Gernot Hoffmann

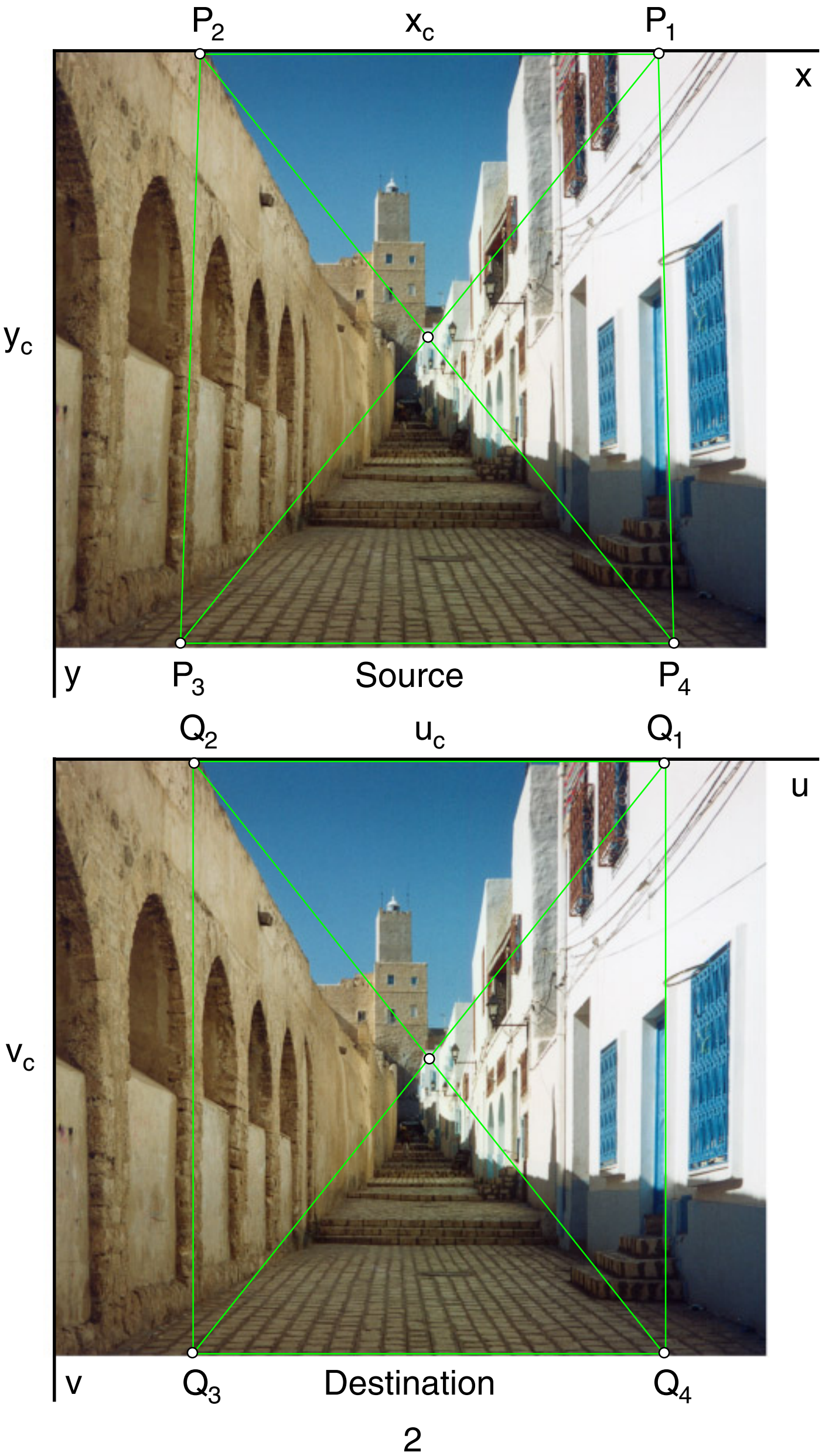
Perspective Rectification for Images

Table of Contents

1. Introduction 2
2. Mathematics 4
3. Perspective Transform by PostScript 5
4. References 13

1. Introduction

A general quadrilateral is mapped to a rectangle (or to another quadrilateral). This defines the transformation for the whole image.
The destination image is filled by regular scanlines, taking the values from the source image by bicubic interpolation.



2.1 Mathematics / Mapping

General perspective mapping is well-known. The formulas are valid for mapping from 3D to 2D (11 parameters) or like here for mapping from 2D to 2D (8 parameters).

$$\mathbf{x} = (x,y)^T$$
$$\mathbf{u} = (u,v)^T$$
$$\mathbf{a} = (a_u,a_v)^T$$
$$\mathbf{b} = (b_u,b_v)^T$$
$$\mathbf{c} = (c_u,c_v)^T$$

$$x = \frac{a_0 + \mathbf{a}^T\mathbf{u}}{1+\mathbf{c}^T\mathbf{u}}$$

$$y = \frac{b_0 + \mathbf{b}^T\mathbf{u}}{1+\mathbf{c}^T\mathbf{u}}$$

$$a_0 + \mathbf{a}^T\mathbf{u} + 0 + 0 + 0 - x\,\mathbf{c}^T\mathbf{u} = x$$
$$0 + 0 + 0 + b_0 + \mathbf{b}^T\mathbf{u} - y\,\mathbf{c}^T\mathbf{u} = y$$

Mapping four points delivers eight linear equations

$$\mathbf{p} = (a_0, a_u, a_v, b_0, b_u, b_v, c_u, c_v)^T$$
$$\mathbf{r} = (x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4)^T$$

$$\mathbf{A}\mathbf{p} = \mathbf{r}$$

$$\mathbf{A} = \begin{bmatrix} 1 & u_1 & v_1 & 0 & 0 & 0 & -x_1u_1 & -x_1v_1 \\ 1 & u_2 & v_2 & 0 & 0 & 0 & -x_2u_2 & -x_2v_2 \\ 1 & u_3 & v_3 & 0 & 0 & 0 & -x_3u_3 & -x_3v_3 \\ 1 & u_4 & v_4 & 0 & 0 & 0 & -x_4u_4 & -x_4v_4 \\ 0 & 0 & 0 & 1 & u_1 & v_1 & -y_1u_1 & -y_1v_1 \\ 0 & 0 & 0 & 1 & u_2 & v_2 & -y_2u_2 & -y_2v_2 \\ 0 & 0 & 0 & 1 & u_3 & v_3 & -y_3u_3 & -y_3v_3 \\ 0 & 0 & 0 & 1 & u_4 & v_4 & -y_4u_4 & -y_4v_4 \end{bmatrix}$$

Simplified center and estimation of the half edge lengths e_u, e_v
The true center is the intersection of the diagonals (next page)

$$u_c = x_c = (x_1 + x_2 + x_3 + x_4)/4$$
$$v_c = y_c = (y_1 + y_2 + y_3 + y_4)/4$$

$$e_u = [(x_1 - x_2) + (x_4 - x_3)]/4$$
$$e_v = [(y_4 - y_1) + (y_3 - y_2)]/4$$

$$u_1 = u_c + e_u \qquad v_1 = v_c - e_v$$
$$u_2 = u_c - e_u \qquad v_2 = v_c - e_v$$
$$u_3 = u_c - e_u \qquad v_3 = v_c + e_v$$
$$u_4 = u_c + e_u \qquad v_4 = v_c + e_v$$

2.2 Mathematics / Center

The true center is the intersection of the diagonals.

$$
\begin{aligned}
\mathbf{x}_{31} &= \mathbf{x}_3 - \mathbf{x}_1 \\
\mathbf{x}_{42} &= \mathbf{x}_4 - \mathbf{x}_2 \\
\mathbf{x}_{21} &= \mathbf{x}_2 - \mathbf{x}_1 \\
\mathbf{x} &= \mathbf{x}_1 + s\mathbf{x}_{31} \\
\mathbf{x} &= \mathbf{x}_2 + t\mathbf{x}_{42} \\
s\mathbf{x}_{31} - t\mathbf{x}_{42} &= \mathbf{x}_{21} \\
s x_{31} - t x_{42} &= x_{21} \\
s y_{31} - t y_{42} &= y_{21}
\end{aligned}
$$

Solve by *Cramer's* rule

$$
\begin{aligned}
D &= -x_{31} y_{42} + x_{42} y_{31} \\
S &= -x_{21} y_{42} + x_{42} y_{21} \\
s &= S/D \\
\mathbf{x}_c &= \mathbf{x}_1 + s\mathbf{x}_{31}
\end{aligned}
$$

2.3 Mathematics / Scaling

If coordinates x,y,u,v are given in millimeters or pixels then it is most likely that the linear equation system is ill-conditioned. Especially products like $x_i u_k$ create huge numbers.
As a result, the determinant of the system might become quite large, e.g 10^{22} for the next example.
Mainly for PostScript, which works in some interpreters by fixed point arithmetic, proper scaling can help to improve the accuracy.

Use **x** and **u** in the units [mm] or [pixel] but use scaled values for the calculation of the parameters, e.g. by s=500:

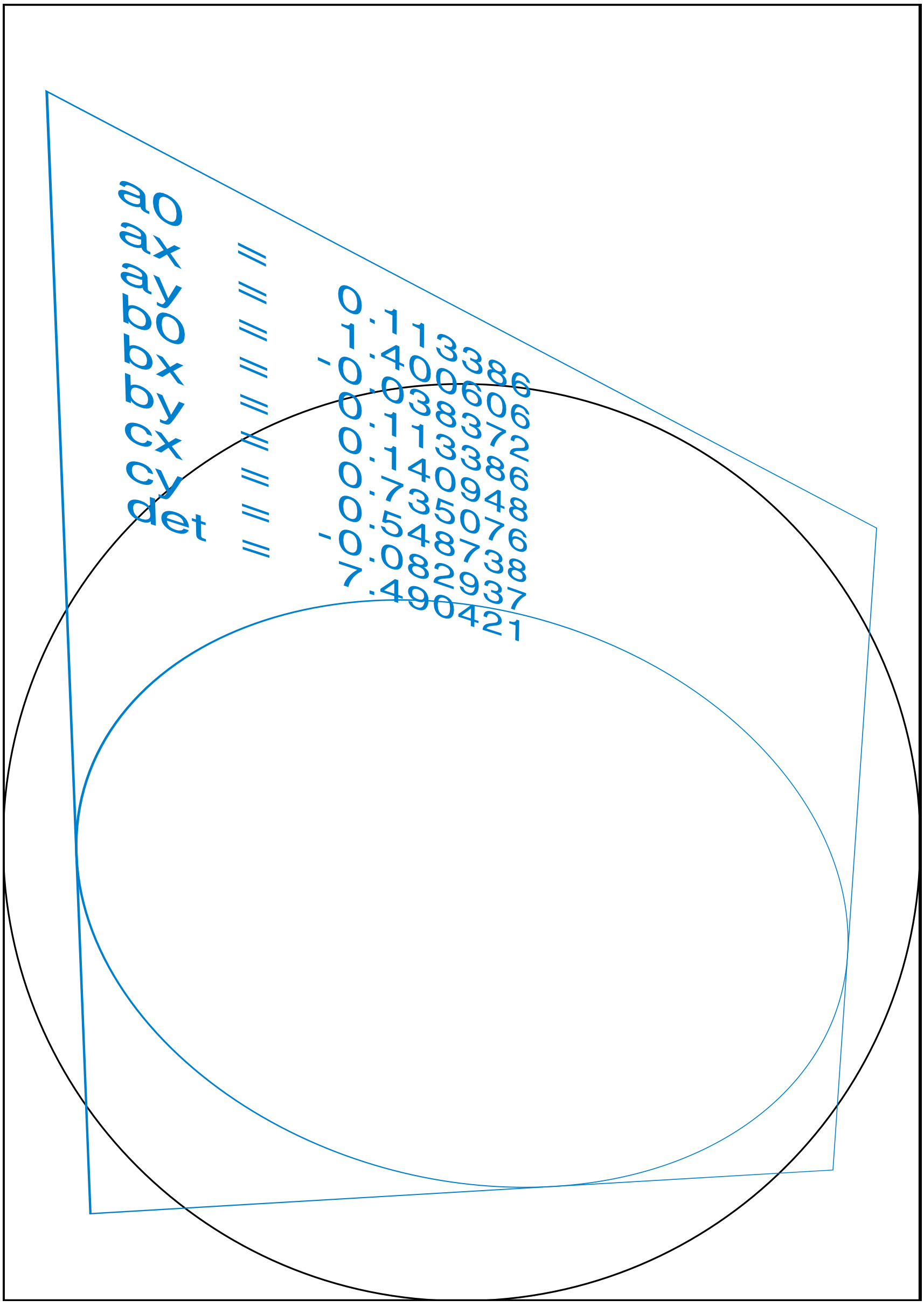
$$
\begin{aligned}
x/s &= \frac{a_0 + \mathbf{a}^T(\mathbf{u}/s)}{1 + \mathbf{c}^T(\mathbf{u}/s)} \\
y/s &= \frac{b_0 + \mathbf{b}^T(\mathbf{u}/s)}{1 + \mathbf{c}^T(\mathbf{u}/s)}
\end{aligned}
$$

Solve linear equation.
For the application use **x** and **u** in the units [mm] or [pixel] but modify the formula:

$$
\begin{aligned}
x &= s\,\frac{a_0 + (\mathbf{a}^T/s)\mathbf{u}}{1 + \mathbf{c}^T\mathbf{u}/s} \\
y &= s\,\frac{b_0 + (\mathbf{b}^T/s)\mathbf{u}}{1 + (\mathbf{c}^T/s)\mathbf{u}}
\end{aligned}
$$

3.1 Perspective Transform by PostScript / Example

PostScript does not have a generic perspective transform. This was established by using homogeneous coordinates, following the algorithm in the previous chapter.
The rendering of paths, including character paths, is a contribution by *Roger Willcocks*.



3.2 Perspective Transform by PostScript / Code

```
%!PS-Adobe-3.0   EPSF-3.0
%%BoundingBox:   0 0 596 842
%%Creator:       Gernot Hoffmann
%%Title:         Gen.PerspecScale
%%CreationDate:  August 15 2005/2003

% Define perspective 2D-2D transform by 4 corners
% of a rectangle in x,y and the quadrilateral in u,v

% General perspective mapping in homogeneous coordinates,

% Draw perspective text

% [u]    [ ax  ay  a0 ] [x]
% [v] = [ bx  by  b0 ]*[y]
% [w]    [ cx  cy  1  ] [1]

% Rearranged
% u = a0+ax*x+ay*y
% v = b0+bx*x+by*y
% w =  1+cx*x+cy*y

% u = u/w
% v = v/w

% /scal 500 def % is defined below

/mm {2.834646 mul } def

/N  8 def
/n1 N 1 sub def

/x1   0  mm def
/y1   0  mm def
/x2 210  mm def
/y2   0  mm def
/x3 210  mm def
/y3 297  mm def
/x4   0  mm def
/y4 297  mm def

/u1  20  mm def
/v1  20  mm def
/u2 190  mm def
/v2  30  mm def
/u3 200  mm def
/v3 177  mm def
/u4  10  mm def
/v4 277  mm def
```

3.3 Perspective Transform by PostScript / Code

```
/FillMat
{% retain x1 y1 ... x4 y4 for box; use r t
/r1 x1 scal div def
/r2 x2 scal div def
/r3 x3 scal div def
/r4 x4 scal div def
/s1 y1 scal div def
/s2 y2 scal div def
/s3 y3 scal div def
/s4 y4 scal div def
/u1 u1 scal div def
/u2 u2 scal div def
/u3 u3 scal div def
/u4 u4 scal div def
/v1 v1 scal div def
/v2 v2 scal div def
/v3 v3 scal div def
/v4 v4 scal div def
/ANN
[  1 r1 s1   0  0  0   r1 u1 mul neg  s1 u1 mul neg
   1 r2 s2   0  0  0   r2 u2 mul neg  s2 u2 mul neg
   1 r3 s3   0  0  0   r3 u3 mul neg  s3 u3 mul neg
   1 r4 s4   0  0  0   r4 u4 mul neg  s4 u4 mul neg
   0  0 0    1 r1 s1   r1 v1 mul neg  s1 v1 mul neg
   0  0 0    1 r2 s2   r2 v2 mul neg  s2 v2 mul neg
   0  0 0    1 r3 s3   r3 v3 mul neg  s3 v3 mul neg
   0  0 0    1 r4 s4   r4 v4 mul neg  s4 v4 mul neg ] def
/YN [ u1 u2 u3 u4 v1 v2 v3 v4 ] def
/XN  N array def
/TParams  [ (a0) (ax) (ay) (b0) (bx) (by) (cx) (cy) ] def
} def

/AParams
{
/a0 XN 0 get def
/ax XN 1 get scal div def
/ay XN 2 get scal div def
/b0 XN 3 get def
/bx XN 4 get scal div def
/by XN 5 get scal div def
/cx XN 6 get scal div def
/cy XN 7 get scal div def
} def

% Path transforms by Roger Willcocks

/path 10000 array def
/pathindex 0 def

% Push value onto path array (marked as executable)

/Tpush
{ cvx
  path pathindex 3 -1 roll put
 /pathindex pathindex 1 add def
} def

% Pop x and y from op stack, transform by ‘mat’ above
% push result onto path array
```

3.4 Perspective Transform by PostScript / Code

```
/Xform
{ 10 dict
  begin
 /y exch def
 /x exch def
  mat 0 get x mul mat 1 get y mul add mat 2 get add
  mat 3 get x mul mat 4 get y mul add mat 5 get add
  mat 6 get x mul mat 7 get y mul add mat 8 get add
 /w exch def
 /v exch def
 /u exch def
  v w div scal mul
  u w div scal mul
  end
  Tpush Tpush
} def

% Transform current path by transform ‘mat’
/Transformpath
{% matrix currentmatrix % save CTM;  Can cause problems !
  % matrix defaultmatrix setmatrix
 /pathindex 0 def
  flattenpath
 { Xform /moveto Tpush }
 { Xform /lineto Tpush }
 { x }
 {/closepath Tpush } pathforall
  newpath
% execute the path array to create transformed path
  path 0 pathindex getinterval cvx exec
  % setmatrix % restore CTM
} def

/stroke
{ strokepath
  Transformpath
//fill
} def

/fill
{ Transformpath
//fill
} def

/show
{ true charpath
  fill
} def

/i   0 def /j   0 def /k 0 def
/dA  0 def /max 0 def /s 0 def
/h   0 def /q   0 def
/aik 0 def /akk 0 def
/bik 0 def /bkk 0 def
/pa  0 def /ca  0 def
```

3.5 Perspective Transform by PostScript / Code

```
/m
% Memory map; input i,k on stack; output N*i+k
{ exch
  N mul add
} bind def

/HoGaussP
{% Solve  ANN*XN=YN for XN and det(ANN)
 % N equations for N unknowns
 % Overwrite all arrays
 % No error detection
 % Originally by H.R.Schwartz: Numerische Mathematik, B.G.Teubner, 1993
 % Modified by G.Hoffmann for zero-based onedimensional arrays for PS
 % Uses external memory mapping function m
/pa N array def
/ca N array def
/n1 N 1 sub def
/dA 1 def
0 1 n1 1 sub
{/k exch def
 /max 0 def
  pa k 0 put
  k 1 n1
  {/i exch def
   /s 0 def
    k 1 n1
    {/j exch def
     /s s ANN i j m get abs add def
    } for %j
     /bik ANN i k m get abs def
     /q bik s div def
      q max gt {/max q def pa k i put } if
  } for %i
    pa k get k ne
    {/dA dA neg def
      0 1 n1
      {/j exch def
       /h ANN k j m get def
        ANN k j m ANN pa k get j m get put
        ANN pa k get j m h put
      } for %j
    } if
   /dA dA ANN k k m get mul def
    k 1 add 1 n1
    {/i exch def
     /aik ANN i k m get def
     /akk ANN k k m get def
     /bkk akk abs def
      bkk 1 lt
      {/bik aik abs def
      } if
      ANN i k m aik akk div put
      k 1 add 1 n1
      {/j exch def
        ANN i j m ANN i j m get ANN i k m get ANN k j m get mul sub put
      } for %j
     } for %i
} for %k
 /dA dA ANN n1 n1 m get mul def
```

3.6 Perspective Transform by PostScript / Code

```
0 1 n1 1 sub
{/k exch def
  pa k get k ne
 {/h YN k get def
   YN k YN pa k get get put
   YN pa k get h put
 } if
} for %k
0 1 n1
{/i exch def
  ca i YN i get put
  0 1 i 1 sub
  {/j exch def
    ca i ca i get ANN i j m get ca j get mul sub put
  } for %j
} for %i
n1 -1 0
{/i exch def
 /s ca i get def
  i 1 add 1 n1
  {/k exch def
   /s s ANN i k m get YN k get mul sub def
  } for %k
 /bik ANN i i m get abs def
  YN i s ANN i i m get div put
} for %i
0 1 n1
{/i exch def
  XN i YN i get put
} for %i
} def % HoGaussP

/Box
{x1 y1 moveto
 x2 y2 lineto
 x3 y3 lineto
 x4 y4 lineto
 closepath
 stroke
} def

/Box2
{ 10 mm  10 mm moveto
 200 mm  10 mm lineto
 200 mm 287 mm lineto
  10 mm 287 mm lineto
 closepath
 stroke
} def
```

3.7 Perspective Transform by PostScript / Code

```
/Shownum
% Draw number by string
% Version May 15 2004 / uses rounding
%
%  tx0 ty0 nu :on stack
%
%  tx0    Position        not overwritten
%  ty0
%  nu     Input number    not overwritten  nu =+-999999
%  fh     Font height     not overwritten
%  tms    Mantissa number of characters    tms=0...6 Global
%  tms=3  Example
%  input  -23.56789  -999.99    0.4567  9999.123456
%  result -23.568    -999.990   0.467   9999.123
%  Postscript number to string is not well defined
%  e.g. 1E-5 instead of 0.00001
%  We use a straightforward BCD conversion.
%  This is always affected by round-off errors
%  because of 32-bit arithmetic
%  Results are different, depending on the interpreter
{ /nu  exch def
  /ty0 exch def
  /tx0 exch def
  /tfw fh 0.6 mul def                    % character distance
  /tna nu abs 10 tms neg exp 0.500001 mul add def
  /tdec 1E5 def
  /tchr 1 string def
   tna 999999.1 lt                       % larger number replaced by #
  /tmm true def                          % sign
  {/tx0 tx0 tfw 6 mul sub def
   /tz 0 def
  1 1 5                                  % first 5 digits, no leading 0
  { pop
   /tk 0 def
   { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
   } loop
    tk 0 ne {/tz tz 1 add def} if
    tz 0 ne
    { tx0 ty0 moveto tk tchr cvs show
    } if
    tz 1 eq nu 0 lt and tmm and          % minus  Correction 28/03/2005
   { tx0 tfw 0.7 mul sub ty0 moveto (-) show
    /tmm false def
   } if
 /tdec tdec 0.1 mul def
 /tx0 tx0 tfw add def
} for
 /tk 0 def                               % leading 0
 { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
 } loop
   tmm nu 0 lt and                       % minus
  { tx0 tfw 0.7 mul sub ty0 moveto (-) show
  } if
  tx0 ty0 moveto tk tchr cvs show
 /tdec tdec 0.1 mul def
 /tx0 tx0 tfw add def
tms 0 gt                                 % for float
{ tx0 ty0 moveto (.) show
 /tx0 tx0 tfw 0.5 mul add def
1 1 tms
```

3.8 Perspective Transform by PostScript / Code

```
  { pop
   /tk 0 def
   { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
   } loop
     tx0 ty0 moveto tk tchr cvs show
   /tdec tdec 0.1 mul def
   /tx0 tx0 tfw add def
  } for
  } if
 }{ tx0 tfw sub ty0 moveto (#) show} ifelse
} def

% Unity transform
/scal 1 def
/mat
[ 1 0 0
  0 1 0
  0 0 1 ] def
0 setgray
0.8 mm setlinewidth
Box
0.4 mm setlinewidth
105 mm 105 mm 105 mm 0 360 arc stroke

% Perspective transform
/scal 500 def
FillMat
HoGaussP
AParams
/mat
[ ax  ay a0
  bx  by b0
  cx  cy  1 ] def
0.0 0.5 0.8 setrgbcolor
Box
105 mm 105 mm 105 mm 0 360 arc stroke

/fh 10 mm def
/Helvetica findfont fh scalefont setfont
/xa  10 mm def
/ya 280 mm def
/buf 20 string def

/d1x fh 2.0 mul def
/d2x fh 4.5 mul def

/tms 6 def

0 1 n1
{/i exch def
  xa ya moveto  TParams i get show
  xa d1x add ya moveto   (=) show
  xa d2x add ya XN i get  Shownum
 /ya ya fh sub def
} for
 xa ya moveto           (det) show
 xa d1x add ya moveto    (=) show
 xa d2x add ya dA Shownum

showpage
```

4. References

More References are in [1]

[1] G.Hoffmann
Planar Projections
http://www.fho-emden.de/~hoffmann/project18032004.pdf

[2] G.Hoffmann
Rectification by Photogrammetry
http://www.fho-emden.de/~hoffmann/sans04012001.pdf

[3] G.Hoffmann
Bicubic Interpolation (and other methods)
http://www.fho-emden.de/~hoffmann/bicubic03042002.pdf

This doc:
Perspective Rectification for Images
http://www.fho-emden.de/~hoffmann/persprect13052005.pdf

Gernot Hoffmann
August 16 / 2005
Website
Load Browser / Click here