords: **3D voxel images, painting, segmentation, data mapping, 3D reconstruction**

EMAP is a research programme at the MRC Human Genetics Unit in collaboration with Biomedical Sciences, University of Edinburgh. Contact Dr Duncan Davidson or Dr Richard Baldock at MRC Human Genetics Unit, Crewe Road, Edinburgh, EH4 2XU, UK.

Abstract

MAPaint is a section viewer for 3D grey-level voxel data. The section-view transform is a rigid body transform to a new coordinate frame and the section plane defined as the plane of constant $z = distance$ in the transformed view. This is too restrictive when we want to consider objects that have non-isotropic voxel sizes and we wish to display the correct aspect ratios or if the input object is defined by an affine transform plus original data. In these cases we need to be able to include an arbitrary additional transform within the section viewing code. The simplest way to achieve this is to convert the geometric definition of the section-view transform to a full affine version. This document details the changes needed for that conversion and con only be understood in the context of the earlier text.

Note this document was written originally in 1996 and has been updated since but the current date reflects when the final version was generated and installed into the CVS repository.

Contents

1 Introduction

This is an addendum document to "A 3D Paint Program for the Mouse Atlas and Gene Expression Database" [1] and replicates the geometric definitions of the sections views in terms of a more general affine transform. For this purpose the Geometry section of that paper is repeated with changes as required.

2 3D Images

See the earlier paper for a review of the 3D image structure. The purpose of this addendum is to allow easy inclusion of a pre-affine transform on an object to be included within the section transform. This makes it easy to include the non-isotropic voxel sizes that are typical of reconstructions and also if we want to align the objects in a different way for example to make the transverse sections genuinely transverse and perhaps to invert an object.

3 Geometry

In this section we discuss the basic geometric transformations used within the program and provide the definitions of viewing parameters.

3.1 Coordinate Transformation

There are many ways to define an arbitrary rotation, scaling and translation of one coordinate frame into another. For the purposes of the paint program we define a set of parameters for viewing arbitrary planes through reconstructions which seem to be the most useful in terms of the way in which the paint program will be used and which should be easy to visualise for the user. The relationship between these transforms and the general affine transform used elsewhere is most easily found be direct comparison of the transform matrix elements.

We define a viewing plane by defining a new set of coordinate axes such that the new z-axis is along the "line-of-sight". This axis is fully determined by defining a single fixed point which is the new coordinate origin. The actual view plane is then defined to be perpendicular to this axis and is determined by a scalar distance parameter along the new axis. In this way the transformation between the original, $\mathbf{r} = (x, y, z)$, and viewing coordinates, $\mathbf{r}' = (x', y', z')$, is determined by a 3D rotation and translation with the viewing plane defined as a plane of constant $z' = d$. These parameters are dipicted in figure 1.

A 3D rotation can be defined in terms of Eulerian angles[2, page 107] which are not consistently defined in the literature, but for which we assume the usual British definition[3, page 9], where a rotation about an axis is *clockwise* in the direction of the axis and the second rotation is about the new y-axis:

1. rotate by angle ξ (xsi) about the z-axis
2. rotate by angle η (eta) about the new y-axis,

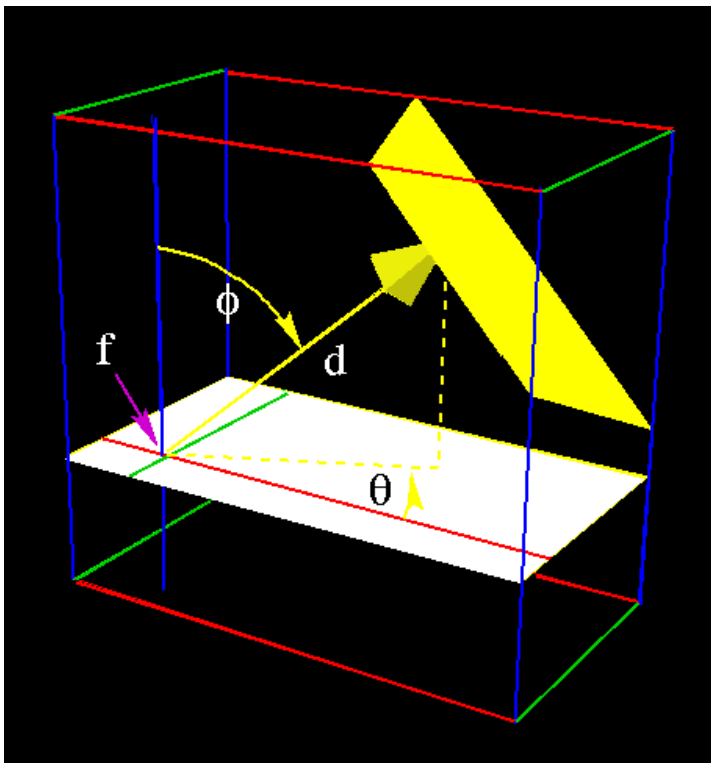

Figure 1: The viewing plane is defined to be perpendicular to the viewing direction given by angles θ and ϕ which are yaw and pitch respectively. The actual plane is distance d from the fixed point $\mathbf{f}$.

3. rotate by angle ζ (zeta) about the new z-axis.

This sequence is depicted in figure 2.

The rotation matrix defined by these angle is most easily determined as a product of three rotations:

$$R = R_\zeta R_\eta R_\xi \tag{1}$$

where in matrix notation

$$R_\xi = \begin{pmatrix} cos(\xi) & sin(\xi) & 0 \\ -sin(\xi) & cos(\xi) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2}$$

$$R_\eta = \begin{pmatrix} cos(\eta) & 0 & -sin(\eta) \\ 0 & 1 & 0 \\ sin(\eta) & 0 & cos(\eta) \end{pmatrix} \text{ and} \tag{3}$$

$$R_\zeta = \begin{pmatrix} cos(\zeta) & sin(\zeta) & 0 \\ -sin(\zeta) & cos(\zeta) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4}$$

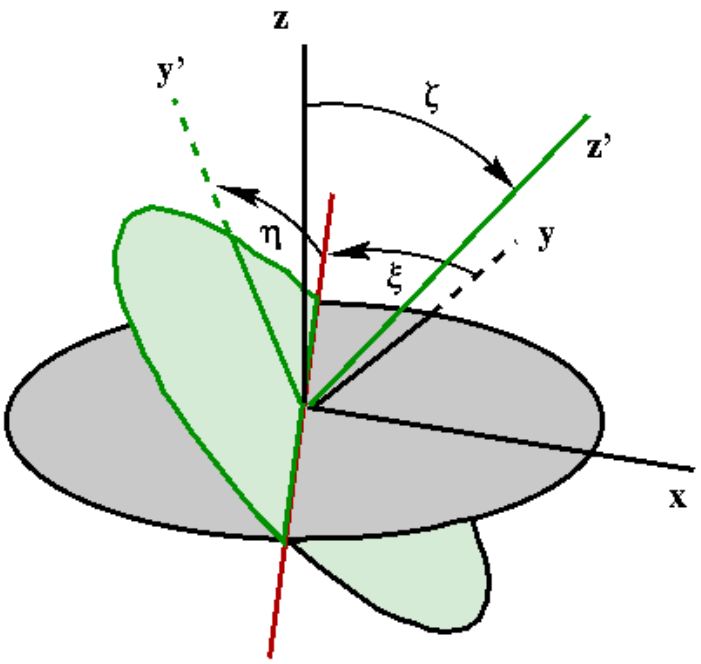

Figure 2: The Eulerian angles. The line of intersection between the original x-y plane and the new x'-y' plane is known as the "line of nodes" and shown extended in red in the figure. The x' axis is below the original plane and not shown.

Multiplying the individual matrices yields

$$R = \begin{pmatrix} cos(\zeta)cos(\eta)cos(\xi) - sin(\zeta)sin(\xi) & cos(\zeta)cos(\eta)sin(\xi) + sin(\zeta)cos(\xi) & -cos(\zeta)sin(\eta) \\ -sin(\zeta)cos(\eta)cos(\xi) - cos(\zeta)sin(\xi) & -sin(\zeta)cos(\eta)sin(\xi) + cos(\zeta)cos(\xi) & sin(\zeta)sin(\eta) \\ sin(\eta)cos(\xi) & sin(\eta)sin(\xi) & cos(\eta) \end{pmatrix}$$

A rigid body transformation can be described as a rotation followed by a translation. In our case we may wish to display the section with magnification and therefore the transformation from screen to object coordinates will also involve scaling. With this in mind we define the transform from object to viewing coordinates as

$$\mathbf{r}' = sR(\mathbf{r} - \mathbf{f}), \tag{5}$$

where s is the scaling and $\mathbf{f}$ is the fixed point in the original coordinates. Most of the calculations we require involve the inverse transform

$$\mathbf{r} = \frac{1}{s}R^{-1}\mathbf{r}' + \mathbf{f}, \tag{6}$$

where $R^{-1} = R^T$ (transposed matrix)[2].

We retain the basic definition of the section view transform in terms of the rotation angles, scale

and fixed point but write it in terms of a general affine T which in augmented coordinates is:

$$T = \begin{pmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

T can be calculated as the product of a scale, T_s, rotation, T_r, and a translation (the fixed point), T_f:

$$T = T_s T_r T_f \tag{8}$$

where

$$T_s = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; T_r = \begin{pmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; T_f = \begin{pmatrix} 1 & 0 & 0 & -f_x \\ 0 & 1 & 0 & -f_y \\ 0 & 0 & 1 & -f_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9}$$

In some cases we may want to display the data with voxel sizes respected or perhaps a transform object which includes an affine transform to define the coordinate frame. In the first case it is convenient to interpose the rescaling after the fixed point transformation so that the fixed point selection can be on the basis of the discrete integer image coordinates. This is not necessary but convenient. For the case of a pre-affine trnasformation is is more appropriate to apply that first. If the voxel re-scaling transform is T_v and the transform object has affine transform T_o (note this could be the voxel rescaling if we want to work in "real" coordinates) then the section transform becomes:

$$T = T_s T_r T_v T_f \quad \text{or} T = T_s T_r T_f T_o \quad \text{and} T_v = \begin{pmatrix} x - size & 0 & 0 & 0 \\ 0 & y - size & 0 & 0 \\ 0 & 0 & z - size & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{10}$$

In practice the voxel sizes may be re-scaled so that the minimum value is such that all voxels are visible when the scaling is unity.

3.2 Viewing Planes

The viewing or normal direction plus the fixed point and distance only define the viewed image to within an arbitrary viewing angle, there is still a choice of the orientation of the image on the screen. There are a number of possibilities here. One is to consider the 3D image as an object in space and display a projection of the viewing plane as seen by an observer in the same 3D space. This has the merit of providing clear feedback of the position of the plane within the whole but is not ideal for painting because for some angles the projection will introduce perspective distortion of the image. There are two solutions adopted by the paint program both of which display the view-plane "flat" on the screen, *i.e.* the view orthogonal to the viewing direction. The difference in the two is the model used to establish the rotation around the viewing direction. In the first versions of the program the plane was displayed as if the viewer was "walking" around the structure on the $x - y$ plane. This proved confusing and therefore a different model has been included which

is to ensure that the projection of an arbitrary but predefined vector **u** (up) onto the viewing plane will be parallel to the y-axis of the displayed plane (actually "up" on the screen is in the direction $(0, -1)$ in screen coordinates because traditionally raster coordinates start at the top left corner of the screen). Both of these have their advantages and disadvantages and therefore the either model can be selected.

3.2.1 Walking Around the Statue

The initial solution adopted in the 3D paint program is to always display the viewing plane "flat" but to orient the image as if the viewer were "walking around" the object. The actual displayed image is then obtained by rotating the viewed plane about an axis parallel to the line of intersection of the view plane and the "horizontal" which is defined to be a plane of constant z. This is depicted in figure 1 where two viewing angles are defined: the angle to the x-axis of the projection of the viewing direction onto the horizontal, θ, and the angle from the viewing direction to the z-axis, ϕ. These are the usual spherical coordinate angles and are known under many aliases *e.g.* azimuth and declination or pitch and yaw. Only two angles are required to define the viewing direction and the third Eulerian angle is determined by using the rule outlined above. It can be seen for example that a $y - z$ plane is displayed with $y' = y$ and $x' = z$, and the $x - z$ plane is displayed with $x' = x$ and $y' = -z$. It is straightforward to show that the Euler angles defined using θ, ϕ and rotation about the line of intersection with the horizontal are

$$\xi = \theta, \tag{11}$$
$$\eta = \phi \text{ and} \tag{12}$$
$$\zeta = -\theta. \tag{13}$$

Substituting these into equation 3.1 yields

$$R = \begin{pmatrix} cos^2(\theta)cos(\phi) + sin^2(\theta) & cos(\theta)sin(\theta)(cos(\phi) - 1) & cos(\theta)sin(\phi) \\ cos(\theta)sin(\theta)(cos(\phi) - 1) & sin^2(\theta)cos(\phi) + cos^2(\theta) & sin(\theta)sin(\phi) \\ -cos(\theta)sin(\phi) & -sin(\theta)sin(\phi) & cos(\phi) \end{pmatrix} \tag{14}$$

For each new view it is necessary to calculate the corresponding point in the reference image for each point on in the view plane. This is done by defining a set of look-up tables so that the trigonometric functions are only calculated once for each viewing direction. Because the transform angles can be arbitrary floating point numbers it is necessary to calculate the corresponding point in the reference image in real (floating point) coordinates. This means that 6 LUT's are required namely a vector LUT for x' and y'. These LUT's, the maximum values for x', y' and z', the view angles, fixed point and distance are each stored in the `ViewStruct` data-structure held with each view window and are used for updating values as the distance or other parameters are modified.

3.2.2 Up is Up

In this model the projection of a predefined direction "up" will always be displayed as the vertical in the section view. If the viewing direction is parallel to this vector then the angle of rotation around the viewing direction is not defined and an arbitrary choice can be made. A consequence of this is that for small changes in viewing direction around the "up" vector may give rise to arbitrarily

large changes in the display orientation. This effect occurs for any rule which determines the screen orientation solely from the static viewing orientation. If a smooth transformation is required then it is necesary to select the orientation not just in terms of the viewing direction but also in terms of the previously viewed section, *i.e. dynamic* information is required. This of course means that the viewing direction does not uniquely define the section orientation.

To establish the viewing transformation we need to calculate the Euler angle ζ so that the projection of the up vector, $\mathbf{u}$, is parallel to the displayed y-axis. To do this we calculate the angle ω between the component of the vector $\mathbf{u}$ perpendicular to the viewing direction and the y-axis in the new coordinate system for the transformation $R(\xi, \eta, \zeta) = R(\theta, \phi, 0)$. The angle ζ is then ω or $\omega - \pi$ depending on the orientation of the y-axis on the screen. This angle can be easily established by rotating the up vector by $R(\theta, \phi, 0)$ and calculating the angle of the projection of this vector to the transformed y-axis. Thus if

$$\mathbf{w} = R\mathbf{u}, \quad \text{where} \tag{15}$$

$$R = \begin{pmatrix} (T^{-1})_{11} & (T^{-1})_{12} & (T^{-1})_{13} \\ (T^{-1})_{21} & (T^{-1})_{22} & (T^{-1})_{23} \\ (T^{-1})_{31} & (T^{-1})_{32} & (T^{-1})_{33} \end{pmatrix} \quad \text{then} \tag{16}$$

$$\omega = \tan^{-1}(\frac{w_x}{w_y}). \tag{17}$$

Given the Euler angles the view transformation is calculated in exactly the same way as for the other viewing mode.

3.2.3 Look up Tables for Display to Image Coordinates

To make the calculation more efficient we can minimize the number of floating point operations by pre-calculating the trigonometric terms and storing them as look-up tables (LUT). In this case we are transforming from the 2D display frame back to 3D image coordinates and to avoid round-off error we require 6 LUTs for each of x' and y' to (x, y, z). Note x' and y' are always integer coordinates in the display frame. In our coordinate system we want the transform from (x', y', d) to the original image coordinates therefore using equation 6 we get

$$\begin{aligned} x &= T_{11}^{-1}x' + T_{12}^{-1}y' + T_{13}^{-1}d + T_{14}^{-1} \ , \\ y &= T_{21}^{-1}x' + T_{22}^{-1}y' + T_{23}^{-1}d + T_{24}^{-1} \ \text{ and} \\ z &= T_{31}^{-1}x' + T_{32}^{-1}y' + T_{33}^{-1}d + T_{34}^{-1} \ . \end{aligned} \tag{18}$$

We define LUTs for x' to (x, y, z) which include the constant terms:

$$\begin{aligned} T_{x'x}(x') &= T_{11}^{-1}x' + T_{13}^{-1}d + T_{14}^{-1} \ , \\ T_{x'y}(x') &= T_{21}^{-1}x' + T_{23}^{-1}d + T_{24}^{-1} \ , \\ T_{x'z}(x') &= T_{31}^{-1}x' + T_{33}^{-1}d + T_{34}^{-1} \ . \end{aligned} \tag{19}$$

and similarly for y' to (x, y, z):

$$\begin{aligned} T_{y'x}(x') &= T_{12}^{-1}y' \ , \\ T_{y'y}(x') &= T_{22}^{-1}y' \ , \\ T_{y'z}(x') &= T_{32}^{-1}y' \ . \end{aligned} \tag{20}$$

To determine the new section image the transformations can be calculated by look-up plus one addition:

$$\begin{aligned} x &= T_{x'x}(x') + T_{y'x}(y') \,, \\ y &= T_{x'y}(x') + T_{y'y}(y') \,, \\ z &= T_{x'z}(x') + T_{y'z}(y') \,. \end{aligned} \tag{21}$$

If the viewing direction does not change and only the distance parameter is incremented by δd then the y' LUTs are unchanged and the x' LUTs become:

$$\begin{aligned} T_{x'x}(x', d+\delta d) &= T_{x'x}(x', d) + T_{13}^{-1}\delta d \,, \\ T_{x'y}(x', d+\delta d) &= T_{x'y}(x', d) + T_{23}^{-1}\delta d \,, \\ T_{x'z}(x', d+\delta d) &= T_{x'z}(x', d) + T_{33}^{-1}\delta d \,. \end{aligned} \tag{22}$$

The LUTs are calculated over the maximum range possible for x' and y' which is determined by forward transform (equation 5) of the bounding box of the original image.

3.3 Fixed Point Constraints

Navigation through the volume can be difficult especially since arbitrary sections give rise to unfamiliar views of well-known structures. To aid the navigation the paint program give a "cartoon" 3D feedback display showing the rough position of each view with respect to the bounding box (see the section on 3D display). A control within the view windows that can provide assistance for navigation is the option of setting fixed-point constraints. The basic idea of the transformation used is that it may often be possible to identify one or more points within the image that the user wishes to be definitely visible and thereby reduce the search space to setting the viewing angles. If one point is fixed then there are two degrees of freedom left to set the view and if there are two fixed points then there is only one degree of freedom.

The transformation is defined so that by setting one fixed point, $\mathbf{f}$, the orientation parameters, θ, ϕ, will rotate the view plane about this point. If two points are fixed then θ and ϕ are dependent and can be represented in parametric form using a third angle parameter, ψ, which corresponds to the angle around the line joining the two fixed points.

The two fixed points $\mathbf{f_1}$ and $\mathbf{f_2}$ define direction vector

$$\mathbf{n_1} = \frac{\mathbf{f_2} - \mathbf{f_1}}{|\mathbf{f_2} - \mathbf{f_1}|}$$

which must remain in the view plane. The values of θ and ϕ which define the plane in which the two fixed points were initially defined, θ_0 and ϕ_0, define an axis perpendicular to $\mathbf{n_1}$

$$\mathbf{n_2} = \frac{T_0^{-1}\mathbf{z} - T_0^{-1}\mathbf{0}}{|T_0^{-1}\mathbf{z} - T_0^{-1}\mathbf{0}|}$$

where T_0 is the section affine transform defined by θ_0, ϕ_0, $\mathbf{z} = (0,0,1)^T$ and $\mathbf{0} = (0,0,0)^T$. A third axis, $\mathbf{n_3} = \mathbf{n_1} \wedge \mathbf{n_2}$, is perpendicular to both $\mathbf{n_1}$ and $\mathbf{n_2}$ and can be used to define the normal to a viewing plane which contains the two fixed points:

$$\mathbf{n} = cos(\psi)\mathbf{n_2} + sin(\psi)\mathbf{n_3},$$

where $0 \leq \psi \leq 2\pi$, and $\psi = 0.0$ is the original fixed plane. If we demand that $\mathbf{f}$ is equal to one of the two fixed points and that the viewing direction is $\mathbf{n}$ then both fixed points will be visible. The corresponding viewing angles are given by:

$$cos(\phi) = n_z, \tag{23}$$

$$tan(\theta) = \frac{n_y}{n_x}. \tag{24}$$

Note if $\mathbf{n} = \mathbf{z}$ then θ is ill-defined. In the program paint the third Euler angle ζ is determined by requiring that the vector $\mathbf{n_1}$ maintains a constant angle to the vertical on the screen. This angle is determined by when the fixed line is first set.

In the 3D paint program the fixed point constraints are implemented so that they remain in force whilst only the angles are being modified. Resetting the first fixed point or changing the distance parameter will cancel the second fixed point constraint.

3.4 Lines of Intersection

To make the painting easier on an undistorted image the arbitrary sections are not displayed with any sort of orthographic or perspective projection but as seen from the viewing angle. This can be somewhat confusing and to help navigation the line of intersection of the view that has the current "input-focus" (usually when the cursor is in the view window) with all other views is displayed and will move as the controls are adjusted. To determine the line of intersection is straightforward geometry and requires solving a set of three simultaneous equations to find a point in each view that is common to both.

For two given planes p_1 and p_2 with viewing angles (θ_1, ϕ_1) and (θ_2, ϕ_2) respectively we want to establish the angle, α, and a point, $\mathbf{p} = (p_X, p_Y)$, of the line of intersection of plane 1 in plane 2. With these two quantities we can then draw the line of intersection in plane 2:

$$(y - p_y) = tan(\alpha)(x - p_x).$$

If $\mathbf{n_1}$ is the normal to viewing plane 1 and $\mathbf{n_2}$ is the normal to viewing plane 2 then the line of intersection, which must lie in both planes, is parallel to the vector $\mathbf{n_1} \wedge \mathbf{n_2}$. To establish the angle in plane 2 we rotate this vector to plane 2 coordinates, $\mathbf{l} = R_2(\mathbf{n_1} \wedge \mathbf{n_2})$, and

$$tan(\alpha) = \frac{l_y}{l_x}.$$

To calculate a point of intersection of the two planes we solve for the point of intersection of a line in plane 1 with plane 2. A line in plane 1 which must intersect plane 2 is a normal to the line of intersection. If T_1 is the affine transform of plane 1 then the normal to the line of intersection which passes through the origin in plane 1 is

$$\mathbf{r} = \mathbf{n_1} \wedge (\mathbf{n_1} \wedge \mathbf{n_2})s_1 + T_1^{-1} d_1 \mathbf{z}, \tag{25}$$

where s_1 is a scalar parameter. Plane 2 can defined in terms of a vector parameter $\mathbf{s_2} = (x_2, y_2, d_2)^T$ where x_2, y_2 are free parameters and d_2 is the distance parameter of plane 2 by the equation

$$\mathbf{r} = T_2^{-1} \mathbf{s_2}. \tag{26}$$

The point of intersection between this plane and the line in plane 1 is found by solving

$$T_2^{-1}\mathbf{s_2} = \mathbf{n_1} \wedge (\mathbf{n_1} \wedge \mathbf{n_2})s_1 + T_1^{-1}d_1\mathbf{z}, \tag{27}$$

for the parameters s_1, x_2 and y_2. This vector equation results in a set of three simultaneous equations which can be expressed in matrix form

$$A \begin{pmatrix} x_2 \\ y_2 \\ s_1 \end{pmatrix} = \mathbf{B} \tag{28}$$

where $\mathbf{B} = T_1^{-1}d_1\mathbf{z} - T_2^{-1}d_2\mathbf{z}$ and

$$A = \begin{pmatrix} (T_2^{-1})_{11} & (T_2^{-1})_{12} & -(\mathbf{n_1} \wedge (\mathbf{n_1} \wedge \mathbf{n_2}))_x \\ (T_2^{-1})_{21} & (T_2^{-1})_{22} & -(\mathbf{n_1} \wedge (\mathbf{n_1} \wedge \mathbf{n_2}))_y \\ (T_2^{-1})_{31} & (T_2^{-1})_{32} & -(\mathbf{n_1} \wedge (\mathbf{n_1} \wedge \mathbf{n_2}))_z \end{pmatrix}.$$

The equations are solved in the paint program using standard linear equations algorithms.

3.5 View Plane from 3D Domain

On occasion it is useful to estimate the view-parameters from a given domain. This makes the assumption that the input domain is indeed planar although the method should find a best fit plane for any roughly planar point set. It is potentially fragile however and is intended to help recover lost view parameters and to be part of a repair process for sections that have somehow "degraded" in the context of the **emage** database.

The method described here first establishes an approximate set of view parameters which are then refined using all of the point-set using a least-squares approach. The point-set could be all the voxel locations for the given domain or some sub-set for example the interval end-points.

We start with a set of voxel locations from the given domain: $\{\mathbf{p_i} : 1 \leq i \leq N\}$. The first step is to select three points and estimate the section viewing parameters by setting the section view -direction (i.e. the perpendicular) parallel to the perpendicular to the plane defined by the points. Any three points will do, here is a method to choose reasonably dispersed points.

3.5.1 Find Three Points

A simple prescription for three points reasonably separated. We use the mean position, the point most distant from the mean and for the third the point with the greatest perpendicular distance to the first two:

$$\mathbf{q_1} = \bar{\mathbf{p}} = \frac{1}{N}\sum_i \mathbf{p_i} \tag{29}$$

$$\mathbf{q_2} = \mathbf{p_k} \quad \text{s.t.} \quad d = \max_i |\mathbf{p_i} - \bar{\mathbf{p}}| \quad \text{for} \quad i = k \tag{30}$$

$$\mathbf{q_3} = \mathbf{p_l} \quad \text{s.t.} \quad D = \max_i \left[\left(\frac{(\mathbf{p_i} - \mathbf{q_1}) \cdot (\mathbf{p_i} - \mathbf{q_2})}{|\mathbf{q_1} - \mathbf{q_2}|} \right)^2 + |\mathbf{p_i} - \mathbf{q_1}|^2 \right] \quad \text{for} \quad i = l \tag{31}$$

3.5.2 View Direction Parameters for a Given Plane

Given three points $\mathbf{q_1}$, $\mathbf{q_2}$ and $\mathbf{q_3}$ then a unit vector perpendicular to the plane defined by these points is

$$\mathbf{r} = \frac{(\mathbf{q_2} - \mathbf{q_1}) \wedge (\mathbf{q_3} - \mathbf{q_1})}{|(\mathbf{q_2} - \mathbf{q_1}) \wedge (\mathbf{q_3} - \mathbf{q_1})|} \tag{32}$$

This corresponds to the transformed z axis which in the original coordinate frame is

$$\mathbf{s} = R^{-1}\mathbf{z} = R^T\mathbf{z} = \begin{pmatrix} \sin\eta\cos\xi \\ \sin\eta\sin\xi \\ \cos\eta \end{pmatrix} \tag{33}$$

Setting $\mathbf{s} = \mathbf{r}$ gives

$$\begin{aligned} \eta &= \cos^{-1}(\mathbf{r} \cdot \mathbf{z}) \\ \xi &= \cos^{-1}\left(\frac{\mathbf{r} \cdot \mathbf{x}}{\sin\eta}\right) \end{aligned} \tag{34}$$

3.5.3 Best Fit Plane Given Initial View Parameters

Using the results above to define an approximate initial view defined by setting the fixed point $\mathbf{f} = \bar{\mathbf{p}}$ and view angles ξ_0 and η_0 using equation 34 we can now refine the view parameters using a least-squares approach. This is possible only if the initial estimate is reasonably close so that the problem can be linearised. With the given fixed point we want to estimate the view parameters ξ, η and distance d to minimise the distance of each point in the domain from the section plane.

Including the distance d along the transformed z-axis, the z coordinate of the point $\mathbf{p_i}$ in the in the view coordinate frame is

$$z_i' = (\sin\eta\cos\xi)X_i + (\sin\eta\sin\xi)Y_i + (\cos\eta)Z_i + d \tag{35}$$

where $X_i = x_i - \bar{x}$, $Y_i = y_i - \bar{y}$, $Z_i = z_i - \bar{z}$, (x_i, y_i, z_i) are the components of $\mathbf{p_i}$ and $(\bar{x}, \bar{y}, \bar{z})$ are the components of $\bar{\mathbf{p}}$.

If we set $\xi = \xi_0 + \delta\xi$, $\eta = \eta_0 + \delta\eta$ and expand to first order then

$$\begin{aligned} \sin\xi &\approx \sin\xi_0 + \delta\xi(\cos\xi_0 - \sin\xi_0) \\ \cos\xi &\approx \cos\xi_0 - \delta\xi(\cos\xi_0 + \sin\xi_0) \\ \sin\eta &\approx \sin\eta_0 + \delta\eta(\cos\eta_0 - \sin\eta_0) \\ \cos\eta &\approx \cos\eta_0 - \delta\eta(\cos\eta_0 + \sin\eta_0) \end{aligned} \tag{36}$$

If we substitute these into equation 35 then

$$z_i' = z_{i0}' + A_i\delta\eta + B_i\delta\xi + d \tag{37}$$

where

$$\begin{aligned} z_{i0}' &= X_i\sin\eta_0\cos\xi_0 + Y_i\sin\eta_0\sin\xi_0 + Z_i\cos\eta_0, \\ A_i &= X_i\cos\xi_0(\cos\eta_0 - \sin\eta_0) + Y_i\sin\xi_0(\cos\eta_0 - \sin\eta_0) - Z_i(\cos\eta_0 + \sin\eta_0) \quad \text{and} \\ B_i &= -X_i\sin\eta_0(\cos\xi_0 + \sin\xi_0) + Y_i\sin\eta_0(\cos\xi_0 - \sin\xi_0). \end{aligned}$$

Defining the squared error

$$\Delta = \sum_i (z_i')^2 \tag{38}$$

we require

$$\begin{aligned} \frac{\partial \Delta}{\partial\, \delta\eta} &= 0, \\ \frac{\partial \Delta}{\partial\, \delta\xi} &= 0 \quad \text{and} \\ \frac{\partial \Delta}{\partial d} &= 0 \end{aligned}$$

which yields the equations

$$\begin{aligned} \sum_i A_i^2 \delta\eta + \sum_i A_i B_i \delta\xi + \sum_i A_i d &= -\sum_i A_i z'_{i0}, \\ \sum_i A_i B_i \delta\eta + \sum_i B_i^2 \delta\xi + \sum_i B_i d &= -\sum_i B_i z'_{i0} \quad \text{and} \\ \sum_i A_i \delta\eta + \sum_i B_i \delta\xi + N d &= -\sum_i z'_{i0} \end{aligned} \tag{39}$$

that can be solved by standard linear algebra techniques.

References

[1] Richard Baldock. A 3d paint program for the mouse atlas and gene expression database. Technical Report EMAP/0010, MRC Human Genetics Unit, Crewe Road, Edinburgh EH4 2XU, UK, 2004. Note this paper was first written in 1996 and parts are out of date.

[2] Herbert Goldstein. *Classical Mechanics*. Addison-Wesley, Reading, MA, 2 edition, 1950.

[3] E T Whittaker. *A Treatise on the Analytical Dynamics of Particles and Rigid Bodies*. Cambridge University Press, London, 3 edition, 1927.