Basic Techniques in Computer Graphics

Winter 2017 / 2018

Visual Computing Institute
RWTH AACHEN UNIVERSITY

The slide comments are not guaranteed to be complete, they are no alternative to the lectures itself. So go to the lectures and write down your own comments!

Projective Maps

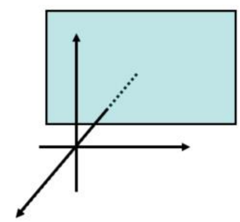
- Standard projection *(perspective division)*

  [x, y, z] → [-x/z, -y/z]

  Center of projection: $[0, 0, 0]^T$

  Projection plane: z = -1

- Geometric interpretation
- Matrix representation ?
  → *homogenous coordinates*



We want to project a point in 3D onto the projection / image plane. In the standard projection, the image plane is parallel to the x-y-plane, placed at z=-1, that is all points on the image plane have a z-value of -1. By utilizing similar triangles (see blackboard), we can derive the x-y-position of the projected point as (x/-z, y/-z).

Unfortunately, we have to use division to compute the projected point (non-linear mapping) and thus cannot use a matrix representation for projective maps.

Notice that the camera looks down the negative z-axis. This is just a convention to keep a right-handed coordinate system (x, y, -z).

Homogenous Coordinates

Points: [x, y, z] → [wx, wy, wz,w], $w \neq 0$
Vectors: [x, y, z] → [x, y, z, 0]

Interpretation:

- points → lines through origin
- vectors → points at infinity
- “stack of affine spaces”

However, by introducing homogeneous coordinates we can again represent projections using matrices. Notice that homogeneous coordinates are an extension of extended coordinates, where we allow an arbitrary value in the fourth component. That is, we represent each 3D point by a line in 4D space. By dividing by the fourth component of a point in homogeneous coordinates, we obtain its Euclidean coordinates in 3D (ignoring the fourth component). This division is called de-homogenization.
Vectors are still represented by a zero in the fourth component. For vectors, de-homogenization can be interpreted as moving a point along the direction of the vector to infinity.

Matrix Representation

Standard projection

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$P: [wx, wy, wz, w]^T \mapsto [wx, wy, wz, -w]^T$$
$$\Leftrightarrow \left[-\frac{x}{z}, -\frac{y}{z}, -1\right]^T$$

Any times we want to do a division, we multiply the fourth component with the divisor, thus accumulating the perspective division such we only have to perform one division in the end. Thus, we can represent a projective mapping by a matrix. Above, the matrix representation of the standard transformation is shown.

Homogenous Coordinates

- Generalization of extended coords
  - all properties are preserved
- Defer all divisions to last step: **“de-homogenization“**
- Translation:

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix} = \begin{bmatrix} wx + wt_x \\ wy + wt_y \\ wz + wt_z \\ w \end{bmatrix}$$

In a 4x4 matrix, the upper left 3x3 matrix represents a linear map, the first 3 components of the last column represents a translation and the last row represents a projection.

General Projections I

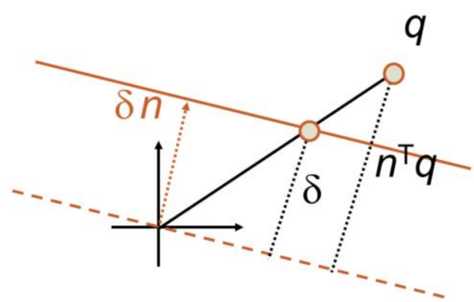
- Center of projection = [0,0,0]
- Focal distance δ
- Normal of image plane n, $||n|| = 1$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{n_x}{\delta} & \frac{n_y}{\delta} & \frac{n_z}{\delta} & 0 \end{bmatrix}$$

The standard projection has the center of projection in the origin and the image plane parallel to the x-y-plane at z= -1. A generalization of that projection allows an arbitary image plane, defined by its normal n and the distance δ of that plane from the origin (also known as focal distance as it is the distance between the center of projection and the image plane).

General Projections I

$$P: q \mapsto \frac{\delta}{n^T q} \cdot q$$



We can again derive the formula for the projection of a point q onto the image plane using the similar triangles shown on the slide: We scale the vector from the origin to q to have unit length in the direction of the normal of the plane and then rescale it by the focal length, which moves the point onto the image plane.

General Projections II

- Image plane through origin, normal *n*.
- Center of projection at -δ*n*
- Focal distance δ

$$P = \begin{bmatrix} 1-n_x^2 & -n_x n_y & -n_x n_z & 0 \\ -n_x n_y & 1-n_y^2 & -n_y n_z & 0 \\ -n_x n_z & -n_y n_z & 1-n_z^2 & 0 \\ \frac{n_x}{\delta} & \frac{n_y}{\delta} & \frac{n_z}{\delta} & 1 \end{bmatrix}$$

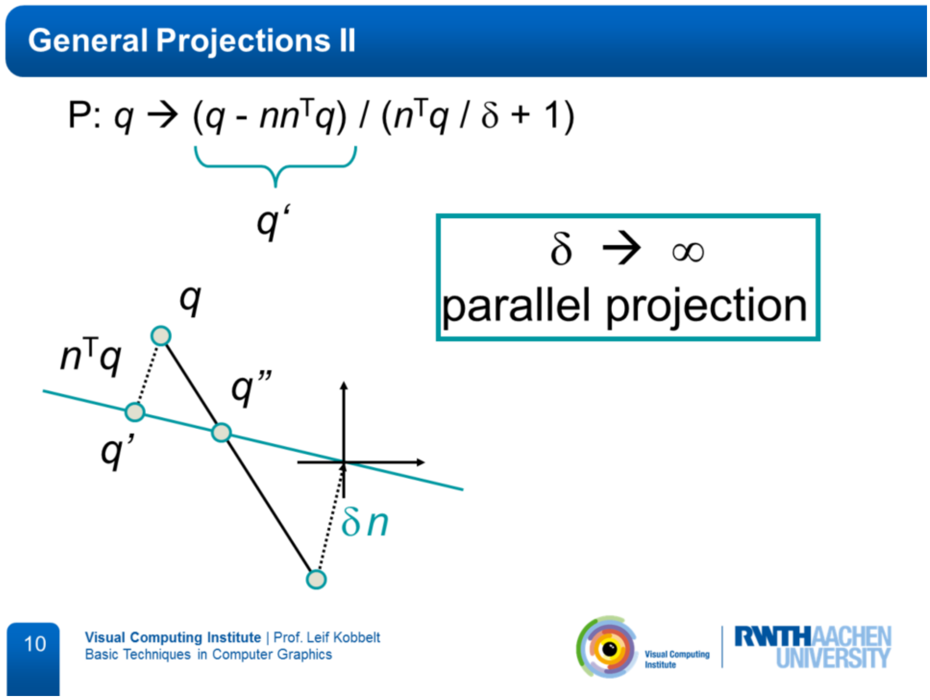
Another way to represent a more general projection is to assume that the image plane contains the origin and that the center of projection is at distance δ from the image plane in the direction of the normal of the plane (see the derivation of that matrix on the next slide).

General Projections II

- Image plane through origin, normal *n*.
- Center of projection at $-\delta n$
- Focal distance δ

$$P = \begin{bmatrix} 1 - nn^T & 0 \\ {n^T}/{\delta} & 1 \end{bmatrix}$$

Notice that we can rewrite the upper left 3x3 part of the matrix as 1-nnt , where 1 is the identity matrix.



As the focal length tends towards infinity, the vectors from the center of infinity towards any point becomes parallel to the normal of the plane.

Vanishing Points

- Lines

$$l(\lambda) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \lambda \cdot \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

- Standard projection

$$P\big(l(\lambda)\big) = \begin{bmatrix} \dfrac{x + \lambda d_x}{-z - \lambda d_z} \\ \dfrac{y + \lambda d_y}{-z - \lambda d_z} \end{bmatrix}$$

Vanishing Points

$d_z = 0$:

$$\begin{bmatrix}(x+\lambda d_x)/(-z-\lambda d_z)\\(y+\lambda d_y)/(-z-\lambda d_z)\end{bmatrix} = \begin{bmatrix}-x/z\\-y/z\end{bmatrix} - \lambda\cdot\begin{bmatrix}d_x/z\\d_y/z\end{bmatrix} \xrightarrow[\lambda\to\infty]{} \text{non-finite point}$$

$d_z \neq 0$:

$$\begin{bmatrix}(x+\lambda d_x)/(-z-\lambda d_z)\\(y+\lambda d_y)/(-z-\lambda d_z)\end{bmatrix} = \begin{bmatrix}-x/(z+\lambda d_z)\\-y/(z+\lambda d_z))\end{bmatrix} + \begin{bmatrix}-d_x/(z/\lambda+d_z)\\-d_y/(z/\lambda+d_z)\end{bmatrix}$$

$$\xrightarrow[\lambda\to\infty]{} \begin{bmatrix}-d_x/d_z\\-d_y/d_z\end{bmatrix}$$

If a line is parallel to the image plane, i.e. dz = 0, the projection of a point moving on that line towards infinity also goes to infinity (point at infinity). In the case where dx or dy is 0 only one coordinate of the VP goes to infinity. (Note that dx and dy can not be 0 at the same time since d is a unit vector).
However, if the line is not parallel to the image plane, the projecting a set of points moving on the line towards infinity converges to a finite point on the image plane, the so-called vanishing point of that line. Since the position of the vanishing point depends only on the orientation of the line, the vanishing point for all lines parallel to that line are the same.

Vanishing Points

- Geometric interpretation:
  - find vanishing point for $\boldsymbol{o} + \lambda \boldsymbol{d}$
  - shoot parallel line from origin: $\lambda \boldsymbol{d}$
  - intersect $\lambda \boldsymbol{d}$ with image plane
- Classify projections wrt # finite VPs (among the coordinate axis directions)

There is a nice geometric interpretation on what the vanishing point of a line with direction d is: The intersection of the image plane and a line in direction d through the center of projection (located in the origin for the standard projection).
Using vanishing points, we can construct perspective images of 3D objects (see the slides on 1-, 2- and 3-point perspectives).

Technical Projections

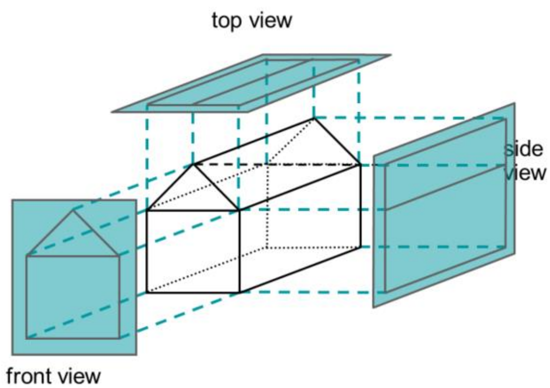
- parallel (0-point)
  - orthogonal
  - oblique

- preserve parallelism
- lengths
- angles if || to image plane

- perspective
  - 1-point
  - 2-point
  - 3-point

- perspective foreshortening
- more realistic appearance
- angles if || to image plane

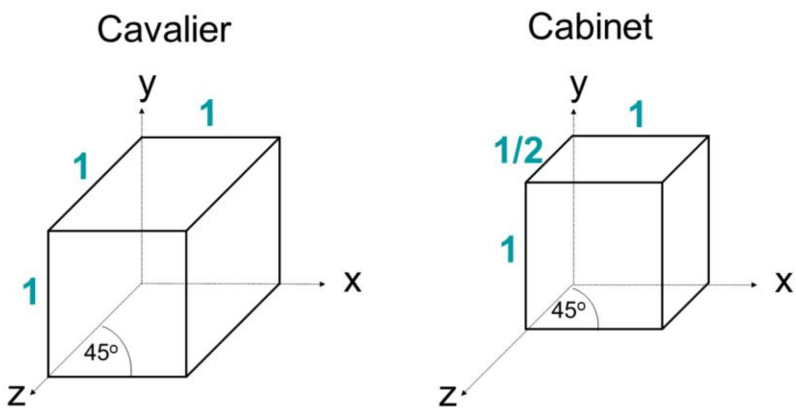
Orthogonal Projections



projection $\perp$ image plane

Orthogonal projections are common e.g. in architecture because it does not change distances or angles.

Oblique Projections



projection not ⊥ image plane

No vanishing points so you can do measurements in 2D as in a parallel projection but this also gives some intuition of the 3D object.

Perspective Projections

- **h** : projection of eye onto image plane
- δ : focal distance
- $\mathbf{z}_i$: vanishing point of coordinate axes
- $\mathbf{d}_i$: vanishing point of diagonals

Visual Computing Institute | Prof. Leif Kobbelt
Basic Techniques in Computer Graphics

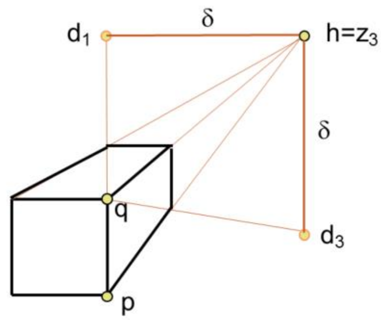
h is also called the principle point („Augpunkt“ in german).

1-, 2-, 3-Point Perspectives

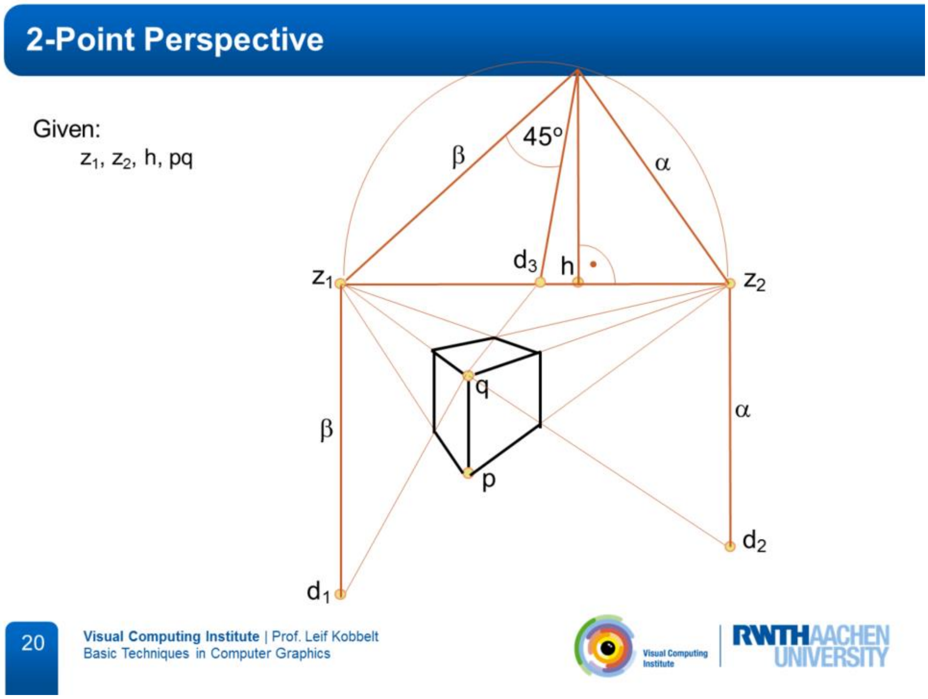
- count the number of vanishing points of the COORDINATE AXES
- count the number of ZERO entries in the fourth row (columns 1 to 3) of the projection matrix

1-Point Perspective

Given: z_3, h, δ, pq



The 1-point perspective gets its name since only one axis has a finite vanishing point z3, while the other two axis are parallel to the image plane. Thus the other lines of a cube (with side length 1) point towards the vanishing point, we only have to determine the length of these sides. To do so, we use the vanishing points of the diagonals of the sides of the cube.

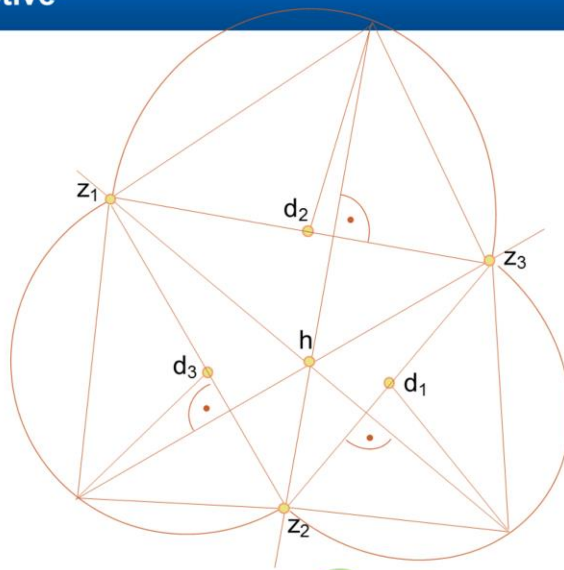


In the 2-point perspective we have two vanishing points for two sides of the cube. The third side is still parallel to the image plane (here: the „up-down“-direction).
The circle can be constructed with Thales’ Theorem (german: Satz des Thales).

3-Point Perspective

Given:

z_1, z_2, z_3, pq

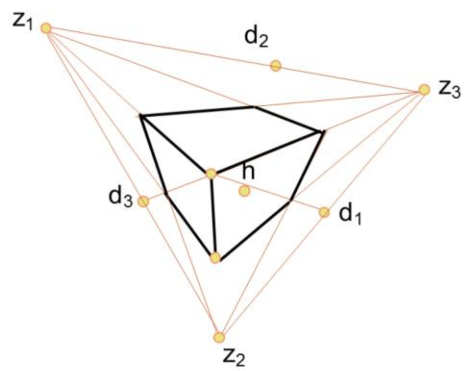


The 3-point perspective is the most general variant. The three semicircles are constructed analog to the semicircle in the 2-point perspective case.

3-Point Perspective

Given:

z_1, z_2, z_3, pq



Transformations of Lines & Polygons

- Line from A to B: $L(\alpha) = (1-\alpha)A + \alpha B$
- Transform line $\rightarrow$ transform endpoints?
- Check for {*affine, projective*} M:

$$\forall \alpha \in R\ \exists \beta \in R :$$
$$M(L(\alpha)) = (1-\beta)\, M(A) + \beta\, M(B)$$

But the aspect ratios are not preserved by projecting the endpoints to 2D.

Transformations of Lines & Polygons

An ideal wide angle lens will not create bended curves from straight lines as it is seen here. The image above was taken with a “fisheye” wide-angle lens which has very obvious distortions. Other lenses have distortions like this as well but much less obvious, we however assume an ideal point-hole camera which does not introduce such distortions.

Affine Transformations

<u>Affine</u> *M (preserves ratios)* :

$\forall \alpha \in R$:

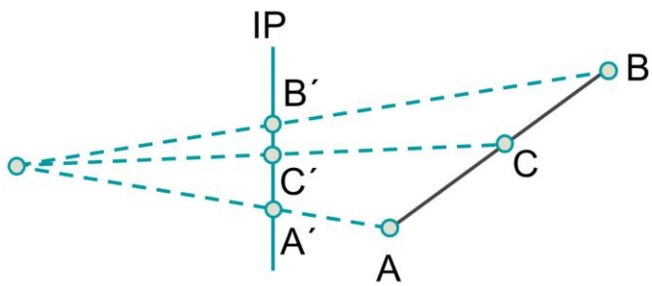
$M(L(\alpha)) = (1-\alpha)\, M(A) + \alpha\, M(B)$

For affine transforms, the ratios are preserved (A/C to C/B ratio is the same after the transform).

Projective Transformations

Projective M *(does not preserve ratios)*:

$A' = M(A), \quad B' = M(B), \quad C' = M(C)$



For perspective transformations however, this is not always true: the A/C to C/B ratio changed with the transformation because of the perspective foreshortening.

Invariants of Mappings

- linear map $\rightarrow$ linear combination

  $$L(\alpha A+\beta B) = \alpha L(A)+\beta L(B)$$

- affine map $\rightarrow$ affine combination

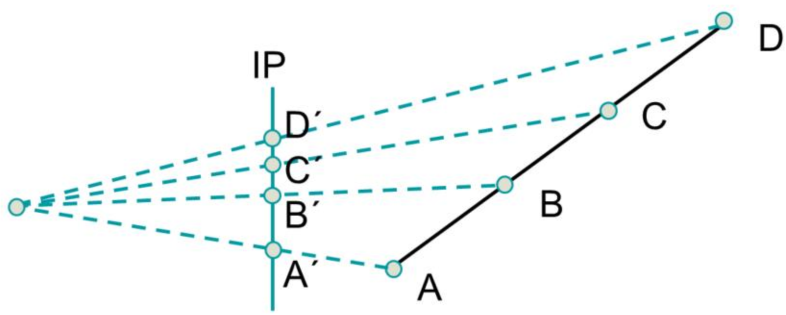
  $$M((1-\alpha)\, A+ \alpha\, B)= L((1-\alpha)\, A+ \alpha\, B) + T$$
  $$= (1-\alpha)\, M(A) + \alpha\, M(B)$$

- projective map $\rightarrow$ cross ratio !?

Projective Transformations

Cross Ratios invariant under projective transforms

$$CR(A,B,C,D) = CR(A',B',C',D')$$

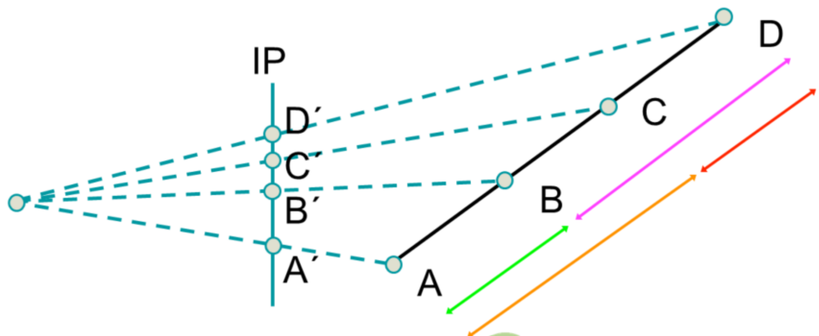


The cross ratio will be preserved and is defined as follows.

Projective Transformations

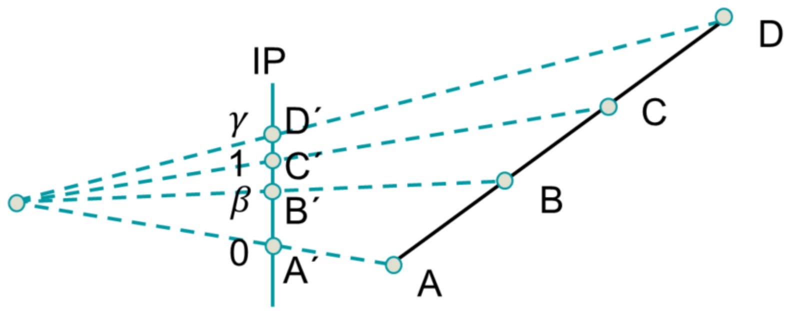
Cross Ratio is a ratio of ratios

$$CR(A,B,C,D) = \frac{ratio(A,B,D)}{ratio(A,C,D)} = \frac{\|\underline{B-A}\|/\|\underline{D-B}\|}{\|\underline{C-A}\|/\|\underline{D-C}\|}$$



Projective Transformations

- If $B = (1\text{-}\alpha)A + \alpha C$ then for which β is $B' = (1\text{-}\beta)A' + \beta C'$?



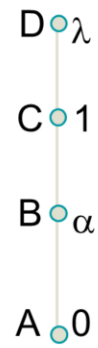
gamma = (D‘ - A‘) / (C‘ - A‘)

Projective Transformations

- If $B = (1-\alpha)A + \alpha C$ then for which β is $B' = (1-\beta)A' + \beta C'$?

- Let $\| C-A \| = 1$ then

$$
\begin{aligned}
CR(A,B,C,D) &= \frac{\|B-A\|/\|D-B\|}{\|C-A\|/\|D-C\|} = \frac{\frac{\|B-A\|}{\|C-A\|}\Big/\frac{\|D-B\|}{\|C-A\|}}{\frac{\|C-A\|}{\|C-A\|}\Big/\frac{\|D-C\|}{\|C-A\|}} \\
&= \frac{\alpha/(\lambda-\alpha)}{1/(\lambda-1)} =: CR(0,\alpha,1,\lambda)
\end{aligned}
$$



Projective Transformations

- If $B = (1-\alpha)A + \alpha C$ then for which β is $B' = (1-\beta)A' + \beta C'$?

- Consider $\lambda \to \infty$

$$CR(0,\alpha,1,\lambda)_{|\lambda\to\infty} = \left.\frac{\alpha/(\lambda-\alpha)}{1/(\lambda-1)}\right|_{\lambda\to\infty} = \alpha$$

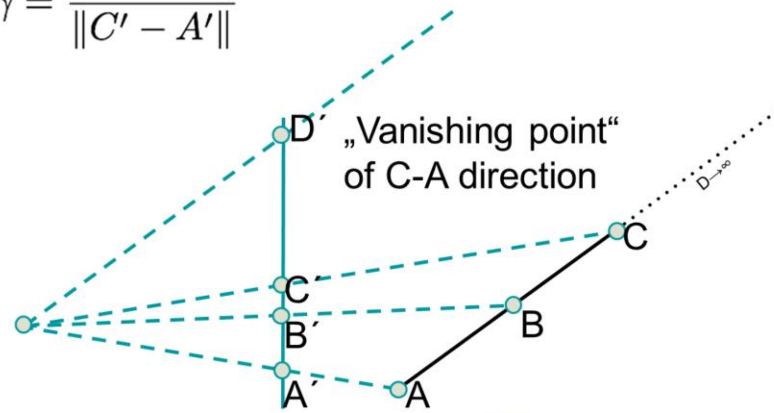
$$\alpha \stackrel{!}{=} CR(0,\beta,1,\gamma) \Leftrightarrow \alpha = \frac{\beta/(\gamma-\beta)}{1/(\gamma-1)} \Leftrightarrow \boxed{\beta = \frac{\alpha\gamma}{\gamma-1+\alpha}}$$

If lambda goes to infinity, D goes to the vanishing point and the cross ratio becomes alpha.

Projective Transformations

$$B' = (1-\beta)A' + \beta C' \text{ with } \beta = \frac{\alpha\gamma}{\gamma - 1 + \alpha}$$

$$\gamma = \frac{\|D' - A'\|}{\|C' - A'\|}$$



Shift the line to the projection center to get the dashed line which crosses the image plane at the vanishing point.

Camera Models

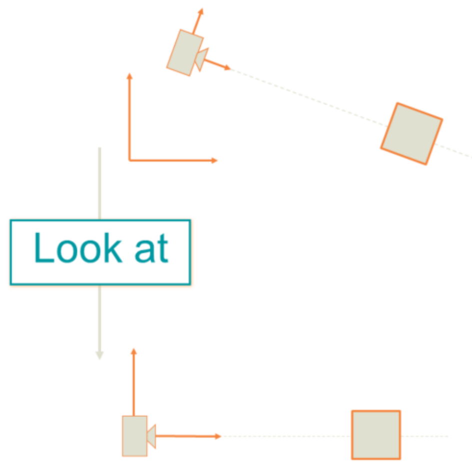
- Look at:
  - camera position and orientation
- Frustum:
  - camera parameters
- Viewport:
  - 2D coordinate system

These three transformations are often done individually because it‘s more easy to handle it this way. Look at defines the camera position/orientation (the extrinsic camera parameters) while the frustum defines the intrinsic parameters that defines for example the field of view. Both are handled by the application. The viewport defines how the unit-cube will get mapped onto the screen and here the screen resolution gets relevant. This will get handled by OpenGL (but the application has to tell OpenGL how large the viewport should be (see glViewport()).

Camera Models

- Parameters:
  - camera position C
  - view direction D
  - up vector U

- Move to standard camera



From the camera position, view direction and the up vector an orthogonal coordinate-system can be defined in which the scene has to be transformed. For practical reasons in some applications the up vector is not perpendicular to the viewing direction (but e.g. is the opposite direction of the gravity), in this case the perpendicular up vector has to be calculated before the transform.

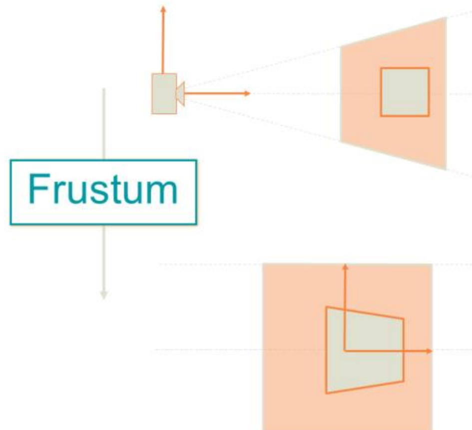
Look At

- Move camera C to origin
  → Translation by -C
- Orthonormal frame:
  – "right" R = D x U
  – "up" U = R x D
- Adjust orientation
  → Rotation [R/||R||, U/||U||, -D/||D||] → [X, Y, -Z]

The new coordinate-system should not scale the world but just ‚move' it, so the final vectors should get scaled to 1 to prevent unwanted scalings of the scene.
D gets a negative sign because by definition we look along the negative Z axis in OpenGL.

Camera Models

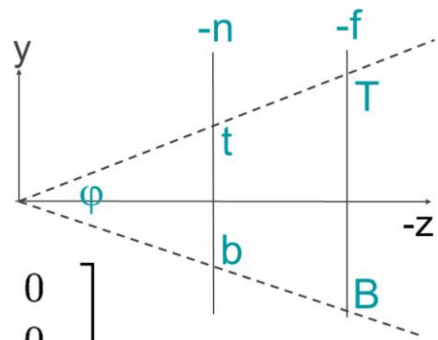
- Parameters:
  - near/far plane
  - opening angles or top / bottom
- Transform frustum to $[-1,1]^3$



The frustum (german: Frustum (if used in computer graphics), literally: Stumpf, hier: Pyramidenstumpf) maps the scene onto a cube.

Frustum

$t = n \cdot \tan(\varphi/2)$
$b = -n \cdot \tan(\varphi/2)$
$T = t \cdot f/n$
$B = b \cdot f/n$



$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

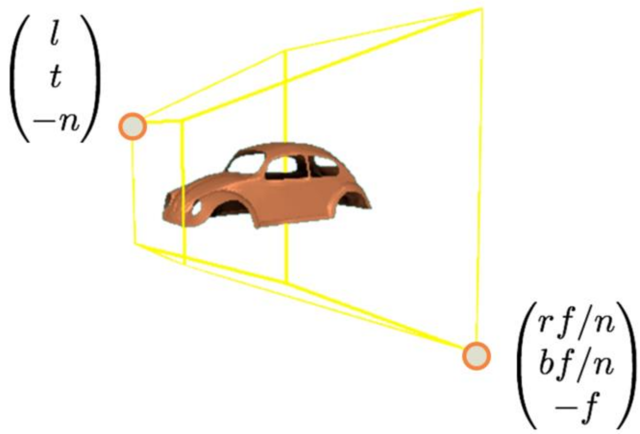
For numerical reasons (we later on have to store the distance to the camera as integers) we need a plane at the far end after which we clip away the geometry (far clipping plane, often just called far plane). We also need a near clipping plane (near plane).

t = top
b = bottom
l = left
r = right

Frustum

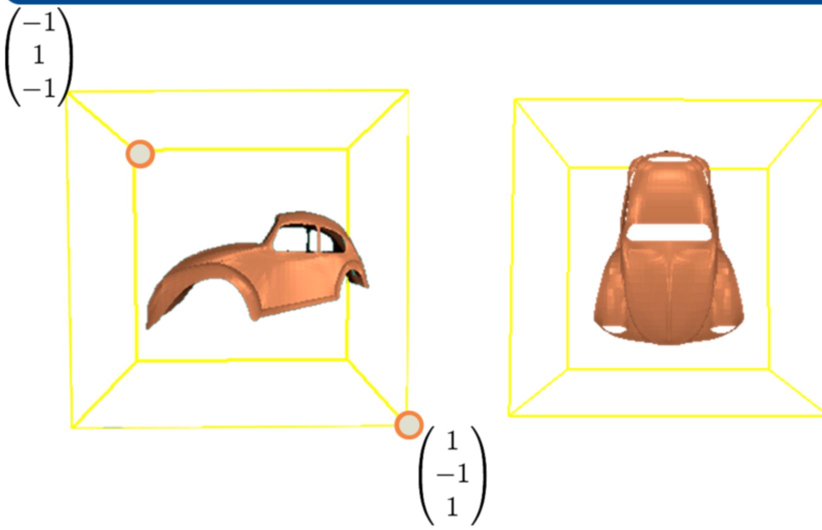
- Central projection → parallel projection eye point transforms to infinity
- Result's z holds depth information
- Use z for visibility determination
  accuracy: $n+\varepsilon$ vs. $f-\varepsilon$

Frustum Transform

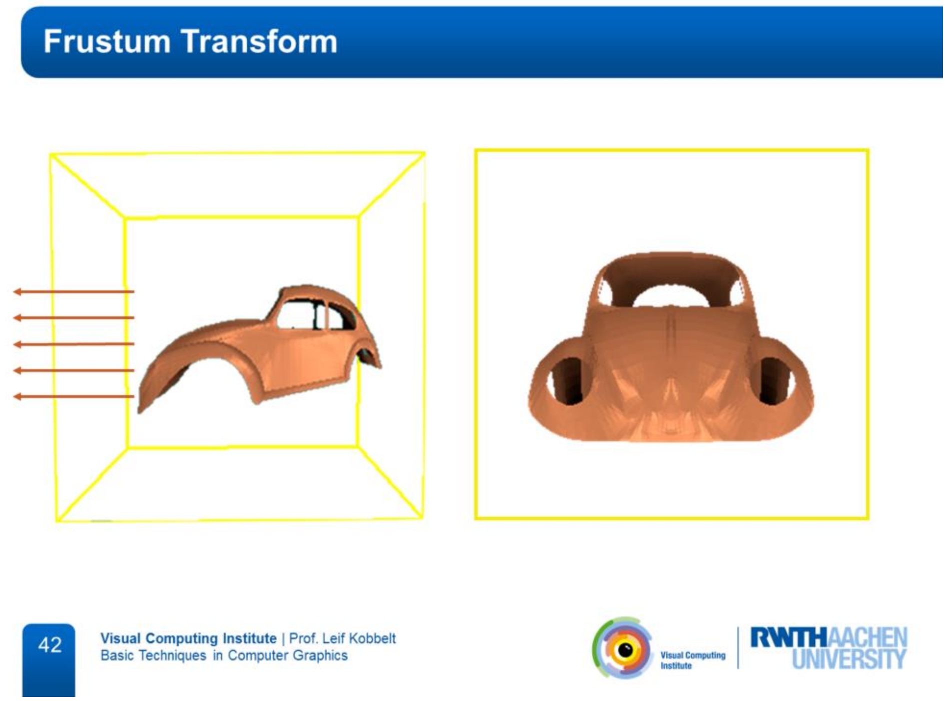


A scene inside of the frustum before the frustum transform.

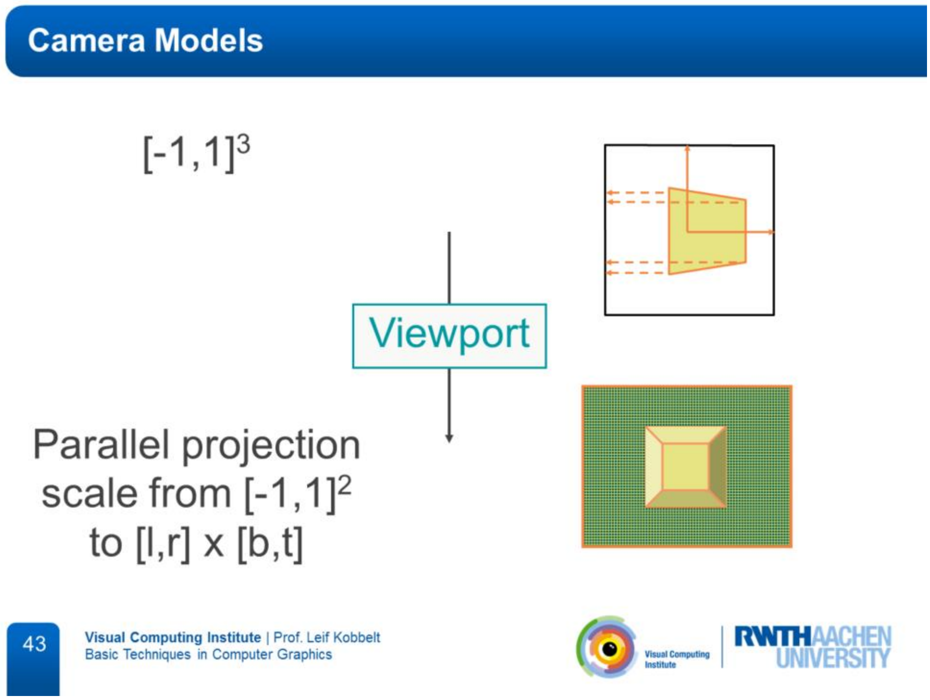
Frustum Transform



The transformed scene (and rendered perspectively again for illustrational reasons).

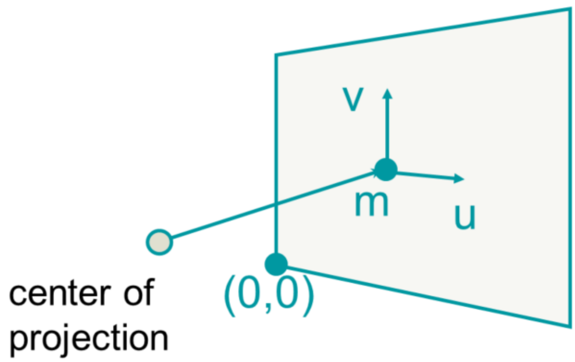


The cube seen from the front without additional perspective projection (just drop the Z coordinate) gives us the perspectively correct image. The Z coordinate is only used to determine visibility from now on (“what is the front most piece of geometry?”).



The last step is to go from the continuous space to the discrete monitor/window with a certain resolution and an aspect ratio of normally != 1.

2D Coordinate System



2D Coordinate System

- Project points onto image plane (3D):

$$n_x\, x + n_y\, y + n_z\, z + d = 0$$
$$\Leftrightarrow [n_x, n_y, n_z, d] \bullet [wx, wy, wz, w]^T = 0$$

- Find 2D coord. system for image plane:

$$E(\alpha,\beta) = m + \alpha\, u + \beta\, v \quad \text{with} \quad n^T u = n^T v = u^T v = 0$$

$$\alpha = u^T\ E(\alpha,\beta) - u^T m$$
$$\beta = v^T\ E(\alpha,\beta) - v^T m$$

$$u^T u = v^T v = 1$$

m,u and v are the vectors from the previous slide. This basistransform form 3D to 2D can be rewritten as a matrix (see next slide)

2D Coordinate System

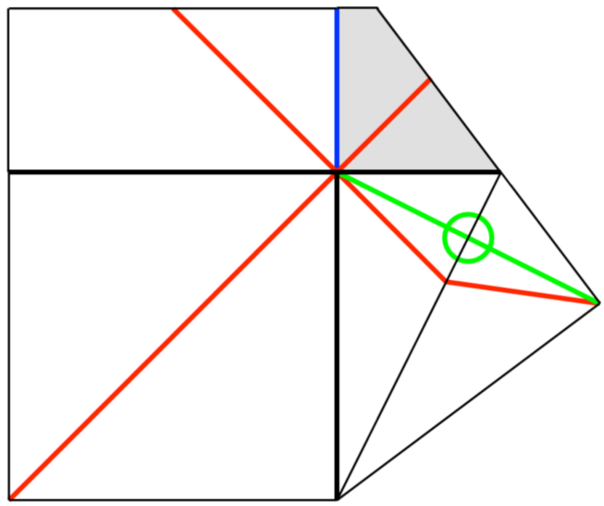
- Matrix representation

$$\begin{bmatrix} w\alpha \\ w\beta \\ w \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & -u^T m \\ v_x & v_y & v_z & -v^T m \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

- Standard projection

$$\begin{bmatrix} w\alpha \\ w\beta \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -m_x \\ 0 & 1 & 0 & -m_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

From 3D homogeneous coordinates to 2D homogeneous coordinates with one matrix.
Now we know everything to map points to the screen, lines and polygons however are not much more complicated. Lines in 3D are mapped to lines in 2D, curved lines from straight objects on photos are curved because of the lens distortion, not because the projection would project lines to curves. That‘s why in fact it is sufficient to map just the edgepoints of lines or polygons to 2D and connect them in the 2D space.



You can print this out and fold it to a corner of a cube (black lines are the edges, red are the diagonals)