Digital Image Processing

BB1603391 116033910045 修宇亮

Problem 6 Requirement

6. **Geometric transform (test image: ray_trace_bottle.tif)**

Develop a geometric transform program that will rotate, translate, and scale an image by specified amounts, using the nearest neighbor and bilinear interpolation methods, respectively.

Problem 6 solution

Scale Result(nearest neighbor VS bilinear interpolation)

main.m

MATLAB

```
orig_img = imread('ray_trace_bottle.tif');
[M,N] = size(orig_img);
scale_params = [4,4];
trans_params = [-10.3,-10.7];
rotate_param = 45;
scale_nn_result = scale(orig_img,'nn',scale_params(1),scale_params(2));
scale_bilinear_result = scale(orig_img,'bilinear',scale_params(1),scale_params(2));
subplot(1,2,1),imshow(scale_nn_result);title('nn result');
subplot(1,2,2),imshow(scale_bilinear_result);title('bilinear result');

trans_nn_result = translate(orig_img,'nn',trans_params(1),trans_params(2));
trans_bilinear_result = translate(orig_img,'bilinear',trans_params(1),trans_params(2));
subplot(1,2,1),imshow(trans_nn_result);title('nn result');
subplot(1,2,2),imshow(trans_bilinear_result);title('bilinear result');

rotate_nn_result = imrotate(orig_img,45,'nearest');
rotate_bilinear_result = imrotate(orig_img,45,'bilinear');
subplot(1,2,1),imshow(rotate_nn_result);title('nn result');
subplot(1,2,2),imshow(rotate_bilinear_result);title('bilinear result');
```

scale.m

```MATLAB
function scale_result = scale(input,type,cx,cy)
    [M,N] = size(input);
    [scale_M,scale_N] = deal(ceil(M*cy),ceil(N*cx));
    scale_result = zeros(scale_M,scale_N);

    for m=1:scale_M
        for n=1:scale_N
            switch type
                case 'nn'
                    recov_m = min(M,max(1,round(m/cy)));
                    recov_n = min(N,max(1,round(n/cx)));
                    scale_result(m,n) = input(recov_m,recov_n);
                case 'bilinear'
                    scale_result(m,n) = bilinear(input,m,n,'scale',containers.Map({'c
            end
        end
    end

    scale_result = uint8(scale_result);
end
```

translate.m

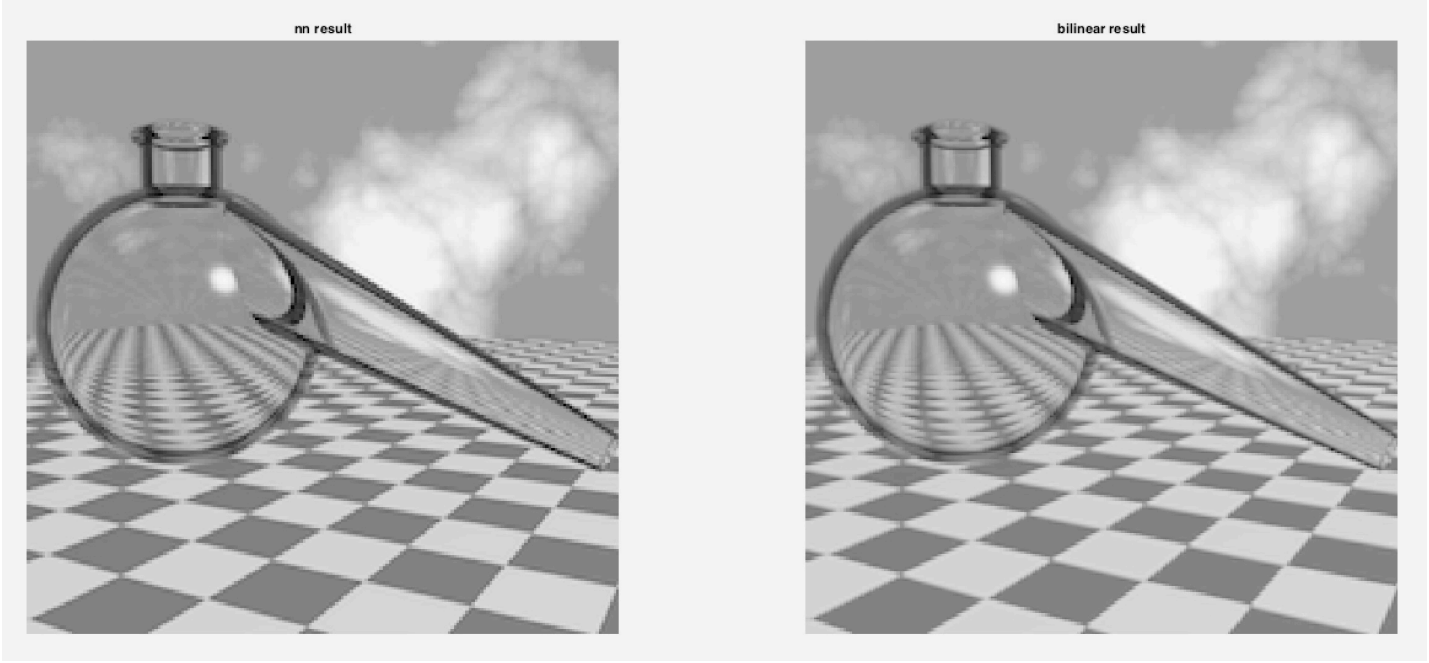
MATLAB

```
function trans_result = translate(input,type,tx,ty)
    [M,N] = size(input);
    [trans_M,trans_N] = deal(ceil(M+2*abs(ty)),ceil(N+2*abs(tx)));
    trans_result = zeros(trans_M,trans_N);
    input_init = zeros(trans_M,trans_N);
    for m=1:M
        for n=1:N
            input_init(m+round(abs(ty)),n+round(abs(tx))) = input(m,n);
        end
    end
    if tx == fix(tx) || ty == fix(tx)
        type = 'nn';
    end


    switch type
        case 'nn'
            for m=1:trans_M
                for n=1:trans_N
                    trans_result(m,n) = input_init(max(1,min(trans_M,m-round(ty))),ma
                end
            end
        case 'bilinear'
            for m=1:trans_M
                for n=1:trans_N
                    trans_result(m,n) = bilinear(input_init,m,n,'trans',containers.Ma
                end
            end
    end

    trans_result = uint8(trans_result);
end
```

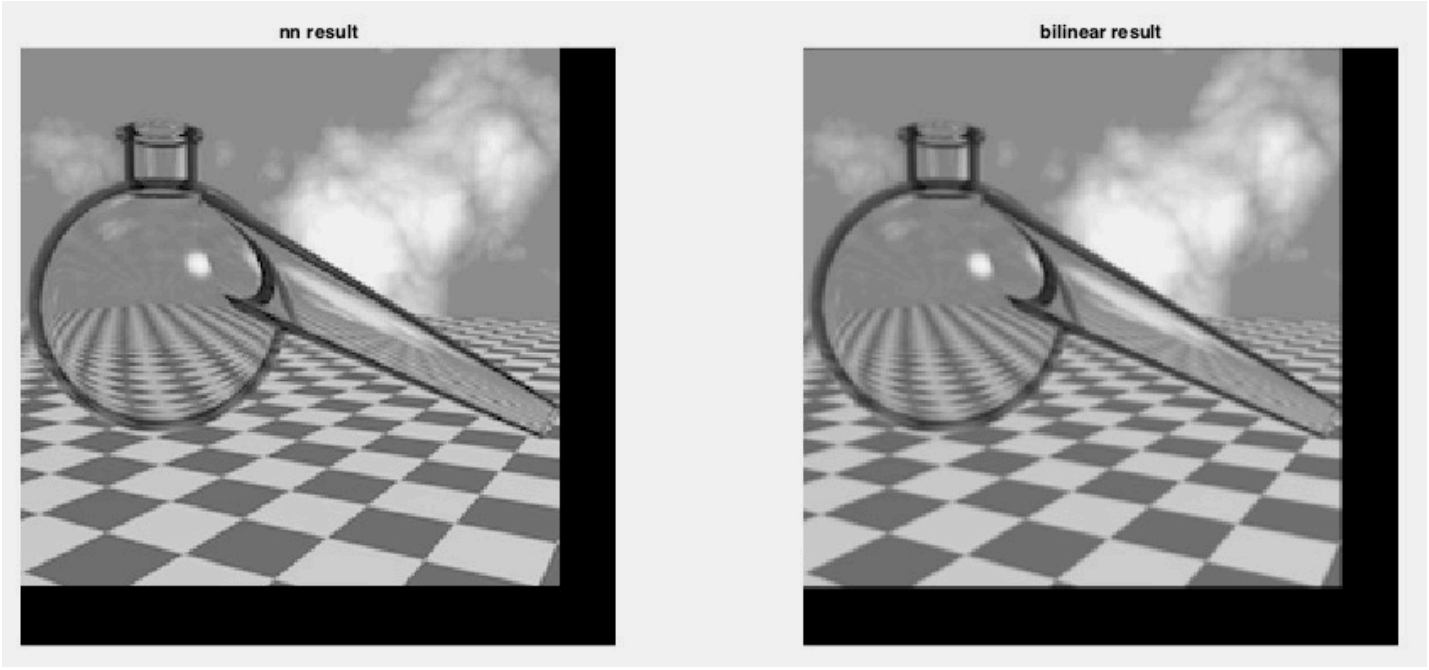
bilinear.m

```MATLAB
function result_gray = bilinear(input,m,n,type,params)
    [M,N] = size(input);
    if strcmp(type,'scale')
        m_value = m/params('cy');
        n_value = n/params('cx');
    elseif strcmp(type,'trans')
        m_value = m-params('ty');
        n_value = n-params('tx');
    end

    lt_pos = [min(M,max(1,floor(m_value))),min(N,max(1,floor(n_value)))];
    ld_pos = [max(1,min(M,ceil(m_value))),min(N,max(1,floor(n_value)))];
    rt_pos = [min(M,max(1,floor(m_value))),max(1,min(N,ceil(n_value)))];
    rd_pos = [max(1,min(M,ceil(m_value))),max(1,min(N,ceil(n_value)))];
    [lt_gray,ld_gray,rt_gray,rd_gray] = deal(input(lt_pos(1),lt_pos(2)),...
        input(ld_pos(1),ld_pos(2)),input(rt_pos(1),rt_pos(2)),input(rd_pos(1),rd_pos(2
    top_gray = (n_value-lt_pos(2))/(rt_pos(2)-lt_pos(2))*(rt_gray-lt_gray)+lt_gray;
    down_gray = (n_value-lt_pos(2))/(rt_pos(2)-lt_pos(2))*(rd_gray-ld_gray)+ld_gray;
    result_gray = (ld_pos(1)-m_value)/(ld_pos(1)-lt_pos(1))*(top_gray-down_gray)+down_
end
```

scale.m



Translate Result(nearest neighbor VS bilinear interpolation)



Rotate Result(nearest neighbor VS bilinear interpolation)