APPLYING HOMOGRAPHIES TO IMAGES

Computer Vision (CS364/AM792)

The MATLAB code below demonstrates how we can use bilinear interpolation to transform an input image A according to a given 3×3 projective transformation matrix H. Every pixel (x, y) in A is transformed in homogeneous coordinates as $[u\ v\ w]^T = H[x\ y\ 1]^T$, so that the corresponding pixel in the output image is given by $(x', y') = (u/w, v/w)$.

```
function B = applyhomography(A,H)
% cast the input image to double precision floats
A = double(A);
% determine number of rows, columns and colour channels of A
m = size(A,1);
n = size(A,2);
c = size(A,3);
% determine size of output image by forward-transforming the four corners of A
p1 = H*[1; 1; 1]; p1 = p1/p1(3);
p2 = H*[n; 1; 1]; p2 = p2/p2(3);
p3 = H*[1; m; 1]; p3 = p3/p3(3);
p4 = H*[n; m; 1]; p4 = p4/p4(3);
minx = floor(min([p1(1) p2(1) p3(1) p4(1)]));
maxx = ceil(max([p1(1) p2(1) p3(1) p4(1)]));
miny = floor(min([p1(2) p2(2) p3(2) p4(2)]));
maxy = ceil(max([p1(2) p2(2) p3(2) p4(2)]));
nn = maxx - minx + 1;
mm = maxy - miny + 1;
% initialize the output with white pixels
B = zeros(mm,nn,c) + 255;
% pre-compute the inverse of H (we'll be applying that to the pixels in B)
Hi = inv(H);
% loop through B's pixels
for x = 1:nn
    for y = 1:mm
        % compensate for the shift in B's origin, and homogenize
        p = [x + minx - 1; y + miny - 1; 1];
        % apply the inverse of H
        pp = Hi*p;
        % de-homogenize
        xp = pp(1)/pp(3);
        yp = pp(2)/pp(3);
        % perform bilinear interpolation
        xpf = floor(xp); xpc = xpf + 1;
        ypf = floor(yp); ypc = ypf + 1;
        if (xpf > 0) && (xpc <= n) && (ypf > 0) && (ypc <= m)
            B(y,x,:) =  (xpc - xp)*(ypc - yp)*A(ypf,xpf,:) ...
                      + (xpc - xp)*(yp - ypf)*A(ypc,xpf,:) ...
                      + (xp - xpf)*(ypc - yp)*A(ypf,xpc,:) ...
                      + (xp - xpf)*(yp - ypf)*A(ypc,xpc,:);
        end
    end
end
% cast the output image back to unsigned 8-bit integers
B = uint8(B);
end
```

This function follows the (x, y) convention for pixel coordinates, which differs from the (*row*, *column*) convention. You will note that y corresponds to row index and x to column index. The matrix H must be set up accordingly.

The size of the output is determined automatically, and the output will contain the entire transformed image on a white background. This means that the origin of the output image may no longer coincide with the top-left pixel. In fact, after executing this function, the true origin $(1, 1)$ will be located at point $(2 - \texttt{minx}, 2 - \texttt{miny})$ in the output image (see if you can figure out why!).

Willie Brink
August 2013